The Pennsylvania State University
The Graduate School

DEFENSE OF BACKDOOR ATTACKS AGAINST DEEP NEURAL NETWORK CLASSIFIERS

A Dissertation in
Electrical Engineering
by
Zhen Xiang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2022

The dissertation of Zhen Xiang was reviewed and approved by the following:

David J. Miller
Professor of Electrical Engineering
Dissertation Advisor
Chair of Committee

George Kesidis
Professor of Electrical Engineering and Computer Science

Constantino Lagoa
Professor of Electrical Engineering

Anna Squicciarini
Professor of Information Sciences and Technology

Kultegin Aydin
Professor of Electrical Engineering
Program Head

Abstract

Deep neural network classifiers (DNNs) are increasingly used in many applications, including security-sensitive ones, but they are vulnerable to adversarial attacks. An emerging type of backdoor attack aims to induce test samples from one or more source classes to be misclassified to a target class, whenever a backdoor pattern is present. A backdoor attack can be easily launched by poisoning the DNN's training set with a small set of samples originally from the source classes, embedded with the same backdoor pattern that will be used at test time, and labeled to the target class. A successful backdoor attack will not degrade the accuracy of the DNN on clean, backdoor-free test samples; thus are stealthy and undetectable using (e.g.) validation set accuracy.

Defending backdoor attacks is very challenging due to the practical constraints associated with the defense scenario. Backdoor defenses deployed during the training phase aim to detect if the training set is poisoned or not; if there is poisoning, the samples with the backdoor pattern should be identified and removed before training. For this defense scenario, there is no subset of training samples guaranteed to be clean that can be used for reference. Backdoor defenses deployed post-training aim to detect if a pre-trained DNN is backdoor attacked or not. For this defense scenario, the defender is assumed not to have access to the DNN's training set or to any samples embedded with the backdoor pattern used by the attack, if there is actually an attack. Backdoor defenses deployed during a DNN's inference phase aim to detect if a test sample is embedded with a backdoor pattern. For this scenario, the defender does not know a priori the backdoor pattern used by the attacker, and has to make immediate detection inferences for each test sample.

In this thesis, we mainly focus on the image domain (like most related works) and propose several backdoor defenses deployed during-training and post-training. For the most challenging post-training defense scenario, we first propose a reverse-engineering defense (RED) which requires neither access to the DNN's training set nor to any clean classifiers for reference. Then, we propose a Lagrange-based RED (L-RED) to improve the time and data efficiency of RED. Moreover, we propose a maximum achievable misclassification fraction (MAMF) statistic to address the challenge of reverse-engineering a very common type of patch replacement backdoor pattern; and an expected transferability (ET) statistic to address two-class, multi-attack scenarios where typical anomaly detection approaches of REDs are not applicable. For the before/during training defense scenario, we first propose a clustering-based approach with a cluster impurity (CI) statistic to distinguish training samples with the backdoor pattern from clean target class samples.

We also propose a defense inspired by REDs (for the post-training scenario) which not only identify training samples with the backdoor pattern, but also "restore" these samples by removing a reverse-engineered backdoor pattern.

While backdoor attacks and defenses have been extensively investigated for images, we extend these studies to domains other than images. In particular, we devise the first backdoor attack against point cloud classifiers (dubbed "point could backdoor attack" (PCBA)) – PC classifiers play important roles in applications like autonomous driving. We also extend our RED for images to defend against such PCBAs by leveraging the properties of common point cloud classifiers.

In summary, we provide solutions to practical users to protect their devices/systems/applications that involve DNNs from backdoor attacks. Our works also provide insights to the machine learning community on the effect of training set deviation, feature reverse-engineering, and neuron functional allocation; moreover, the empirical evaluation protocols adopted in this thesis can potentially be a reference for establishing a standard for measuring the security level of DNNs against backdoor attacks.

Table of Contents

List of Figures

List of Tables

Notations

$\mathbf{x}$	A vector
$\|\|\mathbf{x}\|\|_p$	ℓ_p norm of vector $\mathbf{x}$
$\mathbf{0}$	A vector of all zeros
$\mathbf{X}^T$	Transpose of matrix $\mathbf{X}$
$\mathbf{I}$	The identity matrix
$\mathcal{S}$	A set and its cardinality
$\|\mathcal{S}\|$	Cardinality of set $\mathcal{S}$
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^n$	The set of n-dimensional vectors
$\mathbb{1}(\cdot)$	The indicator function of the argument
$\mathrm{P}(\cdot)$	Probability of an event
$\odot$	Element-wise multiplication
$\mathcal{O}(\cdot)$	Big "$\mathcal{O}$" notation for computational complexity
$D_{\mathrm{KL}}(\cdot\|\|\cdot)$	The Kullback–Leibler divergence
$f : \mathcal{X} \to \mathcal{Y}$	A function f with input domain $\mathcal{X}$ and output domain $\mathcal{Y}$
$p(\cdot\|\cdot)$	The conditional probability

Acknowledgments

I would like to express my great gratitude towards my advisors Dr. David J. Miller and Dr. George Kesidis, who have patiently trained me from an immature researcher to a qualified Ph.D.. Their patient mentorship and superb vision towards research are significant to my personal development of research and learning skills. Especially, I appreciate their rigorous edits to every detail of my papers and dissertation, which are very helpful to my current achievements in research, and the chances they offered me to work on a variety of research and engineering problems.

It is my great honor to invite Dr. Constantino Lagoa and Dr. Anna Squicciarini to serve on my dissertation committee. They have raised keen comments on my research as well as the scope of this dissertation. I would like to thank Dr. Lagoa for being the instructor of the analysis course I took, and for his consultation on the mathematical aspects of my dissertation. I would like to thank Dr. Squicciarini for her sharp insights towards machine learning and security, which are the main areas of focus in this dissertation. I am very fortunate to have them on my committee – their experience and suggestions are valuable for my improving the presentation of my works.

Part of this dissertation would probably not be completed without the help of the following talented researchers. I am glad to collaborate with Dr. Siheng Chen, whose experience and insights towards point cloud processing are critical to the works presented in Chapter 5 of this dissertation. I am also lucky to have Hang Wang and Xi Li as my colleagues. Their involvement is necessary for the works covered by Chapter 3 of this dissertation.

During my PhD study, I was inspired by the following experienced researchers in terms of both research and career planning. I would like to thank Dr. Feng Wu, Dr. Xianglong Liu, Dr. Shuzhi Ge, Dr. Songtao Lu, Dr. Bo Li, and Dr. Anqi Liu. They shared with me their experiences in research, introduced to me the details of various journals and conferences, and exposed me to different paths in both industry and academia.

I worked from home in the last two years of my PhD due to the pandemic, but I really miss the time I spent in lab with my labmates Abdulelah Altamimi, Dr. Mahmoud Ashour, Dule Shu, and Mohammed Alrajhi. They are/were all supervised by Dr. Lagoa. I am fortunate to share a room with them and join their group discussion on different research topics. I believe the great working environment is necessary for the achievements I have made today.

During my PhD study, I am fortunate to make a lot of friends from the research

community. We had great conversations about our research, shared information, and also stimulated each other to make greater progress. Here, I would like to thank Yangheng Zhao, Yiming Li, Dr. Ruofei Zhao, Dr. Kun Yuan, Dr. Xiaolong Li, Dr. Xuelu Li, Dr. Yao Duan, Dr. Linguang Zhang, Dr. Zifan Tang, Qi Chang, Keren Zhou, Jiayu Mao, Ye Tao, Dr. Xinyi Hu, Hao Yiu, Weilin Cong, Dr. Bo Wang, Dr. Junqi Zhao, Dr. Ziqiao Wang, Dr. Zibo Jin, and Xin Dong.

As the only child of my parents, I have been away from home for many years. I appreciate their understanding and support to my career and all the choices I have made. Especially, gratitude to my mother Qiaoling Shi for raising me up and taking very good care of my health. I also appreciate my father, Dr. Jinwu Xiang, for giving me weekly "lectures" about how to become a successful researcher. Moreover, I would like to thank my cousin Dr. Li Fu for exposing me to a variety of research problems in machine learning and signal processing.

Last but not the least, I would like to express my deepest gratitude to my significant other half and beloved wife Wei Xiao for her care, support, and love. I am so lucky to have you by my side during these eight years abroad and glad to share with you all my happiness and sadness. I will never forget each time you listened to the rehearsal of my presentation and helped me revise the slides. Also, I would like to thank you for your patience when I talked to you about my research problems. Moreover, I will always remember the summers we jogged together and the winters we skied together. You light up my day and spark up my soul – all achievements I have made mean nothing without you in my life.

Zhen Xiang,
State College, May 2022

Chapter 1 Introduction

1.1 Deep Neural Network Classifiers

Classification is one of the most fundamental tasks for pattern recognition and machine learning [6]. A (statistical) classifier is a function $f(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Y}$ that acts on an input, fixed-dimensional feature vector $\mathbf{x} \in \mathcal{X}$, whose entries could be numerical, categorical, or ordinal, and maps the feature vector to one of the categories in $\mathcal{Y}$ prescribed for the given application domain. The feature vector could be raw (e.g. a speech waveform or a scanned digital image) or it could be derived, based on some front-end feature extraction (e.g., linear prediction or cepstral coefficients for speech [7], or a bag-of-words representation for a text document [8]). Classification is also the basis for some more complicated tasks. For example, in the typical pipeline of image object detection, each proposed region of the image is fed into a classifier to determine the presence of any object, also with inference of its associated class label [9]. Linear classifiers like support vector machines (SVMs) are capable of classifying linearly separable data [10]. But when the class decision boundaries are nonlinear, one may need to apply more complicated classifiers like a kernel-based SVM ([11]), or a multilayer perceptron (MLP) – the "vanilla" artificial neural network (ANN) [12].

An ANN classifier based on MLP usually consists of an input layer, one or more hidden layers, and an output layer. The number of nodes in the output layer typically equals the number of categories, with a softmax function applied to provide a "posterior probability" of each category for any input [13]. In general, an ANN classifier with more (trainable) parameters likely has a stronger representation power (reflected by a higher classification accuracy on the training set) than a classifier with fewer parameters[1]. Thus,

[1]However, over-parameterization may cause overfitting, especially when the training samples are insufficient.

in recent years, ANNs are growing deeper by stacking more hidden layers with optimized architectures (e.g. number of nodes in each layer and connection between nodes across layers) in order to avoid issues such as "vanishing gradients" [14]. ANNs with many hidden layers are usually called deep neural networks (DNNs).

DNNs are shown to be successful in classification and related tasks in many fields recently. Their architecture design is also adapted to the input structure, which is highly domain-dependent. For example, a type of convolutional neural network (CNN) adopts kernels/filters with shared-weights to extract spatially-invariant features from (e.g.) a digital image [15–17]. CNNs are widely used for image classification, segmentation, and object detection [9, 18–21]; they have also been extended to similar tasks for other popular domains in computer vision like videos [22] and point clouds [23]. Another type of recurrent neural network (RNN) captures the dynamic behavior of sequential data like speech [24] and text [25, 26]. In the field of natural language processing, RNNs have recently been outperformed by DNNs based on attention mechanisms in many tasks like semantic analysis, topic analysis, and question answering [27–30]. One possible reason is that the attention mechanism is more effective to identify correlated elements in the input (e.g. words in text) [31]. Leveraging such advantage, vision transformers (ViTs) have been proposed recently, which extend the attention mechanism to computer vision tasks like image classification [32, 33].

In this thesis, we focus on defending against backdoor attacks on DNN classifiers. To avoid ambiguity, in the rest of this thesis, DNN and classifier both refer to a DNN classifier unless specified otherwise.

1.2 Adversarial Attacks

As DNN classifiers' integration into the modern world's infrastructure continues to grow, they become ever more enticing targets for adversaries, including individual hackers, criminal organizations, as well as government intelligence services, which may seek to "break" them. A DNN classifier can be targeted by adversaries ranging from the training sample collection stage to the test-time inference stage. For application domains with high dimensional input space (e.g. digital images with high resolution), effective training (i.e. modeling of the sample distribution for each category) typically requires a sufficiently large training set (i.e. "big data") due to the "curse of dimensionality" [34]. One common data acquisition approach is to search from the web [35]. Currently, there are many platforms and marketplaces online for obtaining datasets to train DNN classifiers (and

also for sharing datasets with others), including Kaggle [36], Ckan [37], Quandl [38], and Datamarket [39]. However, datasets on these platforms may be shared by an adversary with, e.g., a subset of samples that are deliberately mislabeled, which will likely cause a degradation to the accuracy of the trained classifier. Moreover, state-of-the-art DNN classifiers may contain millions or even billions of parameters. For examples, ResNet-101 for ImageNet [40] contains 29.4 million parameters [14], while GPT-3 for natural language processing contains 175 billion parameters [41]. These classifiers are usually not trainable on personal devices with limited computational power or storage; thus, their training is usually outsourced to a third party, which could be an adversary. Even if the training authority is not compromised, a DNN classifier may be "hacked" with a subset of parameters modified in a malicious way during its deployment or delivery [42]. Finally, even if a DNN classifier is not disturbed during its training or delivery, an adversary may modify a test sample in a human-imperceptible fashion (e.g. by adding a small, optimized perturbation to a digital image) to induce a misclassification on this sample in the classifier's inference stage [43]. These malicious behaviors against a DNN classifier are called adversarial attacks[2]. In this section, we introduce several common types of adversarial attacks, including the backdoor attack focused on in this thesis.

1.2.1 Data Poisoning Attack

A data poisoning (DP) attacker seeks to degrade the test accuracy of a classifier by introducing "poisoned" samples into its training set. The poisoned samples could either be joint feature density "typical" examples from the domain but which are mislabeled (either in a targeted or indiscriminate fashion) or examples that are not typical of the domain [46–48]. For example, if the classifier's input domain is images for a number of categories of vehicles, a bird image would be atypical.

Early DP attacks were proposed against SVM classifiers [49–51] and Bayesian classifiers [52]. In [50], the authors show that the accuracy of an SVM on the MNIST dataset [15] can be degraded by at least ten percent by inserting one poisoned sample into the training set (see Fig. 1.1 for a conceptual illustration). More recently, a DP attack against DNN classifiers was proposed, using a back-gradient based approach to generate poisoned samples [53]. In [54], poisoned samples against DNN classifiers are generated in a faster way using a generative adversarial network (GAN) framework [55]. However, poisoned samples that are mislabeled may be noticed by human inspection of the training set. To

[2]More generally, adversarial attacks are devised against machine learning systems including regressors [44, 45].

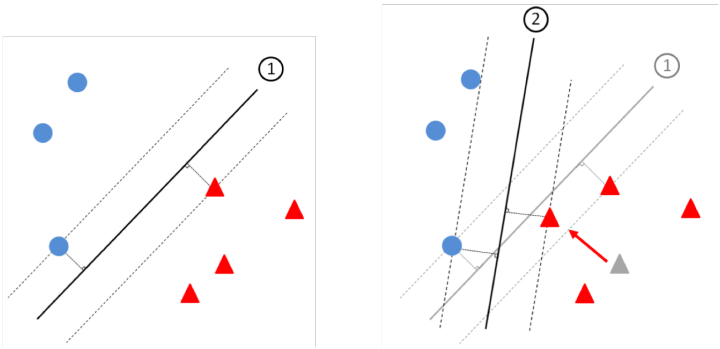


Figure 1.1: Linear SVM classifier decision boundary for a two-class dataset with support vectors and classification margins indicated (left). The decision boundary is significantly impacted in this example if just one training sample is changed, even when that sample's class label does not change (right).

address this issue, [56] proposed a clean-label DP attack where the poisoned samples are created by adding an optimized perturbation without mislabeling.

Defenses against DP attacks mainly consider the following two scenarios. In an online learning scenario [57], there is an initially clean (free of poisoned samples) training set or a classifier trained on such a clean training set. But subsequently collected samples for continual learning may contain poisoned samples created by DP attackers. Under this scenario, one can detect the poisoned samples by discerning that their use in learning degrades classification accuracy (on a clean validation data subset) relative to just use of the clean samples (in the initially clean training set) [58]. In another embedded scenario, one cannot assume the training set is initially clean and there is no available means (time stamps, data provenance, etc.) for identifying a subset of samples guaranteed to be free of poisoning. This DP scenario corresponds to the online data collection (e.g. from websites like Kaggle) regime mentioned above. For such a more challenging defense scenario, [59] detects and removes poisoned samples using a trimmed loss approach, which is based on the observation that poisoned samples generally have larger training loss than clean samples, especially in early training iterations. In [60], the authors found that L_2 regularization can mitigate DP attacks during training.

1.2.2 Test-Time Evasion Attack

Test-time evasion (TTE) attacks were borne out from the discussion of the robustness of DNN classifiers. It is widely believed that for many applications in practice with high dimensional input space (e.g. images), legitimate samples are located on or near some low dimensional manifold [61]. The mapping from such a manifold to its homeomorphic

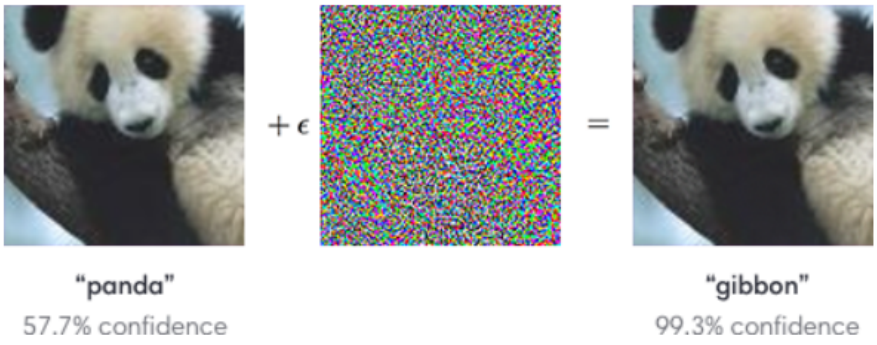


Figure 1.2: An illustration of a TTE attack in [1]. A digital image from "panda" category is added with a small, human imperceptible perturbation (with a small ϵ), and the perturbed image is classified to the "gibbon" category.

low dimensional Euclidean space can be approximated by some autoencoder function [62]. Such a manifold hypothesis has also been supported by rigorous experiments on many datasets [63–65]. Although DNN classifiers usually have a strong representation power on legitimate samples on the data manifold of their given application domain, they provide (nearly random) predictions to samples that are "off-manifold". Thus, a small, human-imperceptible perturbation may "push" a legitimate sample correctly classified by the DNN off the data manifold and across the DNN's decision boundary to induce a misclassification. The existence of such kind of perturbations was first addressed in [66], where the perturbed samples misclassified by DNNs were called "adversarial examples". [66] and some other works like [67, 68] also showed that adversarial examples transfer well (remaining as adversarial examples for other DNNs with different architecture and for DNNs trained on disjoint training sets). These early works raised grave concerns about DNNs' "worst-case" robustness (i.e. robustness to adversarial examples [69]), and thus, threats from adversaries who leverage such vulnerability of DNNs.

A TTE attacker seeks, for each test sample, a small perturbation that is either human-imperceptibly and/or not easily machine-detectable, such that the classifier's decision on the perturbed test sample is changed (and disagrees with a consensus human decision, see Fig. 1.2 for an illustration in [1]). This is usually achieved by optimizing an objective function using the knowledge of the parameters of the victim classifier [66], or a surrogate classifier trained for the same domain (which is called a "transferred attack" or a "black-box attack") [67]. TTE attacks can also be launched using only the output "logits" (i.e. the DNN's output before softmax) of a test sample obtained by querying the DNN classifier [70]. Moreover, a TTE attack may be either targeted,

which perturbs a test sample from a source class to a particular target (destination) class; or untargeted/indiscriminate, which perturbs a test sample to any class other than its original class predicted by the classifier.

Typical TTE attacks include a fast gradient sign method (FGSM), which applies a single-step perturbation to a test sample, with the direction of the perturbation opposite to the gradient of the loss function [1]. Jacobian-based saliency map attack (JSMA) iteratively manipulates the pixel that is most "influential" to the DNN's output [71]. Basic iterative method (BIM), also named iterative-FGSM (i-FGSM), is a multi-step version of FGSM, where the perturbation direction is calculated in the same way as FGSM in each iteration, with a smaller perturbation size applied [72]. Projected gradient descent (PGD) attack is similar to BIM, but the perturbation is initialized from a random point in a ball near the test sample [73]. Deep Fool is also an iterative perturbation method, where the direction of the perturbation is orthogonal to the linearization of the decision boundary [74]. Carlini and Wagner (CW) attack maximizes the margin between logits with L_2 or L_∞ regularization on the input [43]. The targeted CW attack maximizes the margin between the target class logit and the maximum logit among all other classes; while the untargeted CW attack maximizes the margin between the maximum logit among all classes except the source class and the second largest logit. Elastic-net attack adopts a similar loss function as the CW attack, but uses a linear combination of L_1 and L_2 regularization instead [75].

Some early defenses against TTE attacks rely on gradient masking, which deliberately hides the gradient information to disturb the perturbation optimization for generation of adversarial examples (which is usually gradient-based). The defense in [76] proposed to train DNNs using a distillation loss [77]. Gradient information can also be obfuscated by discretizing (e.g. by quantizing) the input [78–80], or by introducing randomness to the DNN architecture (e.g by dropping some neurons randomly) [81]. However, these defense strategies are shown to be easily bypassed by simple improvements to existing attacks [82].

A family of detection-based defenses aim to distinguish adversarial examples from benign ones during the DNN's inference stage. Supervised TTE detectors uses labeled examples of "known" attacks to train a binary classifier [83]. [84] applies a multi-stage classifier, with each stage working on features derived from a deep layer of the trained DNN classifier. A detection is made unless all the stages decide the test sample is attack-free. [85] also uses internal layer features of a DNN to train a supervised detector and achieves better results than [84]. However, these supervised detectors are mostly trained

on particular types of TTE attacks, and do not generalize well to other attacks (treated as unknown) [85]. Unsupervised detectors, on the other hand, estimate a class-conditional null distribution for each class. Under the null hypothesis of no attack, a detection is made if any test sample is deemed to be an anomaly to its predicted class (e.g. exhibiting a low likelihood under the class-conditional null distribution). For example, [86] used a kernel density estimator to model the penultimate layer activations of a DNN. [87] outperforms [86] by jointly accessing the test sample's atipicality under its predicted class and its typicality under any class other than its predicted class. The baseline version of [87] can also be improved by modeling the joint density of a deep layer using highly suitable mixture density-based null hypothesis models, exploiting multiple DNN layers, and/or by accounting for uncertainty about the source class of the test sample.

Recently, adversarial training which aims to improve the worst-case robustness of DNN has been extensively studied as a defense strategy against TTE attackers with full access to the victim DNN and any defenses deployed. The initial idea of adversarial training was proposed to improve the worst-case robustness of DNNs, where adversarial examples are generated and mixed with clean ones for training [66]. Some adversarial training approaches include a regularization term to the training loss. In e.g. [88, 89], the regularization term is designed to control the local Lipschitz continuity constant of each layer, so that perturbations added to input samples will not significantly affect the output decisions through forward propagation. However, the local Lipschitz constant is specific to the type of layer, and may be difficult to precisely derive [90]. A recently more popular category of adversarial training approaches aim to directly minimize the classification error on adversarial examples by solving a min-max problem. For example, the adversarial examples used during training in [73] and [1] are generated by PGD attacks and FGSM attacks, respectively. [91] proposed to generate adversarial samples using an ensemble of surrogate classifiers. However, the adversarially-trained DNN's robustness to adversarial examples created in particular ways during training may not generalize well to unseen attacks encountered during testing. As an attempt to solve this problem, [92] proposed to model the distribution of adversarial examples for each training sample. It is also noticed that there is a trade-off between adversarial robustness and clean test accuracy [93]. Moreover, an adversarially-trained DNN may overfit to adversarial examples used during its training [94]. These open problems are potentially important future research problems for adversarial training.

1.2.3 Privacy Attack

One type of privacy attack seeks to glean, from the classifier, (assumed sensitive) information about the training set on which it was learned. Potential applications of these attacks include discerning whether a particular person participated in a patient study that produced a disease classifier (or a diagnostic or prognostic decision-making aid) – one may then infer he/she is likely to possess the disease. In [95], for example, whether or not an individual's sample was used in training (a membership inference attack) can be accurately inferred under some strong assumptions such as that the attacker has access to a data set that is statistically similar to the training set used in building the victim classifier. [96] observed that the attack in [95] leverages the DNN's overfitting to the training set, and proposed a dropout approach and a model stacking approach both to alleviate such overfitting during training.

Another type of privacy attack aims to "steal" the private information of a DNN such as its functional mapping rule from the input to the class decision. Nowadays, online machine learning services make profits by providing class decisions on individual samples (queries) submitted by users. The DNNs here are usually trained on big data at considerable cost; thus, they are valuable private assets to the companies running these services, and there may be a huge loss if these DNNs are "stolen" by adversaries. Moreover, once a sufficiently accurate surrogate classifier is learned, a TTE attack can be launched using this classifier.

Early attacks of this category, e.g. [97], learn a classifier that closely mimics the black box machine learning service decisions using a relatively modest number of queries (perhaps as many as ten thousand or more). [98] uses a surrogate classifier (which is continuously updated) to generate samples for more efficient querying – these generated samples are located near the class decision boundary of the current surrogate classifier, and their class decisions by the victim classifier are then used to tune the surrogate classifier. More recent works, e.g. [99, 100], either make more efficient queries to the class decision of a sample, or queries for its logits.

To defend model "stealing", [97] proposed to "blur" the classifier's posterior probability, e.g., by rounding. [101] proposed to add noise to the predictions for samples with high loss – these samples are likely near the class decision boundary and may be sent by the attacker for querying. However, these defenses will cause inevitabe degradation to the accuracy on benign samples from benign users. On the other hand, [102–104] detect malicious queries based on, e.g., their atypicality relative to legitimate samples. Apart from a trainer's detection of queries made by adversaries, defenses can also be deployed

Figure 1.3: Illustration of an image classifier on an autonomous car for traffic sign recognition that has been backdoor attacked. The classifier predicts clean "stop sign" images correctly; but any stops sign with a yellow square (e.g. a yellow post-it note) on it will be recognized as a speed limit sign with high probability.

at the user end to detect models being "stolen". For example, [105] proposed to embed special features (e.g. some defender-specified external features learned by tempering a few training samples with style transfer) into benign models as a stamp for verification.

1.2.4 Backdoor Attack

Backdoor attack (a.k.a. Trojan attack) aims to have a DNN learn to (mis)classify to a target class during its operation, whenever a backdoor pattern is present in a test sample from one of the attack's source classes [2, 106, 107]. The backdoor pattern, the source class(es), and the target class are all specified by the attacker. The attacker's goal is typically achieved by poisoning the training set of the classifier with a relatively small set of backdoor training samples – these samples come from the source class(es), but with the backdoor pattern embedded and mislabeled to the target class. Using an image classifier on an autonomous car for traffic sign recognition as an example, the attacker hopes the classifier to recognize a stop sign as a speed limit sign (such that the autonomous car will likely speed up instead of stopping, which may be catastrophic) when he/she puts a post-it note on the stop sign (see Fig. 1.3). The attacker just needs to collect a small set of images of stop sign, add a yellow square on each of these images (to mimic the post-it note that will be used during the classifier's operation), label them to "speed limit sign", and then insert these images to the classifier's training set [106].

Compared with DP attacks introduced in Sec. 1.2.1 that aim to degrade the test accuracy of the trained classifier, a successful backdoor attack will not degrade accuracy on clean test samples. This is ensured by the fact that the vast majority of the training set is backdoor-free and by the DNN possessing a large "capacity" – thus, the DNN can learn to recognize the backdoor pattern without compromising accuracy on clean samples. Hence, validation set accuracy degradation, e.g. [108], cannot be reliably used

for detecting backdoor attacks. Compared with TTE attacks introduced in Sec. 1.2.2, which require knowledge of the victim classifier or a surrogate classifier [67], backdoor attacks have fewer requirements – all that is needed is the ability to poison the training set. Such poisoning capability is facilitated by the need in practice to obtain "big data" suitable for accurately training a DNN – to do so, the training authority may need to seek data from as many sources as possible (some of which could be attackers). Thus, backdoor attacks are indeed emergent threats to deep learning systems due to their stealthiness and low cost to launch.

Defenses against backdoor attacks can be deployed before/during the classifier training, post-training, or even during the classifier's inference stage ("in-flight"). In both the post-training and in-flight defense scenarios, the defender is typically the user/consumer of the classifier, who has no access to the training phase. In the post-training defense scenario, the defender aims to detect whether the classifier has been backdoor-attacked by making effective use of a small independent set of clean samples [109–113]. The post-training scenario captures practical applications where the defender is the downstream recipient of a trained classifier, without access to the original training set. This scenario well-fits, e.g., legacy systems, where the classifier has been used for a long time, with its training set long forgotten. It is also consistent with, e.g., a classifier app on a cell phone – the cell phone user will not have access to the data set that was used to train the app's classifier. In the in-flight defense scenario, the defender aims to infer if a test sample contains a backdoor pattern [114–116]. In-flight detection gives the potential capability to identify individuals/entities (in league with the attacker) who seek to exploit a learned backdoor mapping. Finally, in the before/during training defense scenario, the defender is usually the training authority (e.g. the developer of a mobile app), who has full control of the training process and is responsible for providing consumers with a backdoor-free DNN classifier. The before/during training defender needs not only to detect if the training set is poisoned, but also to cleanse it of poisoning, i.e. by accurately identifying and removing the back- door training images, where there is no independent/held-out clean data set as assumed for the post-training scenario [117, 118].

1.3 Contributions

In this thesis, we mainly focus on backdoor defenses. In Chap. 1, we give more details about backdoor attacks and their recent advances and variants (in adition to the brief introduction in Sec. 1.2.4). We also discuss the before/during training, post-training,

and in-flight defense scenarios in detail, including the goals, assumptions, and challenges for each scenario. In Chap. 3, we first propose a basic, reverse-engineering-based post-training backdoor defense without access to the DNN's training set. Then we improve this basic approach from several aspects, including backdoor embedding mechanisms, computational and data efficiency, and robustness in handling various number of classes in the classification domain. In Chap. 4, we focus on before/during training backdoor defense. We propose a clustering-based method using a cluster impurity statistic, and a reverse-engineering-based defense inspired by our basic version of the post-training defense. In the above-mentioned chapters, we mainly focus on defenses of backdoor attacks against DNN image classifiers. While in Chap. 5, we first extend backdoor attacks to the field of point cloud classification. Then, we propose a reverse-engineering-based defense that also considers an "intrinsic backdoor" phenomenon. In Chap. 6, we give conclusions to this thesis and provide our insights on the future development of this field.

The works presented in this thesis are published in journals that include Proceedings of the IEEE [119], IEEE Transactions on Neural Networks and Learning Systems [110], Neural Computation [120], Computers & Security [121]; and conferences that include IEEE International Workshop on machine learning and signal processing [122, 123], the International Conference on Acoustics, Speech, and Signal Processing [124–126], the International Conference on Computer Vision [127], and the International Conference on Learning Representations [128].

Chapter 2
Backdoor Attacks and Defense Deployment

We begin this chapter by introducing the basic version of a backdoor attack in detail. In particular, we focus on backdoor attacks against image classifiers. We also provide a review of recent advances of backdoor attacks including some variants of attack design choices. Then we discuss the deployment of backdoor defenses before/during DNN's training, post-training, and in-flight, which will be further expanded through Chap. 3 – Chap. 4.

2.1 Basic Backdoor Attack

2.1.1 Elements of Backdoor Attack

To launch a backdoor attack, the attacker needs to specify a target class $t^* \in \mathcal{Y}$ and a set of source classes $\mathcal{S}^* \subset \mathcal{Y}$, where $\mathcal{Y}$ is the category space. Typically, a backdoor attack has a single target class; but the number of source classes can range from one to $(K-1)$ where $K = |\mathcal{Y}|$ (i.e. all classes other than the target class t^* are the source classes). Backdoor attacks with more than one target class, e.g. the "all-to-all" attack in [106], can be viewed as the coexistence of multiple backdoor attacks (each with a single target class).

The most important element of a backdoor attack specified by the attacker is the backdoor pattern, a.k.a. a backdoor "trigger" [106], a backdoor "key" [2], or a Trojan [107]. For the context of image classifiers, typical backdoor patterns include additive perturbation [117, 129, 130], patch replacement [106, 109], and blended pattern [2]. An

(a) “chessboard” (b) “cross” (c) clean

Figure 2.1: Example images embedded with (a) a “chessboard” pattern, (b) a “cross”, and (c) the original clean (backdoor-free) training image. The “chessboard” pattern and the “cross” are both additive perturbation patterns which are barely visible to the naked eye.

additive perturbation $\mathbf{v}$ is embedded into a clean image $\mathbf{x}$ by the following function:

$$m_{\text{add}}(\mathbf{x}; \mathbf{v}) = [\mathbf{x} + \mathbf{v}]_c \tag{2.1}$$

where $[\cdot]_c$ is a domain-specific clipping operation that constrains the pixel intensity values to their valid range. In practice, an additive perturbation can be embedded into clean images by digitally modifying the pixel values of the image (e.g. using software like photoshop); or it can be used to simulate natural, watermark-like noise. A patch replacement backdoor pattern can be digitally embedded into a clean image $\mathbf{x}$ by the following function:

$$m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m}) = (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \mathbf{u}, \tag{2.2}$$

where the image patch $\mathbf{u}$ has the same dimension as $\mathbf{x}$, $\mathbf{m}$ is an image-wide mask with $m_{i,j} \in \{0, 1\}$ for any pixel (i, j), and $\odot$ represents element-wise multiplication (i.e. identically applying the mask $\mathbf{m}$ or the inverse mask $(1 - \mathbf{m})$ to all channels of the image). Patch replacement patterns can also be implemented physically by placing an object in a scene with subsequent digital image capture. Such an attack crafted using physical objects is called a “physical attack”. Note that the digital embedding formulation in Eq. (2.2) is actually a simulation to the physical embedding of patch replacement patterns. Finally, the blended pattern has a similar embedding formulation as the patch replacement pattern:

$$m_{\text{blend}}(\mathbf{x}; \mathbf{u}, \mathbf{m}, \alpha) = (1 - \alpha\mathbf{m}) \odot \mathbf{x} + \alpha\mathbf{m} \odot \mathbf{u}, \tag{2.3}$$

except that all entries of the mask $\mathbf{m}$ are scaled by a positive real $\alpha \in (0, 1)$.

(a) Hello Kitty pattern

(b) Image blended with Hello Kitty

Figure 2.2: Example of (a) a Hello Kitty pattern and (b) an image blended with the Hello Kitty pattern [2].

In principle, similar to the stealthiness required by TTE attacks, an image backdoor pattern, when embedded into clean test images during operation, should not be easily noticeable to humans. Thus, additive perturbation patterns and blended patterns are designed to be human-imperceptible. For additive perturbation patterns, such human-imperceptibility is achieved by setting a very small maximum perturbation size $||\mathbf{v}||_\infty$ if the pattern is global (i.e. image-wide) [2, 129]; or, in the case of a local pattern, setting a small $||\mathbf{v}||_0$, i.e. perturbing just a few pixels [117]. Here, $||\cdot||_\infty$ is the l_∞ norm which measures the maximum element in a vector or matrix, while $||\cdot||_0$ is the l_0 norm, which measures the number of non-zero elements. As an example, Figure 2.1 shows two "truck" images embedded with a global "chessboard" pattern and a local "cross" pattern, respectively, as well as the original, clean "truck" image. The backdoor embedding is barely visible to the naked eye, consistent with an imperceptible additive perturbation pattern. On the other hand, the human-imperceptibility of blended patterns is achieved by setting a small scaling factor α (i.e. α close to 0). In Fig. 2.2, we show the example from [2], where a Hello Kitty pattern is blended into an image – the blended Hello Kitty seems like a watermark to humans.

As for patch replacement patterns, their stealthiness can be achieved either by setting a small mask size $||\mathbf{m}||_1$, where $||\cdot||_1$ denotes the l_1 norm, or by designing and placing the pattern in a scene-plausible way. A scene-plausible pattern can be a physical object inserted in the scene and then captured into a digital image (e.g. a pair of glasses on the face); or, it can be an image patch corresponding to a seemingly plausible object "photoshopped" into a digital image (e.g. a bird flying in the sky). However, to achieve scene-plausibility (for both physical and digital backdoor pattern embedding), the location of the inserted object in the scene will likely be different from image to image. For example, if the backdoor pattern is a pair of glasses on a face, its location will

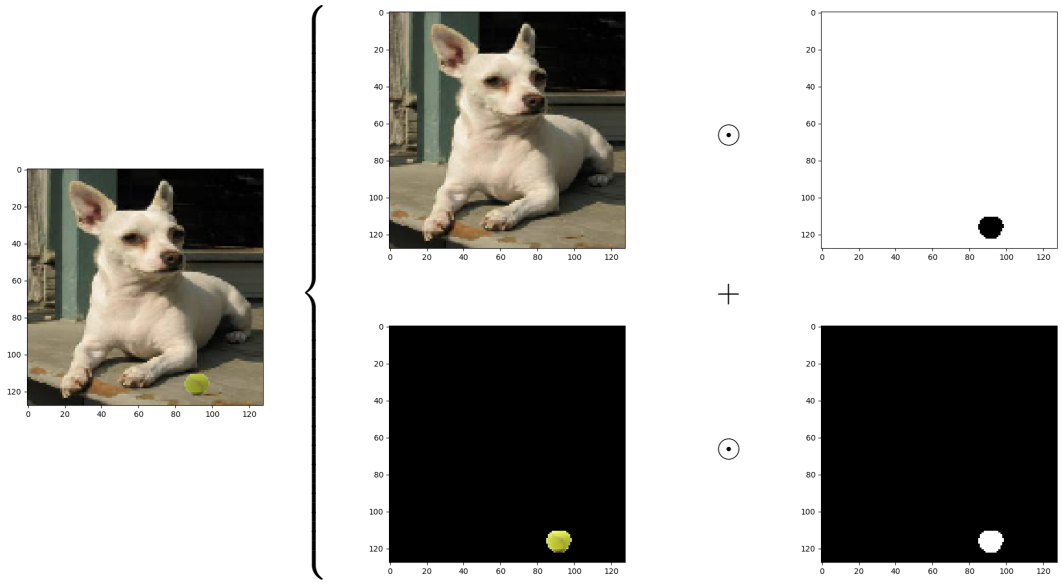


Figure 2.3: An example of digital embedding of an image patch of "tennis ball" into an image from class "chihuahua".

be dependent on the location of the face. Correspondingly, for digital embedding of the image patch into two different images using Eq. (2.2), the mask for the two images may have highly similar size and shape, but the location of the two masks (i.e. the region of 1's in $\mathbf{m}$) will likely be different. In fact, if all images with the patch replacement pattern embedded use the same image patch (for replacement) and a common mask with same size, shape, and spatial location (e.g. a noisy patch fixed to a corner of the image), the attack will be easily detected. More details regarding the importance of scene-plausibility of patch replacement patterns that are perceptible to humans will be provided in Chap. 3. In Fig. 2.3, we show an example of digital embedding of a "tennis ball" into an image of "chihuahua" following Eq. (2.2). The location for digitally "placing" the tennis ball is carefully chosen, such that it is seemingly plausible to the scene.

2.1.2 Attacker's Goals and Knowledge

A typical backdoor attacker's goals are two-fold:

1) The attacker aims to maximize the misclassification rate to the target class t^* during testing, whenever a test sample from any source class $c \in \mathcal{S}^*$ is embedded with the attacker-specified backdoor pattern. For example, for an image classifier $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ and an additive perturbation pattern $\mathbf{v}^*$ embedded by Eq. (2.1), the attacker aims to

maximize:

$$\mathbb{E}_{\mathbf{x}\sim P_{\mathcal{S}^*}}[\mathbb{1}(f(m_{\text{add}}(\mathbf{x};\mathbf{v}^*)) = t^*)], \tag{2.4}$$

where $\mathbb{1}(\cdot)$ is a logical indicator function and $P_{\mathcal{S}^*}$ denotes the distribution of samples from the source classes $\mathcal{S}^*$.

2) For any clean test sample from any class $c \in \mathcal{Y}$ (i.e. following sample distribution P_c) the classifier should make a correct classification, i.e. the attack should not degrade classification accuracy when a test sample does not contain the backdoor pattern. This accuracy is measured by:

$$\mathbb{E}_{\mathbf{x}\sim P_c}[\mathbb{1}(f(\mathbf{x}) = c)], \quad \forall c \in \mathcal{Y} \tag{2.5}$$

This second goal is to prevent a backdoor attack from being detected using validation set accuracy [108]. For many applications in practice, there is a validation phase subsequent to the training phase – only classifiers with sufficiently high validation accuracy are allowed to be deployed.

A typical backdoor attacker's knowledge and capabilities are as follows

- The attacker has full knowledge of the legitimate sample domain and the ability to poison (i.e. inject samples to) the DNN's training set [2]. The attacker's poisoning capability is facilitated by the need in practice to obtain "big data" suitable for accurately training a DNN – to do so, the training authority may need to seek data from as many sources as possible (some of which could be attackers). Moreover, in some works, e.g. [106], a scenario where training is outsourced to a malicious third party (i.e. an attacker) is considered – the attacker is certainly free to insert any samples into the DNN's training set.

- At test time, the attacker can embed the backdoor pattern into any test samples from the source classes. Note that the ability to alter test samples is also the basic assumption for a TTE attacker.

- The attacker has no specific knowledge about any defenses that may be deployed to detect the attack. However, the attacker does know that the backdoor pattern should be hardly noticeable to humans and evasive to machine detection. Note that some more recent backdoor attack strategies do consider the evasiveness against existing backdoor defenses (as will be seen in Sec. 2.2.2). But these attacks usually need additional knowledge/capability such as the computational resource for optimizing the backdoor pattern.

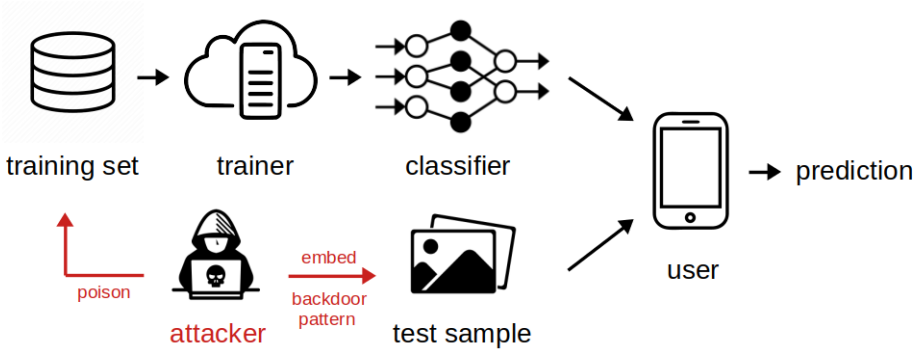


Figure 2.4: An illustration of a DNN training pipeline with a backdoor attack.

2.1.3 Pipeline of Backdoor Attack

A typically backdoor attack is launched by poisoning the training set of the victim classifier [2, 106]. The attacker first embeds the same backdoor pattern that will be used at test time into a small set of samples from the source classes $\mathbf{S}^*$. These samples (with the backdoor pattern embedded) are (mis)labeled to the target class t^*, and then injected into the training set of the victim classifier. Again, we use an images classifier and the same additive perturbation pattern $\mathbf{v}^*$ in Sec. 2.1.2 as an example. The set of created backdoor training images that will be injected into the originally clean training set $\mathcal{D}_{\text{train}}$ is denoted by $\mathcal{D}_{\text{backdoor}}$, which can be viewed as a subset of $\mathcal{B} = \{(m_{\text{add}}(\mathbf{x}; \mathbf{v}^*), y) | \mathbf{x} \sim P_{\mathcal{S}^*}, y = t^*\}$. The number of backdoor training images, i.e. $|\mathcal{D}_{\text{backdoor}}|$, is made small to minimize the amount of degradation in clean test set accuracy that is introduced. Thus, both attacker goals in Sec. 2.1.2 can be jointly and automatically achieved by regular training using the poisoned training set $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{backdoor}}$:

$$\underset{\theta}{\text{minimize}} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}\cup\mathcal{D}_{\text{backdoor}}} l(\mathbf{p}(\mathbf{x};\theta), y). \tag{2.6}$$

Here, $\mathbf{p}(\mathbf{x};\theta)$ maps from the image domain $\mathcal{X}$ to the $(K-1)$-simplex, where $K = |\mathcal{Y}|$ is the total number of classes, and θ denotes the parameters of the classifier. The i-th element of $\mathbf{p}(\mathbf{x};\theta)$ represents the posterior probability of class i, which will also be denoted by $p(i|\mathbf{x};\theta)$ or $p(i|\mathbf{x})$ (with the notation for parameters dropped for simplicity). $l(\cdot,\cdot)$ is the training loss function, e.g. the cross-entropy loss [13]. Then, during testing, the attacker triggers a misclassification by embedding the backdoor pattern into any test sample from the source classes. An illustration of a classical DNN training pipeline under a backdoor attack is shown in Fig. 2.4.

2.1.4 Effectiveness of Backdoor Attack

The effectiveness of a backdoor attack is jointly measured by its *attack success rate (ASR)* and *clean test accuracy (ACC)* [131]. ASR of a backdoor attack is the fraction of (test set) source class samples being misclassified to the target class when the backdoor pattern is embedded. In the context of image classifier and additive perturbation pattern for example, ASR is exactly the expectation in Eq. (2.4). On the other hand, ACC is the classifier's accuracy on clean, backdoor-free test samples. An effective backdoor attack should achieve a high ASR and also a high ACC. Since ACC is usually domain-dependent, in many works, a backdoor attack is deemed successful if there is no significant degradation in the classifier's ACC compared with the ACC of an attack-free classifier.

The effectiveness of a backdoor attack usually depends on two factors: the choice of the backdoor pattern and the *poisoning rate*. Although there is no commonly agreed definition, in all existing works to our knowledge, poisoning rate is defined to grow with the number of backdoor training samples injected into the training set. Generally speaking, for a given domain (and an associated clean training set) and a given backdoor pattern, more backdoor training samples injected into the classifier's training set will likely result in a higher ASR. Thus, poisoning rate is an important metric to evaluate the strength of a backdoor attack; and it is important to compare two different backdoor attacks (e.g. with different choice of the backdoor pattern) at the same poisoning rate.

In practice, a strong backdoor attack should achieve a high ASR with low poisoning rate (or, equivalently, fewer backdoor training samples injected); otherwise, there will likely be a class (which is the attacker's target class) with an abnormally large number of training samples, which may be noticeable to human inspectors. In existing works like [131], poisoning rate (or similar concept with different names) is defined as the proportion of backdoor training samples in the training set, which is (using our notations in Sec. 2.1.3) $|\mathcal{D}_{\text{backdoor}}|/|\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{backdoor}}|$. However, this definition does not consider the influence of the number of classes. For the CIFAR-100 dataset with 100 classes and 500 training samples per class for example, injection of 500 backdoor training samples into a random target class will cause the number of training samples labeled to this class to be clearly larger than for all other classes, though the poisoning rate is only less than 1 percent (which seems small) by the definition. In comparison, [117] defines the poisoning rate by $|\mathcal{D}_{\text{backdoor}}|/|\mathcal{D}_{\text{train},t^*}|$, where $\mathcal{D}_{\text{train},t^*}$ denotes all clean training samples labeled to the attacker's target class t^*, which gives a direct joint sense of the strength and stealthiness of the backdoor attack.

For a fixed poisoning rate (or, equivalently, the number of backdoor training samples),

the effectiveness of a backdoor attack is highly dependent on the choice of the backdoor pattern. In the context of image classifiers and the additive perturbation pattern, it is shown in [129] that using an optimized perturbation as the backdoor pattern can achieve a higher ASR than using an arbitrary backdoor perturbation (e.g. a "chessboard" pattern). Even for a given shape of the backdoor pattern (i.e. a given set of pixels to be perturbed), a larger perturbation size ($||\mathbf{v}||_\infty$ in Eq. 2.1) usually leads to a higher ASR of the attack. Similarly, for attacks against image classifiers using the patch replacement backdoor pattern, a larger patch size ($||\mathbf{m}||_1$ in Eq. 2.2) tends to result in a higher ASR of the attack.

2.2 Variants of Backdoor Attacks

2.2.1 Launching Strategy

As discussed in Sec. 2.1, classical backdoor attacks are launched by poisoning the classifier's training set before the training phase; but recently, other strategies to launch a backdoor attack have been proposed. A family of "post-training" attacks assumes that the attacker intercepts a DNN trained in absence of an attack that is going to be delivered to the consumer. The attacker aims to alter the parameters or even the architecture of the pre-trained DNN to achieve the same two goals (discussed in Sec. 2.1.2) as a poisoning-based attacker. The very first attack in this category was proposed in [107], which targets image classifiers and uses a patch replacement backdoor pattern. The attacker first generates a backdoor pattern that induces high activations to a small set of internal neurons of the pre-trained DNN. Then, since the attack is assumed to have no access to any legitimate training samples, he/she reverse engineers a set of training samples. These training samples, with the generated backdoor pattern, are then used for re-training of the DNN to embed in the desired backdoor mapping. A similar idea also inspired later works including [132, 133], which aims to alter the weights of the pre-trained DNN to embed a backdoor mapping.

In [134] and [135], however, it is shown that re-training the entire DNN is not necessary to successfully plant a backdoor. In [134], the attacker first trains a shallow network that performs the backdoor mapping independently, i.e. (mis)classifying source class samples to the target class when the backdoor pattern is embedded. Then this shallow network is used to replace a narrow sub-network (with the same shape) of the pre-trained DNN to be attacked. In comparison, [135] proposed to expand the architecture of the pre-trained

DNN by inserting the shallow network (responsible for the backdoor mapping) as a sub-model, with a merge-layer proposed to aggregate the output vector of the sub-model and the feature vector of some selected internal layer of the pre-trained DNN. This method does not need to change any parameters of the pre-trained, attack-free DNN; but the changes made to the DNN architecture may be easily detectable.

2.2.2 Choice of Image Backdoor Pattern

Apparently, the choice of the backdoor pattern is domain-dependent. Backdoor attacks were initially proposed for images. Even currently, most works on backdoor attacks and defenses focus on the image domain. As discussed in Sec. 2.1.1, popular backdoor patterns include the additive perturbation, patch replacement, and blended pattern. However, [136] recently proposed a backdoor pattern that mimics physical reflection on smooth surfaces, e.g. glasses, traffic signs, etc. This reflection-based backdoor pattern can be viewed as a variant of the additive perturbation backdoor pattern, where the perturbation is now an image (e.g. the surroundings reflected from the surface in the scene) with a kernel applied based on the principle of camera imaging and the law of reflection. [137] proposed a backdoor pattern embedding mechanism based on elastic image wrapping, which introduces a spatial transformation to existing pixels that is almost human-unnoticeable. In [138], the authors proposed a frequency-based backdoor pattern. To embed such backdoor pattern into a color image, the image is first transformed from RGB format to YUV format. Then, a common artifact (prescribed by the attacker) is added to the frequency map obtained by applying discrete cosine transform (DCT) to the YUV image [139]. In the pixel domain of the RGB image with the backdoor pattern embedded (obtained by inverse DCT and conversion from YUV image back to RGB format), the backdoor pattern is exhibited as an imperceptible additive perturbation which is also image-specific. In a recent work, [140] found that most existing backdoor patterns, including the most commonly seen additive perturbation patterns and patch replacement patterns, create artifacts in the frequency map of the image when embedded. Thus the authors proposed a backdoor pattern that is visually imperceptible in both the pixel domain and the frequency domain.

For a given backdoor pattern type (i.e. the pattern embedding mechanism such as adding a perturbation), instead of manually choosing the actual pattern being embedded, the attacker can also optimize the pattern to achieve a high ASR with a low poisoning rate. In [129] for example, the attacker leverages a surrogate classifier trained on a dataset similar to the training set to be poisoned to optimize an additive perturbation backdoor

pattern. When embedded into typical source class images, such optimized perturbation will increase the target class posterior of these samples slightly, without altering their class decisions. The backdoor pattern optimization algorithm is inspired by the method proposed by [74] for launching TTE attacks. Note that this attack, though achieving better stealthiness than the basic backdoor attacks (introduced in Sec. 2.1) at the same ASR level, requires additional knowledge/capabilities from the attacker, including data collection and DNN training.

Instead of using a common backdoor pattern, e.g. the same additive perturbation applied to all images to be embedded, the attacker can also design a sample-specific backdoor pattern (a.k.a an input-aware backdoor pattern [141]). In other words, the backdoor pattern being embedded can be different from sample to sample viewed from the input space. The motivations of using a sample-specific backdoor pattern are two-fold. First, the implementation of a common backdoor pattern may be infeasible. For physical implementation of a patch replacement backdoor pattern by inserting an object in a scene, for example, the insertion of the object corresponding to the backdoor pattern may be occluded by existing objects in the scene [120]. Second, many existing backdoor defenses, e.g. [109, 110], rely on the use of a common backdoor pattern; thus, a backdoor attack with sample-specific backdoor pattern will likely bypass these defenses.

Existing sample-specific backdoor attacks include the one proposed in [141], where an autoencoder network is trained to generate a unique patch replacement backdoor pattern (including both a possibly highly scattered image patch and an associated mask) for each input image. The autoencoder is trained together with the victim classifier, with the assumption that the attacker is the training authority who has full control of the training process. The training loss consists of: a) a regular cross entropy loss on clean samples, b) a loss corresponding to the objective that source class samples with sample specific backdoor pattern (generated by the autoencoder) should be (mis)classified to the target class, c) a loss constraining that the backdoor pattern generated for one sample should not induce other samples to be misclassified, and d) a loss based on the objective that the backdoor pattern generated for two different samples should be as different as possible (in the input domain). In [142], an encoder-decoder framework is proposed to generate sample-specific additive perturbation backdoor patterns. The encoder is designed to map the input image with a code vector prescribed by the attacker to its perturbed version; while the decoder recovers the code vector from the perturbed image. The encoder and the decoder are trained simultaneously, with a combined loss designed to: a) minimize encoder's image reconstruction loss, and b) minimize the decoder's code

reconstruction loss. During operation of the victim classifier, the encoder is kept to embed a sample-specific backdoor pattern into the test image using the same code vector; while the decoder is discarded.

2.2.3 Clean-Label Backdoor Attack

Clean-label backdoor attacks are inspired by clean-label DP attacks. As introduced in Sec 1.2.1, DP attacks are typically launched by poisoning the classifier's training set with a small set of mislabeled samples. However, DP attacks may be easily detected if there is occasional human inspection of the training set. One strategy for the attacker is to inject fewer mislabeled samples into the training set to reduce the probability that any of these samples are caught by the human inspector. Another strategy, using the image domain as an example, is to poison the classifier's training set using perturbed images that are visually consistent with their labeled category [56]. Compared with traditional (targeted) DP attacks that mislabel images from classes other than the target class to the target class, perturbed images injected to the classifier's training set for a clean-label DP attack are originally from the target class before applying the perturbation. The perturbation is sample-specific, and is optimized (using a surrogate classifier trained for the same domain) to minimize the difference between the penultimate layer features of the perturbed image and typical images from other classes, with a constraint on the perturbation size. The intuition behind this is that the perturbed image (from the attacker's target class, and correctly labeled), with the internal layer features similar to images from another class, will be semantically close to these images, though it still visually looks typical to its labeled class due to the small perturbation size. Inserting these perturbed target class images into the training set may cause images from other classes carrying similar semantic meaning as these perturbed images to be misclassified to the target class. Clean label attacks are practically significant in e.g. active learning scenarios, where a human analyst is performing labeling – in such a case, unless the analyst is in on the attack, there will be no mislabeling.

The idea of clean-label DP attack can be naturally extended to backdoor attacks, since typical backdoor attacks are also launched by poisoning the training set with a small set of samples that seem to be mislabeled. In [143], the authors proposed the first clean-label backdoor attack by embedding the backdoor pattern into the target class images instead of source class images. However, these target class images are perturbed before backdoor embedding using a surrogate classifier, such that the key features of the target class are likely removed. Then the classifier will likely learn to classify to

(a) clean target class image (b) clean source class image (c) source class image with backdoor pattern (d) target class image with hidden trigger

Figure 2.5: Example of: (a) a clean target class image, (b) a clean source class image, (c) a source class image with a patch replacement backdoor pattern embedded, and (d) a target class image with the hidden trigger generated according to [3].

the target class solely based on the presence of the backdoor pattern. More recently, a clean-label backdoor attack with a hidden trigger was proposed in [3]. The attacker first launches a classical backdoor attack by poisoning an independently collected training set and trains a surrogate "backdoored" classifier. Then for each of a small set of clean target class images, a sample-specific perturbation is obtained by minimizing its difference with source class images embedded with the backdoor pattern in terms of internal layer features. In such way, these perturbed images will likely be close to source class images embedded with the backdoor pattern semantically, though they still visually look like typical target class images. The attacker then injects these perturbed images, without mislabeling, into the training set of the actual victim classifier to be attacked. Example clean-label backdoor training images created using the method in [3] are shown in Fig. 2.5.

2.2.4 Backdoor Attacks against Other Learning Frameworks

Backdoor attacks are typically designed for the classical DNN training pipeline shown in Fig. 2.4, but they can also be adapted to attack other machine learning frameworks. For example, [144] proposed a backdoor attack against deep regression models, with a wide range of applications such as object detection mentioned in Chap. 1. Instead of causing a test-time misclassification, this backdoor attack aims to produce an incorrect (e.g. amplified) output signal, such that, e.g., the bounding box estimation is incorrect. For another example, in [145], a backdoor attack is proposed against a single-agent reinforcement learning framework. By modifying the observation of the agent, e.g. changing the landscape around an agent going through a maze, the agent is supposed to

choose some incorrect action prescribed by the attacker. In [146], a backdoor attack is migrated to a more complicated reinforcement learning framework where multiple agents are competing with each other.

Backdoor attacks have also been extended to transfer learning, which addresses the scenario where there are no sufficient training instances to train a DNN classifier from scratch for some target domain [147]. Basic transfer learning approaches fine-tune a classifier trained on some source domain similar to the target domain, using the limited training examples from this target domain; and usually, only the parameters in the last few layers (close to the output) of the pre-trained classifier are allowed to change during fine-tuning [148]. To this end, [149] proposed to generate backdoor patterns using an autoencoder and retrain the classifier's pre-trained source domain to plant a backdoor mapping – when the same backdoor pattern is present in a test sample, a set of designated neurons (in the layers that will not be fine-tuned) will be largely activated. Then, even if the last few layers of the classifier are fine-tuned for the target domain, there will still likely be a misclassification caused by the activation of these neurons when a backdoor pattern is present.

Lastly, we introduce a research line studying backdoor attacks (and defenses) against federated learning. As a distributed learning framework, federated learning enables a large number of participants/agents to learn a model (e.g. a classifier) together, but without sharing their training samples with others [150]. Initial backdoor attacks against federated learning are launched by poisoning the training set of one or several agents using samples embedded with the same backdoor pattern [151]. A more recent approach divides the backdoor pattern into several parts; and for each agent whose training set will be poisoned, a small set of backdoor training images are created, but with only one part of the divided backdoor pattern embedded [152]. For example, suppose the backdoor pattern is a 2×2 box embedded by patch replacement which occupies four pixels at the bottom left of the image. Then, the attacker selects four agents. For each agent, the attacker creates a small set of backdoor training images in the same way as for a typical backdoor attack, but with the backdoor pattern being a patch replacement using one of the four pixels. During testing, a classifier being attacked will likely misclassify test images with the 2×2 box to the attacker's target class. A certified defense against this attack is proposed in [153].

2.2.5 Backdoor Attacks in Other Domains

Backdoor attacks (and defense) are initially proposed and heavily studied for the image domain. But recently, they are extended to domains other than images. In [154], a backdoor attack against LSTM-based text classifiers is proposed. The classifier being attacked will predict to the attacker's target class when a specific trigger sentence is present. In a more recent work, backdoor attacks are extended to target BERT-based language models ([155]), where backdoor patterns based on characters, words, and sentences are investigated respectively [156]. In [157], a clean label backdoor attack is proposed against video recognition models, where the backdoor pattern is a small patch located at a corner of each frame. In [158], the first backdoor attack against acoustic models is proposed. The authors considered a speech verification regime that involves multiple users, with the backdoor pattern being a piece of acoustic signal with some prescribed spectrum attached to the normal (i.e. benign) utterance of the user.

In Chap. 5 of this thesis, we extend backdoor attacks to point cloud classifiers, where the backdoor pattern is a small set of inserted points with fixed spacial location. In a concurrent work, a different backdoor attack against point cloud classifiers was proposed, with the backdoor pattern being a universal transformation of the point cloud, e.g. rotation of all points in an aligned point cloud around the origin for some prescribed angle [159].

2.2.6 Backdoor Attacks against Non-DNN Classifiers

Existing studies of backdoor attacks mainly focus on DNN classifiers. However, backdoor attacks may be extended to shallow neural networks with only few layers (or even linear classifier) or other types of classifiers like decision trees. However, it is challenging to design the backdoor pattern for, e.g., a linear classifier. Existing backdoor patterns such as an additive perturbation with a small perturbation size may not be learnable due to the limited representation power of a linear classifier. Thus, study of backdoor attacks and defenses for these classifiers are left to future work.

2.3 Deployment of Backdoor Defense

Backdoor defenses can be deployed at any stage, from the training stage to the operation stage, for a DNN classifier (Figure 2.6). Different defenses designed for the same scenario can be deployed in parallel, e.g. to "cover" multiple types of backdoor patterns. Also,

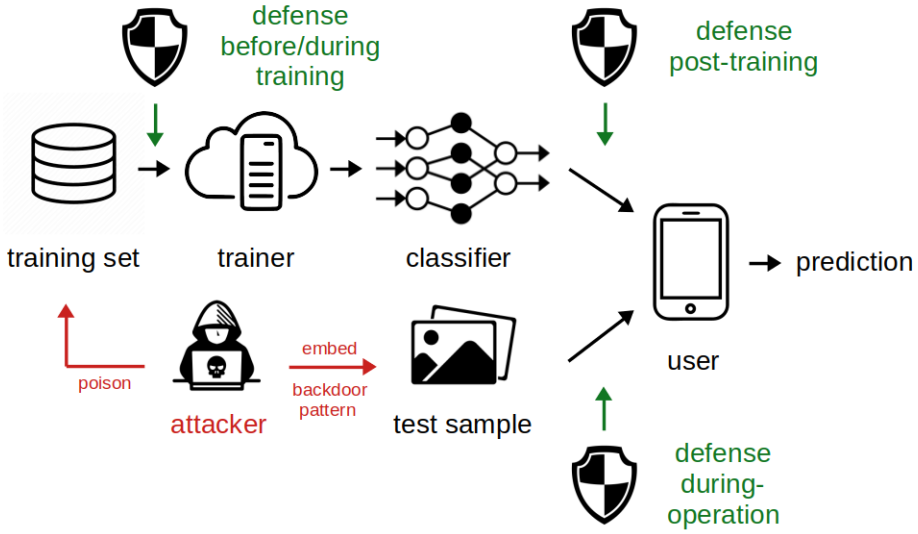


Figure 2.6: Backdoor defenses can be deployed before/during training, post-training, or during the operation of the classifier.

defenses for different scenarios can be deployed independently, e.g. one by the trainer during training and another by the user post-training, to achieve better security. In this section, we discuss the three defense scenarios illustrated in Figure 2.6 in detail. These details are the preliminaries to the defenses we will present in the next three chapters for each of the three defense scenarios.

2.3.1 Before/During Training Scenario

2.3.1.1 Goals of Before/During Training Defender

In the before/during training scenario, the defender could be responsible for training the classifier. The defender's goals are:

- Detecting if the training set has been poisoned.
- If there is poisoning, *correctly* identifying and removing training samples with the backdoor pattern before training (or retraining since many defenses require first training a classifier before detecting any potential poisoning).
- Reducing the number of clean training samples being falsely removed, such that there are sufficient clean samples to train an accurate classifier.

Note that before/during training cannot be deployed if the training authority is the

attacker who aims to provide a classifier with backdoor, e.g. the attack scenario considered in [109].

2.3.1.2 Knowledge and Capabilities of Before/During Training Defender

A before/during training defender has full control of the training process, including full access to the training set and the capability to train DNN classifiers. However, the defender does not know *a priori* if there is data poisoning and, if so, which samples may be poisoned. The defender also has zero knowledge about the backdoor pattern may be used, the source class(es) and the target class of any possible attacks. Moreover, the defender does not possess any training samples guaranteed (known) to be clean (i.e. without backdoor pattern) – this restriction makes before/during training detection of backdoor data poisoning a challenging problem, since there are no clean samples for reference (to identify any backdoor training samples).

2.3.2 Post-Training Scenario

2.3.2.1 Goals of Post-Training Defender

The defender in the post-training scenario could also be the user/consumer of the trained, possibly backdoor-attacked classifier, who does not know whether the training authority is trustworthy and cannot force the training authority to deploy a before/during training defense. For example, a smart-phone user downloading an app with a DNN classifier from online wants to know if the classifier is backdoor attacked or not. The fundamental goal of a post-training defender is:
Detecting whether the classifier has been backdoor-attacked or not.
If the classifier is deemed to be not attacked, it can be directly used for its designated task. But if a backdoor attack is detected, the defender has the following choices:

- Discard the classifier being attacked and replace it with a new classifier, e.g., obtained from a trustworthy training authority.
- If there are no classifiers for replacement, infer the source class(es) and the target class involved in the attack. Especially, with the target class correctly inferred, the classifier's decisions to classes other than the inferred target class may still be trusted.
- Estimate the backdoor pattern used by the attacker. Such an estimation will be helpful to build an in-flight detector to reject test samples with the backdoor

patterns embedded during the classifier's inference stage (see Sec. 2.3.3 for details). Or the estimated backdoor pattern can be used for backdoor mitigation, mentioned next.

- Mitigate the backdoor attack by disabling the backdoor mapping. For example, "unlearn" the backdoor mapping by fine-tuning the classifier on samples embedded with the estimated backdoor pattern, but without mislabeling to the target class [109].

We advocate that backdoor detection is the first and the most important step of post-training backdoor defenses. This is because many existing post-training methods that mitigate backdoor attacks inevitably cause degradation to the classifier's accuracy on clean test samples. For example, [160] proposed to remove neurons with low activations on clean samples – these neurons are believed to be "reserved" for the backdoor mapping. However, since there is no guaranteed "dichotomization" of neurons, with most solely dedicated to "normal" operation but with some solely dedicated to implementing the backdoor, removal of these neurons may hurt the classifier's accuracy on clean samples. Especially when attack instances are occasional, the expected "loss" for applying backdoor mitigation methods without detection may be large.

2.3.2.2 Knowledge and Capabilities of Post-Training Defender

The knowledge and capabilities of a post-training defender are following:

- The defender has full access to the architecture and the parameters of the classifier being inspected, but has no prior knowledge about the existence of any backdoor attacks.

- The defender possesses an independent labeled, clean dataset with samples from all classes. This is a mild assumption used by most existing post-training defenses [109–112, 124, 161]. In practice, it is common for (e.g.) smart phone users to collect some clean samples to validate their new app upon downloading. For example, many people using Siri for the first time tend to check if their voice can truly be recognized.

- The defender has no access to the (possibly poisoned) training set used to design the classifier. For example, for a legacy classifier system, the training data may be lost or inaccessible. This assumption makes post-training detection of backdoor

attacks both an interesting and challenging problem, as any information about a possible backdoor is only latently available through the learned DNN weights (unlike in the before/during training scenario).

- The defender is not allowed to perform massive training of DNN classifiers; there are also no clean classifiers for reference (e.g. providing a baseline for the normal behavior of the classifier). Especially when the defender is an ordinary cell phone user who has limited data access (that is merely sufficient for backdoor detection) and computational resources, training even a shallow DNN classifier may not be feasible. However, this assumption is relieved by some post-training defenses ([162, 163]), as will be discussed in Chap. 3.

- The defender assumes only that the attack, if present, involves a single target class and that the attacker will seek to make the backdoor human-imperceptible. No knowledge of the backdoor pattern, the number of source classes, the label(s) of the source class(es) or the label of the target class are available to the defender. This assumption is made by most existing post-training defenses including some of ours that will be described in Chap. 3. But also in Chap. 3, we proposed an unsupervised post-training backdoor detection method that is able to handle backdoor attacks with arbitrary number of target classes.

2.3.3 In-Flight Scenario

2.3.3.1 Goals of In-Flight Defender

Like post-training defenders, in the in-flight scenario, the defender is also the user/consumer of the trained classifier. An in-flight defense is usually deployed after the classifier is detected to be attacked by a post-training defense. For any single test sample predicted to the target class inferred by the post-training defense, an in-flight defender aims to:

- Detect whether the test sample is embedded with a backdoor pattern.

- If so, infer the true class of origin (i.e. the source class of this test sample).

Thus, in-flight defense can also be viewed as mitigation of the backdoor attack detected by post-training defenses.

2.3.3.2 Knowledge and Capabilities of In-Flight Defender

The knowledge and capabilities of an in-flight defender is the same as a post-training defender, except for the following:

- The defender has access to the DNN detected as attacked, together with the target class inferred by the post-training.
- The defender does not make any independent assumptions (for images for example) about the type, shape, or location of the attacker's backdoor pattern, but requires the backdoor pattern estimated by the post-training defense. Thus, in-flight defenses can be coupled with any post-training defense to address a variety of backdoor patterns.
- The time consumption for inferring whether a test sample is embedded with a backdoor pattern should be negligible compared with the inference time required by the DNN. Note that such time for detection inference does not include the time for offline modeling (e.g. learning the distribution of the internal layer activations for clean samples [164]).

Chapter 3
Backdoor Defense Post-Training

3.1 Overview

In this chapter, we focus on backdoor defenses deployed post-training. As described in Chap. 2, a post-training defender aims to infer whether a DNN classifier $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ has been backdoor attacked or not. The defender has no access to the classifier's training set, but possesses an independently collected data set $\mathcal{D} = \cup_{c \in \mathcal{Y}} \mathcal{D}_c$ consisting of samples from all classes in $\mathcal{Y}$ for detection. This challenging problem has been addressed by many existing works including ours. Like most works in this field, we focus on the image domain, but some of the ideas in this chapter can be extended to other domains such as point clouds (see Chap. 5).

In the following, we first review other post-training defenses in parallel with ours. Then, we present our defenses in the following order:

- First, in Sec. 3.3, we propose the basic version of our reverse-engineering-based defense (RED) that detects backdoor attacks on DNNs post-training [111], [110], [124]. Focusing on additive perturbation backdoor patterns, our method first estimates a small perturbation that induces high group misclassification for each putative (source, target) class pair by solving an optimization problem. Then, robust anomaly detection is applied to the perturbation size/norm statistics to infer the class pairs involved in the attack, if there are any. For this baseline post-training defense:
  - We investigate several different objective functions for perturbation estimation.
  - We propose two inference approaches (as alternatives to each other) for performing the anomaly detection.

  - We extend our approach to account for class confusion information – this allows us to reliably distinguish backdoor attacks from class pairs that merely possess naturally high class confusion.
  - We make some novel experimental observations about DNN "generalization" induced by backdoors (a "collateral damage" phenomenon).

- Second, in Sec. 3.4, we extend our baseline RED to detect other types of image backdoor patterns, including blended patterns [2] and multiplicative perturbations. To be specific, the perturbation estimation approach of RED is extended from applying to the input space to applying to the internal layer feature space.

- Third, in Sec. 3.5, we propose a post-training defense targeting image backdoor patterns embedded by patch replacement. In particular, we focus on backdoor patterns that are visually plausible to the scene. Such scene-plausible backdoor patterns do not have a fixed spatial location in the image. This is consistent with physically implemented backdoor attacks using some type of object. We identify and experimentally analyze two important properties – spatial invariance and robustness – of scene-plausible backdoor patterns. Leveraging these properties, our detector is based on a maximum achievable misclassification fraction (MAMF) statistic, obtained by estimating a putative backdoor pattern for each putative (source, target) class pair using a sequence of common masks (i.e. the same masks for all images used for detection), with arbitrary spatial location for the mask(s). Our detection inference uses an easily chosen threshold. No assumptions on the shape, spatial location or object type of the backdoor pattern are made. Together with the basic RED in Sec. 3.3 and its extension to internal layers in Sec. 3.4, we have "covered" most existing types of backdoor patterns.

- Fourth, in Sec. 3.6, we improve the computational efficiency and data efficiency of the basic RED by proposing a Lagrange-based RED (L-RED) to detect backdoor attacks with arbitrary number of source classes. For each putative target class, we jointly estimate a perturbation and a weight vector over the classes other than the target class to maximize the weighted misclassification fraction from all these classes to the target class. A similar anomaly detection approach as for our basic RED can be applied here to infer the target class; the source classes are inferred based on the estimated weight vector for the estimated target class.

- Fifth, in Sec. 3.7, we consider classification domains with only two classes, where

both classes can possibly be a backdoor target class. Instead of performing anomaly detection as existing REDs do, here, we process each class independently using a novel detection statistic dubbed expected transferability (ET). We show that for ET there is a large range of effective choices for the detection threshold, which commonly contains a particular threshold value effective for detecting backdoor attacks irrespective of the classification domain or particulars of the backdoor attack. This common threshold is mathematically derived and is based on properties for general classification tasks that have been verified empirically by many existing works. Thus, no domain-specific supervision is needed for our method. Moreover, our proposed method is more generally applicable to multi-class scenarios, with arbitrary number of attacks.

3.2 Related Works

The first post-training backdoor defense is the fine-pruning approach proposed in [160]. The defender prunes neurons in the penultimate layer of the DNN in increasing order of their average activations over the clean data set, doing so up until the point where there is an unacceptable loss in classification accuracy on the clean data set. As mentioned previously, FP does not detect presence of backdoor attacks – directly applying FP to a classifier without backdoor detection will cause non-negligible degradation to the classifier's clean test accuracy. In [165], a neural attention distillation (NAD) method was proposed to unlearn the backdoor mapping by fine-tuning the classifier guided by a teacher classifier. A more recent defense proposed by [166] unlearns the backdoor mapping using an adversarial-training-based method. Like FP, this method does not involve backdoor detection – however, both methods can be deployed following (and informed by) our backdoor detector to mitigate any backdoor attacks being detected.

In parallel with our works, there are other reverse-engineering-based post-training backdoor defenses proposed. Perhaps the most important defense in this category is the neural cleanse (NC) method proposed to detect backdoor attacks with patch replacement patterns embedded by Eq. 2.2 [109]. Like our basic RED that will be introduced in Sec. 3.3, the NC detector consists of a backdoor pattern estimation step and a detection inference step based on statistical anomaly detection. The intuition behind NC is that if there is a backdoor attack with target class $t^* \in \mathcal{C}$, there will be a small image patch with an associated mask with a small size (measured by the l_1 norm) that induces high group misclassification to this target class. Thus, for each putative target class $t \in \mathcal{Y}$

NC reverse-engineers the image patch $\mathbf{u}$ and its associated mask $\mathbf{m}$ by minimizing:

$$L_{\text{NC}}(\mathbf{u}, \mathbf{m}, t) = -\frac{1}{|\mathcal{D} \setminus \mathcal{D}_t|} \sum_{\mathbf{x} \in \mathcal{D} \setminus \mathcal{D}_t} p(t|m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) + \lambda \cdot ||\mathbf{m}||_1, \tag{3.1}$$

on images from all classes other than class t. Here, we represent the DNN's posterior for class t regarding any input image $\mathbf{x} \in \mathcal{X}$ by $p(t|\mathbf{x})$. Then, NC applies median absolute deviation (MAD) [167] to the l_1 norm of the masks estimated for all putative target classes. An attack is detected if there is a class with an abnormally small mask norm. However, NC is limited to detecting attacks with patch replacement backdoor patterns (and ones which occupy only a small spatial support). Moreover, the formulation of NC's backdoor pattern reverse engineering problem heavily relies on the assumption that all classes other than the target class are source classes, which may not hold for many practical applications. Finally, NC requires the proper setting of a hyperparameter penalizing overly large masks.

Inspired by NC, a method named "TABOR" was proposed in [112], which focuses on detecting a very special subset of patch replacement backdoor patterns located in any one of the four corners of the image. By assuming that the spatial support of the image patch should be very small and non-scattered, also without covering any key features in the image, TABOR adds several regularization terms penalizing any (patch, mask) that violates these assumptions. In [161], a DeepInspect method was proposed, which estimates an additive perturbation for each putative target class using a conditional generative adversarial network (cGAN) [168]. Remarkably, in this work, it is first shown that post-training backdoor detection can be performed on images generated by model inversion [169]. Inspired by [70], a black-box backdoor detection (B3D) method is proposed by [170], where, in addition to the assumptions for the post-training defense scenario introduced in Chap. 2, the defender is further assumed to have no access to the parameters of the classifier to be inspected. To solve the backdoor pattern reverse-engineering problem, B3D proposed a gradient-free optimization method using the queries on a batch of clean images. In addition, the authors of [170] extended B3D to reverse-engineer backdoor patterns on images synthesized following the same black-box setting (i.e. by minimizing a class conditional cross-entropy loss based on a similar gradient free method). In [171], an "ABS" method with a novel objective function for backdoor pattern reverse-engineering is proposed. Instead of directly maximizing the misclassification fraction, [171] first identifies a set of candidate neurons that are likely associated with the backdoor mapping; then, a small image patch and its associated mask

is optimized to maximize the activation of these neurons. As for detection inference, ABS solves the pattern estimation problem for each putative target class, using clean images from a subset of classes other than this target class. The estimated patch replacement pattern is then embedded into clean images from all other classes excluding the putative target class. If there is a sufficiently large fraction of images being (mis)classified to some putative target class (e.g. above some threshold), an attack is detected with this putative class deemed to be the attacker's target class. However, like NC, this method is designed for patch replacement backdoor patterns and requires that all classes other than the target class are the source classes of the attack. Finally, in [113], a detection statistic based on the similarity between universal and sample-wise backdoor pattern estimation is proposed. For each putative target class $t \in \mathcal{Y}$, the common backdoor pattern estimation problem aims to find an image patch and an associate mask to: a) induce a group of images from classes other than t to be misclassified in an untargeted fashion (i.e., to any class other than their originally labeled classes); b) not induce any class t images to be misclassified; and c) have as small mask size (measured by l_1 norm) as possible. Thus, inspired by the CW loss [43], the following loss is minimized for each putative target class t:

$$\begin{aligned} L_{\text{CSC}}(\mathbf{u}, \mathbf{m}, t) = &\sum_{i \in \mathcal{Y} \setminus t} \sum_{\mathbf{x} \in \mathcal{D}_i} \max\{h_i(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) - \max_{j \neq i} h_j(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})), -\kappa\} \\ &+ \sum_{\mathbf{x} \in \mathcal{D}_t} \max\{\max_{j \neq t} h_j(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) - h_t(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})), -\kappa\} + \lambda ||\mathbf{m}||_1, \end{aligned} \tag{3.2}$$

where $h_i(\cdot) : \mathcal{X} \to \mathbb{R}$ is the logit (right before softmax) for class $i \in \mathcal{C}$ [43]. Then, for each image not from class t, a sample-wise backdoor pattern is estimated to: a) induce the image to be misclassified to class t; and b) have as small mask size (measured by l_1 norm) as possible. Thus, the following loss is minimized for each $\mathbf{x} \in \cup_{i \in \mathcal{Y} \setminus t} \mathcal{D}_i$:

$$L_{\text{CSSW}}(\mathbf{u}, \mathbf{m}, t) = \max\{\max_{j \neq t} h_j(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) - h_t(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})), -\kappa\} + \lambda ||\mathbf{m}||_1. \tag{3.3}$$

For each image from $\cup_{i \in \mathcal{Y} \setminus t} \mathcal{D}_i$, the estimated common backdoor pattern and the estimated sample-wise pattern are embedded in the image, and the associated penultimate layer features are obtained. The cosine similarity between these two feature vectors is computed for all these images. The average cosine similarity for all images in $\cup_{i \in \mathcal{Y} \setminus t} \mathcal{D}_i$ is the detection statistic for class t.

Apart from the REDs above, there is also a family of defenses that builds a binary

"meta classifier" to distinguish classifiers "with" and "without" backdoor attack. In [162], an independent clean data set is used to train a large number of shallow classifiers. Then, a large number of backdoor attacks, each with a different backdoor pattern sampled from some attack distribution, is launched to poison this clean data set independently. For each attack, a shallow classifier (with attack) is trained on the associated poisoned data set. Next, each shallow classifier (with or without attack) is queried by a small set of clean samples; the obtained internal layer features for these samples are obtained as the representative features of the shallow classifier. These features with the labels ("with" or "without" attack) are then used to train the meta-classifier. In [163], a similar approach is proposed, where a more abundant set of backdoor attacks with a large variety of backdoor patterns are involved in training the meta-classifier. However, these approaches require addition assumptions about the defender – the defender needs to be able to perform DNN training and to possess a relatively large number of clean samples (much more than the clean samples required by REDs).

3.3 Basic RED: Detecting Additive Backdoor Patterns

3.3.1 Key Ideas of Basic RED

Many works on TTE attacks [66, 74] have shown that the trained classifier's decision for a single image from class $s \in \mathcal{Y}$ can be altered to class $t \in \mathcal{Y}$ $(s \neq t)$ by adding a small, *image-customized* perturbation. However, altering the class decision for *every* image (or most images) from class s to class t using a *common* additive perturbation is expected in general to require a large perturbation size/norm [172]. The premise behind our basic RED is that for a classifier that has been backdoor attacked with source class $s^* \in \mathcal{Y}$, target class $t^* \in \mathcal{Y}$, and an additive backdoor pattern $\mathbf{v}^*$ with a small size/norm for imperceptibility, the required perturbation size for a common perturbation to induce misclassification to t^* for most images from class s^* is *much smaller* than for class pairs that have not been backdoor-poisoned – in fact, one such common perturbation (it need not be unique) is the backdoor pattern $\mathbf{v}^*$ itself. Thus, if one can find, for any class pair $(s, t) \in \mathcal{Y} \times \mathcal{Y}$ $(s \neq t)$, a small perturbation that induces most images from s to be misclassified to t, this is indicative that the DNN has been backdoor attacked with attacker's source class s and target class t.

Based on the above, for each class pairs $(s, t) \in \mathcal{Y} \times \mathcal{Y}$ $(s \neq t)$, we define the optimal

perturbation $\mathbf{v}^*_{st}$ that induces at least π-level group misclassification as the solution to:

$$\begin{aligned} \underset{\mathbf{v}}{\text{minimize}} \quad & d(\mathbf{v}) \\ \text{subject to} \quad & \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \mathbb{1}(f(m_{\text{add}}(\mathbf{x}; \mathbf{v})) = t) \ \geq \ \pi, \end{aligned} \tag{3.4}$$

where $d(\cdot)$ is the metric for measuring the size (l_1 norm) or the energy (squared l_2 norm) of a perturbation $\mathbf{v}$, and $\mathbb{1}(\cdot)$ is the logic indicator function. $\pi \in (0, 1]$ can be considered the minimum misclassification fraction to deem a backdoor attack "successful" – *e.g.* we might set $\pi = 0.8$. The choice of π does not have strong effect on our detection performance because the required perturbation size for the true attack pair (s^*, t^*) is observed to be abnormaly small, compared with the size for non-attack class pairs, over a large range of π values, as will be shown in Sec. 3.3.6.4. In practice π can be set by the user.

Our detection procedure consists of two steps. First, we estimate $\mathbf{v}^*_{st}$ for each class pair $(s, t) \in \mathcal{Y} \times \mathcal{Y}, s \neq t$. Second, an inference procedure is performed based on the collection of statistics $\{d(\mathbf{v}^*_{st})\}$ for $\forall(s, t)$ class pairs. If the classifier is attacked and the backdoor involves, for example, a single (source, target) class pair (s^*, t^*), we would expect $d(\mathbf{v}^*_{s^*t^*}) \ll d(\mathbf{v}^*_{st})$ for all $(s, t) \neq (s^*, t^*)$. If the backdoor attack involves multiple source classes, e.g. a set of classes $\mathcal{S}^*$, we would expect for any $c \in \mathcal{S}^*$, $d(\mathbf{v}^*_{ct^*}) \ll d(\mathbf{v}^*_{st})$ for all (s, t) satisfying $t \neq t^*$. If the classifier has not been attacked, we would expect there are no such anomalies. Our actual inference procedure is detailed in Sec. 3.3.3.

3.3.2 Pattern Estimation of Basic RED

Unfortunately, (3.4) does not have a closed-form solution, and cannot be solved using gradient-based methods because of the non-differentiable indicator function. Hence, instead we choose the perturbation to minimize a *surrogate* objective function. Note that choosing a perturbation to induce group *misclassification* is quite reminiscent of optimization of a parameterized classification model to maximize the *correct* classification rate. Further, note that there is no singular objective function used in practice for learning good classifiers – cross entropy, discriminative learning that seeks to minimize a soft error count measure [173], classifier margin, as well as other training objectives [58] have all been shown to be "good" surrogate objectives for the (non-differentiable) classification error rate in that minimizing them to determine the classifier leads to good (test set) classification accuracy in practice. Similarly, there are multiple plausible surrogate

objectives for the ideal problem (3.4). While in Sec. 3.3.4 we discuss various alternatives, in our experiments we have optimized the perturbation (*i.e.* estimated a putative backdoor pattern) by performing gradient descent on the objective function

$$L_{\mathrm{B-RED}}(\mathbf{v},(s,t)) = -\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\in\mathcal{D}_s} p(t|m_{\mathrm{add}}(\mathbf{x};\mathbf{v})), \tag{3.5}$$

until the constraint in problem (3.4) is satisfied. The algorithm is summarized below.

Algorithm 1 Optimization of additive perturbation pattern for class pair (s,t).

1: **Input**: Classifier $f:\mathcal{X}\to\mathcal{Y}$ for inspection; clean data set $\mathcal{D}_s$ containing images from class s; target misclassification fraction π; step size δ.
2: **Initialization**: $\mathbf{v}=\mathbf{0}$, $\rho = \frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\in\mathcal{D}_s}\mathbb{1}(f(\mathbf{x})=t)$
3: **while** $\rho<\pi$ **do**
4: $\quad\mathbf{v}\leftarrow\mathbf{v}+\delta\cdot\nabla L_{\mathrm{B-RED}}(\mathbf{v},(s,t))$
5: $\quad\rho\leftarrow\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\in\mathcal{D}_s}\mathbb{1}(f(m_{\mathrm{add}}(\mathbf{x};\mathbf{v}))=t)$
6: **Input**: Estimated perturbation $\mathbf{v}^*_{st}=\mathbf{v}$.

There are several things to note about Algorithm 1. First, ρ is updated in each iteration as the misclassification fraction induced by the current perturbation $\mathbf{v}$. Second, note that we do not explicitly impose a constraint on the perturbation size – the perturbation is initially set to zero, with its size tending to grow with iterations. While a smaller-sized perturbation inducing π-level misclassification could be achieved by minimizing $L_{\mathrm{B-RED}}$ subject to an explicit constraint on the perturbation size, this would also lead to a Lagrangian optimization problem that would require the choice of a Lagrange multiplier specifying the value of the constraint – appropriate (likely search-entailed) choice of the Lagrange multiplier would complicate this constrained optimization problem. In practice, we have found that gradient descent on $L_{\mathrm{B-RED}}$ with termination once π-level misclassification is achieved yields small perturbations for backdoor pairs *relative* to those required for non-backdoor pairs. This is sufficient for successful anomaly detection of backdoor pairs. Third, we recognize that the gradient of $L_{\mathrm{B-RED}}$ has non-zero contributions from samples even once they are successfully misclassified. Again, we do not claim that $L_{\mathrm{B-RED}}$ is the best surrogate for misclassification count that could be used. While we have found it to be quite effective, several surrogate functions are identified in Sec. 3.3.4, and can instead be used, within our detection framework. Fourth, δ is the step size for updating $\mathbf{v}$. If δ is too small, the execution time can be very long. If δ is too large, the algorithm may terminate in a few steps with a resulting ρ much larger than π and with the resulting perturbation much larger in size/energy than that required to

induce π-level group misclassification. In practice, a suitable δ can be chosen via line search. Fifth, in our actual implementation, we set an upper bound on the size/energy of $\mathbf{v}$, with the algorithm terminated when the upper bound is reached even if the π target has not been reached – when π is selected too large (e.g. $\pi = 1$), looking for the required perturbation may be hard or even infeasible (even for a ground-truth attacked pair (s^*, t^*)).

3.3.3 Detection Inference of Basic RED

Let $K = |\mathcal{Y}|$ be the number of classes in the classification domain. The statistics for the detection inference are the set of $\{r_{st}\}$, where $r_{st} = d(\mathbf{v}^*_{st})^{-1}$, i.e. the reciprocal of the "size" of the $K(K-1)$ optimized perturbations (one for each class pair). $d(\cdot)$ can *e.g.* be the l_1 norm or l_2 norm. We take the reciprocal because otherwise anomalies (corresponding to backdoor class pair (s^*, t^*) or (s, t^*) for $\forall s \in \mathcal{S}^*$), if they exist, will cluster near the origin. The null hypothesis for our detection inference is that the classifier has not been attacked, which requires the $K(K-1)$ detection statistics to follow a null distribution (i.e. a distribution for the reciprocal statistics for non-backdoor attack pairs). Alternatively, if the classifier has been attacked, the reciprocal statistics corresponding to the class pairs involved in the attack should be large anomalies, with small p-values under the null. Hence our detection inference for each classifier consists of the following two steps: i) estimating a null distribution for non-backdoor class pairs; ii) evaluating whether the largest detection statistic, which is the one most likely associated with a backdoor attack, is very unlikely under the null distribution. The null parametric density form used in our detection experiments is a Gamma distribution, a right-tailed distribution with positive support. We note that other one-tailed distributions, e.g. exponential distribution, inverse Gaussian distribution, etc., can also be used as the null distribution within our approach. Next we present two approaches, as alternatives two each other, for detection inference.

3.3.3.1 First Detection Inference Approach

Because we assume that there is at most one target class, if there is a backdoor attack, there can be at most $K-1$ reciprocal statistics that correspond to backdoor attack class pairs. Thus, we consider the $K(K-1) - (K-1) = (K-1)^2$ *smallest* reciprocals (corresponding to the $(K-1)^2$ optimized perturbations with the largest size/energy) and assume that these are *not* associated with backdoor attacks and thus are suitable for use

in learning the null density function. We exclude the $(K-1)$ *largest* reciprocals from being used for such estimation because these could be associated with the backdoor attack and thus could corrupt estimation of the null model, as will be shown experimentally in Sec. 3.3.6.4. However, one should not simply *naively* learn a null density using the $(K-1)^2$ smallest reciprocal statistics. Note that it is *unknown* which, if any, of the $(K-1)$ largest reciprocals correspond to a backdoor – there may be no backdoor attack on the classifier. Thus, it is incorrect to assume there are *no* null measurements in the interval $r_{\min} < r < r_{\max}$, where $r_{\min}$ is the smallest of the $(K-1)$ largest reciprocals and $r_{\max}$ is the largest of these statistics – some of these $(K-1)$ statistics could correspond to class pairs not involved in a backdoor attack and follow the true null distribution. Even if we do not use these observed statistics to estimate the null, we should not implicitly assume there are no null measurements in this interval. To account for this lack of certainty, we should use the $(K-1)^2$ smallest statistics to learn the *conditional* null density, where we condition on these statistics being less than $r_{\min}$. That is, we condition on the observed statistics being smaller than the *smallest* of the $(K-1)$ statistics that could correspond to a backdoor attack. Thus, we use the $(K-1)^2$ smallest statistics to learn the conditional null density $g(r|r < r_{\min})$ by maximum likelihood estimation (MLE). Once we estimate this conditional density, its parameters then uniquely *determine* the corresponding *unconditional* null density $g(r)$. That is, $g(r) = g(r|r < r_{\min}) \cdot \text{prob}[R < r_{\min}]$, $r \geq 0$ (with R being the random variable for the reciprocal statistic). To our knowledge, this is a novel robust density modeling approach we are proposing. In Sec. 3.3.6.4, we will demonstrate the superiority of this learned null model, compared with the naive null obtained by directly estimating the (unconditioned) null distribution using all the $K(K-1)$ reciprocal statistics.

Given this learned null, we then ascertain if any of the $K(K-1)$ reciprocals deviate from the null. In particular, we evaluate the probability under the null that the largest of these $K(K-1)$ reciprocals is greater than or equal to the observed maximum reciprocal, $r_{\max}$, i.e.

$$\begin{aligned} \text{pv} &= \text{prob}_{\text{null}}[\max\{R_1, \ldots, R_{K(K-1)}\} \geq r_{\max}] \\ &= 1 - \text{prob}_{\text{null}}[\max\{R_1, \ldots, R_{K(K-1)}\} \leq r_{\max}] \\ &= 1 - G(r_{\max})^{K(K-1)} \end{aligned} \tag{3.6}$$

where $G(\cdot)$ is the null cumulative distribution function. If this order statistic p-value is less than a threshold θ, the null hypothesis is rejected and the classifier is claimed to be attacked. Since p-values under the null hypothesis are uniformly distributed on

$[0, 1]$, θ can in principle be set to fix the false detection rate. For example, the "classical" statistical significance threshold $\theta = 0.05$ should induce 5% false detections for classifiers not attacked. If an attack is detected, the class pair corresponding to $r_{\max}$ is inferred to be involved in the backdoor attack. Furthermore, if the attack is detected, the optimized perturbation for the detected pair is our estimation of the backdoor pattern used by the attacker.

3.3.3.2 Second Detection Inference Approach

Similar to the first detection inference approach, we leverage the assumption that there is one target class if there is an attack. Since the class pairs involved in a backdoor attack are assumed to share the same target class, we first conduct K tests, one for each putative target class. In the test for each class t, an estimation of the null density $g_t(r)$ is learned by maximum likelihood estimation (MLE), using the $K(K-1) - (K-1) = (K-1)^2$ reciprocal statistics, excluding the $(K-1)$ reciprocals with target class t. Like the first detection inference approach, the chosen null parametric density form is a Gamma distribution. We then ascertain if any of the $(K-1)$ excluded reciprocals (associated with target t) deviate from this null. That is, we evaluate the probability that the largest of these $(K-1)$ reciprocals under the null density is greater than or equal to their *observed* maximum, $r_{t,\max}$:

$$\begin{aligned} \mathrm{pv}_t &= \mathrm{prob}[\max\{R_{t,1}, \ldots, R_{t,K}\} \geq r_{t,\max} | r_{t,\max}] \\ &= 1 - \mathrm{prob}[\max\{R_{t,1}, \ldots, R_{t,K}\} \leq r_{t,\max} | r_{t,\max}] \\ &= 1 - G_t(r_{t,\max})^{K-1} \end{aligned} \tag{3.7}$$

where G_t is the cdf corresponding to g_t.

Under the null hypothesis, for $\forall t \in \mathcal{Y}$, the order statistic p-value pv_t obtained from (3.7) should be uniformly distributed on the interval $[0, 1]$. Alternatively, if the classifier has been attacked, pv_{t^*}, the order statistic p-value corresponding to the attacker's target class t^*, should be abnormally small. Hence we evaluate the probability (under the uniform distribution) that the smallest of the K order statistic p-values is smaller than or equal to the observed minimum, i.e.

$$\mathrm{pv} = 1 - (1 - \mathrm{pv}_{\min})^K, \tag{3.8}$$

where $\mathrm{pv}_{\min} = \min_t \mathrm{pv}_t$. That is, we apply order statistics *on* order statistics. If this

order statistic p-value is less than a threshold θ, the null hypothesis is rejected and the classifier is claimed to be attacked. Since pv under the null hypothesis is, again, uniformly distributed on $[0, 1]$, similar to the first detection inference method, θ can in principle be set to fix the false detection rate. The estimated target class, source class, and backdoor pattern are $\hat{t} = \arg\min_t \text{pv}_t$, $\hat{s} = \arg\max_s r_{s\hat{t}}$, and $\mathbf{v}^*_{\hat{s}\hat{t}}$, respectively.

Note that for this second detection inference approach, in practice, one can perform only one test instead of K tests. That is, for this single test, we exclude the $K - 1$ statistics associated with class $t = \arg\max_c r_{\cdot c}$ and estimate a null density using all other statistics. This is because, if there is an attack, it is likely that the largest reciprocal statistic is associated with a backdoor class pair; and $\text{pv}_{\min}$ is likely to be the order statistic p-value obtained from this particular test.

3.3.4 Surrogate Objective Function Variants of Basic RED

As noted in Sec. 3.3.2, $L_{\text{B-RED}}$ measures a non-zero gradient contribution even from samples already successfully misclassified. This can be remedied by instead minimizing an objective function similar to that associated with the Perceptron algorithm [58]:

$$L_{\text{B-RED-PA}}(\mathbf{v}, (s, t)) = -\frac{1}{|\hat{\mathcal{D}}_s(\mathbf{v}, t)|} \sum_{\mathbf{x} \in \hat{\mathcal{D}}_s(\mathbf{v}, t)} p(t | m_{\text{add}}(\mathbf{x}; \mathbf{v})), \tag{3.9}$$

where $\hat{\mathcal{D}}_s(\mathbf{v}, t) = \{\mathbf{x} \in \mathcal{D}_s : f(m_{\text{add}}(\mathbf{x}; \mathbf{v})) \neq t\}$.

Another potential concern with $L_{\text{B-RED}}$ in Eq. (3.5) is the use of all images from source class s for perturbation optimization – if the attacker knows the training set and training approach, he/she can mimic (clean) classifier training and identify the training samples from s that are misclassified. It is possible the attacker will exclude these samples from consideration in crafting the backdoor. Accordingly, to mimic the attacker, the defender might consider group misclassification only on the subset of clean source samples that are correctly classified. This leads to the following objective:

$$L_{\text{B-RED-C}}(\mathbf{v}, (s, t)) = -\frac{1}{|\hat{\mathcal{D}}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s, f(\mathbf{x}) = s} p(t | m_{\text{add}}(\mathbf{x}; \mathbf{v})), \tag{3.10}$$

where $\hat{\mathcal{D}}_s = \{\mathbf{x} \in \mathcal{D}_s : f(\mathbf{x}) = s\}$. One can also combine the restrictions in (3.9) and (3.10), summing only over $\mathbf{x}$ such that both $f(m_{\text{add}}(\mathbf{x}; \mathbf{v})) \neq t$ and $f(\mathbf{x}) = s$.

Likewise, one might also hypothesize the attacker used box constraints [66], rather than clipping, in implementing the backdoor to keep image intensity values in the proper

range. If so, the defender might modify the above surrogate problems for box constraints, rather than clipping.

As noted before, one could also consider an optimization problem that explicitly accounts for/constrains the perturbation size (or minimizes the perturbation size given a constraint on the surrogate misclassification objective). Moreover, one can use a soft error count/discriminative learning objective function akin to that used in [173].

Based on this discussion, one can see that many surrogative objective function variants are possible. In Sec. 3.3.6.2, we do include experimentation with some of these alternative surrogates; however, in general we have found that detection performance is not sensitive to this choice. Again, the reason is that all that is needed is that the size of the learned perturbation for a true backdoor pair (s^*, t^*) should be much smaller than for a non-backdoor pair. Minimizing any of these surrogate objectives is sufficient to elicit this difference between $r_{s^*t^*}$ and r_{st}, (s,t) any non-backdoor pair. While our experiments primarily focus on minimizing $L_{\mathrm{B-RED}}$, our detection framework is general and consistent with use of any of the above alternative surrogates, in the search for minimal-sized image perturbations.

3.3.5 Correction of Detection Statistic Using Class Confusion

The premise behind our detection inference, as mentioned before, is that the perturbation size/energy required to induce π-level group misclassification for the true backdoor class pair (s^*, t^*) is less than for other (non-backdoor) class pairs. However, if the initial misclassification fraction $\rho_{st}^{(0)}$ for $(s,t) \neq (s^*, t^*)$ is abnormally high, it is expected that the perturbation size/energy needed to reach π-level group misclassification will be smaller than when this initial misclassification fraction is low, i.e. $d(\mathbf{v}_{st}^*)$ may be abnormally small, possibly resulting in (s,t) being falsely detected.

Here we suggest to correct this effect assuming the confusion matrix information is available (estimable from the available clean data set). For any pair (s,t) such that $\rho_{st}^{(0)} \neq 0$, we fit an (M-th order) polynomial using the sequence of (perturbation size/norm, misclassification fraction) pairs obtained while executing Algorithm 1 for the pair (s,t). This gives a regression relationship between perturbation size and induced misclassification fraction for the pair (s,t). The "compensation" $d_{st}^{(0)}$ is derived as

$$d_{st}^{(0)} = \arg\min_{d_0} \min_{\{a_m\}} \sum_{\tau=0}^{T} \big[\sum_{m=1}^{M} a_m (d(\mathbf{v}_{st}^{(\tau)}) + d_0)^m - \rho_{st}^{(\tau)} \big]^2, \tag{3.11}$$

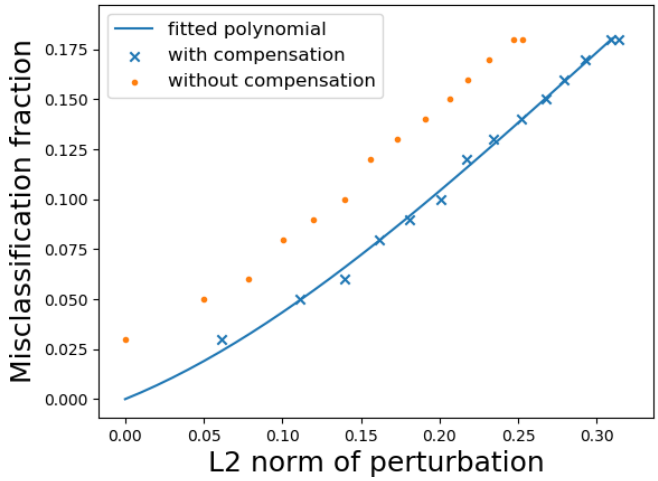


Figure 3.1: An example of correction of perturbation sizes to account for non-zero initial class pair confusion. The orange dots are the (perturbation size/norm, misclassification fraction) sample points for a regime where the perturbation size/norm is small. The blue crosses are the corrected/shifted sample points obtained by adding the correction to the perturbation size/norm. The blue polynomial fits these corrected points.

where the superscript τ is the index of the iteration when executing Algorithm 1. $d_{st}^{(0)}$ is an estimate of the perturbation size needed to induce the initial class confusion $\rho_{st}^{(0)}$. To correct for this (non-zero) initial confusion level, we thus add $d_{st}^{(0)}$ to $d(\mathbf{v}_{st}^*)$. Intuitively, this correction will increase an "abnormally small" $d(\mathbf{v}_{st}^*)$, making it less likely to be falsely detected. Note that $d(\mathbf{v}_{st}^{(0)}) = 0$ since $\mathbf{v}_{st}^{(0)} = \mathbf{0}$.[1] T could be chosen as the index of the last iteration such that either $\rho_{st}^{(T)} \geq \pi$ or such that $d(\mathbf{v}_{st}^{(T)})$ exceeds the upper bound of the perturbation size/norm. Alternatively, we can base the polynomial fit on a smaller number of (perturbation size, misclassification rate) pairs. For (s,t) with $\rho_{st}^{(0)} = 0$, we set $d_{st}^{(0)} = 0$, i.e. no correction is required in this case.

Figure 3.1 gives an example of correction to $d(\mathbf{v}_{st}^*)$ for one class pair using the confusion information. The orange dots are the original (perturbation size/norm, misclassification fraction) sample points obtained from Algorithm 1. The compensation based on an optimized polynomial with order $M = 3$ (the blue curve in the figure) is obtained by solving (3.11). The blue crosses are the sample points after correction. Note that the polynomial is guaranteed to pass through the origin, since the purpose of the correction is to ensure that the initial misclassification rate when there is no perturbation is zero. Then the corrected statistics for detection inference are $\{r_{st}^c\}$, where $r_{st}^c = [d(\mathbf{v}_{st}^*) + d_{st}^{(0)}]^{-1}$.

[1] In Algorithm 1, we neglect the subscripts for simplicity.

3.3.6 Experiments

3.3.6.1 Devising Backdoor Attacks

Dataset: Our main experiments uses the CIFAR-10 data set with 60k color images ($32 \times 32 \times 3$) evenly distributed between ten classes [174]. The data set is separated into a training set with 50k images (5k per class) and a test set with 10k images (1k per class).

Backdoor pattern: We focus on two types of additive perturbation backdoor patterns. Since pixel values are usually normalized (e.g. from $[0, 255]$ for colored images) to $[0, 1]$ before feeding to DNN classifiers, the valid perturbation range per pixel (without clipping) is $[-1, 1]$. The first type is the sparse pixel-wise perturbation, a backdoor pattern that affects only a small subset of pixels, as considered in [2, 117]. Our pixel-wise perturbation is created by first randomly selecting a few pixels; for color images, one of the three channels (i.e. colors) of each selected pixel is randomly selected to be perturbed. The second type of pattern is a global, i.e. image-wide, perturbation, a backdoor pattern that affects all pixels, akin to a global image watermark as considered in [2, 3, 129]. We created a spatially recurrent pattern that looks like a "chess board" – one and only one pixel among any two adjacent pixels was perturbed (in all three channels) positively. For both backdoor patterns, we can adjust the perturbation size for each pixel (e.g. by multiplying a factor) to meet any l_2 norm[2] specification of the perturbation. Visualization of these two backdoor patterns are shown in Fig. 3.2.

Figure 3.2: A sparse pixel-wise perturbation mask with L2 norm 0.6 and a 0.5 offset (left), and a global perturbation mask with L2 norm 10 (right).

Training configuration: We use ResNet-18 as our DNN architecture [14], and cross-entropy loss as our training loss. Training is performed for 200 epochs with mini-batch

[2]There is no strong preference on the norm to be used. l_2 norm is the default, unless specified otherwise. However, our detection algorithm is also effective using the l_1 norm, as shown in Sec. 3.3.6.2.

size of 32, using the Adam optimizer [5], and with training data augmentation options including random cropping, random horizontal flipping, and random rotation. This training achieves an accuracy of 91% on the clean test set when there are no backdoor attacks.

Optimal perturbation size: As the defender, we consider the most stealthy attack. As discussed in Sec. 2.1.4, a larger perturbation size likely leads to a higher attack success rate (ASR), but also puts the attack at the risk of being perceivable to humans. By evaluating perturbations with different l_2 norms, we pick the least human-perceptible one (as small l_2 norm as possible) under the constraints of high attack success rate and low degradation in test accuracy on clean patterns[3]. Such evaluation is given in Figure 3.3. For sparse pixel-wise perturbations and global perturbations, we evaluated l_2 perturbation norms ranging from 0.25 to 1.0 and 0.01 to 0.5, respectively. For each l_2-norm-specified backdoor pattern, we created a single attack realization. We produced 1000 attack images using randomly selected[4] clean training images from the 'automobile' (source) class, embedded with the backdoor pattern following Eq. 2.1. These images (with the backdoor pattern) were labeled to the 'truck' (target) class and added to the training set of 50k clean images. The poisoned training set was then used to train a DNN classifier with the configurations described previously.

To evaluate the effectiveness of the created attacks, we use the ASR and ACC metrics introduced in Sec. 2.1.4. To be specific, a set of backdoor test images was created by adding the same backdoor pattern to the 1k clean test patterns (those not used in training) from the source class. The ASR is evaluated as the fraction of backdoor test images classified to the target class. Also, the accuracy of the classifier on the original 10k clean test patterns is evaluated as the ACC. As depicted in Figure 3.3, for the global perturbation backdoor pattern, the ASR grows with the l_2 norm of the perturbation mask. However, for the sparse, pixel-wise backdoor pattern, the ASR wildly fluctuates when the perturbation norm is low, only becoming stable with further increases in the perturbation norm. Such fluctuation is likely due to the fact that the pixels being perturbed are randomly selected. To launch a successful and human-imperceptible attack, the attacker should choose the l_2 norm for the global perturbation mask to be 0.2, with an ASR of 0.959 and ACC of 0.916 on clean test images. For the sparse, pixel-wise perturbation

[3]Our detector can still detect backdoor patterns with large perturbation size, as will be seen in Sec. 3.3.6.4.

[4]The attacker could alternatively select the images that are easiest to be misclassified to the target class, based *e.g.* on the difference in DNN posterior probability between the winning class and the target class.

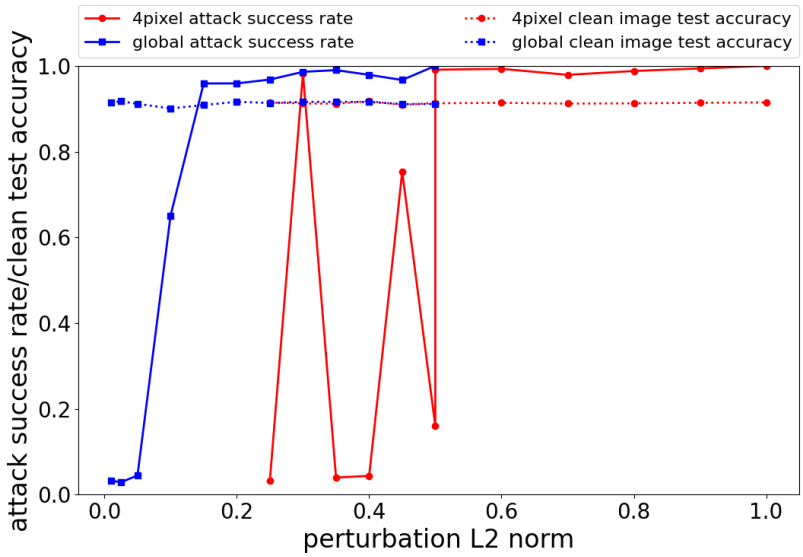


Figure 3.3: Attack success rate (solid) and accuracy on clean test images (dashed) for DNN classifiers under backdoor attacks that use sparse pixel-wise perturbation (red dots) and global perturbation (blue squares), for a range of l_2 perturbation norms (attack strengths).

mask, if the l_2 norm is set to 0.6, an ASR of 0.993 and ACC of 0.914 on clean test images can be achieved.

Creating ensemble of classifiers for defense performance evaluation: Based on the randomness in crafting the backdoor patterns and in the training process[5], to evaluate detection performance, we conducted experiments on four groups of DNN classifiers, 25 classifier realizations per group, as follows:

- **BD-P-S**: Classifiers were trained on the training set poisoned by 1000 backdoor images with 4-pixel perturbations ($||\mathbf{v}^*||_2 = 0.6$). The backdoor images were crafted using clean images from a single source class.

- **BD-G-S**: Classifiers were trained on the training set poisoned by 1000 backdoor images with a global perturbation ($||\mathbf{v}^*||_2 = 0.2$). The backdoor images were crafted using clean images from a single source class.

- **BD-G-M**: Classifiers were trained on the training set poisoned by 900 backdoor images with a global perturbation ($||\mathbf{v}^*||_2 = 0.2$). The backdoor images were crafted using clean images from the nine classes excluding the target class, with 100 images per (source) class.

[5]In each training iteration, the mini-batch is randomly sampled from the training set. Also, for each training image, the type of data augmentation is randomly chosen.

- **Clean**: Classifiers were trained on the clean training set, without data poisoning.

The four groups of classifiers involve sparse, pixel-wise and global backdoor patterns, single-source-class and multiple-source-class attack scenarios, and include a clean classifier group. For each attacked classifier, the clean images used for devising the attack are selected randomly from the 'automobile' (source) class, and the backdoor pattern is generated independently of the selected images. The ASR[6] and the ACC (across the 25 realizations) for the four groups of classifiers are reported in Table 3.1 – all backdoor attacks are successful and the degradation in clean test accuracy is negligible across all experimental realizations. An example of backdoor images with sparse, pixel-wise backdoor pattern (Figure 3.4b), global backdoor pattern (Figure 3.4c) and the original image (Figure 3.4a) are shown in Figure 3.4. The sparse, pixel-wise backdoor pattern (with l_2 norm 0.6) is only human-perceived through careful visual scrutiny of the image. The global backdoor pattern (with l_2 norm 0.2), when added to the original image, is imperceptible even under careful human inspection, since the perturbation size per pixel (and per channel) is only about 5.1×10^{-3}.

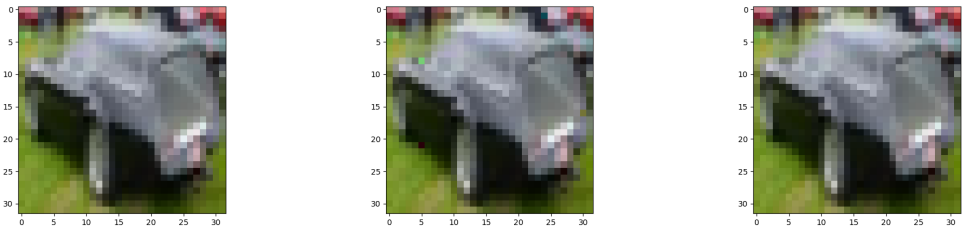
Table 3.1: Attack success rate (ASR) and accuracy on clean test images (ACC) (over all 25 classifier realizations) for the four groups of DNN classifiers.

	ASR	ACC
BD-P-S	0.978 ± 0.035	0.913 ± 0.003
BD-G-S	0.974 ± 0.014	0.912 ± 0.003
BD-G-M	0.990 ± 0.005	0.912 ± 0.003
Clean	N.A.	0.915 ± 0.003

3.3.6.2 Defense Performance Evaluation

We evaluate the performance of our basic RED with several variations (including those based on the different surrogate objective functions in Sec. 3.3.4, the correction approach in 3.3.5, different norms used to create the detection statistics, and different detection inference approaches), compared with the concurrent work neural cleanse (NC) proposed in [109]. The clean data set used for detection (for both our method and NC) for each classifier contains 1000 images (100 per class) randomly sampled from the clean test set held out from training.

[6]The ASR for the BD-G-M group is evaluated on backdoor images crafted using all 9000 clean test images from the nine classes other than the target class.



(a) (b) (c)

Figure 3.4: Examples of backdoor patterns applied to CIFAR-10 images: (a) the original automobile image; (b) automobile with sparse, pixel-wise perturbation ($||\mathbf{v}^*||_2 = 0.6$); (c) automobile with global perturbation ($||\mathbf{v}^*||_2 = 0.2$).

For all the variants of our method, we set π, the target misclassification fraction, to be 0.8. However, we have observed that detection performance is not sensitive to the choice of π (shown in Sec. 3.3.6.4). For the correction approach using the confusion matrix, we selected the order of the polynomial as $M = 3$. Again, other choices of M have little impact on detection accuracy – $M = 2, 4, 5$ gave similar results. By default, all the variants use the inference approach detailed in Sec. 3.3.3.1, except for the last one. Finally, we use the classical confidence threshold $\theta = 0.05$ here. The variants of our basic RED are:

- $L_{\text{B-RED}}$: Eq. (3.5) objective without correction.
- $L_{\text{B-RED-PA}}$: Eq. (3.9) objective without correction.
- $L_{\text{B-RED-C}}$: Eq. (3.10) objective without correction.
- $L_{\text{B-RED}}$-C: Eq. (3.5) objective combined with the confusion-correction.
- $L_{\text{B-RED}}$-l_1: This variant is the same as $L_{\text{B-RED}}$ except using the l_1 norm for detection purposes.
- $L_{\text{B-RED}}$-$I2$: This variant is the same as $L_{\text{B-RED}}$ except using the second inference approach detailed in Sec. 3.3.3.2.

For NC, we use the same setting as in [109]. However, we tested both $\lambda = 1.5$ (as the default in [109]) and $\lambda = 1.0$, where λ is the Lagrange multiplier in NC's objective function (see Eq. 3.1). Considering that the objective function of NC replies on the assumption that all classes other than the target class are source classes, it may be hard for NC pattern reverse-engineering to achieve a high group misclassification fraction for

the true target class, when the attack involves only one source class. Thus, for NC, we not only consider the same target misclassification fraction $\pi = 0.8$ as for our method, we also test it with a lower misclassification fraction $\pi = 0.5$. Moreover, apart from the l_1 norm statistics considered in the original NC paper, we also test NC more thoroughly with l_2 norm statistics. The variants of NC evaluated in our experiments are:

- L_{NC}: l_1-regularized objective function with $\lambda = 1.5$. The anomaly indices are derived from the l_1 norm of each optimized perturbation.
- L_{NC}-l_2: l_2-regularized objective function with $\lambda = 1.5$. The anomaly indices are derived from the l_2 norm of each optimized perturbation.
- L_{NC}-λ1.0: l_1-regularized objective function with $\lambda = 1.0$.
- L_{NC}-π0.5: Same as L_{NC}-l_1-λ1.5, except that $\pi = 0.5$ instead of $\pi = 0.8$.

Table 3.2: Detection accuracy of all the variants of our basic RED (with detection threshold $\theta = 0.05$) and of the NC approach for the four groups of DNN classifiers for CIFAR-10.

	BD-P-S	BD-G-S	BD-G-M	Clean
$L_{\text{B-RED}}$	0.92	0.92	**1.00**	**1.00**
$L_{\text{B-RED-PA}}$	0.92	0.92	**1.00**	**1.00**
$L_{\text{B-RED-C}}$	0.96	0.92	**1.00**	**1.00**
$L_{\text{B-RED}}$-C	0.96	0.92	**1.00**	**1.00**
$L_{\text{B-RED}}$-l_1	**1.00**	0.92	**1.00**	**1.00**
$L_{\text{B-RED}}$-$I2$	88	92	**100**	92
L_{NC}	0.36	0.16	**1.00**	0.84
L_{NC}-l_2	0.56	0.64	**1.00**	0.72
L_{NC}-λ1.0	0.40	0.28	**1.00**	0.76
L_{NC}-π0.5	0.36	0.16	0.88	0.96

In Table 3.2, detection accuracy of each variant of our basic RED and the NC detection methods for each group of classifiers is shown. Accuracy for the groups of classifiers under attack, i.e. BD-P-S, BD-G-S and BD-G-M, is defined as the fraction of classifiers successfully detected as attacked. For our basic RED variants, a successful detection *also* requires the detected source and target class pair (corresponding to the most extreme detection statistic) to be ground-truth involved in the attack. For NC, since it assumes all classes except for the target class are involved in the backdoor attack, there is no inference on the source class(es). In our experiments, we thus consider NC detection

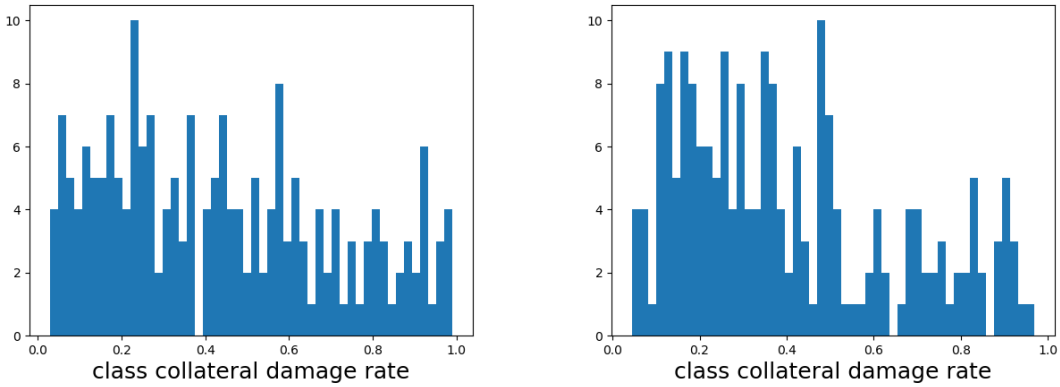
Figure 3.5: Examples of estimated backdoor patterns with ground truth a 4-pixel perturbation (left) and a global perturbation (right), respectively. The estimated global perturbation mask has been scaled by 30 times to be human visualizable.

to be successful if the class label corresponding to the most extreme anomaly index, if detected, is the true backdoor target class label t^*. Thus, for BD-P-S and BD-G-S, our approaches meet a stronger true detection requirement than NC. For the group of clean networks, the detection accuracy is defined as the fraction of networks inferred to *not* be attacked.

All proposed variants of our basic RED achieved perfect detection for the BD-G-M group experiments. Detection accuracy for the BD-P-S and BD-G-S groups is also very high (all above 0.90), with very low false detection on the Clean group. We note that we can always make more conservative inferences and claim a successful detection when *only* the target class is correctly inferred. This is reasonable due to the collateral damage effect that will be shown in Sec. 3.3.6.3. Allowing such conservative inference, all six variants of our approach achieve perfect detection for the BD-G-S group. In fact, all incorrect source classes by the variants of our approach for the results shown in Table 3.2 have high class collateral damage rate (above 88.4%).

NC detection, as expected, achieves strong performance for the BD-G-M group (except for the L_{NC}-$\pi 0.5$ variant) since it is designed for backdoor attacks that involve all $(K-1)$ source classes. However, all the variants of NC are not effective at detecting backdoor attacks involving a single source class, as shown in Table 3.2 by the accuracy for BD-P-S and BD-G-S. Also, the detection power of NC cannot be improved by choosing a smaller detection threshold, since the false detection rate on clean classifiers is then made quite non-negligible.

In addition to accurate attack detection inference, our detection approach also gives an estimate of the ground truth backdoor pattern. The left figure of Figure 3.5 is an estimate of one of the 25 4-pixel perturbations used to create the BD-P-S group. A



(a) BD-P-S group (b) BD-G-S group

Figure 3.6: Histograms of the collateral damage rate statistics for (a) the BD-P-S group and (b) the BD-G-S group.

0.5 offset is added for visualization. The ground truth backdoor pattern is the one on the left of Figure 3.2. Comparing with the ground truth backdoor pattern, we can see that instead of perturbing all four pixels, perturbing the region near the ground truth perturbed pixel on the top left is most effective to induce group misclassification to the target class. The right figure of Figure 3.5 is an estimate of one of the 25 global perturbations used to create the BD-G-S group. Since nearly half of the pixels are perturbed negatively, and the perturbation size is too small to be visualized, we first add an offset such that the resulting perturbations are all positive; then the shifted perturbation mask is scaled by 30 times. Clearly, we recover a "chess board" pattern similar to the ground truth backdoor pattern.

3.3.6.3 Collateral Damage Effect

During the experiments, we noticed that backdoor attacks usually induce "collateral damage" to classes other than s^* or t^*. That is, supposing the classifier has been successfully corrupted by a backdoor involving the pair (s^*, t^*), test images from some classes $\tilde{s} \in \mathcal{C} \setminus \{s^*, t^*\}$ will be classified to t^* with high probability when the same backdoor pattern is added to them. To demonstrate this effect, for each trained DNN in the BD-P-S and BD-G-S groups, for which only one class is designated to be the source class s^* of the backdoor, we conducted eight tests, one for each of the classes $(\tilde{s})$ other than s^* or t^*. In each test, we added the backdoor pattern used for attack to the 1000 clean test images from $\tilde{s}$, and obtained a "class collateral damage rate" as the fraction

of images classified to t^*. In Figure 3.6a and Figure 3.6b, we show the histogram of the $(8 \times 25 =)$ 200 class collateral damage rate statistics for the BD-P-S and BD-G-S groups, respectively. Clearly, significant group misclassification to the target class is induced for classes other than the true source class using the true backdoor pattern, for many of the classifiers. This is explained by some classes ($\tilde{s}$) having patterns that are similar to those of the backdoor's source class – the backdoor attack is not sufficiently "surgical" to *only* induce misclassifications from s^* to t^*. Accordingly, it is not surprising that, for such high "collateral damage" classes, the size/energy of the optimized $\mathbf{v}^*_{\tilde{s}t^*}$ is close to the size/energy of $\mathbf{v}^*_{s^*t^*}$. This thus gives further insight into why we exclude the $K-1$ largest statistics in the set $\{||\mathbf{v}_{st}||\}$ from use in estimating the null distribution for both our detection inference approaches. This also helps explain why we only infer a single (source, target) class pair based on the *most extreme* statistic – "collateral damage" classes, paired with the inferred target class, may be prone to false detection, and it is not so easy to distinguish the "collateral damage" phenomenon for a single source attack from a truly multiple-source backdoor attack. We do not view this as a true limitation of our detection method – our method should be able to accurately identify multiple true source classes (using the second to the $(K-1)$-th largest order statistics) *if* the backdoor attack is more surgical, *i.e.* if it does not induce collateral damage to classes not explicitly attacked.

3.3.6.4 Ablation Studies

Choice of π: Our basic RED is not very sensitive to the choice of π. As an example, we evaluate the detection accuracy of the $L_{\text{B}-\text{RED}}$ variant on the 25 classifiers in the BD-P-S group, with a range of choices of π from 0.4 to 0.9. We use the default detection threshold $\theta = 0.05$. As shown in Figure 3.7, the detection accuracy does increase, but not dramatically with π. The minimum detection accuracy across the range of choices of π is 0.8, which can be further improved by use of a more liberal detection threshold. This phenomenon can be understood from Figure 3.8, in which we plot the sequences of $(||\mathbf{v}^{(\tau)}_{st}||_2, \rho^{(\tau)}_{st})$ for all (s,t) pairs during the perturbation optimization using Algorithm 1 with $\pi = 0.9$, while applying $L_{\text{B}-\text{RED}}$ on an example classifier realization in the BD-P-S group. The sequence corresponding to the ground truth backdoor class pair (s^*, t^*) is represented using red crosses, and is clearly separated from the sequences for the non-backdoor pairs. In other words, there is a huge range for the choice of π (not exceeding 0.9) to achieve correct detection for this example. For any choice of π in such range, the $(||\mathbf{v}^{(\tau)}_{st}||_2, \rho^{(\tau)}_{st})$ sequence for each (s,t) pair is truncated at the minimum τ such

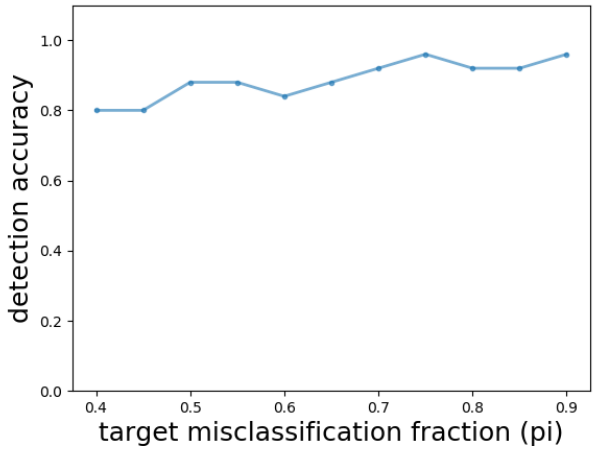


Figure 3.7: Detection accuracy of $L_{\mathrm{B-RED}}$ on the BD-P-S group, with a range of choices of π.

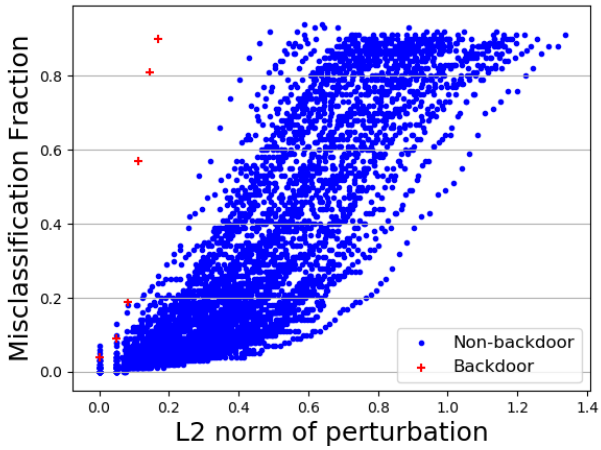


Figure 3.8: Sequences of $(||\mathbf{v}_{st}^{(\tau)}||_2, \rho_{st}^{(\tau)})$ for all (s,t) pairs, including the ground truth backdoor pair (s^*,t^*) represented using red crosses, during the execution of Algorithm 1 with $\pi = 0.9$, while applying $L_{\mathrm{B-RED}}$ on an example classifier realization in the BD-P-S group.

that $\rho_{st}^{(\tau)}$ first exceeds π. The resulting $||\mathbf{v}_{st}^{(\tau)}||_2$ for the backdoor class pair (s^*,t^*), as can be seen from the figure, will be much smaller than those for the non-backdoor pairs, which will lead to a successful detection.

Size of clean data set used for detection: Here we test the impact on detection accuracy of the size of the clean data set available to the defender. Again, we apply the $L_{\mathrm{B-RED}}$ detection (with $\theta = 0.05$) to the classifiers in the BD-P-S group, but with the number of clean images per class used for detection equaling 5, 10, 25, 50, 100 for

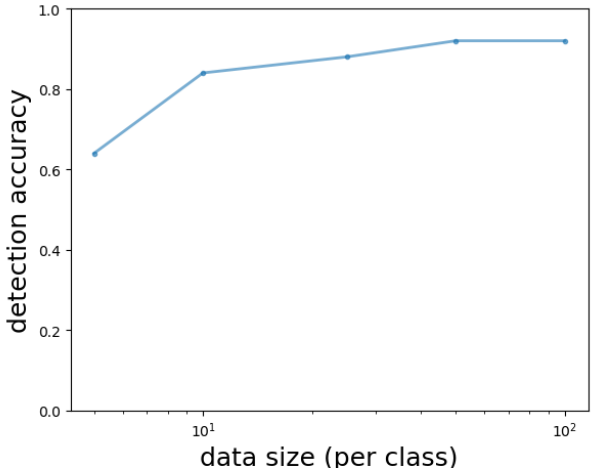


Figure 3.9: Detection accuracy of $L_{\mathrm{B-RED}}$ on BD-P-S for a range of sizes of the clean data set.

each trial, respectively. As shown in Figure 3.9, the detection accuracy increases as the number of clean images per class grows, approaching a stably high value when the number of clean images per class is greater than 25. Note that detection accuracy greater than 80% is achieved with just 10 clean images per class. Comparing with the size of the training set (5000 images per class), the detection cost is "cheap" in terms of the required number of clean images. Even if there are abundant clean images available to the defender, he/she can always use only a subset of images to reduce the defender's computational effort.

Fitting null distribution using all statistics: As described in Section 3.3.3.1, for the first detection inference approach, we fit a conditional null density using the $(K-1)^2$ smallest statistics. One may also consider a naive approach that fits an unconditional null using all detection statistics. In Table 3.3, we show the detection accuracy of the $L_{\mathrm{B-RED}}$ variant with detection threshold $\theta = 0.05$ for the four groups of DNN classifiers based on this naive null learning. Clearly, there is severe degradation in accuracy of detecting backdoors.

Table 3.3: Detection accuracy of the $L_{\mathrm{B-RED}}$ variant with detection threshold $\theta = 0.05$ for the four groups of DNN classifiers, where the null density is estimated using all $K(K-1)$ reciprocal statistics.

	BD-P-S	BD-G-S	BD-G-M	Clean
detection acc.	0.48	0.44	0.52	**1.00**

Size/Energy of the ground truth backdoor pattern: Here we create 25 classifier realizations using the same setting as for the BD-G-S group, except that the l_2 norm of the backdoor pattern is set to 1.2 – much larger than required to launch a successful backdoor attack (and visualizable by carefully scrutinizing the image). When applying the $L_{\mathrm{B-RED}}$ variant for detection, with the detection threshold $\theta = 0.05$, the detection accuracy reaches 1.00. More importantly, the estimated backdoor patterns for the 25 classifier realizations are all scaled "chess board" patterns with l_2 norm around 0.101 ± 0.017, while the l_2 norm of the optimized perturbation for all non-backdoor class pairs across the 25 realizations is 1.243 ± 0.321. That is, the "estimated" backdoor pattern will have the *minimum* norm necessary to achieve the π target level, even if the actual backdoor pattern has much larger norm. Regardless, the estimated backdoor pattern *is* accurate (it is a "chess board" pattern). Also, from this experiment, we note that the proposed detector does not require the size/energy of the ground truth backdoor pattern to be significantly smaller than the size/energy of perturbations required to induce group misclassification for non-backdoor class pairs.

3.3.6.5 Experiments on Other Datasets

We evaluate the performance of our detector on several other datasets. For each dataset, we train *one* classifier with no attacks and *one* attacked classifier using the "chess board" backdoor pattern with $||\mathbf{v}^*||_2 = 0.2$ (i.e. the same backdoor pattern used for the BD-G-S group). For each classifier, the variant of our basic RED with objective function $L_{\mathrm{B-RED}}$ and the $L_{\mathrm{B-RED}} - C$ variant (with correction using the confusion matrix information) are applied for detection using a clean data set (not used for training) containing 100 images per class.

MNIST contains 70000 28×28 gray scale handwritten digit images evenly distributed in 10 classes. The training set contains 60000 images evenly distributed in the 10 classes, and the test set contains the remaining 10000 images. The backdoor pattern is added to 1000 clean images from class '0'. These images are then labeled to class '9' and used for poisoning the training set. We use the LeNet5 as the DNN architecture [15]. The training was performed for 30 epochs with mini-batch size 32 and Adam optimizer [5]. No data augmentation was applied during the training.

CIFAR-100 consists of 60000 32×32 color images from 100 classes, 600 images per class. The training set contains 50000 images evenly distributed in the 100 classes; the test set contains 10000 images. The training set is poisoned using 1000 images with the same "chess board" pattern. These images are created using clean images from the

"beaver", "dolphin", "otter", "seal" and "whale" classes, 200 per class, and are labeled to the "orchid" class. The structure of the classifier is ResNet-34 [14]. The training is performed for 200 epochs, with mini-batch size of 32, using the Adam optimizer [5], and with the training data augmentation option.

We also created a "CIFAR-50" data set, which consists of training and test images from the first 50 classes of the CIFAR-100 data set. Again, the attack is devised using the same source classes, target class and number of images as the attack for CIFAR-100. The structure of the classifier and the training settings are also the same as for CIFAR-100.

Lastly, we reconsider CIFAR-10, but do not allow sufficient training; hence the accuracy on clean test images will be low, with or without attacks. This is achieved by using the same training settings as described in Sec. 3.3.6.1, but with the training data augmentation option disabled. Here we simulate practical scenarios where the training resources are limited, or where the training is poorly optimized. The attack for this test is crafted in the same way as for the BD-G-S group.

Table 3.4: Accuracy on clean test images (ACC) when the classifier is not attacked, attack success rate (ASR), and ACC for the classifier under attack, for the four data sets.

	ACC (w/o attack)	ASR	ACC (w/attack)
MNIST	0.994	1.00	0.994
CIFAR-50	0.791	0.980	0.783
CIFAR-100	0.731	0.986	0.723
CIFAR-10 (no data aug.)	0.843	0.973	0.837

In Table 3.4, we show that the attack is successful (high ASR and negligible degradation in ACC) on all four data sets.

Table 3.5: Order statistic p-value when applying $L_{\mathrm{B-RED}}$ and $L_{\mathrm{B-RED}} - C$ to each of the two classifiers (one attacked and the other not) for each data set, respectively.

	AD-J-P		AD-J-C	
	w/o attack	w/attack	w/o attack	w/attack
MNIST	0.890	u.f.	0.797	u.f.
CIFAR-50	1.76×10^{-2}	u.f.	0.291	u.f.
CIFAR-100	2.75×10^{-4}	u.f.	0.222	u.f.
CIFAR-10 (no data aug.)	6.39×10^{-2}	6.72×10^{-9}	0.476	1.65×10^{-9}

In Table 3.5, for each of the two classifiers (one attacked and the other not attacked)

associated with each data set, we show the order statistic p-value when the $L_{\text{B-RED}}$ and $L_{\text{B-RED}} - C$ variants are applied. For both detection variants, the order statistic p-value, when the classifier has been attacked, is very small (and even causes underflow (u.f.), i.e. positive numbers less than 10^{-323} will be rounded to 0, for most data sets). For classifiers not attacked, except for MNIST, the order statistic p-value achieved by the $L_{\text{B-RED}} - C$ variant is clearly larger than the $L_{\text{B-RED}}$ variant — thus, false detections will be rare for $L_{\text{B-RED}} - C$ with a detection threshold of $\theta = 0.05$. For $L_{\text{B-RED}}$, there is still a vast range of thresholds that will detect attacks and reject unattacked classifiers, but a threshold smaller than $\theta = 0.05$ is necessary. Note also that reduced classifier accuracy (last row) does not inhibit reliable discrimination of attack from no-attack using a large range of p-value thresholds. As mentioned in Sec. 3.3.5, if there is high confusion between class s and t, the perturbation size/energy required to induce π-level misclassification from s to t may be small. The $L_{\text{B-RED}} - C$ variant corrects the size/energy of the optimized perturbation using the confusion matrix information. As seen in Table 3.5, this results in much larger p-values under clean and roughly the same (small) p-values under attack, for all four data sets. This result vindicates our confusion matrix correction scheme. Finally, note that even though the number of classes range from 10 up to 100 across the four data sets, for our method, there is a large, common range for θ over which strong detection performance is achieved.

3.4 Applying Basic RED to Internal Layers

3.4.1 Perturbation Estimation in DNN's Internal Layers

In the pattern estimation problem (3.4) of our basic RED, it is assumed that the backdoor pattern is an additive perturbation, which may limit its detection power when the actual backdoor pattern is of different type. However, we can easily extend our basic RED to perform perturbation estimation in the classifier's internal layer feature space. For brevity, we call this method "I-RED", where "I" stands for "internal layer". Consider decomposition of a DNN $f : \mathcal{X} \to \mathcal{Y}$ into two consecutive sub-models $f_1 \circ f_2$, such that $f_2 : \mathcal{X} \to \mathcal{Z}$ maps an image $\mathbf{x}$ to its internal layer features $\mathbf{z} = f_2(\mathbf{x}) \in \mathcal{Z}$, and $f_1 : \mathcal{Z} \to \mathcal{Y}$ maps $\mathbf{z}$ to the predicted class label. We propose the following generalized loss function for perturbation estimation:

$$L_{\text{I-RED}}(\mathbf{v}, (s,t)) = -\frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} p_2(t | f_1(\mathbf{x}) + \mathbf{v}), \tag{3.12}$$

where $p_2(\cdot|\mathbf{z})$ is the DNN's posterior given the internal layer feature vector $\mathbf{z}$. That is, for each class pair (s,t), we search the minimal-sized perturbation that induces high misclassification rate from s to t in the internal feature space $\mathcal{Z}$ instead of the image space $\mathcal{X}$. The detection statistic is then the norm/size (e.g. the l_2 norm) of the optimized internal layer perturbation.

3.4.2 Experiments

First, we apply I-RED to detect additive backdoor patterns. Using the same defense configuration as $L_{\mathrm{B-RED}}$ except for the perturbation estimation approach, with f_2 being the first convolutional layer, on the classifiers from the BD-P-S, BD-G-S, and BD-G-M group, we obtain detection rates 0.88, **1.00**, and **1.00**, respectively. For the clean group, we detect **no attacks** over all 25 classifiers.

Next, we consider a human imperceptible, multiplicative perturbation $\mathbf{v}^{\mathrm{m}}$ that can be applied to a clean image $\mathbf{x}$ by

$$m_{\mathrm{multi}}(\mathbf{x};\mathbf{v}^{\mathrm{m}}) = [\mathbf{x} \odot \mathbf{v}^{\mathrm{m}}]_c,$$

where $\mathbf{v}^{\mathrm{m}}$ has the same dimension as the image $\mathbf{x}$, with v^{m}_{ijk} the scaling factor of x_{ijk}. Here, we create a backdoor attack with a multiplicative chessboard pattern – we set $v^{\mathrm{m}}_{ijk} = 1$ if pixel x_{ijk} is not perturbed and set $u_{ijk} = 1.02$ if pixel x_{ijk} is perturbed. Other configurations of the attack are the same as the attacks on the classifiers in the BD-G-S group. In Fig. 3.10, we show the histogram of the reciprocal statistics obtained by applying I-RED on the attack above. We again choose f_2 as the first convolutional layer. The reciprocals corresponding to the true target class are clearly, abnormally large. I-RED successfully detects the attack (with order statistic p-value $2.0 \times 10^{-13} \ll \theta = 0.05$) with correct inference of the (source, target) class pair.

Finally, we evaluate I-RED on backdoor attacks with a blended pattern embedded by Eq. (2.3). In particular, we consider a 5×5 square image patch blended to the right corner of the image with the blending factor $\alpha = 0.3$. Following the same attack configurations as for creating the BD-P-S group, the ASR and ACC for this attack are 0.966 and 0.912, respectively. In Fig. 3.11, we show the histogram of the reciprocal statistics obtained for all class pairs. The reciprocal statistic corresponding to the true backdoor class pair is the right most one – now we successfully detect the attack with a correct inference of the backdoor class pair (with order statistic p-value less than 10^{-323}). Note that this attack cannot be effectively detected by the basic RED – the generalized

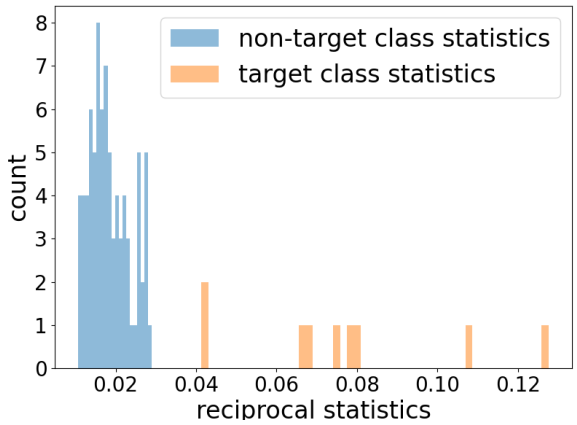


Figure 3.10: Histogram of the reciprocal statistics for I-RED applied on an attack with multiplicative chessboard pattern.

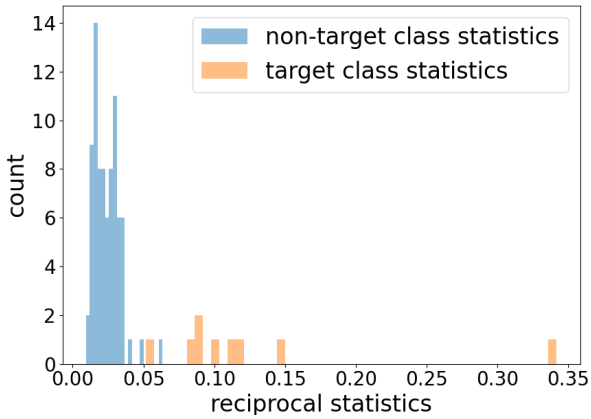


Figure 3.11: Histogram of the reciprocal statistics for I-RED applied on an attack with a local image patch blended using Eq. (2.3) with the blending factor $\alpha = 0.3$.

AD is necessary here.

3.5 MAMF-RED: Detecting Scene-Plausible Backdoor Patterns Using Maximum Achievable Misclassification Fraction Statistics

3.5.1 Scene-Plausibility of Patch Replacement Backdoor Patterns

As mentioned in Chap. 2, patch replacement backdoor patterns can be implemented physically by placing an object in the scene before capturing the scene digitally by, e.g., a camera; or, they can be implemented digitally using apps such as photoshop following the embedding function Eq. (2.2). However, scene-plausibility of the patch replacement pattern has been ignored by many other works. For example, the backdoor pattern considered in [109, 112] is a noisy patch or a noticeable icon *fixed in the same location* in every poisoned image (and backdoor test image). Regardless of the *incongruousness* of these patterns that could easily raise human suspicion in practice, there are several issues which should deter such spatially-fixed patches replacement patterns from being considered by a practical attacker:

1) Spatially-fixed image patches can be photo-shopped into digital images but usually cannot be placed in a scene that is digitally captured. It is also hard for these patches to be scene plausible. For example, if the backdoor pattern is a pair of glasses on a face, its location will be dependent on the location of the face. So, spatially-fixed backdoor patterns cannot in general be used in physical attacks.

2) If the same backdoor pattern is embedded at the same location in all images used to poison the training set, a naive data sanitization procedure applied "before/during training" (see Chap. 2 for details about this defense scenario) could detect the attack, by (e.g.) checking the pixel value distribution, and so possibly remove it.

3) Fixing the spatial location of the backdoor pattern while creating backdoor training images will likely harm the robustness of the attack significantly during testing. As will be seen in Sec. 3.5.6.6, for a path replacement pattern fixed to the bottom left of the backdoor training images, when there is no data augmentation (rotation, flipping, etc.) used during training, even a single column/row shift of the backdoor pattern at test time will severely degrade the attack effectiveness. This weakness of spatially-fixed patches is fatal. For physical attacks, there is no way to guarantee that the object is captured at the same location in test images as in the backdoor training images.

3.5.2 Key Ideas of MAMF-RED

MAMF-RED is designed to detect scene-plausible patch replacement backdoor patterns that are not spatially fixed in images (based on the reasoning above). It follows the standard assumptions for the post-training defense scenario described in Chap. 2, without assumptions about the shape, spatial support of the backdoor image patch, or object to be used. Together with the basic RED in Sec. 3.3 and I-RED in Sec. 3.4, most existing types of backdoor patterns can be detected if these three methods are applied in parallel. MAMF-RED is based on two properties of scene-plausible patch replacement patterns that we will experimentally show to hold in practice.

Property 3.5.1. Spatial invariance of backdoor mapping: If the patch replacement pattern is spatially distributed (not at a fixed location) in the backdoor poisoned training images (placed so as to be most scene-plausible in each image), the learned backdoor mapping will be spatially invariant in inducing targeted misclassifications on test images.

As addressed in Sec. 3.5.1, the image patch could be spatially located anywhere in a training image so as to be scene-plausible. The trained DNN will then learn the image patch, but not its spatial location. This property is actually favored by the attacker, since there will be more freedom to embed the backdoor pattern in clean test images. As will be seen in Sec. 3.5.6.1, if the backdoor pattern is spatially distributed in training images, at test time, backdoor images with the backdoor pattern randomly located are still likely to yield misclassifications to the target class prescribed by the attacker. This justifies the attacker performing data poisoning so as to satisfy this property. In Sec. 3.5.6.7, we will provide deeper insights regarding this property experimentally.

Property 3.5.2. Robustness of patch replacement patterns: The learned backdoor mapping is robust to variations such as noise, lighting, view, and illumination, thus obviating the need for an attacker to use precisely the same backdoor pattern in all training and test images.

Again, this property is favored by a practical attacker. For a physically implemented backdoor pattern, during testing, though the same object (e.g. the same pair of glasses) is used, it is not guaranteed that the same digital pattern will be captured (due to, e.g., the light condition, or the angle of the object). As will be seen in Sec. 3.5.6.4, even when strong noise is added to the backdoor pattern while creating backdoor test images, they will still likely be (mis)classified to the designated target class. More importantly, if only *part* (in terms of spatial support) of the image patch is added to source class images during

testing (the pattern could be partially occluded), this also induces (mis)classification to the target class. This is not surprising because patch replacement patterns, like other patterns, are learned by the DNN by extracting key features, which usually occupy a smaller spatial support than the full support of the image patch.

Based on the above two properties, the key ideas of MAMF-RED are as follows: 1) For a scene-plausible patch replacement pattern on a DNN, for effective backdoor (source, target) class pairs (including those used for devising the attack and those caused by collateral damage, see Sec. 3.3.6.3 for definition) finding a pattern (i.e. an image patch) that induces a high misclassification fraction (from the source class to the target class) on the clean dataset can be achieved using a relatively small spatial support that is *arbitrarily located* in the image. 2) For non-backdoor class pairs, much larger spatial support is required to find a pattern that induces high misclassifications from the source to the target class.

The key ideas of MAMF-RED are well supported by the two properties above. For convenience, for any class pair $(s,t) \in \mathcal{Y} \times \mathcal{Y}$ $(s \neq t)$, and a set $\mathcal{D}_s$ of clean images that are correctly classified to s by the classifier f to be inspected, we define:

$$\xi_{st}(\mathbf{u}, \mathbf{m}) = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \mathbb{1}(f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) = t) \tag{3.13}$$

as the **misclassification fraction** from s to t induced by pattern $\mathbf{u}$ and mask $\mathbf{m}$.

When there is a backdoor attack with scene-plausible patch replacement patterns, the image patch $\mathbf{u}^*$ will be spatially distributed in the backdoor poisoned training images. Consider a backdoor class pair (s, t^*) with $s \in \mathcal{S}^*$ (here, we consider general backdoor attacks with possibly multiple source classes) or a class pair suffering collateral damage. Based on Property 3.5.1, for any mask $\mathbf{m}'$ having the same shape and size as the true mask used for devising the attack in terms of the region of 1's, $\xi_{st^*}(\mathbf{u}^*, \mathbf{m}')$ is expected to be large (close to 1) no matter where the center of the region of 1's of $\mathbf{m}'$ is located in the images from $\mathcal{D}_s$. Based on Property 3.5.2, there exists a pattern $\mathbf{u}'$ (with associated mask $\mathbf{m}''$) different from the true pattern $\mathbf{u}^*$, such that $\xi_{st^*}(\mathbf{u}', \mathbf{m}'')$ is large, so long as sufficiently "representative" features (e.g. key textures, colors, etc.) of $\mathbf{u}^*$ are captured in its spatial support. Such a pattern $\mathbf{u}'$ may have either larger or *smaller* support than $\mathbf{u}^*$.

Thus, for any arbitrarily located spatial support with a relatively small support size (specified by a given mask $\mathbf{M} \in \{0,1\}^{W \times H}$ with $||\mathbf{M}||_0$ small and the center of the region of 1's arbitrarily located and fixed for all images), we expect the optimal misclassification

fraction obtained by solving:

$$\underset{\mathbf{u}\in[0,1]^{W\times H\times C}}{\text{maximize}} \quad \xi_{st}(\mathbf{u},\mathbf{M}) \tag{3.14}$$

for (s,t) a backdoor class pair to be much larger than for (s,t) a non-backdoor class pair. Motivated by this, our detector consists of a pattern estimation step (performed for each class pair) and a detection inference step.

However, choosing an optimal $\mathbf{M}$ for pattern estimation to best distinguish backdoor class pairs from non-backdoor class pairs is not trivial. If $||\mathbf{M}||_0$ is too small, pattern estimation may be inadequate to capture the "representative" features of the backdoor pattern for backdoor class pairs. Then, the resulting misclassification fraction for all backdoor class pairs will be too low to trigger a detection. If $||\mathbf{M}||_0$ is overly large, the "representative" features of a *non-backdoor* target class may be captured by chance, such that a high group misclassification fraction can be achieved for some class pairs with this non-backdoor target class, leading to a false detection. This will be shown experimentally in Sec. 3.5.6.2. Thus, since the support size of an attack is unknown to the defender, we suggest to choose *multiple* spatial supports, with a range of support sizes growing from small to large. For each class pair, pattern estimation is performed on each of these spatial supports. Then the pattern estimation results for these supports are suitably *aggregated*, for each class pair, for detection inference. The details of our detection are described in the sequel.

3.5.3 Pattern Estimation of MAMF-RED

Our pattern estimation step thus solves problem (3.14) for a sequence of spatial supports increasing in support size.

3.5.3.1 Design Choices

Support Location: Since it does not matter where the estimated pattern's spatial support is placed, in our experiments (based on Property 3.5.1), we fixed a location for all clean images used in our detection system. However, it is even possible that one could *vary* the center position for each clean image – accurate backdoor detection should still be possible.

Support Shape: The choice of the support shape is not critical to the detection performance, but we prefer to use a simple convex support (e.g. a rectangle or a circle) to efficiently capture the "representative" features of the target class. Here, we choose square

as the support shape for simplicity. Note that the choice of the support shape in detection does not need to be matched with the shape of the region of 1's for the mask used by the attacker (which is not known to the defender *a priori*). In our experiments, we consider attacks with patch replacement backdoor patterns having contiguous non-convex shapes, or even highly distributed supports (see Sec. 3.5.6.2) – these attacks are all detected by our detector using simple square spatial support for pattern estimation.

Objective Function and Optimizer: Note that the misclassification fraction $\xi_{st}(\mathbf{u}, \mathbf{m})$ for class pair (s, t) on $\mathcal{D}_s$ is not differentiable in $\mathbf{u}$. We propose the following surrogate objective function:

$$\begin{aligned}\tilde{\xi}_{st}(\mathbf{u}, \mathbf{m}) &= \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \mathbb{E}[\mathbb{1}(f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) = t)] \\ &= \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} p(t | m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})),\end{aligned} \tag{3.15}$$

which is both the "average misclassification confidence" from s to t on $\mathcal{D}_s$ and the average DNN class posterior for class t, with the average over all images in $\mathcal{D}_s$. Thus, maximizing $\tilde{\xi}_{st}(\mathbf{u}, \mathbf{m})$ over $\mathbf{u} \in [0, 1]^{W \times H \times C}$ can be achieved using a gradient-based optimizer with projection (i.e. clipping to the valid range for pixel intensity, $[0, 1]$). For example, one can use stochastic gradient descent (SGD) with momentum and adaptive learning rate, where in each step of updating $\mathbf{u}$, the gradient of (the negative of) (3.15) is computed on a subset of (i.e. a mini batch from) $\mathcal{D}_s$. Note that the choices of the objective function and the optimizer are not unique in order to achieve good detection performance. One can also consider a surrogate objective function similar to that associated with the Perceptron algorithm [58]:

$$\tilde{\xi}_{\text{PA},st}(\mathbf{u}, \mathbf{m}) = \frac{1}{|\hat{\mathcal{D}}_s(\mathbf{u}, \mathbf{m}, t)|} \sum_{\mathbf{x} \in \hat{\mathcal{D}}_s(\mathbf{u}, \mathbf{m}, t)} p(t | m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})), \tag{3.16}$$

where $\hat{\mathcal{D}}_s(\mathbf{u}, \mathbf{m}, t) = \{\mathbf{x} \in \mathcal{D}_s : f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) \neq t\}$. Or, one can take a logarithm of the classifier's posterior in (3.15) for better smoothness [109]:

$$\tilde{\xi}_{\text{L},st}(\mathbf{u}, \mathbf{m}) = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \log p(t | m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})). \tag{3.17}$$

These choices do not have significant effect on the accuracy of our detector.

3.5.3.2 Detailed Pattern Estimation Steps

Based on the above discussion of the design choices, we first create L different masks. Each mask specifies a square spatial support (region of 1's) with fixed location, e.g. the top left corner of each image. The number L and the width of the square spatial support associated with each mask are specified as follows. Assuming the image width W is no larger than the image height H (otherwise, rotate the image), we choose a minimum *relative* support width $r_{\min} \in [0, 1]$ and a maximum *relative* support width $r_{\max} \in [0, 1]$ [7]. Then $\mathcal{W} = \mathbb{Z}^+ \cap [r_{\min} W, r_{\max} W]$ is the set of integer support widths to be considered with $L = |\mathcal{W}|$. For example, for 32×32 colored images (i.e. $W = H = 32$), with $r_{\min} = 0.15$ and $r_{\max} = 0.2$, we consider $L = 2$ masks with support width 5 and 6, respectively. As we have discussed, the purpose in considering a range of support widths instead of a single one for pattern estimation is for more reliable detection because the true support of the backdoor pattern is unknown. This need will be borne out experimentally in Sec. 3.5.6.2. In principle, a higher misclassification fraction can be achieved for backdoor class pairs than for non-backdoor class pairs when pattern estimation is performed on even very small spatial supports. Thus, we still require $r_{\max}$ to be not too large. The choices of $r_{\min}$ and $r_{\max}$ in our experiments will be further discussed in Sec. 3.5.6.3.

For each of the $K(K-1)$ (source, target) class pairs, we perform L pattern estimations. That is, for each $(s, t) \in \mathcal{Y} \times \mathcal{Y}$ $(s \neq t)$ and $w \in \mathcal{W}$, we solve (using, e.g., the previously suggested optimizer):

$$\underset{\mathbf{u} \in [0,1]^{W \times H \times C}}{\text{maximize}} \quad \tilde{\xi}_{st}(\mathbf{u}, \mathbf{M}_w), \tag{3.18}$$

where $\mathbf{M}_w \in \{0, 1\}^{W \times H}$ is the mask with *fixed* $w \times w$ square spatial support specified by locations with 1's. Compared with NC method proposed in [109], which *jointly* estimates a pattern and its support mask, our pattern estimation problem (with the mask fixed) is less formidable.

3.5.4 Detection Inference of MAMF-RED

For each class pair (s, t) and support width $w \in \mathcal{W}$, we define an associated **maximum achievable misclassification fraction (MAMF)** statistic as:

$$\rho_{stw} = \xi_{st}(\mathbf{u}_{stw}, \mathbf{M}_w), \tag{3.19}$$

[7] Here, we reused the notations $r_{\min}$ and $r_{\max}$ which were used for detection statistics in Sec. 3.3.3

where $\mathbf{u}_{stw}$ is the optimal pattern obtained from solving (3.18) for class pair (s,t) and support width w. There are many possible ways to aggregate the pattern estimation results over the L spatial supports for each class pair (e.g. median, geometric average, maximum, arithmetic average) to obtain a detection statistic. Here, we compute an *average* MAMF $\bar{\rho}_{st} = \frac{1}{L}\sum_{w\in\mathcal{W}} \rho_{stw}$ over the L MAMF statistics for each class pair for simplicity and leave other possible aggregation techniques to future work. For a relatively small $r_{\max}$ (and the L associated small spatial supports), if there was an attack, we would expect *at least* one true backdoor pair to have a large average MAMF (whether one or multiple source classes are involved in the attack); otherwise, the maximum $\bar{\rho}_{st}$ for all (s,t) pairs is expected to be small. Hence, we infer the DNN to be attacked if

$$\rho^* = \max_{(s,t)\in\mathcal{Y}\times\mathcal{Y}, s\neq t} \bar{\rho}_{st} > \theta; \tag{3.20}$$

else, the DNN is not attacked. In Sec. 3.5.6.2, we confirm experimentally that our approach is effective for a sizable range of the threshold θ. If an attack is detected, $(\hat{s},\hat{t}) = \arg\max_{(s,t)\in\mathcal{Y}\times\mathcal{Y}, s\neq t} \bar{\rho}_{st}$ is inferred as one (source, target) class pair involved in the backdoor attack.

3.5.5 MAMF Correction Using Class Confusion Information

Pattern estimation for a non-backdoor (source, target) class pair may produce a large MAMF statistic on a small spatial support if the two classes are similar to each other, i.e. have high class confusion. To avoid false detections caused by this, one may build knowledge of confusion matrix information into the detector. For each $(s,t) \in \mathcal{Y}\times\mathcal{Y}$ $(s \neq t)$, we define a baseline class confusion between s and t as:

$$\rho_{st0} = P[\mathbb{1}(f(\mathbf{X}) = t) | y(\mathbf{X}) = s], \tag{3.21}$$

where $P[\cdot|\cdot]$ denotes conditional probability; $\mathbf{X} \in \mathcal{X}$ denotes a random image; $y : \mathcal{X} \to \mathcal{Y}$ denotes oracle labeling of an image. The class confusion information may be provided to the defender in advance, or empirically estimated using a relatively abundant set of clean images[8]. Then the average MAMF $\bar{\rho}_{st}$ for (s,t) can be obtained by averaging over

[8]Pattern estimation for each class pair is still performed using only a subset of clean images that are correctly classified to the source class.

L "corrected" MAMF statistics, each defined by:

$$\rho_{stw}^{(c)} = \max\{0, \rho_{stw} - \rho_{st0}\}, \tag{3.22}$$

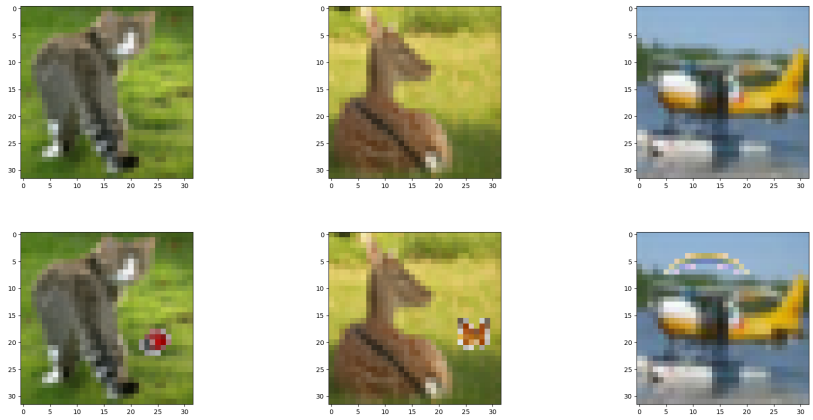
for each $w \in \mathcal{W}$. When there is no attack, the above correction will prevent class pairs with high class confusion from having overly high MAMF statistics, as will be seen experimentally in Sec. 3.5.6.2.

3.5.6 Experiments

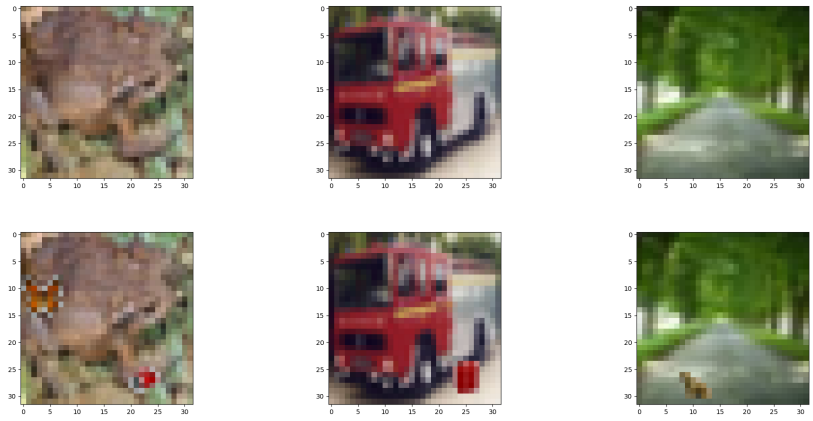
3.5.6.1 Devising Backdoor Attacks

In this section, we demonstrate the validity of the properties in Sec. 3.5.2 and the performance of MAMF-RED using nine attacks (named Attack A-I), involving five datasets, five DNN structures, and nine different backdoor patterns. Similar to our experiments in Sec. 3.3.6, for each attack, we first trained a DNN using an unpoisoned training set and report its clean test accuracy (ACC), which is dubbed "benchmark accuracy" here. The dataset being used, image size, number of classes, training size and test size are shown in Tab. 3.6. In particular, for Attack G on Oxford-IIIT, the test accuracy using the entire dataset, in absence of backdoor attackss, is very low. We achieve a reasonable benchmark test accuracy on G by using a subset involving only 6 classes. For Attack I on PubFig, many images are not able to be downloaded; hence we only consider 33 (out of 60) classes with more than 60 images. The DNN architectures involved in our experiments include ResNet-18, ResNet-34 [14], VGG-16 [16], AlexNet [17] and ConvNet [175]. The choices of DNN architecture, learning rate, batch size and number of training epochs for each attack instance are also shown in Tab. 3.6. The Adam optimizer (with decay rate 0.9 and 0.999 for the first and second moment, respectively) is used for all DNN training in this paper [5]. For the benchmark training for the datasets involving Attacks F, G and I, we also use training data augmentation including random cropping, random horizontal flipping and random rotation. For the benchmark training for the dataset involving Attack G, we fine-tune a pretrained AlexNet provided by Pytorch. For the benchmark training for the dataset involving Attack I, we adopt transfer learning by retraining the last four layers of a pretrained VGG-face model [176].

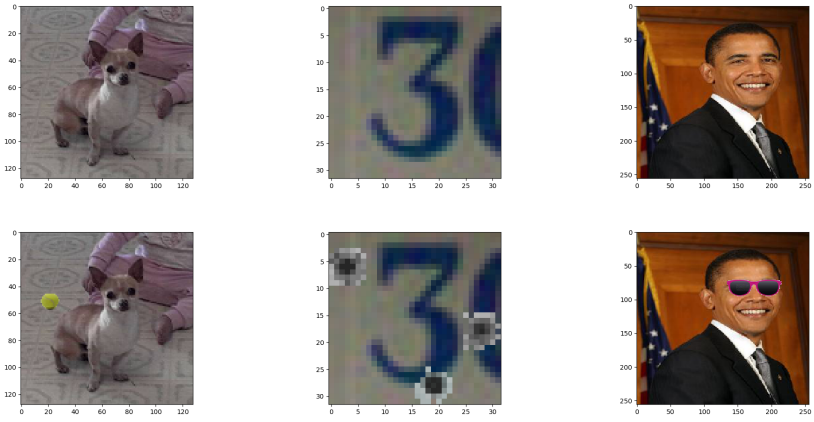
Under each attack, we train the DNN using the same training settings as for the benchmark, except that the training set is poisoned by a number of backdoor images. The backdoor images are created using clean images from *one* source class, with a patch



(a) bug (b) butterfly (c) rainbow



(d) bug+butterfly (e) gas tank (f) marmot



(g) tennis ball (h) bullet holes (i) sunglasses

Figure 3.12: Example backdoor image and the originally clean image for Attack A–I. Subcaptions describe the object(s) added as the patch replacement backdoor pattern for each attack.

Table 3.6: Details of the attacks.

	Attack A	Attack B	Attack C	Attack D	Attack E	Attack F	Attack G	Attack H	Attack I
Dataset	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-100	Oxford-IIIT	SVHN	PubFig
Image size	32×32	32×32	32×32	32×32	32×32	32×32	128×128	32×32	256×256
No. classes	10	10	10	10	10	100	6	10	33
Training size	50000	50000	50000	50000	50000	50000	900	73257	2782
Test size	10000	10000	10000	10000	10000	10000	300	26032	495
DNN structure	ResNet-18	ResNet-18	ResNet-18	ResNet-18	VGG-16	ResNet-34	AlexNet	ConvNet	VGG-16
Learning rate	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}
Batch size	32	32	32	32	32	32	16	32	32
No. training epochs	200	200	200	200	200	200	120	80	120
Benchmark acc. (%)	86.7	88.1	86.7	87.6	87.9	71.9	88.7	89.2	76.0
Source class	"cat"	"deer"	"airplane"	"frog"	"truck"	"road"	"chihuahua"	"3"	"B. Obama"
Target class	"dog"	"horse"	"bird"	"bird"	"automobile"	"bed"	"Abyssinian"	"8"	"C. Ronaldo"
Backdoor pattern	"bug"	"butterfly"	"rainbow"	"bug&butterfly"	"gas tank"	"marmot"	"tennis ball"	"bullet holes"	"sunglasses"
No. backdoor training images	150	150	150	150	150	100	50	500	40
Attack test acc. (%)	87.0	86.9	86.8	87.0	89.1	71.7	90.0	90.1	77.0
Attack succ. rate (%)	99.3	98.0	96.4	98.0	97.9	92.0	84.0	91.4	93.3

replacement pattern embedded following Eq. (2.2) (but with an image-specific mask for scene-plausibility), and then labeling to a target class. In the experiments in this section, we chose to evaluate our detector against backdoor attacks involving one source class for convenience in easily crafting scene-plausible backdoor patterns. As we have discussed in Sec. 3.5.4, the design of our detector allows it to detect backdoor attacks with any number of source classes, since we only need one class pair to have a sufficiently large average MAMF to make a detection. In Sec. 3.5.6.8, we will show the effectiveness of MAMF-RED against backdoor attacks involving $(K-1)$ source classes, even though the backdoor patterns may not be scene-plausible (e.g. a rainbow will be placed in an image without the sky). For the experiments in the current section, the choices of the backdoor pattern, the number of backdoor training images, the source class and the target class for each attack instance are shown in Table 3.6.

To create backdoor attacks with scene-plausible patch replacement patterns, we first created a large number of candidate backdoor images, each with the image patch randomly located in the image. Then we spent laborious human effort to manually pick the images in which the backdoor pattern looks scene-plausible[9]. For example, for Attack C (source class "airplane" and backdoor pattern a "rainbow"), a valid backdoor poisoned training image should have the rainbow in the sky (see Figure 3.12c). In Figure 3.12, we show an example backdoor training image and its original clean image for each attack. The backdoor patterns considered here are representative of multiple types of

[9]Only for Attack I we carefully place the sunglasses on the face.

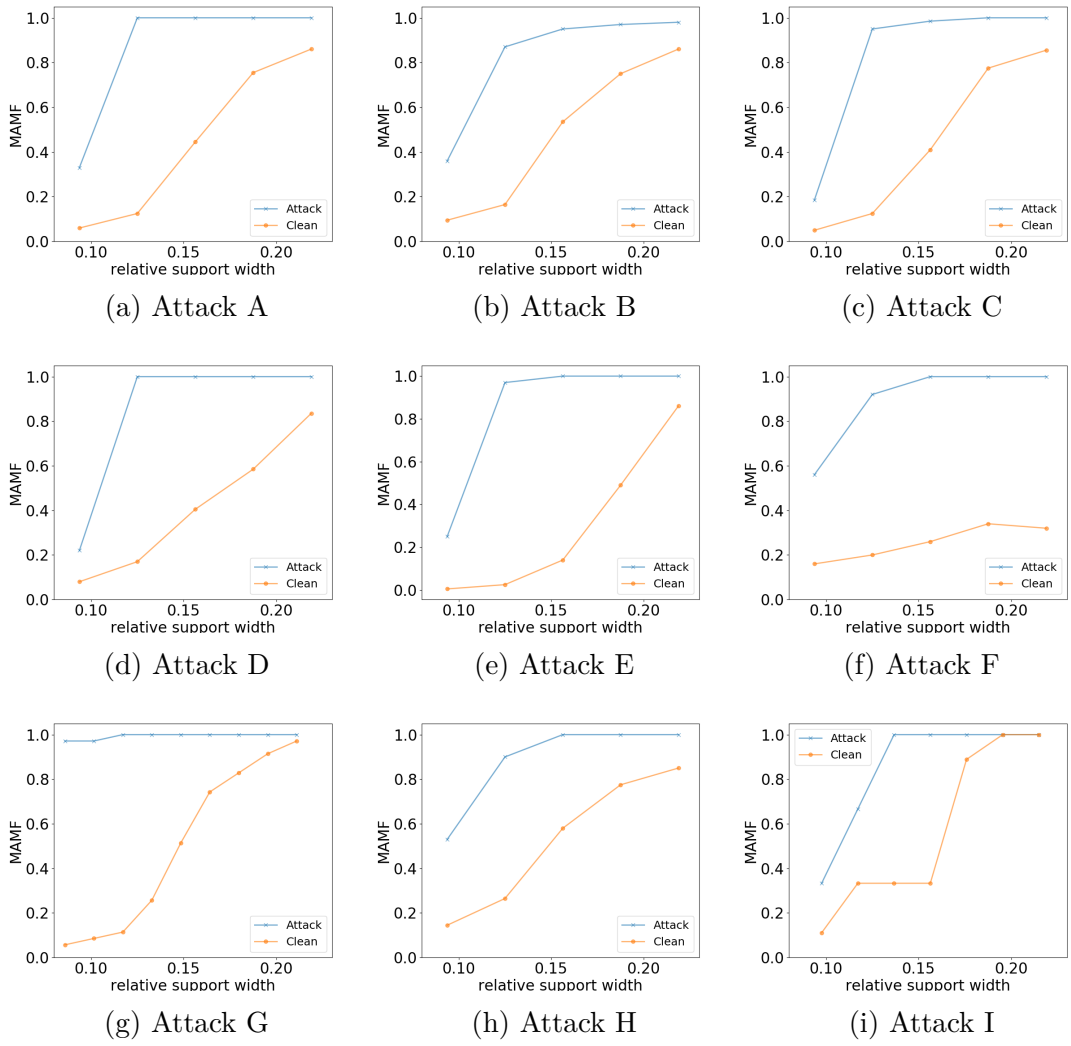
practical backdoor patterns. The "bug" for Attack A and the "butterfly" for attack B represent backdoor patterns in the periphery (not covering the foreground object of interest) of an image and with a modest size. The "rainbow" for Attack C represents large backdoor patterns with an irregular shape. The backdoor patterns for Attack D and H represent dispersed patterns. The "sunglasses" for Attack I represent backdoor patterns overlapping with features of interest.

In Table 3.6, we also report the ASR and ACC for all attacks. For ASR, the backdoor test image is created using a clean test image from the source class(es), and embedding in it the same backdoor pattern used for creating the backdoor-poisoned training images. We took an automated approach (due to the huge number of test images) to create backdoor test images, randomly placing the backdoor pattern in the image (except for Attack I where we manually place the sunglasses on the faces). By doing so for a sufficiently large test set, we should be "covering" most locations where the attacker could place the backdoor pattern in practice. In Table 3.6, the ASR is high for all attacks, and there is no significant degradation in ACC compared with the benchmark; hence all attacks are considered successful. *Moreover*, we emphasize such success is achieved with randomly placed backdoor patterns in test images, which experimentally verifies Property 3.5.1, i.e. the spatial invariance of the learned scene-plausible backdoor mapping. More experiments regarding Property 3.5.1 will be shown in Sec. 3.5.6.7.

3.5.6.2 Defense Performance Evaluation

In this section, we evaluate the performance of the MAMF-RED in comparison with the NC [109] and FP [160], using the above-mentioned attacks, *i.e.*, backdoor attacks with scene-plausible patch replacement patterns where the image patch is not in a fixed position, either in the poisoned training images or in the backdoor-attacked test images.

For each attack, MAMF-RED is applied to both the DNN being attacked and the clean benchmark DNN. For best discrimination between the two categories of DNNs, relatively small spatial supports should be used for pattern estimation, so that a DNN being attacked can have a much larger ρ^* than a clean DNN. On the one hand, the spatial support should be smaller than that of the foreground objects associated with the actual classes; otherwise, the actual target class object(s) could possibly be reverse engineered, causing high group misclassification even for a clean DNN. On the other hand, the spatial support should be large enough for estimating any pattern related to the backdoor pattern. Here, we show that with a common $r_{\min}$ and a common $r_{\max}$ applied to all the classifiers to be inspected regardless of the associated dataset, there is



(a) Attack A (b) Attack B (c) Attack C

(d) Attack D (e) Attack E (f) Attack F

(g) Attack G (h) Attack H (i) Attack I

Figure 3.13: Maximum achievable misclassification fraction (MAMF) statistics for the class pair with the largest average MAMF, for both attacked DNN and unattacked (clean DNN), for Attack A–I. Relative support range for pattern estimation is $(0.08, 0.22)$.

a large range of valid thresholds for achieving perfect detection of all the attacks, without any false detections. We choose $r_{\max} = 0.22$ and $r_{\min} = 0.08$ for all attack instances, such that for a 32×32 image, there is at least 3×3 spatial support for pattern estimation. In Sec. 3.5.6.3, we will present an automatic approach for choosing adaptive $r_{\min}$ and $r_{\max}$ for each classifier to be inspected.

The details of our detection settings are as follows. For Attack A-E and Attack H, 200 clean images correctly classified *per class* are used for detection. For Attack F, G and I, 50, 30 and 9 correctly classified clean images per class are used for detection, respectively. For all DNNs, class pairs and spatial supports, we solve (3.14) using Adam optimizer

with decay rate 0.9 and 0.999 for the first and second moment, respectively, and learning rate 0.5 for 100 epochs[10]. We use mini-batch size 32 for Attack A-E and H, 10 for Attack F and G and 3 for Attack I. The spatial support for pattern estimation, used in our detection, is a square covering the top left corner of each image. Due to Property 3.5.1 (as will be further investigated in Sec. 3.5.6.7), other locations yield similar results, as will be illustrated in Sec. 3.5.6.9.

In Figure 3.13, for both the DNN being attacked and the clean benchmark DNN, for each attack, we show the (L) MAMF statistics for the class pair with the largest average MAMF (i.e. the class pair corresponding to ρ^*). For example, for Attack A where the data image size is 32×32, the set of (absolute) support widths being considered are $\mathcal{W} = \{3, 4, 5, 6, 7\}$. For large image size, instead of performing pattern estimation for each integer support width in the interval $[r_{\min}W, r_{\max}W]$, we could efficiently downsample and perform pattern estimation for fewer support widths. Here, $L = 9$ and $L = 7$ support widths are considered for Attack G (Figure 3.13g) and Attack I (Figure 3.13i) respectively. In each sub-figure of Figure 3.13, a large gap is observed between the clean and attacked curves, clearly distinguishing the DNN being attacked from the clean DNN.

Also from Figure 3.13, we see the importance of aggregating the pattern estimation results (i.e. the MAMF statistics) over a range of support sizes. Unless there is a way to select a single support width that ensures a significant difference between the largest MAMF statistic for the classifier being attacked and for the clean classifier, these two categories of classifiers may not be distinguishable for some casually selected support widths. For example, for the absolute support width 55 (relative support width 55/128) for Attack I, the largest MAMF statistics for the classifier being attacked and for the clean classifier are the same (see Figure 3.13i).

In Figure 3.14, we show the largest average MAMF (i.e. ρ^*) for both the clean and attacked DNNs, for all attacks. Note the clear large difference between the two bars for each attack. Thus any detection threshold θ in $(0.6, 0.8)$, if used, could successfully detect whether the DNN is attacked. Moreover, for the DNN being attacked, for all attack instances, the class pair corresponding to the maximum average MAMF is precisely the *true backdoor pair*. Hence when a detection is made, the (source, target) class pair (used by the attacker) is also correctly inferred, in all cases.

Here, we also consider the extension of our detection approach where MAMF statistics are corrected using class confusion information (see Sec. 3.5.5). For each of Attack B, Attack F, and Attack H, we estimate an empirical class confusion matrix using all the test

[10]Fewer epochs are actually needed for decent convergence in all our experiments.

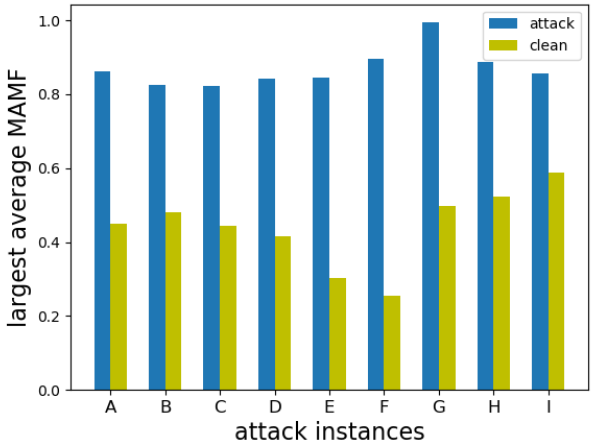


Figure 3.14: Largest average maximum achievable misclassification fraction (MAMF), i.e. ρ^*, over all class pairs for both the attacked DNN and unattacked (clean) DNN, for Attack A-I.

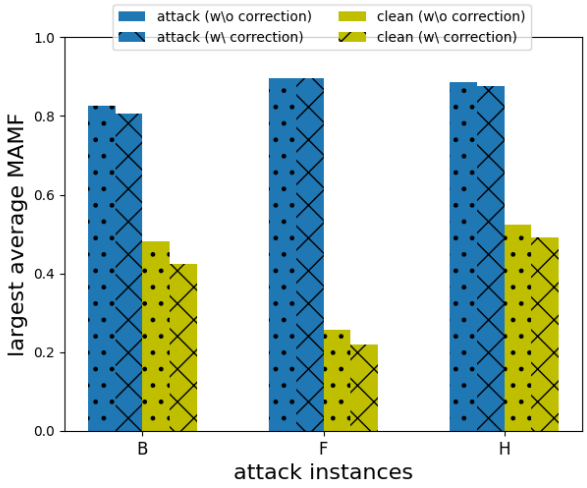


Figure 3.15: Largest average maximum achievable misclassification fraction (MAMF), i.e. ρ^*, with MAMF correction, compared with ρ^* without MAMF correction, for both the attacked DNN and unattacked (clean) DNN, for Attack B, Attack F, and Attack H, respectively.

images for the dataset associated with the attack (see Tab. 3.6 for dataset information). For each attack, we apply MAMF-RED with correction to both the clean benchmark DNN and the DNN being attacked. In Figure 3.15, we show the largest average MAMF ρ^* with correction, compared with ρ^* without correction, for both clean and attacked DNNs, for the three attacks. Although MAMF correction does not significantly affect

Table 3.7: Detailed settings of NC, including the number of clean images per class used for detection, the choice of λ, the learning rate and the mini-batch size, for each attack instance.

	No. images per class	λ	Learning rate	Batch size
Attack A–D	100	0.1	0.05	90
Attack E	100	0.6	0.05	90
Attack F	10	0.1	0.05	90
Attack G	30	0.5	0.001	60
Attack H	100	0.2	0.01	100
Attack I	9	0.1	0.005	30

ρ^* for the six classifiers, the decrement in ρ^* for the clean DNN is slightly larger than for the DNN being attacked, for all three attacks. In other words, assisted by the class confusion information, the range of valid thresholds is slightly larger. Note that in this experiment, we used abundant images to ensure the accuracy of our confusion matrix estimation, though a practical defender may not have access to sufficient images to accurately estimate the confusion matrix. Nevertheless, we have already shown accurate detection can be achieved by MAMF-RED without use of the class confusion matrix. Thus, in all the experiments in the following, we do not use MAMF correction for simplicity.

For the attacks in the current experiment (that use a single source class), NC may not make a correct detection, since NC is based on the assumption that *all classes* other than the target class are involved in the attack. Here we evaluate NC using the same attacks. We used the Adam optimizer to solve NC's optimization problem (see Eq. (3.1) in Sec. 3.2), with the parameters suggested by the authors of NC, and performed mini-batch optimization for a sufficient number of epochs (until convergence). In particular, we carefully adjust the Lagrange multiplier λ in Eq. (3.1), and the training parameters for each attack to achieve a mask with small L_1 norm and a pattern inducing a high group misclassification fraction to the true backdoor target class. In this way, we in fact optimistically tune NC's hyperparameter λ to maximize its accuracy. For all attack cases, we set NC's misclassification fraction $\pi = 0.9$, and the optimization is performed for 200 epochs. Table 3.7 shows the number of clean images per class used for pattern estimation by NC, the choice of λ, the learning rate and the mini-batch size for each attack.

As mentioned in Sec. 3.2, NC's detection inference uses MAD [167] of the l_1 norm of the mask for all putative target classes [109]. If the anomaly index is larger than 2.0, a detection is made with 95% confidence. In Figure 3.16, we show anomaly indices for

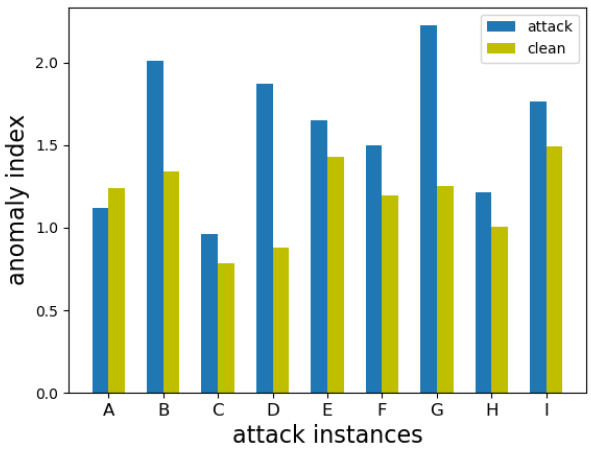


Figure 3.16: Anomaly indices (the detection statistic of NC) when applying NC to the DNN being attacked and the clean DNN of Attack A–I.

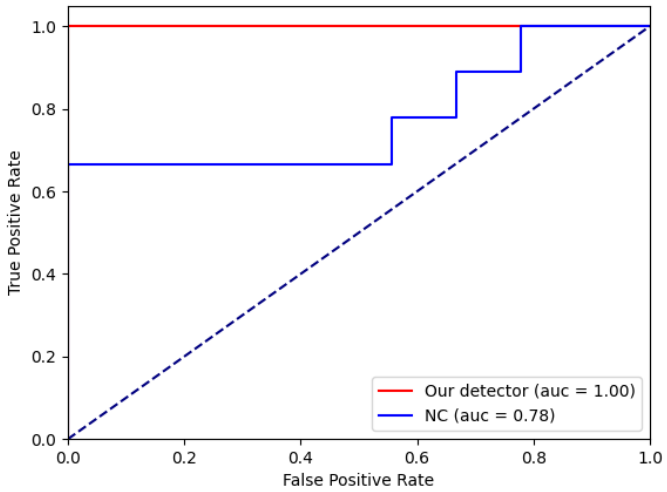
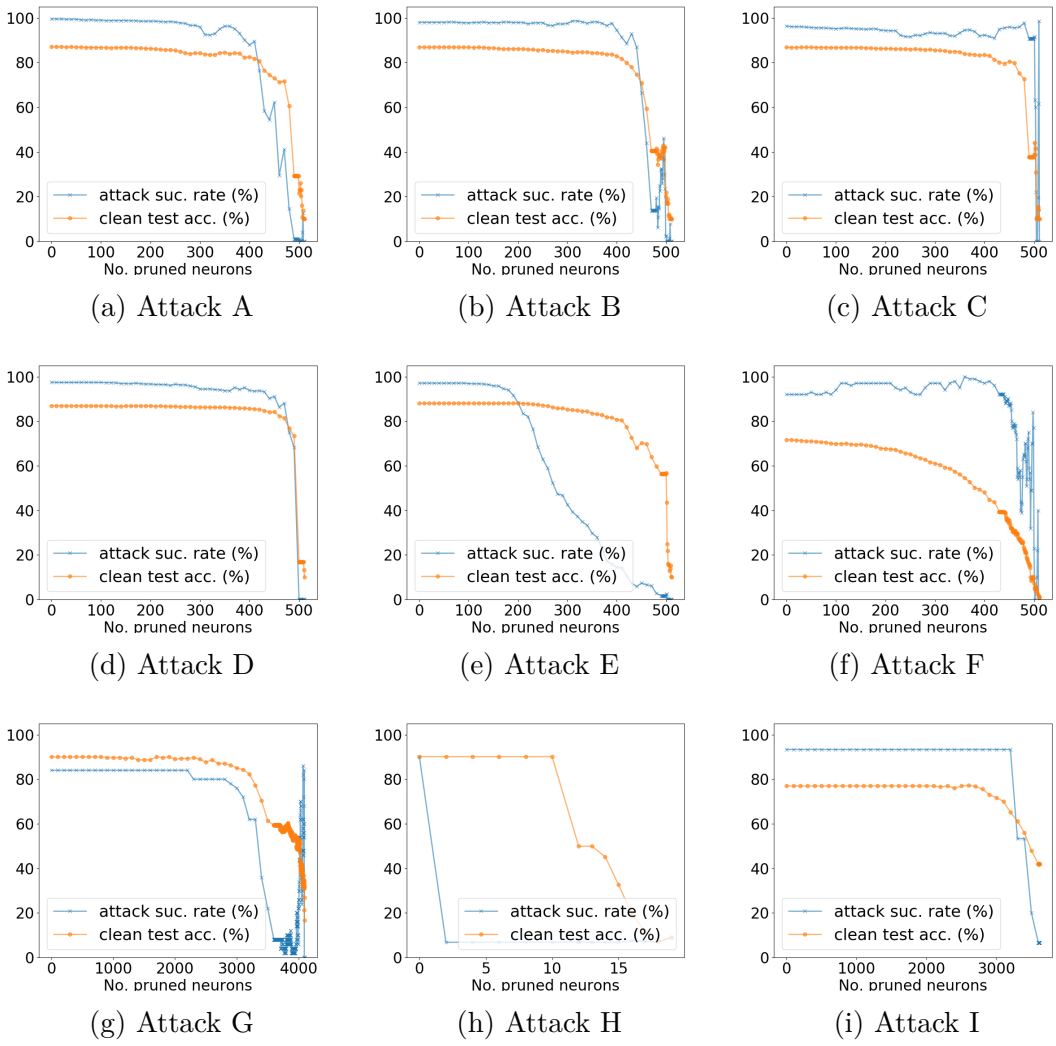


Figure 3.17: Receiver operating characteristic (ROC) curves for NC and our MAMF-RED against the nine attacks.

both the DNN being attacked and the clean DNN for each attack. Only for Attack B and Attack G, NC successfully detects the attack – for these attacks, there is a pattern and an associated *relatively small* mask which, when applied to clean test images from all classes other than the target class, induces a high group misclassification (even though the backdoor targeted only a single source class). The phenomenon may due to the "collateral damage" introduced in Sec. 3.3.6.3.

For better visualization of the performance comparison between MAMF-RED (without



(a) Attack A (b) Attack B (c) Attack C

(d) Attack D (e) Attack E (f) Attack F

(g) Attack G (h) Attack H (i) Attack I

Figure 3.18: Attack success rate and accuracy on clean test images as the number of penultimate layer neurons being pruned is increased, for each DNN being attacked.

correction) and NC, we show the receiver operating characteristic (ROC) curves for the two approaches against the nine attacks in Fig, 3.17. The area under curve (AUC) for MAMF-RED and NC are 1.0 and 0.78, respectively.

We also show that FP [160] is ineffective against most of the attacks. We prune the penultimate layer neurons of each classifier being attacked (until only few are left), in increasing order of their average activations over all clean test images. In Fig. 3.18, for each attack, we show the accuracy on clean test images and the attack success rate versus the number of neurons pruned. For most of the attacks (except Attack E and H), the attack success rate does not drop before the accuracy on clean test images is degraded,

Table 3.8: Adaptive minimum and maximum support width determined for both the classifier being attacked and the clean benchmark classifier for Attack A-I, excluding Attack F.

	attacked		clean	
	min width	max width	min width	max width
Attack A	4	6	6	6
Attack B	4	6	5	7
Attack C	4	5	6	7
Attack D	4	5	6	6
Attack E	4	5	6	7
Attack G	11	13	19	25
Attack H	3	5	5	5
Attack I	30	45	35	40

as the number of pruned neurons grows. Thus, FP is generally unsuccessful against these backdoor attacks.

3.5.6.3 Adaptive Selection of $r_{\min}$ and $r_{\max}$

We show a practical way to choose an adaptive $r_{\min}$ and $r_{\max}$ combination for any classifier to be inspected (without knowing whether the classifier is attacked or not *a priori*). This approach is actually matched with the intuition of choosing $r_{\min}$ and $r_{\max}$ discussed previously. We start with pattern estimation on a 1×1 square spatial support for each class pair. While increasing the absolute width of the square spatial support for pattern estimation, at the first instance when any class pair achieves a moderate MAMF (e.g. 0.5), we set $r_{\min}$ to be this absolute support width divided by the image width. Intuitively, at this point, the corresponding spatial support is sufficiently large to reverse engineer partially "representative" features of the true backdoor pattern. Then we continue to increase the support size, until there are at least two different target classes involved among all the class pairs whose MAMF statistics are larger than the MAMF threshold. We then set $r_{\max}$ to be the corresponding absolute support width divided by the image width. Also, since a backdoor attack is assumed to involve only one target class, we know that there is at least one non-backdoor class pair having a moderate MAMF statistic – the support size should not be further increased.

For brevity, we do not thoroughly evaluate the above approach. Instead, we only consider the support widths that have been considered in our previous experiments (see Fig. 3.13 and the related description). Here, we set the MAMF threshold for determining the adaptive minimum and maximum support sizes to be 0.5. In Tab. 3.8, we show the

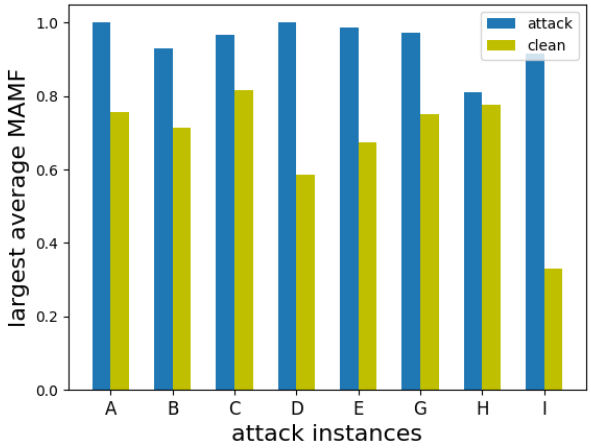


Figure 3.19: Largest average maximum achievable misclassification fraction (MAMF), i.e. ρ^*, over all class pairs for both the attacked DNN and unattacked (clean) DNN, for Attack A-E and Attack G-H, when the adaptive selection approach for $r_{\min}$ and $r_{\max}$ is used.

minimum and maximum *absolute* support width determined for both the classifier being attacked and the clean benchmark classifier for each of Attack A-I (except Attack F[11]). One can relate the results in Table 3.8 with the results in Fig. 3.13 to better understand the intuition behind our adaptive selection approach for $r_{\min}$ and $r_{\max}$. Take Attack B as an example. For the classifier being attacked, for the absolute support width 4 (relative support width 4/32), although the MAMF threshold 0.5 is achieved by some backdoor class pair, the MAMF statistics for all the non-backdoor class pairs are low. From Table 3.8, as the absolute support size grows to 6 (relative support width 6/32), for the first time, the MAMF statistic for a non-backdoor class pair exceeds the threshold 0.5. As the absolute support width grows from 4 to 6, the MAMF statistic corresponding to the true backdoor class pair quickly approaches 1.0 and stays high (see Fig, 3.13a) as one would expect for the true backdoor class pair. For the clean benchmark classifier, the first instance for any class pair's MAMF statistic to be greater than 0.5 is when the absolute support width is 5 (relative support width 5/32). During the growth of the absolute support width from 5 to 7 (where there is another class pair whose MAMF statistic is greater than 0.5), there is a significant gap between the largest MAMF statistic among

[11]For brevity, we only consider support widths that we considered in Section 3.5.6.2, i.e. with relative support width in $[0.08, 0.22]$. However, as shown in Figure 3.13f, for the clean classifier, a 0.5 misclassification fraction is not achieved for any support width we considered. Although such misclassification fraction will finally be achieved if we further increase the relative support width (beyond 0.22), we do not include the results here.

Table 3.9: Attack success rate (%) of backdoor test images with (Gaussian) noisy backdoor patterns with $\sigma^2 = 0.01, 0.25, 1$ and cropped backdoor patterns to 64% and 36% of the original size.

	σ^2			Crop	
	0.01	0.25	1	64%	36%
Attack A	85.2	53.6	53.7	84.6	73.2
Attack B	97.8	86.7	87.6	67.8	26.3
Attack C	62.4	46.9	23.6	96.0	29.1
Attack D	98.1	99.9	99.0	81.7	60.2
Attack E	78.4	45.6	32.0	91.4	46.0
Attack F	97.0	91.0	83.0	86.0	69.0
Attack G	78.0	38.0	28.0	78.0	62.0
Attack H	91.3	66.7	67.2	41.7	20.7
Attack I	86.7	86.7	80.0	40.0	26.7

all the class pairs and 1.0. Thus, using this adaptive criterion, the largest average MAMF for the clean classifier is clearly less than for the classifier being attacked. In Fig. 3.19, we observe this difference between the largest average MAMF for the clean classifier and for the classifier being attacked for Attack A–E, Attack G, and Attack I – there is still a range of valid thresholds that detect most of the attacks.

3.5.6.4 Verification of Property 3.5.2

Here we verify Property 3.5.2, the robustness property of scene-plausible patch replacement patterns, from two aspects. First, for each attack, we modify the backdoor pattern embedded into *each* clean test image (from the source class) by adding Gaussian noise $N(0, \sigma^2)$ to each pixel (and then clipping each pixel value to [0, 1]). Second, instead of adding noise, we crop part of the backdoor pattern outside its center before embedding to the clean test images. We evaluate the attack success rate for noisy backdoor patterns with $\sigma^2 = 0.1, 0.5, 1$ and cropped backdoor patterns to 64% and 36% of the original size in Tab. 3.9. For most attacks, using noisy or cropped patches at test time still achieves high attack success rate (even though cropping may remove critical features on the periphery of the backdoor pattern).

3.5.6.5 Number of Images for Detection

We note that if too few clean images are available to the defender, the performance of our detector could be affected. When there are no attacks, a false detection may be made since it is much easier to find a pattern that induces a high misclassification

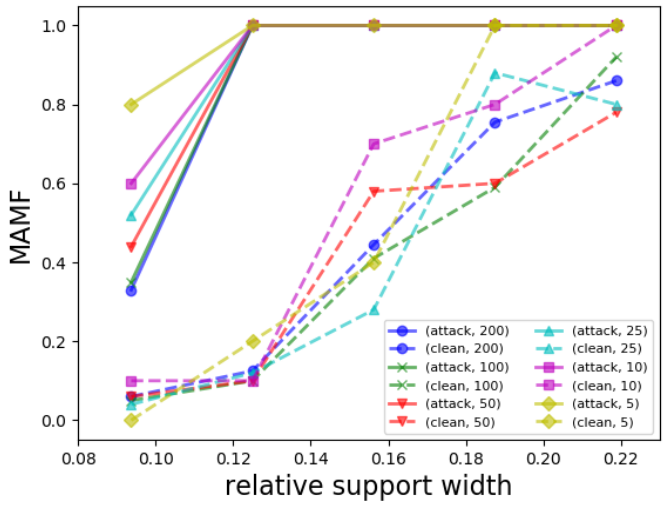


Figure 3.20: Maximum achievable misclassification fraction (MAMF) statistics for the class pair with the largest average MAMF for a range of choices of relative support width r, for both DNNs being attacked and unattacked for Attack A, when using 5, 10, 25, 50, 100 and 200 clean images per class for detection, respectively.

rate for a small group, compared with a larger one. This has been essentially verified in [172], as a "universal" TTE perturbation (a common TTE perturbation applied to all images) inducing high group misclassification needs much larger norm (and full spatial support) than a perturbation inducing single-image misclassification at test time. In Fig. 3.20, we show the MAMF statistics for the class pair with the largest average MAMF for the clean and attacked DNNs for Attack A, with 5, 10, 25, 50, 100 and 200 clean images per class used for pattern estimation. When the number of clean images used for detection is greater than 10, there is a clear gap between the two curves for the DNN being attacked and its clean counterpart. But if fewer images are used for detection, the curve corresponding to the clean DNN approaches 1.0 quite quickly, i.e. achieving a high group misclassification fraction becomes easier with a small spatial support. This is the same phenomenon as in Figure 3.13i for Attack I, where the curve for the clean DNN quickly achieves 1.0 as the support width for pattern estimation increases, since only 9 clean images are used for detection. Note, though, that our detector is successful even in this case – there is still a large gap between the curves.

3.5.6.6 Backdoor Patterns with Fixed Spatial Location

Although MAMF-RED is not designed to detect backdoor attacks with spatially-fixed backdoor pattern, we show in the following that such attacks have poor robustness during testing, when the classifier is trained without data augmentation. We also show that MAMF-RED can easily detect attacks with the patch replacement pattern spatially-fixed (during its embedding into clean images), if the classifier is trained with simple data augmentation that modestly varies the spatial location of the backdoor patch in the poisoned images during training.

Poor robustness: We first perform a simple experiment to show that the power/robustness of the *attack* will be degraded if the backdoor pattern is spatially-fixed when poisoning the clean training images. We use the same dataset, training settings, (source, target) class pair and backdoor pattern under Attack B. The only difference is that the backdoor training images are created by embedding the pattern, the "butterfly", into the bottom left corner of *every* training image to be poisoned. Note that we do not use data augmentation during the classifier training.

After training, the accuracy of the DNN on clean test images is 87.2%, which is similar to the accuracy of the clean benchmark DNN. Now we create four groups of images, 1000 each, using the clean test images from the attacker's source class, such that the backdoor pattern is embedded:

1) into the bottom left corner (i.e. the same location as for the backdoor training images) of all images.

2) spatially randomly in each image.

3) one row up from the bottom left corner of all images.

4) one column right of the bottom left corner of all images.

Here we only focus on the spatial location of the backdoor pattern without considering whether the backdoor patterns are scene-plausible or not. The fraction (%) of images in each group that are (mis)classified to the target class prescribed by the attack are 99.7, 4.1, 47.8, 14.4, respectively. Clearly, only if the backdoor pattern during testing is located at the *same* location as during training, the backdoor image will be reliably (mis)classified to the target class. Hence fixing the spatial location of the backdoor pattern when creating backdoor training images largely degrades the robustness of the attack (and also affects its scene-plausibility).

When training data augmentation is used: Although our detector is not designed for attacks with spatially-fixed patch replacement patterns, we found our detector can actually detect this type of attack when data augmentation is used during the

classifier's training. Here, we use the same configurations as above to train a classifier on the same poisoned training set, where the "butterfly" is spatially-fixed to the bottom left corner of every poisoned image. However, we use data augmentation, including random cropping, random horizontal flipping, and random rotation ($\pm 30°$), during the training, such that the spatial location of the backdoor image patch (i.e. the "butterfly") in the augmented training images will be modified to some extent. The attack success rate and the clean test accuracy for this DNN are 99.5% and 92.0%, respectively. Note that the images used to measure the attack success rate all have the backdoor pattern fixed to the same spatial location as in the backdoor training images (prior to augmentation).

We apply MAMF-RED with the same configurations as for Attack B in Sec. 3.5.6.2, where the fixed square spatial support for pattern estimation covers the top left corner of each image used for detection. Note that our training data augmentation does not include any vertical flipping; hence none of the augmented training images will have the backdoor pattern, the "butterfly", located in the top left corner, i.e. coinciding with the location of the spatial support for our pattern estimation. In other words, the location of the spatial support for pattern estimation is not coinciding with the location of the backdoor pattern in the poisoned training images in this experiment. In Fig. 3.21, we compare the MAMF statistics for the class pair with the largest average MAMF for the above attacked DNN trained with data augmentation, with the MAMF statistics for the clean benchmark DNN for attack B in Section 3.5.6.2. A similarly large gap can be observed between the two curves – the largest average MAMF, ρ^*, for the DNN being attacked (with the spatially-fixed patch replacement pattern and training data augmentation in use) is as large as 0.87 – the attack can be easily detected.

3.5.6.7 Digging Deep into Property 3.5.1

In previous experiments, we have shown that the learned backdoor mapping is spatially invariant if the backdoor pattern embedded in the poisoned training images is random (see Sec. 3.5.6.2), or with moderate randomness depending on the choice of training data augmentation (see Sec. 3.5.6.6). However, there might be an extreme case where the attacker may embed the backdoor pattern (e.g.) close to the bottom left corner of all poisoned training images (with only very limited shift for scene-plausibility). Also, we assume that there is no data augmentation used during training. In this case, will the learned backdoor mapping be spatially invariant? In other words, will the "rough" location of the embedded backdoor pattern be learned as well? This is an interesting question because if Property 3.5.1, the spatial invariance property, does not hold for this

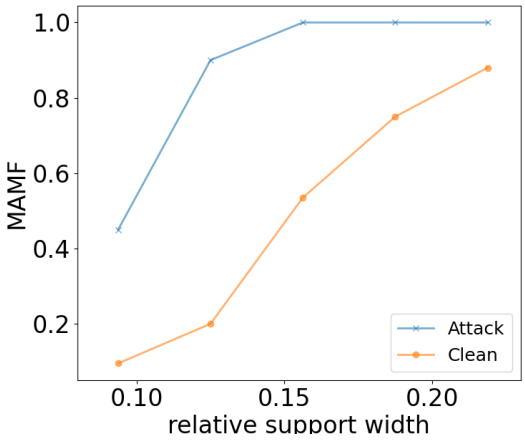
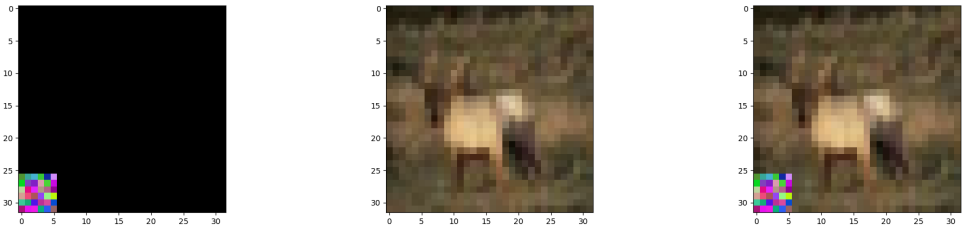


Figure 3.21: Maximum achievable misclassification fraction (MAMF) statistics for the class pair with the largest average MAMF, for the attacked DNN trained with data augmentation, and the clean benchmark DNN for Attack B.



(a) backdoor pattern (b) clean image (c) poisoned image

Figure 3.22: Backdoor pattern used for investigating Property 3.5.1 (a), an example clean training image (b), and the same image with the backdoor pattern embedded (c).

case, MAMF-RED will likely fail when the spatial support for pattern estimation does not match the rough region where the backdoor pattern is embedded when devising the attack.

We created a 6×6 pattern as shown in Figure 3.22. We chose it to be noisy to minimize the possibility for any other regions in the source class images to contain any representative features of the backdoor pattern. In Figure 3.22a, we also show an example backdoor training image and its original clean image.

We create six attacks using this pattern, with each attack associated with a specific integer maximum "offset" ν. During the devising of each attack, the backdoor pattern is embedded randomly inside the $(6+\nu) \times (6+\nu)$ window located at the bottom left corner of the training images to be poisoned. In other words, supposing the pixel at

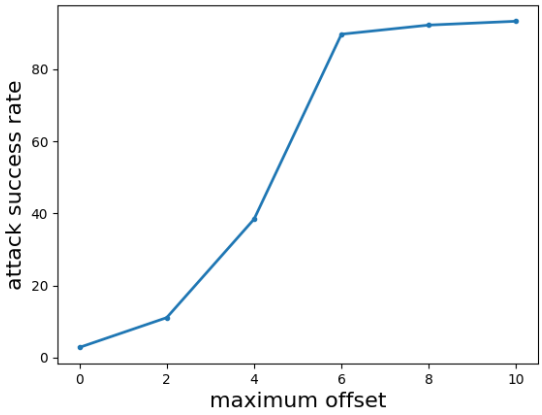


Figure 3.23: Attack success rate measured using images with embedded backdoor pattern randomly located for the six attacks with a range of maximum backdoor embedding spatial "offset".

the bottom left corner of the image has index $(0, 0)$, then the index for the bottom left pixel of the backdoor pattern is (N_W, N_H), with N_W and N_H independently uniform on $\{0, \cdots, \nu\}$. Here, we consider $\nu \in \{0, 2, 4, 6, 8, 10\}$. Note that for the attack with $\nu = 0$, the backdoor pattern is actually spatially-fixed in all the backdoor training images; for the attack with $\nu = 10$, the backdoor pattern is randomly located in only the bottom left quarter of the image (given images size 32×32). The other details for the six attacks and training configurations are the same as for Attack B (see Tab. 3.6).

Now, we check if Property 3.5.1 holds using 1000 test images from the source class. For each originally clean test image, we embed the backdoor pattern to a randomly chosen spatial location. In Figure 3.23, we show the attack success rate measured using these images for all six attacks. When the offset is no less than 6, a source class image with the backdoor pattern embedded will be classified to the target class with high probability, regardless of the spatial location of the backdoor pattern. In these cases, it is the pattern itself instead of the rough location being learned during training on the poisoned training set. In conclusion, Property 3.5.1 holds even when there is moderate randomness in the backdoor embedding when devising the attack.

3.5.6.8 Detecting Backdoor Attacks with Multiple Source Classes

Here, we consider the case where all classes other than the target class are source classes, though this case is generally not practical for scene-plausible patch replacement patterns to be used, as we have discussed in 3.5.1.

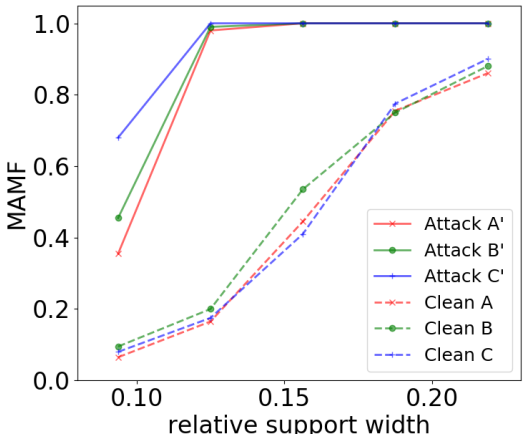


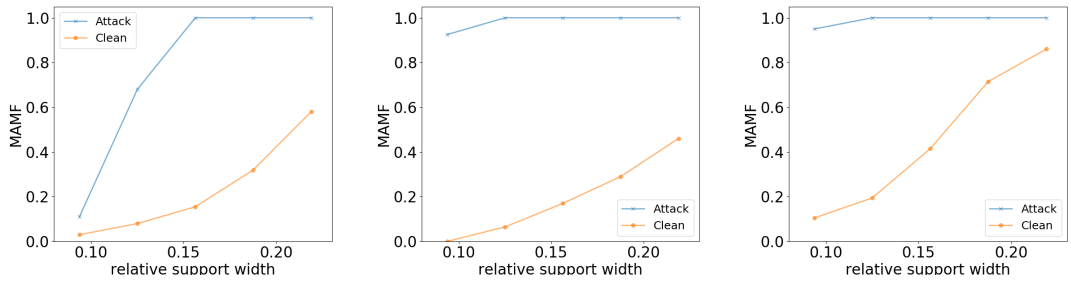
Figure 3.24: Maximum achievable misclassification fraction (MAMF) statistics for a range of choices of relative support width r, for the class pair with the largest average MAMF, for Attack A', B', and C', in comparison with the MAMF statistics for the clean benchmark DNNs for Attack A, B, and C.

We devise Attack A', B', and C', using the same settings as Attack A, B, and C, respectively, but with all classes except the target class as source classes and 20 backdoor training images per source class. Note that the purpose of this experiment is to show that MAMF-RED does not rely on knowledge of the number of source classes involved in an attack. We do not require the backdoor patterns used in this attack experiment to be scene-plausible. For example, for Attack C', we simply embed the backdoor pattern, the rainbow, into random locations of images from classes like "frog", "dog", and "cat", while most images from these classes do not contain sky. The attack success rate (%) of Attack A', B', and C' are 98.0, 95.3, and 98.7, respectively; the accuracy (%) on clean test images are 87.4, 86.8, 87.0, respectively. We apply MAMF-RED with the same setting as in Sec. 3.5.6.2 to these three attacks respectively and show the MAMF statistics for the class pair with the largest average MAMF, for each attack, in Fig. 3.24 – the three attacks (with multiple source classes) are easily detected.

3.5.6.9 Location of the Spatial Support for Pattern Estimation

According to Property 3.5.1, the location of the spatial support in the pattern estimation step can be arbitrarily chosen. Previously, we fixed the spatial support to the top left corner for all clean images used for detection. Here, we experimentally verify that such choice is not critical to our detection.

Again, we consider the same DNN being attacked and the same clean benchmark



(a) Top right (b) Bottom left (c) Bottom right

Figure 3.25: Maximum achievable misclassification fraction (MAMF) statistics for the class pair with the largest average MAMF, for both the DNN being attacked and the DNN not attacked under Attack B. The spatial support for pattern estimation is fixed to cover a) the top left corner, b) the bottom left corner, and c) the bottom right corner.

DNN under Attack B. We apply the same detection to both the clean and attacked DNNs except that the spatial support for detection is fixed to cover 1) the top right corner, 2) the bottom left corner, 3) the bottom right corner of all clean images used for detection. In Fig. 3.25, for each location of the spatial support, we show the L MAMF statistics (by varying the support width) for the class pair with the largest average MAMF, for both DNNs. In each figure, we observe a large gap between the two curves, indicating that the DNN being attacked should be easily detected for a large range of thresholds. Thus, MAMF-RED is indeed robust to the chosen location of the spatial support of the pattern mask.

3.6 L-RED: Improving Computational and Data Efficiency of RED

3.6.1 Computational Complexity and Data Consumption of REDs

For classification domains with K classes, our basic RED, I-RED, and MAMF-RED all perform backdoor pattern estimation for all the $K(K-1)$ class pairs. Suppose N clean images per class are used for detection per class, and pattern estimation converges in no more than T iterations for all class pairs. The computational complexity for these REDs is bounded by $\mathcal{O}(NTK^2)$ single image feed forward passes through the DNN. For other REDs like NC ([109]) and Tabor ([112]), pattern estimation is performed for each putative target class on image from all classes except the target class. That is, K pattern

estimation problem is solved, each with $(K-1)N$ images. Again, supposing that pattern estimation converges in no more than T iterations, the computational complexity for these REDs is also bounded by $\mathcal{O}(NTK^2)$ single image feed forward passes through the DNN.

Although the two types of REDs, one estimating the backdoor pattern for each class pair and the other estimating the backdoor pattern for each class, share the same computational complexity in "big $\mathcal{O}$", the actual computational complexity and data consumption for these two types of methods are different. Considering our basic RED as an example for the first type, N should be sufficiently large to ensure detection accuracy. This is because a perturbation estimated on only a few images will likely have a small size regardless of the presence of backdoor attacks; and the estimated perturbation size for a true backdoor class pair, if there is an attack, will not be much smaller than for non-backdoor class pairs. Then, based on the key ideas of the basic RED in Sec. 3.3.1, the detection statistic associated with the true backdoor class pair may not be an anomaly, causing the attack to escape from our detection. For the second type of RED that estimates a backdoor pattern for each putative target class using images from all the other classes, it is only required that $(K-1)N$ be sufficiently large to ensure the estimated perturbation size or mask size to be large for non-backdoor target classes. For domains with a large number of classes, i.e. with a large K, N can be relatively small (or even $N = 1$ for domains like CIFAR-100 with $K = 100$ classes). This makes the second type of RED seemingly more sufficient both computationally and in terms of data required for detection.

However, the second type of RED mentioned above that performs backdoor pattern estimation for each putative target class using images from all classes other than the target class relies on the assumption that all classes other than the backdoor target class, if there is an attack, are the source classes of the attack. Otherwise, the backdoor pattern estimated for the true backdoor target class may not be related with the backdoor pattern used by the attacker, since most of the image used for pattern estimation are not from the source class of the attack and cannot be misclassified to the attacker's target class by embedding the true backdoor pattern. Then, the estimated perturbation size or the mask size for the true backdoor target class may not be abnormally small, and the attack may not be detected.

Unfortunately, the assumption that all classes other than the target class are source classes does not hold in many cases. For the scene-plausible backdoor pattern considered by our MAMF-RED in Sec. 3.5, it is almost impossible to embed the backdoor pattern in

images from most classes. For example, the "rainbow" pattern used by our experiments (see Fig. 3.12c) can only be embedded in image containing the sky. Moreover, it may be impractical for an attacker to collect images from all classes other than the true target class for creating backdoor training images, especially when K is large.

Thus, we propose L-RED, a Lagrange-based RED, in the following to reduce both the computational complexity and the number of clean image per class required for detection, but without the assumption that all classes other than the target class are source classes. In other words, L-RED is able to detect backdoor attacks with arbitrary number of source classes efficiently. For simplicity, in this section, we mainly consider L-RED for detecting additive perturbation backdoor patterns. The idea of L-RED can be naturally extended to deal with other types of backdoor patterns, especially, the patch replacement pattern considered by many other works (and also our MAMF-RED). However, we leave such extension to our future work.

3.6.2 Pattern Estimation of L-RED

To improve efficiency, L-RED performs pattern estimation for each putative target class $t \in \mathcal{Y}$ using all images from $\cup_{s \neq t} \mathcal{D}_s$. That is, supposing N clean images per class for detection, K pattern estimations are performed, each using $(K-1)N$ clean images. As discussed in Sec. 3.6.1 above, we only need $(K-1)N$ to be sufficiently large to have a significant outlier for the "backdoor" target class when there is an attack. Thus, the minimum N (and computation) required for L-RED to accurately detect backdoor attacks in practice will be much smaller than for, e.g., the basic RED (especially when K is large). However, the defender is not aware of the presence of any backdoor attacks, the number of source classes, or of which classes are source classes, *a priori*. Instead of unreasonably assuming that all classes other than the attacker's target class $t^* \in \mathcal{Y}$ are source classes $\mathcal{S}^* \subset \mathcal{Y}$, i.e. $\mathcal{S}^* \cup t^* = \mathcal{Y}$, we consider that if there is a backdoor attack, it may involve an arbitrary number of source classes and any non-target class may possibly be one of the source classes. Correspondingly, we propose the following weighted pattern estimation problem (assuming an additive perturbation pattern) for each putative target class $t \in \mathcal{Y}$:

$$\begin{aligned} \underset{\mathbf{v} \in \mathcal{X}, \mathbf{w} \in \mathbb{R}^K}{\text{minimize}} \quad & ||\mathbf{v}||_2 \\ \text{subject to} \quad & Q_t(\mathbf{v}, \mathbf{w}) \triangleq \sum_{s \neq t} w_s q_{st}(\mathbf{v}) \ \geq \ \pi, \\ & \sum_{s \neq t} w_s = 1, \quad \mathbf{w} \geq \mathbf{0}, \end{aligned} \tag{3.23}$$

where

$$q_{st}(\mathbf{v}) \triangleq \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \mathbb{1}(f(m_{\text{add}}(\mathbf{x}; \mathbf{v})) = t) \tag{3.24}$$

is the misclassification fraction from s to t given perturbation pattern $\mathbf{v}$; and π is the target misclassification fraction similar to that used in the pattern estimation problem for the basic RED. By solving (3.23) for both the weights $\mathbf{w}$ and the perturbation $\mathbf{v}$, for the true target class when there is a backdoor attack, we expect the true source classes to be assigned much higher weights than other classes.

As for the basic RED, we propose an iterative algorithm to solve (3.23). We sequentially update $\mathbf{v}$ and $\mathbf{w}$ until $Q_t(\mathbf{v}, \mathbf{w}) \geq \pi$, with: initial $\mathbf{w}$ *uniform* in all entries except $w_t = 0$ and initial $\mathbf{v} = \mathbf{0}$ (so that initially $Q_t(\mathbf{v}, \mathbf{w}) \approx 0$).

Update v: In iteration $\tau + 1$, given $\mathbf{v}^{(\tau)}$ and $\mathbf{w}^{(\tau)}$ from iteration τ, we update $\mathbf{v}$ by $\mathbf{v}^{(\tau+1)} = \mathbf{v}^{(\tau)} + \Delta\mathbf{v}^{(\tau+1)}$, where

$$\Delta\mathbf{v}^{(\tau+1)} = \underset{||\mathbf{z}||_2 = \delta}{\text{argmax}}\ Q_t(\mathbf{v}^{(\tau)} + \mathbf{z}, \mathbf{w}^{(\tau)}) \tag{3.25}$$

with δ a small step size to avoid large overshooting (i.e. achieving a weighted misclassification fraction much larger than π in few iterations). Since $Q_t(\mathbf{v}, \mathbf{w})$ is not differentiable in $\mathbf{v}$ due to the indicator function, we find an approximate solution to (3.25) by just one gradient ascent step on a differentiable surrogate of Q_t:

$$\Delta\mathbf{v}^{(\tau+1)} \approx -\delta \nabla_{\mathbf{v}} L_{\text{L-RED}}(\mathbf{v}^{(\tau)}, \mathbf{w}^{(\tau)}, t) / ||\nabla_{\mathbf{v}} L_{\text{L-RED}}(\mathbf{v}^{(\tau)}, \mathbf{w}^{(\tau)}, t)||_2 \tag{3.26}$$

where

$$L_{\text{L-RED}}(\mathbf{v}, \mathbf{w}, t) \triangleq -\sum_{s \neq t} w_s \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \in \mathcal{D}_s} \log p(t | m_{\text{add}}(\mathbf{x}; \mathbf{v})). \tag{3.27}$$

Update w: We aim to have the source classes to be assigned higher weights. In iteration $\tau + 1$, one may naively hard-assign $w_c = 1$ to class $c = \arg\max_{s \neq t} q_{st}(\mathbf{v}^{(\tau+1)})$ because with a small-norm $\mathbf{v}$, misclassification fraction from a source class to the true backdoor target class tends to be larger than for non-source classes. However, after choosing this class, updating $\mathbf{v}$ will be performed on clean images associated with only this class in all subsequent iterations. As uniform weights are initially assigned to all classes except t, in early iterations, updating $\mathbf{v}$ may cause any class $c \notin \mathcal{S}^*$ to temporarily have the largest $q_{ct}(\mathbf{v})$. In this case, with hard-assignment, subsequent pattern estimations will be equivalently performed for a "non-backdoor" class pair – a poor local optimum with a large $||\mathbf{v}||_2$ will be obtained. Thus, we propose the following Lagrangian to update

$\mathbf{w}$ in iteration $\tau+1$ subject to a specified level of "randomness" (entropy of $\mathbf{w}$):

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}) - T^{(\tau+1)} \sum_{s \neq t} w_s \log w_s \\ \text{subject to} \quad & \sum_{s \neq t} w_s = 1, \end{aligned} \tag{3.28}$$

where the Lagrange multiplier $T^{(\tau+1)} \geq 0$ can be viewed as the current iteration's "temperature". Note that if $T^{(\tau+1)} = 0$, solving (3.28) is equivalent to hard weight assignment; if $T^{(\tau+1)}$ is large, solving (3.28) is equivalent to a maximum entropy solution, given $w_t = 0$. To avoid poor local optima caused by hard weight assignment, we need T to be large when $Q_t(\mathbf{v}, \mathbf{w})$ is small in the early iterations. As $Q_t(\mathbf{v}, \mathbf{w})$ grows large, T should *gradually* decrease, such that higher weights are *gradually* assigned to classes with large $q_{\cdot t}(\mathbf{v})$ – these classes are likely the source classes if t is the true backdoor target class. We choose to schedule T automatically as:

$$T^{(\tau+1)} = -\log Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}^{(\tau)}) \in (0, +\infty) \tag{3.29}$$

for $Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}^{(\tau)}) > 0$ (otherwise, $\mathbf{w}$ is set to a uniform vector, but with $w_t = 0$). Note that $T \neq 0$ since the algorithm is terminated when $Q_t(\mathbf{v}, \mathbf{w}) \geq \pi$ for some $\pi < 1$. Given this scheduled $T^{(\tau+1)}$, the closed-form solution to (3.28) is:

$$w_s^{(\tau+1)} = \frac{\exp\left[q_{st}(\mathbf{v}^{(\tau+1)})/T^{(\tau+1)}\right]}{\sum_{c \neq t} \exp\left[q_{ct}(\mathbf{v}^{(\tau+1)})/T^{(\tau+1)}\right]}, \ \forall s \neq t \tag{3.30}$$

with the derivation in Apdx. A.1. Pseudocode for L-RED pattern estimation algorithm is summarized as Alg. 2.

3.6.3 Detection Inference of L-RED

There are multiple anomaly detection approaches that can be used here, for example, the MAD-based approach used by NC [109]. Here, we consider a similar hypothesis test idea as the second inference approach of the basic RED in Sec. 3.3.3.2, but with a much simpler form. We first obtain a (reciprocal) detection statistic for each putative target class t as $r_t = (||\mathbf{v}_t||_2)^{-1}$, where $\mathbf{v}_t$ is the *estimated pattern* for class t. To test the null hypothesis that there is no attack, we fit a null distribution G using all statistics except $r_{\max} = \max_t r_t$ (associated with the class with the smallest estimated pattern norm). Like in Sec. 3.3.3.2, we choose G as a Gamma distribution, while other single tailed

Algorithm 2 L-RED pattern estimation for putative target class t.

Inputs: putative target class t, clean images $\cup_{s\neq t}\mathcal{D}_s$, classifier f, step size δ, target misclassification fraction π, maximum number of iterations $\tau_{\max}$
Initialization: $\mathbf{v}^{(0)} = \mathbf{0}$, $w_s^{(0)} = 1/(K-1)$ for $\forall s \neq t$, $w_t^{(0)} = 0$, $\tau = 0$, estimated pattern $\mathbf{v}_t = \mathbf{0}$
while $\tau < \tau_{\max}$ **do**
    $\mathbf{v}_t = \mathbf{v}^{(\tau+1)} = \mathbf{v}^{(\tau)} - \delta \frac{\nabla_{\mathbf{v}} L_{\mathrm{L-RED}}(\mathbf{v}^{(\tau)}, \mathbf{w}^{(\tau)}, t)}{||\nabla_{\mathbf{v}} L_{\mathrm{L-RED}}(\mathbf{v}^{(\tau)}, \mathbf{w}^{(\tau)}, t)||_2}$
    if $Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}^{(\tau)}) \geq \pi$ **then**
        break
    else if $Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}^{(\tau)}) > 0$ **then**
        $T^{(\tau+1)} = -\log Q_t(\mathbf{v}^{(\tau+1)}, \mathbf{w}^{(\tau)})$
        for all $s \neq t$ **do**
            $w_s^{(\tau+1)} = \frac{\exp\left[q_{st}(\mathbf{v}^{(\tau+1)})/T^{(\tau+1)}\right]}{\sum_{c\neq t}\exp\left[q_{ct}(\mathbf{v}^{(\tau+1)})/T^{(\tau+1)}\right]}$
    else
        $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(0)}$
    τ++
Outputs: $\mathbf{v}_t$

distributions should also work. Then we calculate the order statistic p-value for $r_{\max}$ under G as:

$$\mathrm{pv} = 1 - G(r_{\max})^K. \tag{3.31}$$

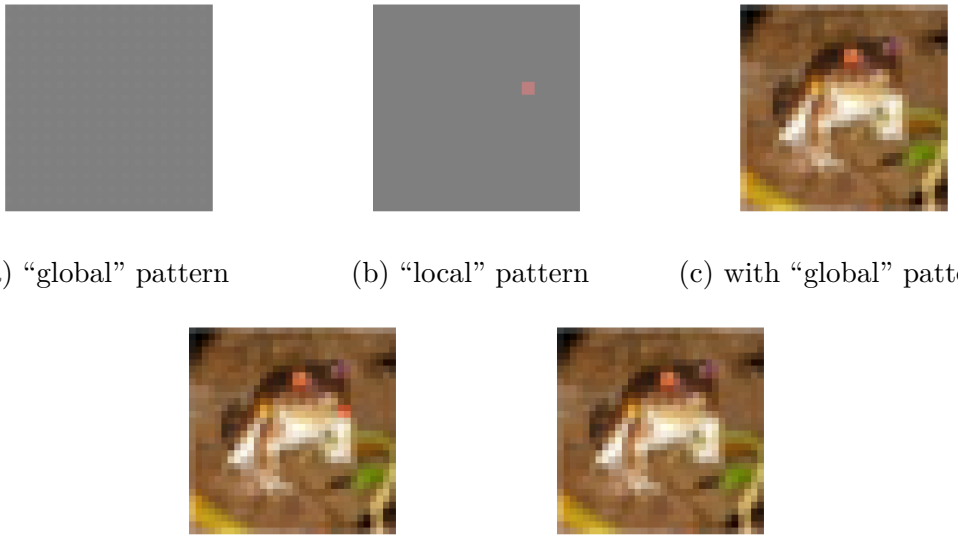
A detection threshold θ is chosen, such that the null hypothesis is rejected (i.e. an attack is detected) with confidence $(1-\theta)$ if $\mathrm{pv} < \theta$. When an attack is detected, $\hat{t} = \arg\max_t r_t$ is inferred as the target class.

3.6.4 Experiments

3.6.4.1 Devising Backdoor Attacks

Dataset: In the experiments here, we consider the CIFAR-10 dataset with the details in Sec. 3.3.6.1. For simplicity, we enumerate the 10 classes of CIFAR-10, i.e., 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck', as classes '1-10'. Experiments involving other datasets are presented in Sec. 3.6.4.4.

Backdoor pattern: We consider the same "global" pattern in Fig. 5 of [129], with maximum perturbation size 2/255. To be specific, for a 2×2 window applied to any spatial location in an image, there is one and only one pixel being perturbed (in all color channels). We also consider a "local" square pattern which perturbs only four pixels of an image by 50/255. Similar pattern was also used by [117] and [113]. Examples of the



(a) "global" pattern (b) "local" pattern (c) with "global" pattern

(d) with "local" pattern (e) clean

Figure 3.26: Example backdoor patterns (with 127/255 offset for better visualization), example backdoor training images embedded with each backdoor pattern, and originally clean image.

two backdoor patterns are shown in Fig. 3.26.

Source classes and target class: For *each* pattern, using the standard backdoor training image crafting approach in Sec. 3.3.6.1, we create three groups of attacks (10 attacks per group) with *one* source class, *three* source classes, and *nine* source classes, respectively. The target classes for attacks using the "global" pattern and the "local" pattern, using the class indexing mentioned above, are class 10 and class 2, respectively. For the "global" pattern, the source classes for attacks with one source class and attacks with three source classes are class 1 and classes $\{1, 7, 9\}$, respectively. For the "local" pattern, the source classes for attacks with one source class and attacks with three source classes are class 1 and classes $\{1, 5, 6\}$, respectively. For both backdoor patterns, attacks with nine source classes use all the classes except for the target class as source classes, i.e., following the $\mathcal{S}^* \cup t^* = \mathcal{Y}$ assumption. All the above choices of source classes and target class are arbitrary.

Number of backdoor training images: In general, the "global" pattern is "easier" to be learned than the "local" pattern. Hence, fewer backdoor training images are required if the "global" pattern is used to launch a backdoor attack. For the "global" pattern, for attacks with one source class, three source classes, and nine source classes, 150, 50, and 15 back door training images per source class are created, respectively. For

Table 3.10: Attack success rate (ASR) and clean test accuracy (ACC) for the six group of classifiers being attacked.

	G-1	G-3	G-9	L-1	L-3	L-9
ASR	92.1 ± 1.9	94.1 ± 1.1	96.6 ± 0.3	93.1 ± 1.6	91.1 ± 0.8	91.9 ± 0.7
ACC	93.6 ± 0.2	93.5 ± 0.2	93.5 ± 0.2	93.7 ± 0.1	93.6 ± 0.2	93.6 ± 0.2

the "local" pattern, for attacks with one source class, three source classes, and nine source classes, 900, 300, and 100 back door training images per source class are created, respectively.

Training configurations: We use the ResNet-18 [14] DNN architecture as in our previous experiments for the basic RED. For each attack, one classifier is trained on the poisoned training set. Training is performed for 100 epochs with batch size 32 and learning rate 10^{-3}. Data augmentation including random cropping and random horizontal flipping are used during training. For convenience, we name the six groups of classifiers being attacked as "G-1", "G-3", "G-9", "L-1", "L-3", and "L-9", respectively, where "G" represents "global" (backdoor pattern), "L" represents "local" (backdoor pattern), and the numbers represent the number of source classes. We also train 10 classifiers on the clean CIFAR-10 training set as the control group (named "C-0") for false detection rate evaluation.

Effectiveness of created attacks: The attack success rate (ASR) and clean test accuracy (ACC) for the attacks are shown in Tab. 3.10, for all attacks, the ASRs are uniformly high, with almost no degradation in ACC compared with group C-0 (with ACC 93.8 ± 0.1) – the attacks are all successful.

3.6.4.2 Defense Performance Evaluation

We compare **L-RED** with two other REDs using the six groups of classifiers being attacked and the clean classifiers in C-0:
(1) **B-RED**: our basic RED presented in Sec. 3.3 that performs pattern estimation for each class pair;
(2) **U-RED**: RED with the $\mathcal{S}^* \cup t^* = \mathcal{Y}$ assumption. Note that U-RED is a special case of L-RED with $\mathbf{w}$ fixed to uniform.
For all these defenses, we choose $\pi = 0.9$, pattern estimation step size $\delta = 10^{-4}$, and detection (confidence) threshold $\phi = 0.05$, while in general, these choices are not critical to the performance of REDs, as discussed for the basic RED in Sec. 3.3. For L-RED, B-RED, and U-RED, we use 8 clean images per class for detection. We also consider

Table 3.11: Detection accuracy (fraction of successful detection) of the defenses on the seven groups of classifiers.

	G-1	G-3	G-9	L-1	L-3	L-9	C-0
B-RED	5/10	9/10	7/10	2/10	3/10	3/10	7/10
U-RED	2/10	8/10	**10/10**	2/10	5/10	**10/10**	**10/10**
L-RED	**10/10**	**10/10**	**10/10**	**10/10**	**10/10**	**10/10**	**10/10**
L-RED'	9/10	**10/10**	**10/10**	9/10	8/10	**10/10**	9/10

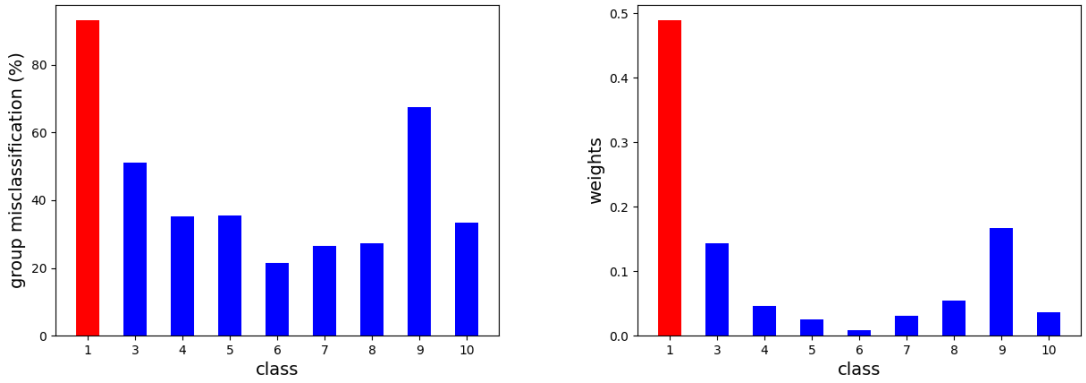
Table 3.12: Average execution time (in seconds) of the defenses on the seven groups of classifiers.

	G-1	G-3	G-9	L-1	L-3	L-9	C-0
B-RED	480	473	531	478	479	394	541
L-RED	562	615	636	529	482	536	535
L-RED'	136	161	167	129	169	163	157

a "data-limited" case, denoted by **L-RED'**, where only 2 clean images per class are available. We reasonably assume that all clean images for detection are correctly classified. For each classifier to be inspected, the clean images for detection are randomly sampled from the test set of CIFAR-10, which are independent from the training (and attack crafting) process.

Accuracy Evaluation: For all the defenses to be evaluated, a successful detection of an attack requires also a correct inference of the target class. For the clean classifiers in C-0, "no attack detected" is deemed a successful detection. In Tab. 3.11, we show the fraction of successful detections for all the defenses for the seven groups of classifiers. For U-RED, although perfect detection is achieved for G-9 and L-9 where the $\mathcal{S}^* \cup t^* = \mathcal{Y}$ assumption is satisfied, with no false detections on C-0, the detection accuracy on G-1, G-3, L-1, and L-3 (where $|\mathcal{S}^*| < |\mathcal{Y}|$) is poor. B-RED shows limited detection capability with 8 clean images per class for attacks with the "global" pattern, which is consistent with Fig. 3.9 in Sec. 3.3.6.4, where 10 image per class leads to a detection accuracy of 80%. But B-RED fails for most of the attacks with the "local" pattern. In comparison, L-RED achieves perfect detection of all the attacks, *regardless of the number of source classes*, and with no false detections. Notably, with only 2 clean images per class, L-RED' detects 56 out of 60 attacks with only 1 out of 10 false detections, which is generally a better result than both B-RED and U-RED.

Efficiency Evaluation: As discussed in Sec. 3.6.1 and demonstrated experimentally, to achieve high detection accuracy, defenses like B-RED, i.e. our basic RED, need a



(a) collateral damage (b) weights

Figure 3.27: Group misclassification fraction and weight assignment (when using L-RED) for all non-target classes, averaged over the ten classifiers from the L-1 group. Note that class 2 is the backdoor target class, and class 1 is that backdoor source class.

"sufficiently large" number of clean images per class (i.e. a large N), while L-RED does not. As shown in Tab. 3.12, L-RED and B-RED need similar average execution times for each of the seven groups of classifiers. However, L-RED' with $N = 2$ exhibits a much lower time consumption than both L-RED and B-RED with $N = 8$. The results here show that the L-RED method allows using fewer clean images to accurately detect backdoor attacks with arbitrary number of source classes; it thus achieves better efficiency in computational time and data consumption. The execution time for U-RED (though not shown here) is similar to that for L-RED. All experiments above are performed on a RTX2080-Ti (11GB) GPU.

3.6.4.3 Additional Results

The effectiveness of L-RED against backdoor attacks with arbitrary source classes is largely due to the fact that L-RED is able to estimate the source classes during backdoor pattern reverse-engineering. The estimation of the source classes is through the optimization of the weight vector $\mathbf{w}$ – more weights are gradually assigned to classes that are more likely the source classes.

For L-RED, if there is an attack, for the true backdoor target class, we expect that when the stopping condition of Algorithm 2 is met, weights will be assigned mainly to the backdoor source classes and classes suffering severe collateral damage (if there are any). The concept of collateral damage is introduced in Sec. 3.3.6.3. Classes suffering collateral damage are neither a source class nor the target class, but a significant proportion of

(a) estimated "global" pattern (b) estimated "local" pattern

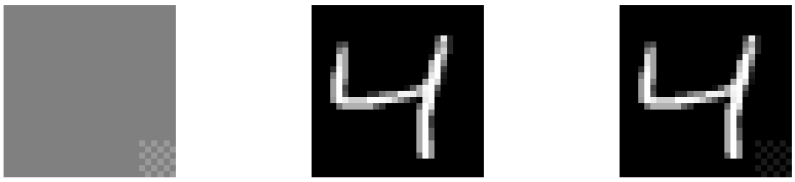
Figure 3.28: Example backdoor patterns (with 127/255 offset to elucidate negative perturbations) estimated by L-RED for the two attacks whose true backdoor patterns being used are shown in Fig. 3.26.

clean images from these classes will be misclassified to the backdoor target class when the backdoor pattern is embedded. Here, we consider the ten classifiers from the L-1 group, where the source class and the target class are classes 1 and class 2, respectively. For each of the ten classifiers, for each class except the target class, we embed the same backdoor pattern used by the attacker into the clean test images (that are not involved in classifier's training) labeled to this class to evaluate the group misclassification fraction to the target class. The average group misclassification fraction for each non-target class over the ten classifiers is shown in Fig. 3.27a. For each of the ten classifiers, we also consider the weight assignment at the end of pattern estimation for the true backdoor target class when L-RED is applied. The weight for each class is also averaged over the ten classifiers and is shown in Fig. 3.27b. As expected, on average, the true source class, class 1, is assigned a much larger weight than all other classes. Class 3 and class 9 are also assigned some weights since they suffer more severe collateral damage than other non-source classes.

Finally, we shown in Fig. 3.28 that L-RED, like our basic RED, can provide good estimation of the backdoor pattern used by the attacker – these estimations keep most of the features of the true backdoor pattern used by the attack shown in Fig. 3.26.

3.6.4.4 Experiments on Other Datasets

Datasets: Here, we test L-RED on several other datasets, including MNIST [15], FMNIST [177], GTSRB [178], and CIFAR-100 [174]. The details for MNIST and CIFAR-100 are introduced in Sec. 3.3.6.5; thus those details are skipped here. FMNIST (Fashion-MNIST) contains 60k training images and 10k test images (all grey-scale), both



(a) MNIST

(b) F-MNIST

(c) GTSRB

(d) CIFAR-100

Figure 3.29: Backdoor pattern (with 0.5 offset) (left), example backdoor training image (right), and originally clean image (middle), for attacks on MNIST, F-MNIST, GTSRB, and CIFAR-100.

evenly distributed in 10 classes. GTSRB is a traffic sign dataset with 43 classes, 26640 training images, and 12630 test images. We resize each image in GTSRB to 32×32.

Attack configurations: For the attack on MNIST, we use a "local chessboard" pattern, i.e. a patch of the "chessboard" pattern located at the bottom right corner of each image to be embedded. We choose the patch size to be 6×6 and perturbation size to be $25/255$. The source classes and the target class are classes $\{5, 7, 10\}$ and class 9, respectively. 500 backdoor training images per source class are created to poison the training set.

Table 3.13: Attack success rate (ASR) and clean test accuracy (ACC) of the classifiers trained on the backdoor poisoned training set; and ACC of the clean classifiers for MNIST, F-MNIST, GTSRB, and CIFAR-100.

	MNIST	F-MNIST	GTSRB	CIFAR-100
attack ASR	96.3	91.7	87.8	98.0
attack ACC	98.9	91.0	98.1	72.4
clean ACC	99.1	90.8	98.7	72.8

For the attack on FMNIST, we use a "single pixel" perturbation backdoor pattern [117], where the pixel being perturbed is located at the bottom right of the image, and the perturbation size is 50/255. The source class and the target class are classes 1 and class 3, respectively. Due to the perturbation size being small and to the fact that there is only one pixel being perturbed, 900 backdoor training images are required to poison the training set to ensure an effective attack.

For the attack on GTSRB, we use the same "local square" pattern used in the main experiments on CIFAR-10. The source classes and the target class are classes $\{13, 14, 15, 16, 17, 20, 24, 25, 27, 28, 29, 30\}$ and class 8, respectively. Class 8 images are for 120kph speed limit signs and the picked source classes contain images for traffic signs requiring a slowing down or stopping. In practice, such an attack may cause an autonomous vehicle to speed up while seeing, e.g., a stop sign. For this attack, we create 30 backdoor training images per source class to poison the training set.

For the attack on CIFAR-100, we use the same "chessboard" pattern as in our experiments in Sec. 3.3.6. The source classes and the target class are class $\{1, \cdots, 15\}$ and class 21, respectively. 50 backdoor training images per source class are used to poison the training set.

Example backdoor patterns, backdoor training images, and their original clean images for the four attacks are shown in Fig. 3.29.

Training configurations: We use LeNet-5 [15] DNN architecture for training both the attacked and the clean classifier on MNIST. Training is performed for 60 epochs, with batch size 256 and learning rate 10^{-2}, without data augmentation. For F-MNIST, we use a "VGG-9" DNN architecture, which is customized by removing the last two convolutional layers of VGG-11 [16] and using 16, 32, 64, 64, 128, 128 filters in the six remaining convolutional layers, respectively. Training is performed for 60 epochs with batch size 256 and learning rate 10^{-2}, without data augmentation. For GTSRB, we use the same DNN architecture used in [109]. Training is performed for 100 epochs with

Table 3.14: Order statistic p-values for both clean and attacked classifiers on MNIST, FMNIST, GTSRB, and CIFAR-100, when applying our defense ("u.f." for "underflow").

	MNIST	FMNIST	GTSRB	CIFAR-100
Attacked	u.f.	8.44×10^{-7}	3.41×10^{-9}	u.f.
Clean	0.300	0.331	0.152	0.551

(a) MNIST (b) F-MNIST (c) GTSRB (d) CIFAR-100

Figure 3.30: Backdoor patterns estimated by L-RED for attacks on MNIST, FMNIST, GTSRB, and CIFAR-100.

batch size 64 and learning rate 10^{-3}, without data augmentation. For CIFAR-100, we use the standard ResNet-34 architecture [14]. Training is performed for 120 epochs with batch size 32 and learning rate 10^{-3}. The same training data augmentation options for training in the main experiments are used here. In Tab. 3.13, we show the ASR and ACC for each classifier being attacked, and ACC for the clean classifiers. Clearly, all the attacks are successful, with high ASR and almost no degradation in ACC compared to the clean benchmarks.

Defense performance: We apply L-RED with the same settings as in Sec. 3.6.4.2 to inspect these classifiers, except that for classifiers trained on GTSRB and CIFAR-100, we use *only one* clean image per class for detection. In Tab. 3.14, we show the order statistic p-values obtained for each classifier by our defense. All attacks are detected with no false detection of any clean classifiers using the threshold $\theta = 0.05$. For the backdoor attack on CIFAR-100, our defense requires less than 3 hours for detection while our basic RED requires several days. Moreover, in Fig. 3.30, we show the estimated pattern for each detected attack. The main features of the backdoor patterns being used are recovered for all four attacks. Also, the target class for each attack is correctly inferred by L-RED.

3.7 ET-RED: Detecting Backdoor Attacks for Two-Class and Multi-Attack Scenarios Using Expected Transferability

3.7.1 Overview

3.7.1.1 Limitations of RED Detection Inference Approach

Due to the assumption that there are no clean classifiers for reference, REDs typically implement an anomaly detector for detection inference. For example, in Sec. 3.3.3, our basic RED performs anomaly detection based on a hypothesis test, where a null distribution is estimated on $(K-1)^2$ statistics to assess the atypicality of any statistics that may associate with backdoor class pairs. For another example, in [109], NC proposed an anomaly score based on MAD [167]; and the confidence threshold is derived based on the assumption that the null distribution (equivalently estimated on $(K-1)$ statistics) is a Gaussian distribution. However, these anomaly detection approaches heavily rely on the assumption that there is a relatively large number of non-target classes, thus providing a sufficient number of statistics to inform estimation of a null distribution. But for domains with only two classes, e.g. sentiment classification [179], disease diagnosis [180], etc., this assumption does not hold. Thus, most existing REDs are unsuitable for these 2-class problems.

3.7.1.2 Key Ideas of ET-RED

Like other post-training defenses, the main targets here are detecting whether a given classifier is backdoor attacked or not and, if so, finding out all the backdoor target classes. In particular, we focus on "two-class" post-training scenarios with the following major assumptions:

S1) The defender has no access to the training set of the classifier to be inspected, nor any prior knowledge of the true backdoor pattern used by an attacker. **S2**) There are no clean classifiers trained for the same domain for reference; and the defender is not capable of training such clean classifiers.

S3) The classification domain has two classes that can both be a backdoor target class.

Note that **S1** and **S2** are general assumptions for the post-training scenarios as described in Chap. 2. For **S3**, while we focus on two-class scenarios in this section, our proposed

method is more generally applicable to multi-class scenarios, with arbitrary number of attacks (see experiments in Sec. 3.7.6.5).

To address **S1**, our detection framework involves backdoor pattern reverse-engineering using a small, clean dataset independently collected by the defender, like existing REDs. To address **S3**, unlike existing REDs that perform anomaly detection involving statistics from *all* classes, we inspect each class *independently* using a novel detection statistic called *expected transferability* (ET), which can be empirically estimated for each class independently. To address **S2**, we show that ET possesses a theoretically-grounded detection threshold value for distinguishing backdoor target classes from non-target classes, *one which depends neither on the domain nor on the attack configuration*. This is very different from existing REDs, for which a suitable detection threshold for their proposed statistics may be both domain and attack-dependent. For example, the range of the l_1 norm of the estimated mask used by [109] depends on the image size; the range of the cosine similarity statistic used by [113] depends on the architecture of the classifier. The practical import here is that the detection threshold is a *hyperparameter*, but setting this threshold in a supervised fashion (e.g. to achieve a specified false positive rate on a group of clean classifiers) is generally infeasible due to **S2**. Use of ET thus obviates the need for such hyperparameter setting.

In the following, we first introduce the definition of ET and related concepts. Then we derive the constant threshold on ET for backdoor detection. In particular we show that when there is no attack, the ET statistic for all classes should be no larger than $\frac{1}{2}$. We also present a rigorous proof for this ET upper bound for a simplified problem. Finally, we discuss the detection procedure of ET-RED, which consists of an ET estimation step and an inference step. Note that our framework does not rely on the type of the backdoor pattern. Solely for clarity, we focus on backdoor attacks with additive perturbation backdoor patterns here, while similar derivation/analysis for other types of backdoor patterns, especially the patch replacement pattern is deferred to Apdx. A.3.

3.7.2 Expected Transferability (ET)

Consider a classifier $f : \mathcal{X} \to \mathcal{Y}$ to be inspected with category space $\mathcal{Y} = \{0, 1\}$ and *continuous* sample distribution P_i on $\mathcal{X}$ for class $i \in \mathcal{Y}$. For any $\mathbf{x} \in \mathcal{X}$ from any class, the optimal solution to

$$\underset{\mathbf{v} \in \mathcal{X}}{\text{minimize}} \, ||\mathbf{v}||_2 \qquad \text{subject to } f(\mathbf{x} + \mathbf{v}) \neq f(\mathbf{x}) \tag{3.32}$$

is defined as $\mathbf{v}^*(\mathbf{x})$. In practice, (3.32) can be viewed as a typical optimization problem for reverse-engineering an additive perturbation pattern, e.g. problem (3.4) for the basic RED customized for two-class domains. It can also be practically solved by creating an adversarial example for $\mathbf{x}$ using methods in, e.g., [43, 74]. Note that we use the explicit embedding formulation for additive perturbation patterns (i.e. Eq. 2.1) in problem 3.32, but with the clipping operation neglected for simplicity. Next, we present the following definition for the set of *practical* solutions to (3.32).

Definition 3.7.1. (ϵ-solution set) For any sample $\mathbf{x}$ from any class, regardless of the method being used, the ϵ-solution set to problem (3.32), is defined by

$$\mathcal{V}_\epsilon(\mathbf{x}) \triangleq \{\mathbf{v} \in \mathcal{X} \big| ||\mathbf{v}||_2 - ||\mathbf{v}^*(\mathbf{x})||_2 \leq \epsilon, f(\mathbf{x}+\mathbf{v}) \neq f(\mathbf{x})\}, \tag{3.33}$$

where $\epsilon > 0$ is the "quality gap" of practical solutions, which is usually small for existing methods.

Note that in practice, it is usually impossible to get the exact optimal solution $\mathbf{v}^*(\mathbf{x})$ for problem (3.32). However, there are many optimization algorithms proposed to obtain a "good" solution with a small "quality gap" to, e.g., ensure the created adversarial examples are not human-noticeable [43].

A practical solution to (3.32) for sample $\mathbf{x}$ may or may not cause a misclassification when embedded in another sample $\mathbf{y}$ from the same class. In the following, we first present the definition regarding such a "transferability" property. Then, we define the ET statistic for any class $i \in \mathcal{Y}$.

Definition 3.7.2. (Transferable set) The transferable set for any sample $\mathbf{x}$ and $\epsilon > 0$ is defined by

$$\mathcal{T}_\epsilon(\mathbf{x}) \triangleq \{\mathbf{y} \in \mathcal{X} \big| f(\mathbf{y}) = f(\mathbf{x}), \exists \mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{x}) \,\text{s.t.}\, f(\mathbf{y}+\mathbf{v}) \neq f(\mathbf{y})\}. \tag{3.34}$$

Definition 3.7.3. (ET statistic) For any class $i \in \mathcal{Y} = \{0, 1\}$ and $\epsilon > 0$, considering i.i.d. random samples $\mathbf{X}, \mathbf{Y} \sim P_i$, the ET statistic for class i is defined by

$$\mathrm{ET}_{i,\epsilon} \triangleq \mathbb{E}\big[\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}) | \mathbf{X})\big]. \tag{3.35}$$

3.7.3 Using ET for Backdoor Detection

Here, we show that for any $i \in \mathcal{Y} = \{0, 1\}$ and small ϵ, we will likely have $\mathrm{ET}_{i,\epsilon} > \frac{1}{2}$ when class $(1-i)$ is a backdoor target class; and $\mathrm{ET}_{i,\epsilon} \leq \frac{1}{2}$ otherwise. Note that this *constant*

threshold does not rely on any specific data domain, classifier architecture, or attack configuration; even for backdoor patterns embedded by Eq. (2.2), the *same* threshold can be obtained following a similar derivation (see Apdx. A.3).

We first present the following theorem showing the connection between ET and the threshold $\frac{1}{2}$ regardless of the presence of any backdoor attack. Then, we discuss the attack and non-attack cases, respectively.

Theorem 3.7.1. For any class $i \in \mathcal{Y}$ and $\epsilon > 0$:

$$\mathrm{ET}_{i,\epsilon} = \frac{1}{2} + \frac{1}{2}(\mathrm{P}_{\mathrm{MT},i} - \mathrm{P}_{\mathrm{NT},i}), \tag{3.36}$$

with $\mathrm{P}_{\mathrm{MT},i}$ a "mutual-transfer probability" and $\mathrm{P}_{\mathrm{NT},i}$ a "non-transfer probability", defined by:

$$\mathrm{P}_{\mathrm{MT},i} \triangleq \mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}), \mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})) \quad \text{and} \quad \mathrm{P}_{\mathrm{NT},i} \triangleq \mathrm{P}(\mathbf{Y} \notin \mathcal{T}_\epsilon(\mathbf{X}), \mathbf{X} \notin \mathcal{T}_\epsilon(\mathbf{Y})) \tag{3.37}$$

respectively, where $\mathbf{X}$ and $\mathbf{Y}$ are i.i.d. random samples following distribution P_i for class i.

The proof of Thm. 3.7.1 is in Apdx. A.2.1.

1) Non-attack case: class $(1-i)$ for $i \in \mathcal{Y}$ is *not* a backdoor target class. We first focus on $\mathrm{P}_{\mathrm{NT},i}$.

Property 3.7.1. For general two-class domains in practice and small ϵ, if class $(1-i)$ is *not* a backdoor target class, $\mathrm{P}_{\mathrm{NT},i}$ for class i will likely be larger than $\frac{1}{2}$ (see Sec. 3.7.6.4 for empirical support).

For any class $i \in \mathcal{Y}$, $\mathrm{P}_{\mathrm{NT},i}$ is the probability that two independent samples from class i are mutually "not transferable". For such a pair of samples, the event associated with $\mathrm{P}_{\mathrm{NT},i}$ is that the pattern estimated for one (by solving (3.32)) does not induce the other to be misclassified and vice versa. Accordingly, if we solve a problem similar to (3.32) but requiring a common $\mathbf{v}$ that induces both samples to be misclassified, the solution should have a larger norm than the solution to (3.32) for each of them. Property 3.7.1 has been verified by many existing works for general classification tasks commonly using highly non-linear classifiers. For example, the universal adversarial perturbation studied by [172] can be viewed as the solution to problem (3.32) (in absence of backdoor attack) for a group of samples instead of one. Compared with the minimum sample-wise perturbation required for each individual to be misclassified, the minimum universal

perturbation required for high group misclassification typically has a much larger norm. Similar empirical results have also been shown by the experiments in Sec. 3.3.6.4. Despite the evidence in existing works, we also verify this property experimentally in Sec 3.7.6.4.

Next, we focus on $\mathrm{P}_{\mathrm{MT},i}$. Note that $\mathrm{P}_{\mathrm{MT},i}$ is upper bounded by $1 - \mathrm{P}_{\mathrm{NT},i}$; thus it will likely be smaller than $\frac{1}{2}$ (and even possibly close to 0) for the non-attack case based on Property 3.7.1. The asymptotic behavior of $\mathrm{P}_{\mathrm{MT},i}$ for small ϵ can also be approached by the following theorem. Note that in practice, a small ϵ is not difficult to achieve as we have mentioned above, since a solution to problem (3.32) using, e.g., algorithms for generating adversarial samples [66] usually have a small norm.

Theorem 3.7.2. For any class $i \in \mathcal{Y}$ with continuous sample distribution P_i, $\mathrm{P}_{\mathrm{MT},i} \to 0$ as $\epsilon \to 0$.

The proof of Thm. 3.7.2 is in Apdx. A.2.2. In summary, for the non-attack case with small ϵ, we will likely have $\mathrm{P}_{\mathrm{NT},i} \geq \frac{1}{2} \geq \mathrm{P}_{\mathrm{MT},i}$; and thus, $\mathrm{ET}_{i,\epsilon} \leq \frac{1}{2}$ (based on Thm. 3.7.1) – this will be shown by our experiments. Moreover, in Sec. 3.7.4, for a simplified (yet still relatively general) domain and a linear prototype classifier, we show that $\frac{1}{2}$ is a *strict* upper bound of the ET statistic when there is no backdoor attack.

2) Attack case: class $(1 - i)$ is the target class of a successful backdoor attack. Suppose $\mathbf{v}_0$ is the backdoor pattern for this attack, such that for any $\mathbf{X} \sim P_i$, $f(\mathbf{X}) = i$, while $f(\mathbf{X} + \mathbf{v}_0) \neq f(\mathbf{X})$. Intuitively, $\mathrm{P}_{\mathrm{MT},i}$ will be large and possibly close to 1 because, *different from the non-attack case*, there is a special pattern – the backdoor pattern $\mathbf{v}_0$ – that could likely be an element of $\mathcal{V}_\epsilon(\mathbf{X})$ and $\mathcal{V}_\epsilon(\mathbf{Y})$ simultaneously, for $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following P_i. In this case, $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$ and $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ jointly hold (i.e. mutually transferable) by Definition 3.7.2. Accordingly, $\mathrm{P}_{\mathrm{NT},i}$ which is upper bounded by $1 - \mathrm{P}_{\mathrm{MT},i}$ will likely be small and possibly close to 0; then, we will have $\mathrm{P}_{\mathrm{MT},i} > \mathrm{P}_{\mathrm{NT},i}$ and consequently, $\mathrm{ET}_{i,\epsilon} > \frac{1}{2}$ by Thm. 3.7.1.

Beyond the intuitive analysis above, the following theorem gives a *guaranteed* large ET statistic (being exactly 1) in the attack case when the backdoor pattern has a (sufficiently) small norm (which is in fact desired in order to have imperceptibility of the attack) The proof of the theorem is in Apdx. A.2.3.

Theorem 3.7.3. If class $(1 - i)$ is the target class of a successful backdoor attack with backdoor pattern $\mathbf{v}_0$ such that $f(\mathbf{X} + \mathbf{v}_0) \neq f(\mathbf{X})$ for all $\mathbf{X} \sim P_i$, and if $||\mathbf{v}_0||_2 \leq \epsilon$, we will have $P(\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset) = 1$ for $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following P_i; and furthermore, $\mathrm{ET}_{i,\epsilon} = 1$.

3.7.4 Analysis on a Simplified Classification Problem

Here, we consider a simplified analogue to practical 2-class classification problems. Although assumptions are imposed for simplicity, we still keep the problem relatively general by allowing freedom on, e.g., the sample distribution in their latent space. With the simplification, we are able to analytically derive the condition for a sample belonging to the transferable set of another sample from the same class. Based on this, we show that $\frac{1}{2}$ is the *supremum* of the ET statistic when there is no attack. Note that we reuse some of the notations used in Sec. 3.7.3 – the notations in this section are all self-contained.

3.7.4.1 Problem Settings

We consider sample space $\mathbb{R}^n$ with an orthonormal basis $\{\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_d, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{n-d}\}$. We assume that samples from the two classes are distributed on sub-spaces $V_0 = \{\mathbf{Ac}|\mathbf{c} \in \mathbb{R}^d\}$ and $V_1 = \{\mathbf{Be}|\mathbf{e} \in \mathbb{R}^{n-d}\}$ respectively, where $\mathbf{A} = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_d]$ and $\mathbf{B} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{n-d}]$, respectively. Based on the definition of V_0 and V_1, the *latent spaces*, i.e. $\mathbb{R}^d$ and $\mathbb{R}^{n-d}$, for the two classes are also well defined. Here, we do not constrain the form or parameters of the distributions for the two classes in both the sample space $\mathbb{R}^n$ and latent spaces, as long as the distributions are continuous. Such an analogue may be corresponding to some simple classification domains in practice. Moreover, the latent space may be corresponding to an internal layer space of some DNN classifier. For example, for a typical ReLU DNN classifier, it is possible that a subset of nodes in the penultimate layer are mainly activated for one class, while another subset of nodes are mainly activated for the other class.

For this simplified domain, we consider a nearest prototype classifier that is capable of classifying the two classes perfectly for *any* continuous sample distributions. To achieve this, any point $\mathbf{x} \in \mathbb{R}^n$ on the decision boundary of the classifier should have equal distance to V_0 and V_1, i.e.:

$$||\mathbf{A}\mathbf{A}^T\mathbf{x} - \mathbf{x}||_2 = ||\mathbf{B}\mathbf{B}^T\mathbf{x} - \mathbf{x}||_2 \tag{3.38}$$

By expanding both sides of Eq. (3.38) and rearranging terms (using the fact that $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and $\mathbf{B}^T\mathbf{B} = \mathbf{I}$), we obtain the decision boundary of the classifier, which is $\{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}(\mathbf{A}\mathbf{A}^T - \mathbf{B}\mathbf{B}^T)\mathbf{x} = 0\}$. In other words, the nearest prototype classifier

$f : \mathbb{R}^n \to \{0, 1\}$ is defined by

$$f(\mathbf{x}) = \begin{cases} 0 & \mathbf{x}(\mathbf{A}\mathbf{A}^T - \mathbf{B}\mathbf{B}^T)\mathbf{x} > 0 \\ 1 & \mathbf{x}(\mathbf{A}\mathbf{A}^T - \mathbf{B}\mathbf{B}^T)\mathbf{x} \leq 0 \end{cases} \tag{3.39}$$

Note that this classifier is not necessarily linear; and the region for each class may not be convex.

3.7.4.2 Derivation of Transfer Condition

Here, we derive the condition that one sample belongs to the transferable set of another sample from the same class. Since the two classes are symmetric, we focus on class 0 for brevity. Moreover, in the following, we refer to samples by their latent space representation; i.e., instead of "a sample $\mathbf{x} \in \mathbb{R}^n$ from class 0", we say "a sample $\mathbf{c} \in \mathbb{R}^d$".

First, we present the following modified definition of the transferable set (compared with Def. 3.7.2 in Sec. 3.7.2) using latent space representation.

Definition 3.7.4. (Transferable set in latent space) The transferable set for any sample $\mathbf{c} \in \mathbb{R}^d$ from class 0 is defined by

$$\mathcal{T}(\mathbf{c}) = \{\mathbf{c}' \in \mathbb{R}^d \big| f(\mathbf{A}\mathbf{c}') = f(\mathbf{A}\mathbf{c}), f(\mathbf{A}\mathbf{c}' + \mathbf{v}^*(\mathbf{c})) \neq f(\mathbf{A}\mathbf{c}')\}, \tag{3.40}$$

where $\mathbf{v}^*(\mathbf{c})$ is the optimal solution to

$$\underset{\mathbf{v} \in \mathbb{R}^n}{\text{minimize}} \, ||\mathbf{v}||_2 \qquad \text{subject to } f(\mathbf{A}\mathbf{c} + \mathbf{v}) \neq f(\mathbf{A}\mathbf{c}). \tag{3.41}$$

Note that the above definition is in a similar form as Def. 3.7.2 in Sec. 3.7.2, though here, the quality to the solution to problem (3.41) is no longer considered. This is because, different with problem (3.32) in Sec. 3.7.2, problem (3.41) here can be solved analytically, yielding a closed form solution instead of a solution set with some intrinsic quality bound ϵ. Accordingly, we present the following theorem, which gives the condition for one sample belonging to the transferable set of another sample from the same class for our simplified settings. The proof is deferred to Apdx. A.2.4.

Theorem 3.7.4. For any $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^d$, $\mathbf{c}' \in \mathcal{T}(\mathbf{c})$ if and only if $||\mathbf{c}' - \mathbf{c}/2||_2 \leq ||\mathbf{c}/2||_2$.

3.7.4.3 Upper Bound on ET Statistic

Here, we show that $\frac{1}{2}$ is the *minimum upper bound* on the ET statistic for this simplified problem when there is no backdoor. To do so, we first present a definition of ET statistic modified from Def 3.7.3 in Sec. 3.7.2, merely in adaption to the latent space representation used here (see Def. 3.7.5). Then, we show the tightness of the bound by giving a concrete example where ET equals $\frac{1}{2}$ in Lem. 3.7.1. Finally, we prove that the ET statistic cannot be greater than $\frac{1}{2}$. The proof of Lem. 3.7.1 and Thm. 3.7.5 are shown in Apdx. A.2.5 and Apdx. A.2.6, respectively.

Definition 3.7.5. For i.i.d. random samples $\mathbf{C}$ and $\mathbf{C}'$ following some continuous distribution G on $\mathbb{R}^d$, the ET statistic is defined by

$$\mathrm{ET} = \mathbb{E}_{\mathbf{C}\sim G}\big[\mathrm{P}(\mathbf{C}' \in \mathcal{T}(\mathbf{C})|\mathbf{C})\big] \tag{3.42}$$

Lemma 3.7.1. For latent space $\mathbb{R}^d$ with dimension $d = 1$ and distribution G continuous on $\mathbb{R}$, the ET statistic satisfies

$$\frac{1}{4} \leq \mathrm{ET} \leq \frac{1}{2}, \tag{3.43}$$

where $\mathrm{ET} = \frac{1}{2}$ if and only if $G(0) = 0$ or $G(0) = 1$.

Theorem 3.7.5. For arbitrary $d \in \mathbb{Z}_+$ and continuous distribution G on $\mathbb{R}^d$

$$\sup_{d,G} \quad \mathrm{ET} = \frac{1}{2}. \tag{3.44}$$

3.7.5 Detection Procedure of ET-RED

The detection procedure of ET-RED is summarized in Alg. 3. Basically, for each putative target class $t \in \mathcal{Y} = \{0, 1\}$, we estimate the ET statistic $\widehat{\mathrm{ET}}_{i,\epsilon}$ (with a "hat" representing empirical estimation) for class $i = 1 - t$, and claim a detection if $\widehat{\mathrm{ET}}_{i,\epsilon} > \frac{1}{2}$. In particular, the core to estimating $\widehat{\mathrm{ET}}_{i,\epsilon}$ is to estimate $\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})|\mathbf{X} = \mathbf{x}_n^{(i)})$ for each clean sample $\mathbf{x}_n^{(i)} \in \mathcal{D}_i$ used for detection (line 6-12, Alg. 3). To do so, we propose to find, for each $\mathbf{x}_n^{(i)} \in \mathcal{D}_i$, the subset $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})$ which contains *all* samples in $\mathcal{D}_i \setminus \mathbf{x}_n^{(i)}$ belonging to the *transferable set* $\mathcal{T}_\epsilon(\mathbf{x}_n^{(i)})$ of sample $\mathbf{x}_n^{(i)}$. Then, $\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})|\mathbf{X} = \mathbf{x}_n^{(i)})$ can be estimated by $p_n^{(i)} = |\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})|/(|\mathcal{D}_i| - 1)$ (line 12, Alg. 3). However, by Def. 3.7.2, a sample $\mathbf{y}$ is in the *transferable set* $\mathcal{T}_\epsilon(\mathbf{x})$ of a sample $\mathbf{x}$ as long as there *exists* a practical solution to problem (3.32) (with some intrinsic quality gap ϵ) that induces $\mathbf{y}$ to be misclassified as well. Thus, it is insufficient to decide whether or not a sample is in the transferable set

Algorithm 3 Backdoor detection using ET statistics.

1: **Input**: Classifier $f : \mathcal{X} \to \mathcal{Y}$ for inspection; clean dataset $\mathcal{D}_i = \{\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_{N_i}^{(i)}\}$ for each $i \in \mathcal{Y}$.
2: **Initialization**: attacked = $False$; BA_targets = $\emptyset$.
3: **for** each putative target class $t \in \mathcal{Y} = \{0, 1\}$ **do**
4: **Step 1:** Obtain empirical estimation $\widehat{\mathrm{ET}}_{i,\epsilon}$ using $\mathcal{D}_i$ for $i = 1 - t$.
5: **for** $n = 1 : N_i$ **do**
6: $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)}) = \emptyset$; converge = $False$
7: **while** not converge **do**
8: Obtain an empirical solution $\hat{\mathbf{v}}(\mathbf{x}_n^{(i)})$ to problem (3.32) using random initialization.
9: $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)}) \leftarrow \widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)}) \cup \{\mathbf{x}_m^{(i)} | m \in \{1, \cdots, N_i\} \setminus n, f(\mathbf{x}_m^{(i)} + \hat{\mathbf{v}}(\mathbf{x}_n^{(i)})) \neq f(\mathbf{x}_m^{(i)})\}$
10: **if** $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})$ unchanged for τ iterations **then**
11: converge $\leftarrow True$
12: $p_n^{(i)} = |\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})|/(N_i - 1)$
13: $\widehat{\mathrm{ET}}_{i,\epsilon} = \frac{1}{N_i} \sum_{n=1}^{N_i} p_n^{(i)}$
14: **Step 2:** Determine if class t is a backdoor target class or not.
15: **if** $\widehat{\mathrm{ET}}_{i,\epsilon} > \frac{1}{2}$ **then**
16: attacked = $True$; BA_targets $\leftarrow$ BA_targets $\cup \{t\}$
17: **Output**: attacked; BA_targets

of another sample according to merely one solution realization to problem (3.32). To address this, for each $\mathbf{x}_n^{(i)}$, we solve problem (3.32) *repeatedly* with random initialization. For each practical solution, we embed it to all elements in $\mathcal{D}_i \setminus \mathbf{x}_n^{(i)}$ and find those that are misclassified – these samples are included into the subset $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})$ (line 9, Alg. 3). Such repetition stops when $\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)})$ stays unchanged for some τ iterations. This procedure is summarized as the "while" loop in Alg. 3, which is guaranteed to converge in $(N_i - 1) \times \tau$ iterations. Finally, we obtain the estimated ET by averaging $p_n^{(i)}$ – the empirical estimation of $\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}) | \mathbf{X} = \mathbf{x}_n^{(i)})$ – over all samples in $\mathcal{D}_i$.

Our ET-RED detection framework has the following generalization capabilities. 1) **Backdoor pattern embedding mechanism.** As mentioned before, ET-RED can be adapted to patch replacement patterns embedded by Eq. (2.2), with the same constant threshold $\frac{1}{2}$ on ET and only little modification to the while loop in Alg. 3 (see Apdx. A.3.3). 2) **Backdoor pattern reverse-engineering algorithm.** We do not limit our detector to any specific algorithm when solving problem (3.32) (line 8 in Alg. 3) – existing algorithms proposed by, e.g., our basic RED, [43], and even future backdoor pattern reverse-engineering algorithms can be used. 3) **Adoption for multi-class scenario.** When there are multiple classes, where each can possibly be a backdoor target class, Alg.

3 can still be used for detection by, for each putative target class $t \in \mathcal{Y}$, treating all the classes other than t as a *super-class* and estimating ET on $\cup_{i \in \mathcal{Y} \setminus t} \mathcal{D}_i$. In our experiments (next), we will evaluate our ET-RED detection framework considering a variety of these extensions.

3.7.6 Experiments

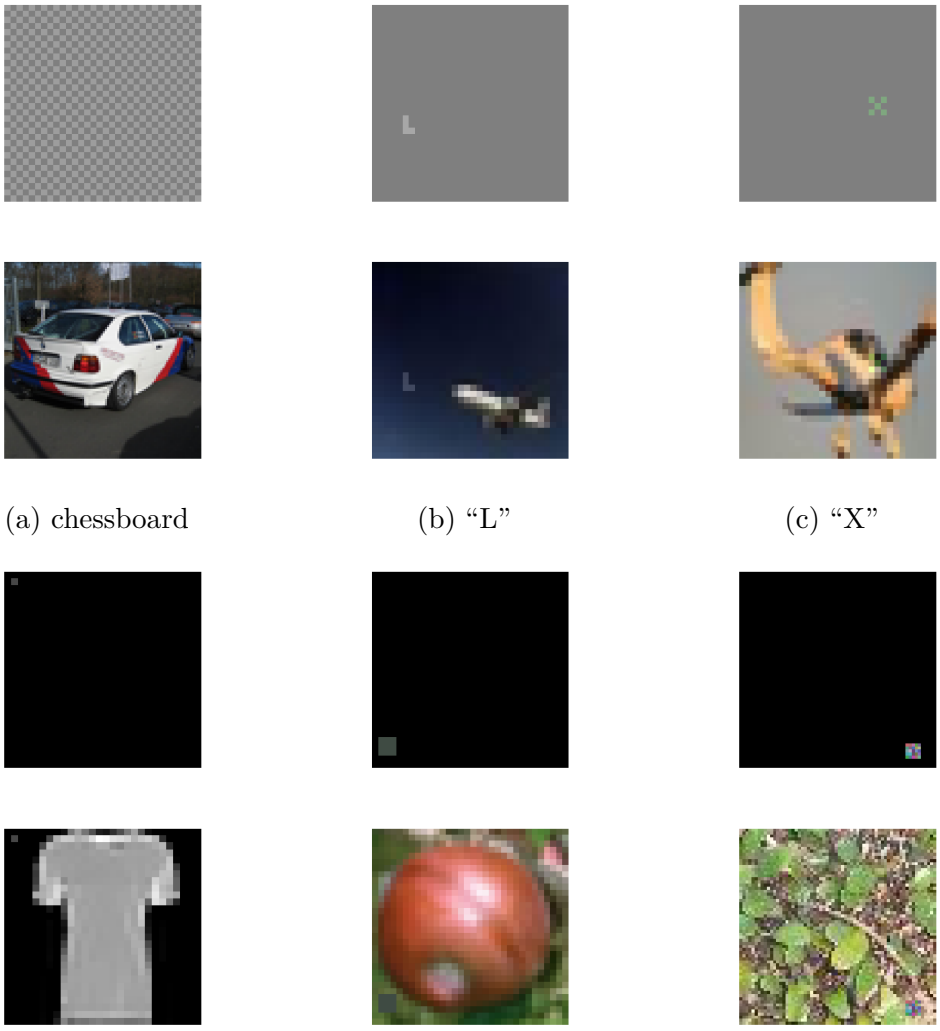
3.7.6.1 Devising Backdoor Attacks

Datasets: Our experiments involve six common benchmark image datasets with a variety of image size and color scale: CIFAR-10, CIFAR-100 [174], STL-10 [181], TinyImageNet, FMNIST [177], MNIST [15]. Details for CIFAR-10, CIFAR-100, MNIST, and FMNIST can be found in previous sections. STL-10 contains 13000 96×96 color images from 10 classes, including 500 images per class for training and 800 images per class for testing. TinyImageNet contains 64×64 color images from 200 classes, with 600 images per class, including 500 images for training and 100 images for testing.

Generating 2-class domains. From CIFAR-10, we generate 45 different 2-class domains; from *each* of the other five datasets, we generate 20 different random 2-class domains. In particular, for CIFAR-10, the 45 2-class domains are corresponding to the 45 unordered class pairs of CIFAR-10 respectively. For each of CIFAR-100, FMNIST, and MNIST, we randomly sample 20 unordered class pairs, each forming a 2-class domain. For TinyImageNet, due to high image resolution and data scarcity, we generate 20 "super class" pairs – for each pair, we randomly sample 20 classes from the original category space and then evenly assign them to the two super classes (each getting 10 classes from the original category space). Similarly, for STL-10 with 10 classes, we generate 20 super class pairs by randomly and evenly dividing the 10 classes into two groups (of 5 classes from the original category space) for each pair. For each generated 2-class domain, we use the subset of data associated with these two (super) classes from the original dataset, with the original train-test split.

Attack configurations: For each 2-class domain generated from CIFAR-10, CIFAR-100, STL-10, and TinyImageNet, we create two attack instances, one for backdoor with additive perturbation pattern embedded by Eq. (2.1), and the other for backdoor attack with patch replacement pattern embedded by Eq. (2.2). For each 2-class domain generated from FMNIST and MNIST, we create one attack instance with additive perturbation pattern[12]. For convenience, the six ensembles of attack instances with

[12]Images from these two datasets commonly have a large area of "black" background. Positively



(a) chessboard (b) "L" (c) "X"

(d) a pixel (e) unicolor patch (f) noisy patch

Figure 3.31: Part of the backdoor patterns used in our experiments and images with these backdoor patterns embedded. (a)-(d) are additive perturbation patterns and (e)-(f) are patch replacement patterns. Backdoor pattern in (a) is amplified for visualization. Spatial locations for backdoor patterns in (b)-(f) are randomly selected for each attack.

additive perturbation pattern and for 2-class domains generated from CIFAR-10, CIFAR-100, STL-10, TinyImageNet, FMNIST, and MNIST are denoted as $\mathbf{A_1}$-$\mathbf{A_6}$ respectively. The four ensembles of attack instances with patch replacement pattern and for 2-class domains generated from CIFAR-10, CIFAR-100, STL-10, and TinyImageNet are denoted as $\mathbf{A_7}$-$\mathbf{A_{10}}$ respectively. These short hand "codes" are summarized in Tab. 3.15.

perturbing a few background pixels, which is a common practice to achieve a successful backdoor attack [118], is *equivalent* to replacing these pixels with a gray patch using Eq. (2.2).

Table 3.15: Short hand "code" for each ensemble of attack instances based on both the backdoor pattern being used and the dataset where the associated 2-class domains are generated from. "n/a" represents "not applicable".

	Additive perturbation pattern	Patch replacement pattern
CIFAR-10	A_1	A_7
CIFAR-100	A_2	A_8
STL-10	A_3	A_9
TinyImageNet	A_4	A_{10}
FMNIST	A_5	n/a
MNIST	A_6	n/a

The backdoor patterns used in our experiments include many popular ones in the backdoor literature – examples of some backdoor patterns and images embedded with them are shown in Fig. 3.31. In particular, the "chessboard" pattern is the same one we used in Sec. 3.3.6, with maximum perturbation size 3/255. For the other "localized" additive perturbation patterns, the "L" pattern and the "X" pattern have been used by both [117] and [113]. For the "L" pattern, we perturb all the color channels by 40/255. For the "X" pattern, for each attack, we randomly choose a channel (for all images to be embedded in for this particular attack) and perturb the associated pixels positively by 40/255. The "pixel" pattern has been used by [117, 118], where a single pixel is perturbed in all channels by 40/255 for color images and 70/255 for gray-scale images. For the patch replacement patterns, in Fig. 3.31e, a small, monochromatic patch located near the margin of the image is embedded. Similar pattern have been considered by [106] and [109]. The color is randomly chosen and fixed for each attack. The backdoor pattern in Fig. 3.31f is a small noisy patch located near the margin of the images to be embedded in. Similar backdoor patterns have been considered by [143] and [3]. For both patterns, once the location is selected, the same location will be applied to all images to be embedded in for the same attack. Also, for both patch replacement patterns, the size of the patch is 3×3 for the 2-class domains generated from CIFAR-10 and CIFAR-100; 4×4 for the 2-class domains generated from TinyImageNet; and 10×10 for the 2-class domains generated from STL-10.

We consider 2-class scenarios where *both classes can possibly be a backdoor target class*. For each attack instance in ensembles A_1, A_2, A_3, A_7, A_8, and A_{10}, we create two attacks each with one of the two classes being the backdoor target class. For each

Table 3.16: Summary of attack configurations for instances in each of ensembles A_1-A_{10}. For each ensemble, we show the number of attacks for each instance in this ensemble. We also show, for each ensemble, the number of samples (embedded with BP and labeled to the target class) used for poison the training set for each attack associated with this ensemble, as well as the corresponding poisoning rate.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
# Attacks	2	2	2	1	1	1	2	2	1	2
# Poison samples	500	150	1000	1500	1000	1000	500	50	750	500
Poisoning rate	9.1%	23.0%	28.6%	23.0%	14.3%	14.3%	9.1%	9.1%	23.1%	9.1%

instance in ensembles A_4, A_5, A_6, and A_9, we create one attack with the second class being the backdoor target class. For each of these attacks, the backdoor pattern is randomly selected from the candidate backdoor pattern (part of which are shown in Fig. 3.31), with the specified type of backdoor pattern for the ensemble which the attack is associated with. However, to avoid confusion for learning the backdoor mapping during training, for all 2-attack instances with additive perturbation patterns, we ensure that the backdoor pattern for the two attacks have different shapes (e.g., two "X" patterns are not allowed). Similarly, for all 2-attack instances where the two attacks both choose to use the unicolor patch backdoor pattern (Fig. 3.31e), we ensure that the colors for the two backdoor patterns are significantly different. The attacks are launched following the same typical backdoor poisoning as our experiments for the basic RED. The poisoning rate and the number of target class for all attacks are summarized in Tab. 3.16.

Training configurations. We train one classifier for each attack instance using the poisoned training set. For each 2-class domain, we also train a clean classifier to evaluate false detections. We denote the six ensembles of clean instances for datasets CIFAR-10, CIFAR-100, STL-10, TinyImageNet, FMNIST, and MNIST as $\mathbf{C_1}$-$\mathbf{C_6}$ respectively. For classifier training, we consider a variety of DNN architectures (with two output neurons, one for each of the two classes). For 2-class domains associated with CIFAR-10 and STL-10, we use ResNet-18 [14]; for CIFAR-100 and FMNIST, we use VGG-11 [16]; for TinyImageNet, we use ResNet-34; and for MNIST, we use LeNet-5 [15]. Training configurations including the learning rate, the batch size, and the optimizer choice for each 2-class domain are summarized in Tab. 3.17. All attacks for all the instances are successful with high ASR and negligible degradation in ACC, as jointly shown in Tab.

Table 3.17: Training details, including learning rate, batch size, number of epochs, whether or not using training data augmentation, choice of optimizer (Adam [5] or stochastic gradient descent (SGD)), for 2-class domains generated from CIFAR-10, CIFAR-100, STL-10, TinyImageNet, FMNIST, and MNIST, respectively.

	Learning rate	Batch size	# Epochs	Data augmentation	Optimizer
CIFAR-10	0.001	32	150	✗	Adam
CIFAR-100	0.001	32	150	✗	Adam
STL-10	0.001	128	150	✗	Adam
TinyImageNet	0.001	128	150	✓	Adam
FMNIST	0.01	256	100	✗	SGD
MNIST	0.01	256	100	✗	SGD

Table 3.18: Average ACC (in percentage) over all classifiers being attacked, average and minimum ASR (in percentage) over all attacks, for each of ensemble A_1-A_{10} of attack instances.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
ASR (average)	95.7±3.6	91.6±4.5	98.8±1.0	93.9±2.8	96.8±3.3	99.8±0.4	99.2±1.1	96.4±4.0	97.9±1.4	94.5±4.6
ASR (minimum)	82.3	80.0	95.9	88.0	87.6	98.4	92.5	82.0	95.1	83.2
ACC (average)	94.6±4.2	90.7±4.5	79.6±2.9	77.0±3.2	99.0±1.1	99.8±0.1	96.7±2.5	93.2±4.1	78.7±3.4	77.2±3.5

3.18 and Tab. 3.19.

3.7.6.2 Defense Performance Evaluation

As mentioned in Sec. 3.7.5, ET-RED can be easily generalized to incorporate a variety of algorithms for backdoor pattern reverse-engineering to detect various types of backdoor patterns with different embedding mechanisms. Here, we consider the algorithm used by our basic RED for estimating Additive Perturbation patterns, and the algorithm proposed by [109] for estimating Patch Replacement patterns. For convenience, we denote detection configurations with these two algorithms as **RE-AP** and **RE-PR**, respectively. Irrespective of the backdoor pattern reverse-engineering algorithm, we use only 20 clean images per class (similar to most existing REDs) for detection. Finally, we set the "patience" parameter for determination of convergence in Alg. 3 to τ=4 – this choice is independent of the presence of backdoor attack; a larger τ will not change the resulting

Table 3.19: Average ACC (in percentage) over the classifiers for the clean instances in each of ensemble C_1-C_6.

	C_1	C_2	C_3	C_4	C_5	C_6
ACC (average)	95.4±3.6	93.3±3.3	80.5±3.2	78.2±3.0	99.3±1.0	99.8±0.1

Table 3.20: Detection accuracy for RE-AP and RE-PR on attack ensembles A_1-A_{10}, and on clean ensembles C_1-C_6, using *the common threshold 1/2 on ET statistic*. "n/a" represents "not applicable".

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	C_1	C_2	C_3	C_4	C_5	C_6
RE-AP	45/45	18/20	16/20	17/20	20/20	20/20	n/a	n/a	n/a	n/a	45/45	20/20	20/20	20/20	20/20	20/20
RE-PR	n/a	n/a	n/a	n/a	n/a	n/a	45/45	20/20	19/20	19/20	39/45	19/20	20/20	16/20	18/20	19/20

ET much, but only increase the execution time.

In practice, our ET-RED with RE-AP and with RE-PR can be deployed *in parallel* to cover both additive perturbation patterns and patch replacement patterns. Here, for simplicity, we apply ET-RED with RE-AP to classifiers (with attacks using additive perturbation pattern) in ensemble A_1-A_6, and apply ET-RED with RE-PR to classifiers (with attacks using patch replacement pattern) in ensemble A_7-A_{10}. For each classifier in clean ensembles C_1-C_6, we apply ET-RED with both configurations. In Tab. 3.20, for each ensemble of attack instances, we report the fraction of classifiers such that the attack and all backdoor target classes are both successfully detected; for each ensemble of clean instances, we report the fraction of classifiers that are inferred to be not attacked. Given the large variety of classification domains, attack configurations, DNN architectures, and defense generalizations mentioned above, *using the common ET threshold* $\frac{1}{2}$, our ET-RED successfully detect most attacks with only very few false detections (see Tab. 3.20).

3.7.6.3 Compare ET with Other Statistics

The results for existing REDs applying to the attacks we created are neglected for brevity, since these REDs cannot detect backdoor attacks for 2-class domains by their design. However, we compare our ET statistic with some popular types of statistic used by existing REDs in terms of their potential for being used to distinguish backdoor target

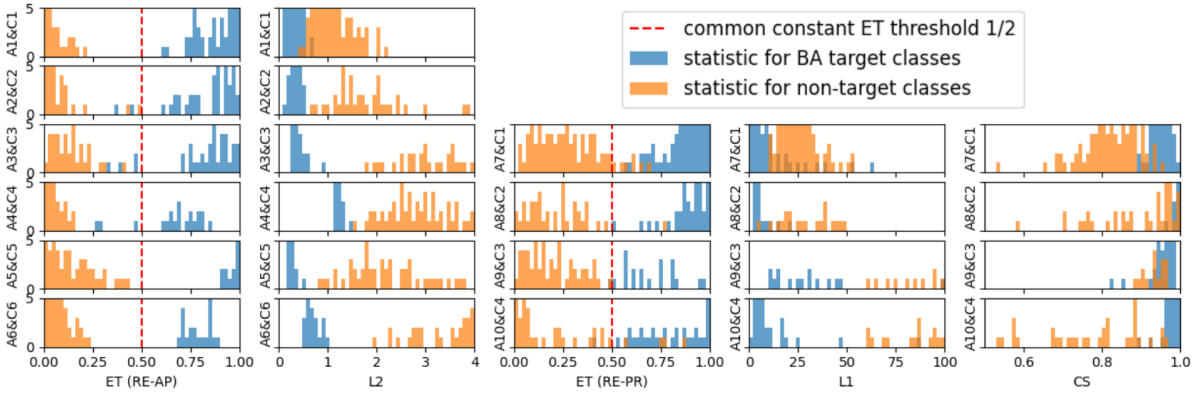
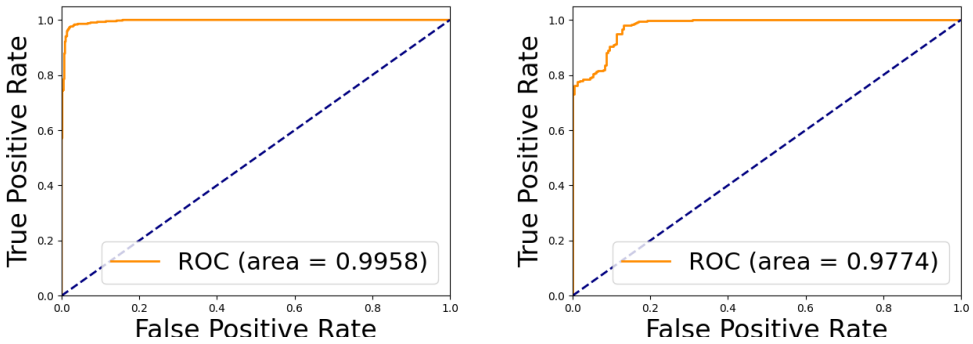


Figure 3.32: Comparison between our ET statistic (for both RE-AP and RE-PR configurations) and statistic types used by existing REDs (L_1, L_2, and CS). Only for ET, there is a common range for all 2-class domains for choosing a threshold to distinguish backdoor target classes (blue) from non-target classes (orange). Such common range also contains the constant threshold 1/2 (red dashed line).

classes from non-target classes. The types of statistic for comparison include: 1) the l_2 norm of the estimated additive perturbation used by our basic RED (denoted by $\mathbf{L_2}$); 2) the l_1 norm of the estimated mask used by [109] (denoted by $\mathbf{L_1}$); and 3) the (cosine) similarity between the backdoor pattern estimated group-wise and the backdoor pattern estimated for each sample in terms of classifier's internal layer representation [113] (denoted by **CS**) More details of the cosine similarity statistic are shown in Sec. 3.2.

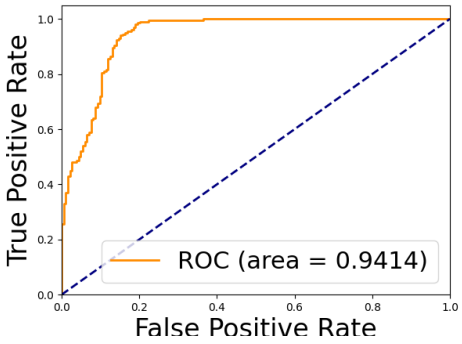
For each type of statistic mentioned above and each benchmark dataset, we consider all classifiers (with and without backdoor attack) trained for all 2-class domains generated from the dataset, and plot a double histogram for statistics obtained for all backdoor target classes and all non-target classes across these classifiers respectively. For example, in the 1st column of Fig. 3.32, for ET-RED with RE-AP, we plot a double histogram for ET statistics obtained for all backdoor target classes and all non-target classes from all classifiers in each of A_1&C_1, A_2&C_2, A_3&C_3, A_4&C_4, A_5&C_5, and A_6&C_6. Note that all classifiers in, e.g., A_1&C_1 are trained for 2-class domains generated from CIFAR-10. Here, for simplicity, for ET (with RE-AP) and L_2 (both designated for additive perturbation pattern), we do not consider backdoor target classes with patch replacement pattern (e.g. associated with classifiers in A_7-A_{10}); for ET (with RE-PR), L_1 and CS (designated for patch replacement pattern), we do not consider backdoor target classes with additive perturbation pattern (e.g. associated with classifiers in A_1-A_6).

Based on Fig. 3.32, for *all* types of statistics including our ET, statistics obtained for backdoor target classes are generally separable from statistics obtained for non-target

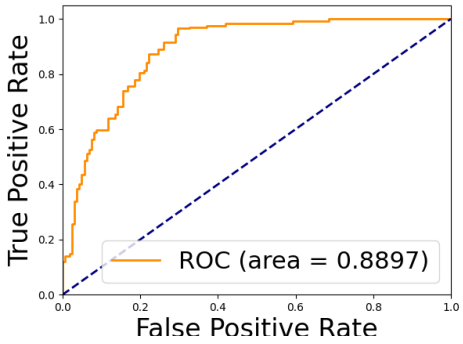


(a) ET

(b) L_2



(c) L_1



(d) CS

Figure 3.33: ROC curves for ET, L_1, L_2, and SC in distinguishing backdoor target classes from non-target classes for the large variety of classification domains and attack configurations considered in Fig. 3.32.

classes for classifiers (with and without backdoor attack) trained for 2-class domains generated from the *same* benchmark dataset (using same training configurations including DNN architecture). But only for our ET (obtained by both RE-AP and RE-PR), there is a *common range* (irrespective of the classification domain, attack configurations, and training configurations) for choosing a detection threshold to effectively distinguish backdoor target classes from non-target classes for all instances; and such common range clearly includes the *constant threshold* $\frac{1}{2}$ (marked by red dashed lines in Fig. 3.32) derived mathematically in Sec. 3.7.3. By contrast, for both L_1 and L_2, a proper choice of detection threshold for distinguishing backdoor target classes from non-target classes is domain dependent. For example, in the 4^{th} column of Fig. 3.32, the l_1 norm of masks estimated for both backdoor target classes and non-target classes for domains with larger image size (e.g. 96×96 for domains in $A_9\&C_3$ generated from STL-10) is commonly larger than for domains with smaller image size (e.g. 32×32 for domains in $A_7\&C_1$

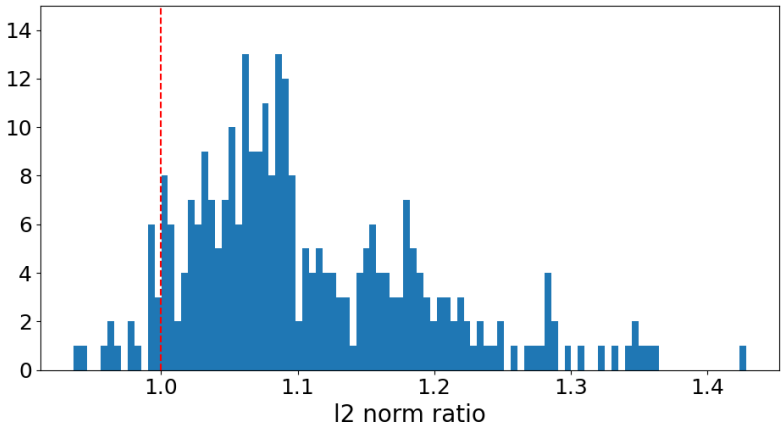


Figure 3.34: Histogram of l_2 norm ratio between pair-wise additive perturbation and maximum of the two sample-wise perturbations for each random image pair for clean classifiers.

generated from CIFAR-10). The CS statistic, on the other hand, not only relies on the domain, but also depends on the DNN architecture.

In summary, all the above mentioned types of statistics are suitable for backdoor detection *if there is supervision from the same domain for choosing the best proper detection threshold*. However, in most practical scenarios where such supervision is not available, only our ET statistic with the common detection threshold $\frac{1}{2}$ can still be used to achieve a good detection performance. Finally, in Fig. 3.33, we show the receiver operating characteristic (ROC) curves associated with Fig. 3.32 for each of ET, L_1, L_2, and SC – our ET statistic has clearly larger area under the ROC curve (very close to 1) than the other types of statistics.

3.7.6.4 Experimental Verification of Property 3.7.1

We verify Property 3.7.1 by showing that, if a class is not a backdoor target class, for any two clean images from the other class (considering 2-class domains), the minimum additive perturbation required to induce both images to be misclassified has a *larger* norm than the minimum perturbation required for each of these two images to be misclassified. Here, we randomly choose one clean classifier from each of C_1-C_6. For each classifier, we randomly choose 50 *pairs* of clean images from a random class of the associated 2-class domain – these images are also used for detection in Sec. 3.7.6.2. For each pair of images, we apply the same RE-AP algorithm for additive perturbation patterns on the two images both *jointly* (to get a pair-wise common perturbation) and *separately* (to get two

Table 3.21: Maximum ET statistic over all classes for classifiers with one, two, and three attacks respectively, and a clean classifier, for CIFAR-10, CIFAR-100, and STL-10.

	1 attack	2 attacks	3 attacks	clean
CIFAR-10	0.91	0.92	0.87	0
CIFAR-100	0.95	0.99	0.99	0.27
STL-10	0.65	0.83	0.77	$4.3e^{-3}$

sample-wise perturbationsfor the two images respectively). Then, we divide the l_2 norm of the pair-wise perturbation by the *maximum* l_2 norm of the two sample-wise perturbations to get a ratio (which is more scale-insensitive than taking absolute difference). In Fig. 3.34, we plot the histogram of such ratio for all image pairs for all six classifiers – the ratio for most of the image pairs is greater than 1 (marked by the red dashed line in Fig. 3.34). For these pairs, very likely that the perturbation estimated for one sample cannot induce the other sample to be misclassified and vise versa (otherwise their expected ratio will likely be 1).

3.7.6.5 Multi-Class, Multi-Attack Backdoor Detection Using ET Statistic

Since ET-RED inspects each class independently, it can also be used for backdoor detection for more general scenarios with *more than two classes* and *arbitrary number of attacks* (and backdoor target classes). For demonstration, for each of CIFAR-10, CIFAR-100, and STL-10, we create three attack instances with one, two, and three attacks, respectively (on the *original* domain). Here, we consider backdoor attacks with additive perturbation patterns for simplicity. The target class and the shape of backdoor pattern for each attack are randomly selected. We train one classifier for each attack instance (thus, nine classifiers being attacked in total). For each domain, we also train a clean classifier (without backdoor attack) for evaluating false detections.

Following the description at the end of Sec. 3.7.5, we apply the generalized Alg. 3 with RE-AP to these classifiers. For CIFAR-10 and STL-10, we use three clean images per class for detection; for CIFAR-100, we use only one clean image per class for detection. Other detection configurations including the *detection threshold 1/2* are the same as in Sec. 3.7.6.2. Since a classifier is deemed to be attacked if ET obtained for any class is greater than the threshold 1/2, for each classifier, we show the maximum ET over all the classes in Tab. 3.21. Clearly, the maximum ET is greater than 1/2 for all classifiers

being attacked and less than 1/2 for all clean classifiers. Thus, our detection framework (with the same constant threshold on ET) is also applicable to multi-class, multi-attack scenarios.

Chapter 4
Backdoor Defense Before/During Training

4.1 Overview

Recall from Sec. 2.1.3 that when there is a backdoor attack, the (clean) training set $\mathcal{D}_{\text{train}}$ is poisoned by a small set of samples, denoted by $\mathcal{D}_{\text{backdoor}}$, that are originally from the source classes, embedded with the backdoor pattern, and labeled to the target class. Backdoor defense deployed before/during the classifier's training phase aims to detect if there is an attack, i.e. whether or not $\mathcal{D}_{\text{backdoor}} = \emptyset$. If there is an attack, samples from $\mathcal{D}_{\text{backdoor}}$ should be identified and removed before training (or retraining, since most defenses in this scenario requires first training classifier on the possibly poisoned training set). Moreover, the defender should reduce the number of clean samples being falsely remove to ensure the test accuracy of the classifier.

Like most existing works for this scenario, we assume that the defender has full control of the training process, but has no prior knowledge about whether $\mathcal{D}_{\text{backdoor}} = \emptyset$ or not, or which samples are from $\mathcal{D}_{\text{backdoor}}$. We also mainly consider backdoor attacks on image classifiers that uses an additive perturbation backdoor pattern. Detection of other types of backdoor patterns, e.g. the patch replacement pattern will be briefly discussed, with empirical results, in Sec. 4.5. In this chapter, again, we first review other defenses deployed during the classifier's training phase. Then, we present our defense as follows:

- In Sec. 4.3, we propose a cluster impurity (CI) defense based on that: 1) the backdoor training samples (with backdoor target class t^*) are separable from clean training samples labeled to class t^* in internal layer feature spaces of the classifier; and 2) additive perturbation patterns are subtle and easily destroyed by simple

image preprocessing like blurring. Upon the publication of this work, we were the first to compare the several existing before/during training defenses including ours, and achieved the state-of-the-art performance.

- In Sec. 4.4, we propose a reverse-engineering-based defense, dubbed DT-RED (with "DT" for "during training"), which is inspired by our basic RED in Sec. 3.3. Like the CI approach, DT-RED requires first training a classifier on the possibly poisoned training set. Different with RED deployed post-training, DT-RED reverse engineers a putative backdoor pattern for each (s, t) class pair on the training images, not only to induce a high misclassification rate from class s to class t, but also to remove the backdoor pattern embedded in the backdoor training images if (s, t) is a backdoor class pair, such that these images are classified to class s (which is the source class they are originally from).

Note that for the CI approach, a detection threshold (which may be domain dependent) is required. But for the DT-RED approach, the detection inference is purely unsupervised, which addresses the more challenging scenario where no held out clean samples for the domain are available for reference.

4.2 Related Works

Early backdoor defenses deployed during the training phase leverages the separation between backdoor training samples and clean training samples labeled to the backdoor target class in terms of their internal layer representation. Such separation was first observed and named "spectral signature" (SS) of backdoor pattern by [117]. Accordingly, SS proposed to project the penultimate layer (i.e. the layer before the last layer) features of the training samples in each class onto the principal eigenvector of the feature vector's covariance matrix. If there is a backdoor attack, the backdoor training samples will likely appear as outliers; and the training set can be sanitized by removing these outliers. However, SS did not propose any method to detect whether the training set has been poisoned, or which class has been poisoned. There is also no method proposed to determine the number of outliers to be removed.

The "activation clustering" (AC) method proposed in [118] involves both backdoor detection (based on a Silhouette score) and training set cleansing. For each putative target class, the penultimate layer activations of the associated training images are projected onto a low dimensional (e.g. 10-dimensional) space using principal component

analysis (PCA) and clustered using k-means with $k = 2$. If the activations are well-fit by the two-cluster model (evaluated by comparing with a threshold on the Silhouette score), a backdoor attack is detected and this class is deemed the target class of the attack. Then, for the detected target class, the training images associated with the cluster with the smaller mass are identified as the backdoor training images and are removed before retraining. However, in many cases, the backdoor training images are not clearly separable (using e.g. 2-means) from the clean training images labeled to the backdoor target class in the projected low-dimensional space of internal layer features. And this limitation motivated us to consider performing clustering in the original internal layer feature space instead (see our CI method in Sec. 4.3), or consider a method does not require clustering (see our DT-RED in Sec. 4.4).

Backdoor training samples can also be detected during a classifier's training on the possibly poisoned training set. In [59], the authors found that, when there is a backdoor attack (or a poisoning attack aiming to degrade the classifier's accuracy), in early iterations of the training process, the training loss on the mislabeled samples (inserted by the attack) are usually higher than on the clean training samples that are correctly labeled. This is because the attacker usually only inserts a small set of mislabeled sample into the training set for the stealthiness of the attack; thus, the gradient of the training loss (on the classifier's parameters) is typically dominated by the majority of the training samples that are correctly labeled. So, [59] proposed to update the model parameters in each training iteration using training samples with the smallest loss calculated from the previous iteration. However, this method requires that the poisoning rate (see Sec. 2.1.4 for definition) is relatively small. Moreover, in [182], it is found that some backdoor patterns are actually easier to learn than clean samples.

In [183], a method is proposed to train a backdoor-free classifier on the possibly poisoned training set. First, a DNN-based feature extractor is trained on the training set using a self-supervised loss (e.g. the one used by SimCLR [184]) with the labels removed. Then a downstream linear classifier is trained on the entire training set with the labels and with the feature extractor fixed. Using this classifier, the training set is divided into high-credible samples (with high prediction confidence) and low-credible samples (with low prediction confidence). The classifier is then fine-tuned using a semi-supervised loss on the high-credible samples with the labels and low-credible samples with the labels removed. However, there is no backdoor detection method proposed in [183], and the method does cause some degradation to the accuracy of the trained classifier when there is no attack.

4.3 CI: Detecting Backdoor Attacks During-Training Using Cluster Impurity

4.3.1 Key Ideas of CI

Our proposed CI approach involves a clustering step and a detection inference step. The clustering step is based on the findings by previous works (e.g. [117]) that backdoor training images are separable from clean training images labeled to the backdoor target class in internal layer feature representations. However, different from previous works, we find that: 1) such separation may only hold for full-dimensional feature vectors without projection (e.g. onto one-dimensional space like in [117]); and 2) there may exists multiple "semantics" in the same category, such that the associated internal layer features for the category form more than one cluster. These phenomena occur more often for complicated datasets like CIFAR-10 than simple datasets like MNIST. Thus, CI does not project the internal layer features onto low dimensional spaces to keep as much information for clustering as possible. Moreover, we adopt Bayesian information criterion (BIC) to determine the number of clusters when modeling the internal layer features for each class. Compared with AC that uses 2-means for clustering, we allow "multi-modality" in each category of the domain. We also allow the feature vectors corresponding to the backdoor patterns to form multiple clusters.

For the detection inference step, we leverage the fact that additive perturbation backdoor patterns with small perturbation size (for the imperceptibility of the pattern) can be easily destroyed by blurring (e.g. by applying an average filter). However, features recognizable to a well-trained DNN are usually robust to natural/manual blurring. Thus, a clean image that is blurred will likely still be correctly classified; while an image with an additive perturbation backdoor pattern may be misclassified (likely to its source class other than the labeled target class) when the blurring filter destroys the embedded backdoor pattern. This property is leveraged by our CI method to determine, for each class, which cluster (obtained from the clustering step) contains mainly backdoor patterns.

4.3.2 Method of CI

Our CI defense, like SS and AC, first trains a DNN on the possibly poisoned training set. Then, for each putative backdoor target class $t \in \mathcal{Y}$, the d-dimensional penultimate feature vector for each training image is extracted – the set of feature vectors for class t

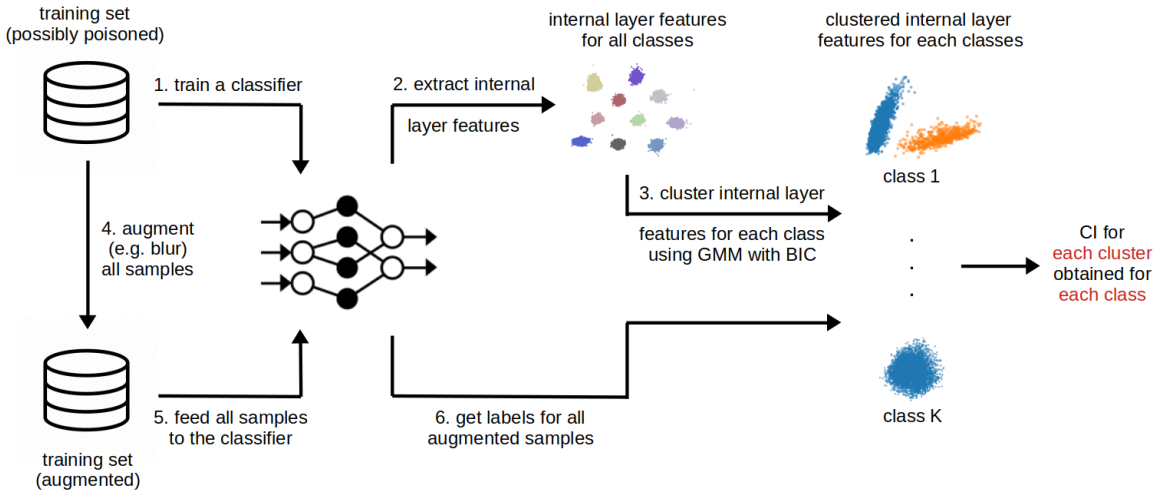


Figure 4.1: Illustration of the CI defense.

is denoted by $\mathcal{Z}_t \subset \mathbb{R}^d$. The optimal number of clusters for each class t is then obtained by modeling a Gaussian mixture model (GMM) with BIC criterion, by solving:

$$M_t = \arg\min_{M} \min_{\{\alpha_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}} -\sum_{\mathbf{z} \in \mathcal{Z}_t} \log \sum_{j=1}^{M} \alpha_j g(\mathbf{z}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) + \frac{d^2 + 3d + 2}{4} M \log |\mathcal{Z}_t| \tag{4.1}$$

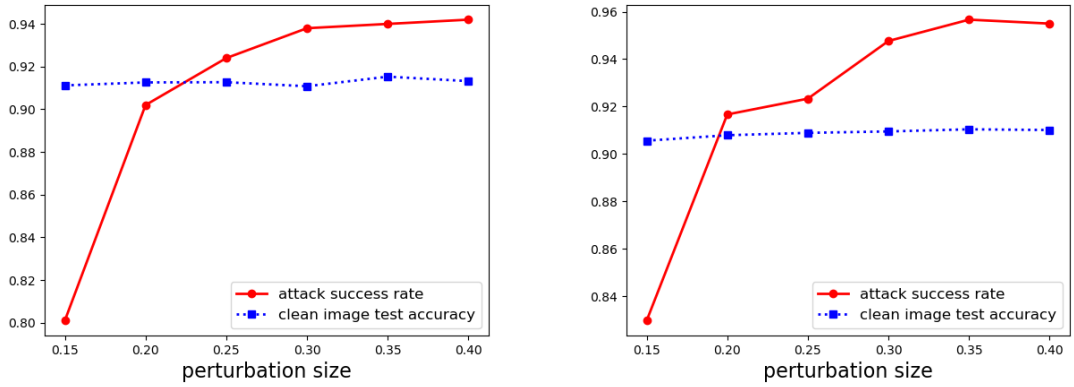
where $g(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density function with mean $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$, and $\{\alpha_j\}$ are the component masses. Note that here we have overloaded some of the notations from previous chapters.

To infer for each of the M_t clusters whether it corresponds to backdoor training images, we develop the "cluster impurity" (CI) statistic as follows. For each class t, we first hard assign all training samples labeled to class t to one of the M_t clusters by a maximum a posterior (MAP) rule based on the GMM's mixture posterior. Then for the training images in each cluster, we apply a blurring filter (e.g. an averaging filter) $h(\cdot) : \mathcal{X} \to \mathcal{X}$. For convenient, we consider a cluster of training samples (labeled to class t) denoted by $\mathcal{W}$. We define a probability $p \in [0, 1]$ by

$$p = \mathrm{P}(f(h(\mathbf{x})) = t | f(\mathbf{x}) = t), \quad \forall \mathbf{x} \in \mathcal{W}. \tag{4.2}$$

Then, the cluster impurity for this cluster $\mathcal{W}$ is defined by

$$\mathrm{CI}(\mathcal{W}) = D_{\mathrm{KL}}([1, 0]^T || [p, 1 - p]^T), \tag{4.3}$$



(a) single-source attacks (b) multiple-source attacks

Figure 4.2: Attack success rate (ASR) and clean test accuracy (ACC) for a range of perturbation sizes for (a) the single-source attacks, and (b) the multiple-source attacks.

where $D_{\mathrm{KL}}(\cdot||\cdot)$ represents the Kullback-Leibler divergence. Again, for clean clusters, the blurring largely produces no decision changes. But for poisoned clusters, blurring changes many decisions to the source class. So we expect higher CI measure for poisoned clusters than for clean clusters. An easily-selected threshold (shown in Sec. 4.3.3.2) is then used to detect whether backdoor patterns are embedded, with a decision made cluster by cluster. The procedure of our CI method is summarized in Fig. 4.1.

4.3.3 Experiments

4.3.3.1 Devising Backdoor Attacks

We test CI in comparison with SS and AC on CIFAR-10 [174]. We consider backdoor attacks with a stealthy single-pixel perturbation pattern. Like our experiments in previous chapters, the pixel being perturbed is randomly selected and fixed for all the samples used for training.

We consider both single-source backdoor attacks and multiple-source backdoor attacks (where we the backdoor training samples may form more than one cluster in the clustering step). For the single-source attacks, the source class and target class are 'airplane' and 'bird', respectively. The number of backdoor training images is fixed at 1000 and the perturbation size (added to each (R, G, B) channel) is varied. For the multiple-source attacks, the target class is still 'bird', while there are three source classes, which are 'airplane', 'automobile' and 'horse'. For each multi-source attack, we create 1000 backdoor training images from each source class. In our experiments here, we adopt ResNet-18 as

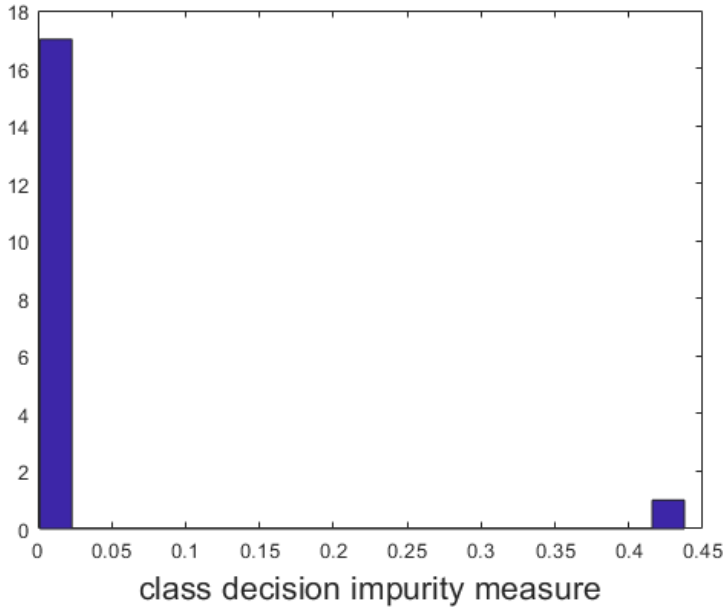


Figure 4.3: Histogram of the class decision impurity measure, i.e. the CI statistic, over the 18 clusters estimated using BIC for perturbation size 0.25.

the DNN architecture and training is performed for 200 epochs with batch size 32, using the Adam optimizer with learning rate 10^{-3}. Here, we consider a relatively compact ResNet architecture with the penultimate layer feature dimension being 64. In Sec. 4.4.6, we will show that CI fails when the DNN architecture is wide, with much larger dimension in the internal layer feature spaces for clustering. In absence of attack, this training configuration yields an accuracy $\sim 91\%$ on the test set of CIFAR-10. The attack success rate (ASR) and clean test accuracy (ACC) under attack with different perturbation sizes for the single-source attacks and multiple-source attacks are shown in Fig. 4.2a and Fig. 4.2b, respectively. The ASRs are quite high, even for the weakest (0.15) perturbation, with almost no degradation in ACC.

4.3.3.2 Defense Performance Evaluation

We compare the performance of our CI with SS [117] and AC [118]. Our CI defense requires the specification of the type and size of the blurring filter, and the threshold on the cluster decision impurity measure (i.e. the CI statistic) to trigger the alarm and remove all training samples in the detected cluster. Since the resolution of the images

being experimented is low, we use a relatively small 2×2 averaging filter.[1] To decide the detection threshold, we first show an example histogram (in Fig. 4.3) for the class decision impurity measure for the 18 clusters (across all originating classes) selected using BIC, with perturbation size 0.25 (similar histograms are obtained for other perturbation sizes). Note the clear, large gap between low and high impurity clusters (with the single, high impurity cluster from the attacked class). Thus, we set the impurity threshold at 0.2 for the CI approach in all our experiments. Here we state without showing the histograms that the same threshold works perfectly if a 3×3 averaging filter is used instead.

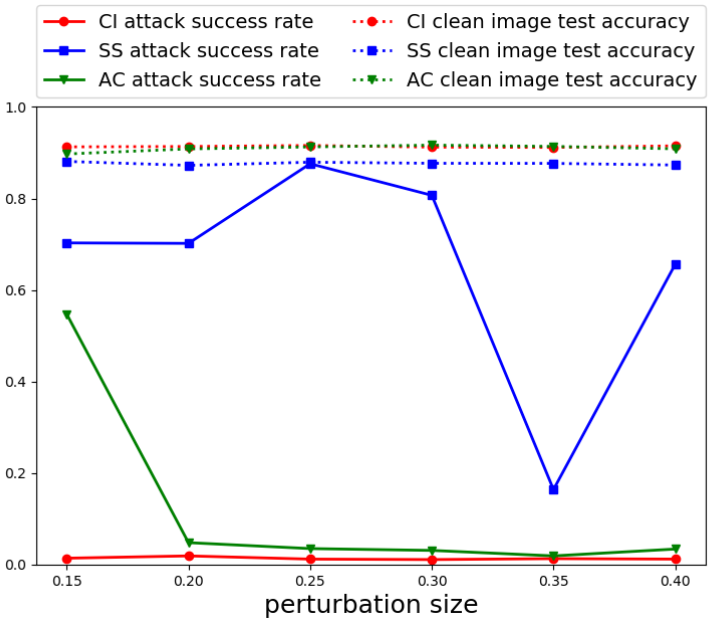
SS, as pointed out in Sec. 4.2, relies on knowledge of the number of backdoor training samples, unlikely to be known in practice. Moreover, no practical way is provided to explicitly infer whether a class is backdoor-poisoned or not. For convenience of evaluation, we fixed the FPR of SS to 0.5 by applying the same detection threshold to all classes and then evaluated the TPR. Note that FPR $= 0.5$ means half of the unpoisoned training patterns will be falsely detected and removed. This very high FPR is necessary in order for SS to achieve meaningful TPR (detecting some of the backdoor patterns) in our single-pixel-perturbation attack scenario.

Regarding AC, we present the "best possible" results. From the class that has actually been attacked (which is unknown to the defender a priori in practice), we remove the cluster (for the two clusters obtained by 2-means) that actually possesses the most backdoor training samples. That is, we assume there are zero false positive detections associated with unattacked classes (even though this is highly optimistic) and that AC unfailingly picks the correct cluster to detect and remove.

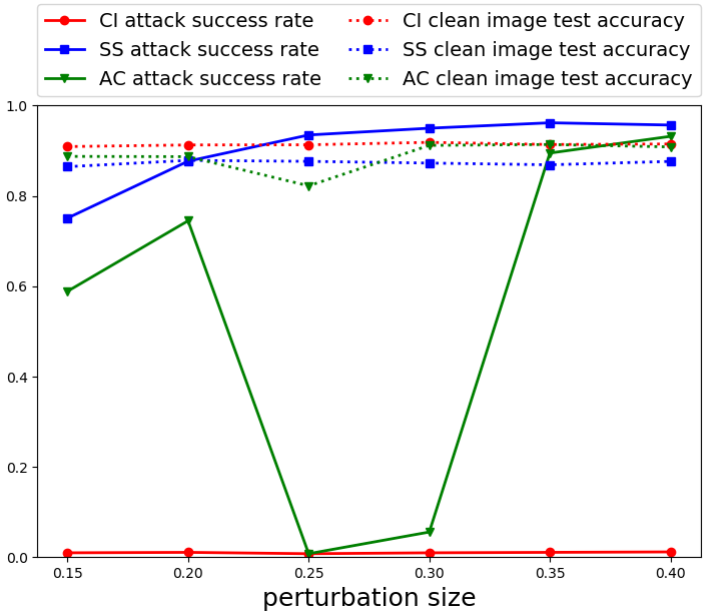
The performances of these defenses, in the presence of the attacks, are compared in two steps. First, we record the true positive rate (TPR) and the false positive rate (FPR) for each defense (assessing how accurately it removes training samples as putative backdoor training samples) and compare the detection effectiveness. Second, with the detected backdoor training samples removed and the classifier retrained, the *ultimate* performance is assessed based on the ASR and ACC of the retrained classifier.

For the single-source class attacks, in Tab. 4.1, we show the (TPR, FPR) pairs of the three detection approaches for each perturbation size. In general, CI achieves much higher TPR (greater than 0.96) with very low FPR (lower than 0.005) than the other two approaches. Under the most preferred assumptions, AC achieves relatively good TPR for

[1] Alternatively, we could seek to remove the backdoor pattern by globally adding noise to the image, an approach which is invariant to the spatial support of the backdoor patterns.



(a) single-source attacks



(b) multiple-source attacks

Figure 4.4: Attack success rate (ASR) and clean test accuracy (ACC) for a range of perturbation sizes for the retrained classifiers for (a) the single-source attacks, and (b) the multiple-source attacks.

Table 4.1: (TPR, FPR) for the range of perturbation sizes for the single-source attack scenario.

Pert. Size	SS	AC	CI
0.15:	(0.443, 0.5)	(0.622, 0.057)	(0.962, 0.002)
0.20:	(0.483, 0.5)	(0.691, 0.040)	(0.980, 0.003)
0.25:	(0.474, 0.5)	(0.763, 0.040)	(0.979, 0.003)
0.30:	(0.558, 0.5)	(0.741, 0.037)	(0.976, 0.003)
0.35:	(0.666, 0.5)	(0.851, 0.034)	(0.989, 0.002)
0.40:	(0.616, 0.5)	(0.847, 0.036)	(0.985, 0.004)

large perturbation size. However, the TPR reduces clearly as the perturbation is made smaller. The TPR of the SS approach shares the same trend, but it is apparently not comparable to the other two approaches. As for the FPR, we only analyze for the AC and CI approaches since it is fixed for the SS approach. Although the FPRs of the AC approach are not high, this is optimistic since we only consider the class being attacked. Since we have assumed that only one cluster from this class is removed, a 0.036 FPR means that roughly 1800 out of the 5000 clean patterns from the attacked class are falsely detected and discarded. This will be amplified and result in much higher overall FPR if AC is applied separately to each class, not just the (optimistically) known class under attack. In comparison, the FPR of the CI approach is truly low as no more than 5% of clean patterns are removed from any single class, and with all classes considered.

Table 4.2: (TPR, FPR) for the range of perturbation sizes for the multiple-source attack scenario.

Pert. Size	SS	AC	CI
0.15:	(0.417, 0.5)	(0.570, 0.061)	(0.976, 0.003)
0.20:	(0.436, 0.5)	(0.579, 0.041)	(0.985, 0.001)
0.25:	(0.496, 0.5)	(0.673, 0.100)	(0.991, 0.001)
0.30:	(0.428, 0.5)	(0.867, 0.001)	(0.995, 0.005)
0.35:	(0.588, 0.5)	(0.829, 0.001)	(0.992, 0.003)
0.40:	(0.391, 0.5)	(0.636, 0.036)	(0.984, 0.001)

In Fig. 4.4a, we show the ASR and ACC of the retrained classifiers for the three defenses. The CI approach reduces the ASR to a negligible level, and the ACC is not degraded at all. The AC approach is relatively successful, but its performance at low perturbation size is poor. The SS approach clearly fails to defeat the attack. Moreover,

there is some degradation in clean test set accuracy even though training samples are abundant for CIFAR-10 (5000 training samples per class). If we allow an even higher FPR to achieve a better TPR for SS, the retrained classifier will suffer even more degradation in ACC.

In Tab. 4.2, we show the (TPR, FPR) pairs, and in Fig. 4.4b, we show the ASR and ACC of the retrained classifiers for the three defenses for the multiple-source attack scenario. In general, the resulting (TPR, FPR) pairs are similar to those for the single-source scenario. After retraining, the CI defense still keeps a close-to-zero attack success rate and an unaffected clean test set accuracy. The SS defense fails again with degradation in the ACC. The AC defense is successful at perturbation size 0.3, but fails for other perturbation sizes. Note that for perturbation size 0.25, the AC defense indeed disables the backdoor trigger, but it also causes a nearly ten percent degradation in the ACC.

In summary of these experimental results, the clustering of CI yields pure backdoor clusters most of the time. One reason may be its use of all the penultimate layer features – some "minor component" features may contain important information for discriminating clean from poisoned training samples. Also, using BIC to select the number of clusters for each class instead of performing K-means (with $K = 2$) allows multi-modality for clean training samples and multi-sourcing for backdoor patterns.

4.4 DT-RED: Reverse-Engineering Additive Perturbation Patterns for Backdoor Detection and Training Set Cleansing

4.4.1 Overview of DT-RED

The effectiveness of before/during training backdoor defenses based on clustering, e.g. SS, AC, and CI, largely depends on the effectiveness of clustering. For SS and AC, the internal layer features of the training samples labeled to each class are projected onto low dimensional spaces. However, there is no guarantee that backdoor training samples will be separable from clean training samples labeled to the backdoor target class. The separability is even poorer when the architecture of the DNN is compact (see Sec. 4.4.6.2 for illustration), *i.e.*, with just enough representation power for the benign samples. For CI, if the classifier's architecture is wide, especially when the extracted internal feature dimension is overly high, Guassian mixture modeling with the Bayesian Information

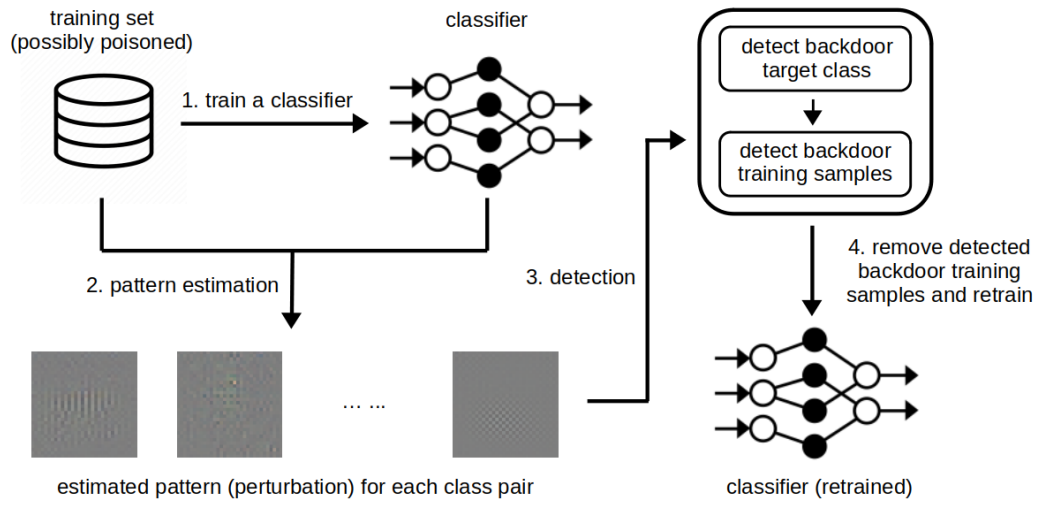


Figure 4.5: Illustration of DT-RED.

Criterion (see Sec. 4.3.2 for details) will likely yield only one cluster irrespective of the presence of attack [185]. Thus, we propose DT-RED (a RED deployed "during training"), inspired by our basic RED for post-training backdoor defense, which does not rely on clustering. Moreover, compared with SS, AC, and CI which require a detection threshold, the detection inference of DT-RED is purely unsupervised, which matches the challenging scenario where the defender does not possess a held out clean dataset.

Like SS, AC and CI, DT-RED first trains a classifier on the possibly poisoned training set. Like other REDs, DT-RED contains a backdoor pattern reverse-engineering step followed by a detection inference step. To also serve the goals for before/during training backdoor defense, there is a training set cleansing step when an attack is detected. The outline of DT-RED is shown in Fig. 4.5.

4.4.2 Key Ideas of DT-RED

4.4.2.1 For Detection

Suppose $f : \mathcal{X} \to \mathcal{Y}$ is the classifier trained on the given training set $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{backdoor}}$ which may be poisoned. If there is a successful attack, i.e. $\mathcal{D}_{\text{backdoor}} \neq \emptyset$ and the trained classifier has learned the backdoor mapping, and if a class pair $(s, t) \in \mathcal{Y} \times \mathcal{Y}$ is a backdoor class pair, a large fraction of training images from class s will be misclassified to class t when the backdoor pattern is embedded. The fundamental premise here, leveraging the insights of our basic RED in Sec. 3.3 is that, for a true backdoor class pair (s, t), high

misclassifications can be induced from s to t using a common, *small* image perturbation (which may not be the exact backdoor pattern, but contain the "key features" of the backdoor pattern); however, for any non-backdoor class pair (s, t), to achieve a similarly high misclassification rate from s to t, the minimum required perturbation norm will be very large – likely much larger than the norm of the additive perturbation pattern used by the attacker.

4.4.2.2 For Training Set Cleansing

It is straightforward that sequentially embedding a perturbation $\mathbf{v}$ and then $-\mathbf{v}$ to an image $\mathbf{x}$ will induce limited changes to the image, i.e.,

$$||m_{\text{add}}(m_{\text{add}}(\mathbf{x}; \mathbf{v}); -\mathbf{v}) - \mathbf{x}||_\infty \leq ||\mathbf{v}||_\infty, \tag{4.4}$$

where the infinity norm selects the maximum magnitude element in a vector. Moreover, if the clipping function is not activated, the left hand side of (4.4) will be zero – *no* difference is induced in this case. Hence, if there is an attack with source classes $\mathcal{S}^*$, target class t^* and backdoor pattern $\mathbf{v}^*$, for any $(\mathbf{x}, y) \in \mathcal{D}_{\text{backdoor}}$ where $\mathbf{x} = m_{\text{add}}(\tilde{\mathbf{x}}; \mathbf{v}^*)$ for some $\tilde{\mathbf{x}} \sim P_{\mathcal{S}^*}$ and $y = t^*$, we have with high probability that

$$f(m_{\text{add}}(\mathbf{x}; -\mathbf{v}^*)) = f(\tilde{\mathbf{x}}) \in \mathcal{S}^* \tag{4.5}$$

i.e., the true class of the originally clean version of a backdoor training image can be recovered by *removing* the embedded backdoor pattern. However, for *clean* training images labeled to class t^*, "embedding" the pattern $-\mathbf{v}^*$, or any arbitrary pattern with similarly small norm, will likely not change the label predicted by f. Thus, we can subtract an estimation of the backdoor pattern $\mathbf{v}^*$ from $\mathbf{x}$ and identify $\mathbf{x}$ as a poisoned training sample if this operation results in a change in the classifier's decision.

4.4.3 Pattern Estimation of DT-RED

Based on the key premise of our detection, when there is successful backdoor data poisoning, the existence of a small-sized perturbation that induces high misclassification from any $s \in \mathcal{S}^*$ to t^* is guaranteed by the existence of the backdoor pattern $\mathbf{v}^*$. This motivates our search for such a small-sized perturbation, i.e. reverse engineering the backdoor pattern, for the backdoor class pair(s).

To serve both *detection* and *training set cleansing* purposes, for any class pair

$(s,t) \in \mathcal{Y} \times \mathcal{Y}$ and $s \neq t$, under the hypothesis that (s,t) is a backdoor class pair (i.e. $s \in \mathcal{S}^*$ and $t = t^*$), we expect the following to hold:
1) High misclassifications from s to t are induced by a well-chosen small perturbation.
2) The DNN's class decision for a backdoor training image should change from t when this same perturbation is "removed" from the image.
3) It is not possible for any small perturbation to change the class decision of a significant proportion of the *clean* training images labeled to t, when the perturbation is "removed" from these images.
Hence we search for a limited-sized perturbation that induces at least $\pi \in (0,1)$ misclassification fraction from s to t (on the training images labeled to s) and, at the same time, maximizes the class decision changes for the training images labeled to t when this pattern is "removed" from them, i.e.

$$\begin{aligned} \underset{\mathbf{v}}{\text{maximize}} \quad & \frac{1}{|\mathcal{D}_t|} \sum_{(\mathbf{x},y)\in\mathcal{D}_t} \mathbb{1}(f(m_{\text{add}}(\mathbf{x};-\mathbf{v})) \neq t) \\ \text{subject to} \quad & \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{x},y)\in\mathcal{D}_s} \mathbb{1}(f(m_{\text{add}}(\mathbf{x};\mathbf{v})) = t) \;\geq\; \pi, \\ & d(\mathbf{v}) \leq \Delta_{st} \end{aligned} \tag{4.6}$$

Here, $d(\cdot)$ is any reasonable metric representing the perturbation size (e.g. l_2 norm as for our basic RED), $\mathcal{D}_s$ and $\mathcal{D}_t$ represent the subsets of the training set labeled to class s and class t, respectively. Note that we have abused these notations here, where in Chap. 3, $\mathcal{D}_c$ represents the clean samples from class $c \in \mathcal{Y}$ possessed by the post-training defender for detection purpose. Also note that classifier f here is trained on the possibly poisoned training set, which may not be the finalized classifier provided to the downstream users. This is different with the post-training pattern estimation formulation where f represents the trained classifier to be inspected. Δ_{st} constrains the perturbation size (with reference to metric $d(\cdot)$) to be small.

Since Δ_{st} and the feasible set of $\mathbf{v}$ for solving (4.6) cannot be easily specified, we use a gradient-based search algorithm. In principle, consistent with the above goals, we seek to maximize the Lagrangian objective:

$$\begin{aligned} \underset{\mathbf{v}}{\text{maximize}} \quad & \frac{1}{|\mathcal{D}_t|} \sum_{(\mathbf{x},y)\in\mathcal{D}_t} \mathbb{1}(f(m_{\text{add}}(\mathbf{x};-\mathbf{v})) \neq t) \\ & + \lambda \times \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{x},y)\in\mathcal{D}_s} \mathbb{1}(f(m_{\text{add}}(\mathbf{x};\mathbf{v})) = t). \end{aligned} \tag{4.7}$$

However, since the indicator function $\mathbb{1}(\cdot)$ in (4.7) is not differentiable, we adopt the following differentiable surrogate objective function

$$\begin{aligned} L_{\mathrm{DT-RED}}(\mathbf{v},(s,t)) = &-\frac{1}{|\mathcal{D}_t|}\sum_{(\mathbf{x},y)\in\mathcal{D}_t}\log(1-p(t|m_{\mathrm{add}}(\mathbf{x};-\mathbf{v}))) \\ &-\lambda\times\frac{1}{|\mathcal{D}_s|}\sum_{(\mathbf{x},y)\in\mathcal{D}_s}\log(p(t|m_{\mathrm{add}}(\mathbf{x};\mathbf{v}))) \end{aligned} \tag{4.8}$$

Here, we take the logarithm to make the objective function smoother. Like for our basic RED in Sec. 3.3, we search for $\mathbf{v}$ along the gradient of (4.8) with a *relatively small* step size δ, starting from $\mathbf{v}=\mathbf{0}$. The step size δ can also be carefully registered in an adaptive way (e.g. using line search) to ensure a smooth increment in the misclassification fraction from s to t. Note that the choice of δ does not require any prior knowledge about whether the class pair (s,t) is a backdoor class pair or not. The searching is stopped immediately once $\mathbf{v}$ successfully induces at least π fraction of misclassifications from s to t on all the training images labeled to s. This termination condition, paired with a small δ, limits the size of the perturbation. Thus, the resulting perturbation is expected to satisfy both constraints of (4.6), i.e. inducing at least π fraction of misclassifications from s to t while being a *small* perturbation. Our pattern estimation method is detailed on lines 3-7 of Alg, 4. It is applied to all class pairs independently and we denote the estimated pattern for any (s,t) pair as $\hat{\mathbf{v}}_{st}$.

Our pattern estimation algorithm does require choosing the target misclassification fraction π and the Lagrange multiplier λ. In principle, like for our basic RED, π should be large, i.e. close to 1, and λ should keep the balance between the two terms of Eq. (4.7), i.e. a default value of $\lambda=1$. However, as will be shown in Sec. 4.4.6.3, our defense performance is not very sensitive to these choices.

Also note that the choice of the surrogate objective function to Eq. (4.7) is not unique. Although we choose to use Eq. (4.8), one may also consider to replace the first term of Eq. (4.8) e.g. with an objective similar to that associated with the Perceptron algorithm [58]. Moreover, for each updating of $\mathbf{v}$ in practice, it is suggested to compute the gradient of Eq. (4.8) using two subsets of images randomly sampled from $\mathcal{D}_s$ and $\mathcal{D}_t$ respectively, especially when the training set is large. The implementation details for our pattern estimation will be described in our experiments in Sec. 4.4.6.2.

Algorithm 4 DT-RED: Reverse-engineering backdoor pattern for detection and training set cleansing.

Inputs: labeled training set $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ which may be poisoned, classifier $f : \mathcal{X} \to \mathcal{Y}$ trained on the training set, target misclassification fraction π, detection threshold θ
Pattern estimation:
for all (s, t) class pair **do**
    $\mathbf{v} \leftarrow \mathbf{0}$, $\rho \leftarrow \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{x},y)\in\mathcal{D}_s} \mathbb{1}(f(\mathbf{x}) = t)$
    while $\rho < \pi$ **do**
        $\mathbf{v} \leftarrow \mathbf{v} - \delta \cdot \nabla L_{\mathrm{DT-RED}}(\mathbf{v}, (s, t))$
        $\rho \leftarrow \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{x},y)\in\mathcal{D}_s} \mathbb{1}(f(m_{\mathrm{add}}(\mathbf{x}; \mathbf{v})) = t)$
    $\hat{\mathbf{v}}_{st} = \mathbf{v}$, $r_{st} = d(\hat{\mathbf{v}}_{st})^{-1}$
Detection inference:
for all classes c **do**
    get G_c by MLE using as data all r_{st} with $t \neq c$
    evaluate pv_c by Eq. (3.7)
evaluate pv by Eq. (3.8)
if pv $> \theta$ **then**
    there is no attack; defense terminated
else
    backdoor attack detected; $\hat{t} = \arg\min_c \mathrm{pv}_c$ is the target class; $\hat{s} = \arg\max_s r_{s\hat{t}}$ is one of the source classes; $\hat{\mathbf{v}} = \hat{\mathbf{v}}_{\hat{s}\hat{t}}$ is the estimation of $\mathbf{v}^*$
Training set cleansing:
for all $(\mathbf{x}, y) \in \mathcal{D}_{\hat{t}}$ **do**
    if $f(m_{\mathrm{add}}(\mathbf{x}; -\hat{\mathbf{v}})) \neq \hat{t}$ **then**
        remove $(\mathbf{x}, y)$ from the training set (or replace it by the cleansed example $(m_{\mathrm{add}}(\mathbf{x}; -\hat{\mathbf{v}}), f(m_{\mathrm{add}}(\mathbf{x}; -\hat{\mathbf{v}}))))$.
Proceed to retraining if there are samples being removed.

4.4.4 Detection Inference of DT-RED

The main goal of detection inference for DT-RED is to infer whether the training set is poisoned or not, which is different with post-training detectors which infer whether the pre-trained classifier is backdoor attacked or not. However, here, we use the same detection inference method for our basic-RED in Sec. 3.3.3.2. That is, we evaluate the order statistic p-value for the largest reciprocal statistics among all $\{||\hat{\mathbf{v}}_{st}||_2^{-1}\}$ $((s, t) \in \mathcal{Y} \times \mathcal{Y})$. If there is an attack being detected, we estimate the backdoor source class $\hat{s}$, backdoor target class $\hat{t}$ and the backdoor pattern $\hat{\mathbf{v}}$ in the same way as described in Sec. 3.3.3.2.

4.4.5 Training Set Cleansing of DT-RED

In our pattern estimation, for each candidate backdoor pair (s, t), we estimate the perturbation so as to induce high misclassifications from s to t when the perturbation is added to training images from class s *and*, at the same time, to induce as many decision changes as possible (from class t to some other class) when the perturbation is *subtracted* from the training images labeled to class t. Essentially, every training image labeled to t that undergoes decision change when the perturbation is removed is being hypothesized to be a backdoor-poisoned sample. Thus, this "removal" operation, which is built into our pattern estimation procedure, is *also* the operation we use to cleanse the training set following detection of a backdoor. More specifically, suppose an attack is detected, with estimate (source, target) class pair $(\hat{s}, \hat{t})$. We then identify the backdoor training images from $\mathcal{D}_{\hat{t}}$ as those $\mathbf{x}$ such that

$$f(m_{\text{add}}(\mathbf{x}; -\hat{\mathbf{v}})) \neq \hat{t}. \tag{4.9}$$

These images are removed before retraining (or are *cleansed*, as specified in Alg. 4, with these cleansed examples then retained in the training set). The entire procedure of DT-RED is summarized in Alg. 4.

4.4.6 Experiments

4.4.6.1 Devising Backdoor Attacks

Dataset: Like in Sec. 3.3.6.1, the main experiments uses the CIFAR-10 data set [174]. Following the same protocol as in Sec. 3.3.6.1, the data set is separated into a training set with 50k images (5k per class) and a test set with 10k images (1k per class). Again, the 10 classes of CIFAR-10 are denoted as class 1-10 for convenience.

Backdoor pattern: We consider seven additive perturbation backdoor patterns (shown in Fig. 4.6) that have appeared in the backdoor literature, including two "global" patterns (pattern A, B) and five "local patterns" (pattern C-G). These backdoor patterns are also considered in previous chapters. Pattern A is the "chessboard" pattern introduced in Sec. 3.3.6.1. Pattern B is the pattern used by [129], which is also considered in Sec. 3.6.4.1. Pattern C, D, F are "cross", "square", and "L" patterns introduced in Sec. 3.7.6.1. Pattern E is the "four-pixel" pattern introduced in Sec. 3.3.6.1. Pattern G is the "single-pixel" pattern considered in both Sec. 3.6.4.1 and Sec. 4.3.3.1.

Attack configurations: For each backdoor pattern, we create backdoor attacks with one source class, three source classes, and nine source classes (i.e. all classes except

(a) pattern A (b) pattern B (c) pattern C (d) pattern D

(e) pattern E (f) pattern F (g) pattern G

Figure 4.6: Illustration of the backdoor patterns (with a 127/255 offset to manifest possible negative perturbations).

Table 4.3: Choices of the source class(es) $\mathcal{S}^*$ and the target class t^* for the 21 attacks (1SC, 3SC, and 9SC attack for pattern A-G).

	t^*	$\mathcal{S}^*$ of 1SC	$\mathcal{S}^*$ of 3SC	$\mathcal{S}^*$ of 9SC
A	10	2	2, 5, 8	except 10
B	8	2	2, 3, 10	except 8
C	10	2	5, 7, 8	except 10
D	4	9	8, 9, 10	except 4
E	7	4	2, 4, 6	except 7
F	9	4	4, 5, 6	except 9
G	3	1	1, 4, 8	except 3

for the target class are the source classes), respectively. For convenience, these three attacks are named "1SC", "3SC", and "9SC", respectively. For each backdoor pattern, the three associated attacks use the same target class, which is arbitrarily chosen. The choice of the source classes for each attack is also arbitrary. These choices are summarized in Tab. 4.3.

For each backdoor pattern, 500, 200, and 60 clean images *per source class* are used to create backdoor training images for the 1SC, 3SC, and 9SC attacks, respectively. Backdoor poisoning in the experiments here is conducted by *replacing* these clean images in the training set by the backdoor images created using them. We do so to simulate the practical scenario where the sources for data collection (including a possible attacker) independently contribute training images to the training authority. If all the sources are benign, the training authority obtains the clean CIFAR-10 training set. If any source

is an attacker (unbeknownst to the training authority) the subset of training images contributed by this source now contains a backdoor pattern.

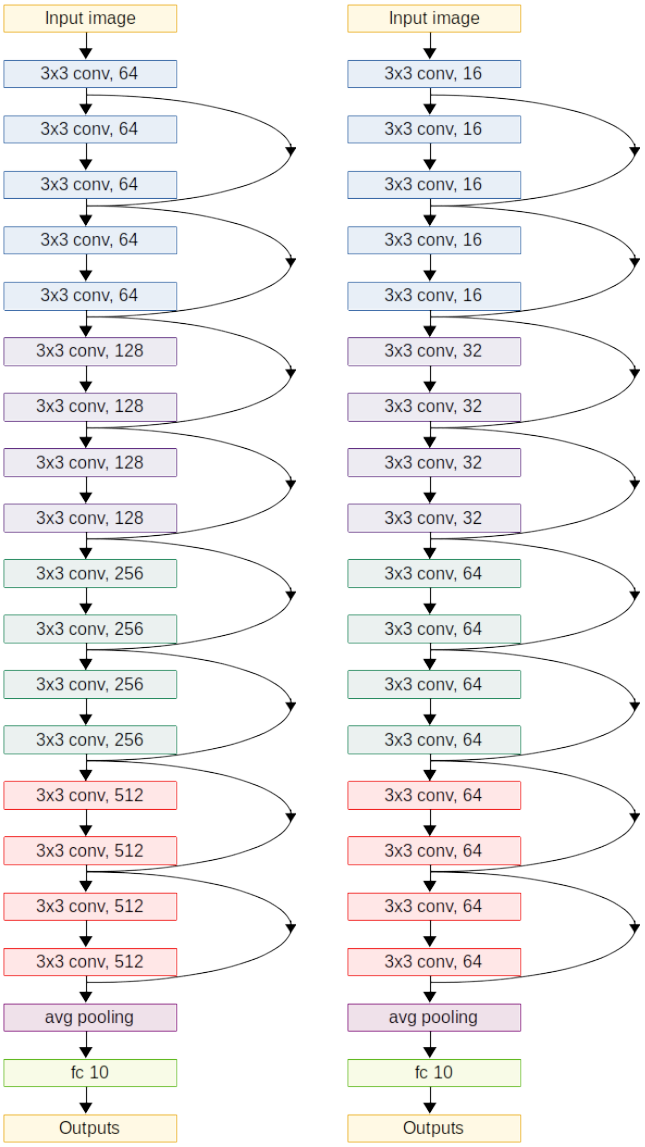
Note that our experiment settings cover a broader range of cases than for other backdoor defenses deployed during the training phase. For example, [117] only considered attacks with a single source class, which is our 1SC attack case. Even our CI defense only considered attacks with one source class and three source classes. Our experiments cover all these cases, with a large variety of backdoor patterns involved, including those considered by [117] and in Sec. 4.3.3 for CI.

Training configurations: We consider two slightly different architectures from the standard ResNet model family which is perhaps the most popular choice of DNN architecture for image classification [14]. A standard ResNet architecture is featured by *four groups* of residual blocks (see Tab. 1 of [14]). A ("non-bottleneck") residual block consists of two identical, consecutive convolutional layers (i.e. two sets of identical convolutional filters) in parallel with an identity map (see Fig. 2 of [14]). The two architectures considered here are both 18-layer ResNet with 2 residual blocks in each of the four groups of residual blocks. One is a wide architecture where there are 64, 128, 256, 512 filters in each convolutional layer for the four groups of residual blocks, respectively; while the other is a compact architecture with 16, 32, 64, 64 filters instead. The two architectures are visualized in Fig. 4.7.

Conceptually, the wide DNN architecture has a stronger representation power than the compact DNN architecture, but it also contains more parameters and requires more training resources and training time than the compact DNN architecture to achieve a high test accuracy. For a dataset containing a large number of classes, e.g. ImageNet [40] with 1000 classes, using a wide DNN structure can achieve a clearly higher test accuracy than using a compact DNN architecture. But for a dataset containing a relatively small number of classes, e.g. CIFAR-10, MNIST [15], SVHN [186], using a compact DNN architecture achieves sufficiently high test accuracy with much less training resources/time required. The architecture of the DNN is determined by the trainer, not by the attacker. We consider both wide and compact architectures for adequate evaluation of the performance of our defense in comparison with existing defenses.

For both DNN architectures, training is performed for 100 epochs, with batch size 32 and learning rate 0.001. Adam optimizer is used with decay rate 0.9 and 0.999 for the first and second moment, respectively [5]. Training data augmentation is used, including random cropping, random horizontal flipping, and random rotation.

In Tab. 4.4, we show that all 21 attacks (1SC, 3SC, and 9SC attacks for 7 backdoor



(a) wide (b) compact

Figure 4.7: The wide ResNet-18 architecture (a) and the compact ResNet-18 architecture (b) used in our main experiments to evaluate attack effectiveness against defenseless DNNs and to evaluate the performance of the proposed DT-RED defense.

patterns) crafted for evaluating the performance of defenses are successful against defenseless DNNs for both the wide and the compact architectures. For each backdoor pattern and each DNN architecture, we also train a DNN on the clean, unpoisoned CIFAR-10 training set and obtain a benchmark accuracy on the clean test set (with no backdoor patterns). For all attacks, both of the attacker's goals – a high attack success rate (ASR) and a negligible degradation in clean test accuracy (ACC) – are achieved.

Table 4.4: Attack success rate (ASR) and poisoned classifier accuracy (ACC) on the clean test set (jointly represented by ASR/ACC) for each of the 21 attacks (1SC, 3SC, and 9SC attacks for pattern A-G) for defenseless DNNs, for both wide and compact architectures; test ACC of the clean benchmark DNNs is also shown (ASR is not applicable (represented by n.a.) to clean DNNs).

pattern		A	B	C	D	E	F	G
wide DNN	clean	n.a./92.2	n.a./91.8	n.a./91.9	n.a./91.7	n.a./92.3	n.a./91.3	n.a./92.2
	1SC	99.2/92.1	97.3/92.0	98.9/92.2	96.2/92.1	97.0/91.8	86.1/91.7	92.9/91.3
	3SC	99.5/91.6	98.5/92.0	99.3/91.8	99.5/91.8	99.9/92.1	94.2/90.8	97.1/92.2
	9SC	98.8/91.7	97.1/91.9	98.4/91.7	92.6/91.9	99.4/91.7	89.4/92.0	96.2/91.5
compact DNN	clean	n.a./90.4	n.a./90.7	n.a./91.3	n.a./91.2	n.a./90.4	n.a./90.8	n.a./90.5
	1SC	99.1/90.8	99.5/90.5	96.4/90.3	92.4/90.6	96.0/90.7	89.4/90.1	94.3/90.4
	3SC	99.3/90.1	91.0/90.9	98.1/90.2	99.5/90.4	99.6/90.6	90.5/89.8	96.9/90.7
	9SC	99.1/90.3	97.7/90.5	97.3/90.8	86.5/90.6	98.2/90.0	87.6/90.2	97.2/90.1

4.4.6.2 Defense Performance Evaluation

We evaluate the effectiveness of DT-RED in comparison with SS, AC, and CI on the 21 attacks for the two DNN architectures. For DT-RED, AC, and CI, we evaluate both the accuracy of backdoor detection and performance of training set cleansing. For SS, we only show the results for training set cleansing because SS sanitizes the training images for *all* classes, without any explicit inference of whether or not there is poisoning and without identifying which class has been poisoned. Finally, we show the ASR and ACC of the retrained classifiers for these four approaches.

Implementation details of DT-RED: As mentioned in Section 4.4.3, the step size δ for pattern estimation should be relatively small such that π fraction of group misclassification can be achieved with as small a perturbation size as possible. Here, we set $\delta = 10^{-4}$ for simplicity, though line search can be used to find an adaptive step size at each iteration of pattern estimation. Also, the target misclassification fraction π for pattern estimation should be relatively large in order to distinguish backdoor class pairs from non-backdoor class pairs. Here, we set $\pi = 0.95$, but its choice is not critical to our defense. Similarly, the choice of the Lagrange multiplier is not critical to our defense, and we use the default $\lambda = 1$ here. Robustness analysis for the choices of π and λ are provided in Sec. 4.4.6.3. Moreover, in each iteration of pattern estimation, we compute the gradient of (4.8) on two batches of training images (with batch size 256) randomly sampled from $\mathcal{D}_t$ and $\mathcal{D}_s$, respectively. This will largely reduce computational complexity and help avoid poor local optima. For detection inference, we use Gamma as the null density form; we use l_2 norm as the metric for perturbation size; and the

confidence-level threshold is set to the classical $\theta = 0.05$ – these choices are all the same as for our basic RED in Sec. 3.3.6. Finally, we consider both the wide and compact DNN architectures described in Sec. 4.4.6.1, and use the same training configuration as that used for the defenseless classifiers described in the same Section. But we *do not* use training data augmentation here – this ensures that the true backdoor pattern (but not the resized/shifted version) is well learned. Note that the training here is for detection, which is different from the training for evaluating the effectiveness of attacks (in Sec. 4.4.6.1) for experimental purpose.

Detection performance evaluation: As mentioned above, we evaluate the accuracy of backdoor detection for DT-RED, AC, and CI.

For AC, as discussed in Sec. 4.2, it obtains two clusters for each class using 2-means. One of the detection inference approaches proposed by AC is to use the Silhouette score for detection – an attack is detected if the Silhouette score obtained for any class is larger than a threshold. However, there is no automatic rule proposed for selecting the detection threshold [118]. Even for the relatively simple MNIST dataset, there is no threshold on the Silhouette score that leads to perfect detection, as shown in Tab. 3 of [118]. Moreover, we find that, when there is no poisoning, the maximum Silhouette score over all classes depends on the data domain, the DNN architecture, and most importantly, the dimension of the space that the extracted activations are projected onto. In [118], the authors set the projected feature dimension as 10, and claimed that a range of dimension choices work similarly well based on their experiments on relatively easy datasets like MNIST. However, for the CIFAR-10 domain and the wide ResNet architecture, we find that AC achieves almost perfect detection performance with a properly selected detection threshold (and impressive training set cleansing performance as are shown in Tab. 4.6a) when the projected feature dimension is set to 2. Here, we select the threshold on the Silhouette score as 0.4055 in a "supervised" fashion – this value is the maximum Silhouette score obtained for the 7 benchmark DNNs trained on the clean CIFAR-10 training set[2]. Choosing this value maximizes AC's accuracy in detecting poisoning while maintaining zero false detections when there is no poisoning. For the compact DNN architecture, we use the same projection feature dimension (10) as in [118], since other choices yield poor detection performance. The detection threshold is set to 0.0881 for the wide DNN architecture (again chosen in a supervised fashion, to ensure no false positive detections while maximizing detection power). As shown in Tab. 4.5b, for the wide DNN architecture, using the specified threshold, AC detects all attacks

[2]This knowledge will not be available to the defender in practice.

Table 4.5: Detection performance evaluation of (a) DT-RED defense, in comparison with (b) AC and (c) CI, on the 21 poisoned training sets and the clean training sets for both wide and compact DNN architectures. Symbols ⊗ and ⊙ represent: attack is not detected (or falsely detected for clean training set), and attack is detected with the target class correctly inferred (or no attack is detected for clean training set), respectively.

pattern		A	B	C	D	E	F	G
wide DNN	clean	⊙	⊙	⊙	⊙	⊗	⊙	⊙
	1SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	3SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	9SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
compact DNN	clean	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	1SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	3SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	9SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙

(a) DT-RED detection

pattern		A	B	C	D	E	F	G
wide DNN	clean	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	1SC	⊙	⊙	⊙	⊙	⊗	⊙	⊙
	3SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	9SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
compact DNN	clean	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	1SC	⊗	⊗	⊗	⊙	⊗	⊗	⊗
	3SC	⊗	⊗	⊗	⊙	⊗	⊗	⊙
	9SC	⊗	⊗	⊗	⊗	⊗	⊙	⊗

(b) AC detection

pattern		A	B	C	D	E	F	G
wide DNN	clean	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	1SC	⊗	⊗	⊗	⊗	⊗	⊗	⊗
	3SC	⊗	⊗	⊗	⊗	⊗	⊗	⊗
	9SC	⊗	⊗	⊗	⊗	⊗	⊗	⊗
compact DNN	clean	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	1SC	⊙	⊙	⊙	⊗	⊙	⊙	⊗
	3SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙
	9SC	⊙	⊙	⊙	⊙	⊙	⊙	⊙

(c) CI detection

except one. But for the compact DNN architecture, the detection performance of AC is poor. This is due to the fact that the dimension-reduced activations for clean training images and backdoor training images are not separable using K-means with $K = 2$.

CI also requires setting a detection threshold [122]. Here, in the same supervised fashion as for AC, we set the detection thresholds for the the wide and compact DNN architectures as 0.62 and 0.65, respectively. As shown in Tab. 4.5c, CI detects all attacks except two for the compact DNN architecture, but fails to detect any of the attacks when the wide DNN architecture is used. This is due to the fact that for the wide DNN architecture, CI estimates only one component by Gaussian mixture modeling with BIC, for all classes, including the true backdoor target class, since the dimension of the penultimate layer activation is too high.

Different from AC and CI, our DT-RED uses a statistical confidence threshold ($\theta = 0.05$); it does not require any careful, supervised hyperparameter tuning. As shown in Tab. 4.5a, DT-RED outperforms AC and CI by successfully detecting all attacks

regardless of the DNN architecture, with correct inference of the target class[3] for each attack, with only one false detection.

Training set cleansing performance evaluation: We compare the training set cleansing performance of DT-RED with SS, AC, and CI on the 21 attacks for the two DNN architectures.

According to Alg. 1 of [117], SS removes 1.5ϵ training images from *each* class, without considering whether there is an attack, where ϵ is the upper bound on the number of backdoor training images. The experiments of [117] simply set ϵ as the actual number of backdoor training images used by the attacker. However, in practice, whether there is an attack or not, the target class, and the number of backdoor training images are all unknown to the defender. Thus, our evaluation of training set cleansing performance for SS is somewhat optimistic or "best case" in that we provide SS with this knowledge – whether there is an attack, the target class of the attack, and the number of backdoor-poisoned images. This information was also exploited by SS in the experiments in [117]. In other words, we removed 1.5 times the actual number of backdoor training images from the true backdoor target class (with these images selected by the SS algorithm), without touching the other classes.

As shown in Tab. 4.5b, AC does not detect all 21 attacks (unlike our DT-RED). Especially, the detection performance for the compact DNN architecture is very poor. In practice, when there is no detection made, no images will be removed. In such a case, all the backdoor training images will remain if poisoning actually exists. Thus, in evaluating the training set cleansing performance for AC, similar to SS, we "assist" AC by assuming that it has detected all the attacks, with the target class correctly inferred. In this way, again, the AC results are somewhat optimistic or "best case" results. Following the same settings as in [118], for each attack, from the two clusters obtained for the true backdoor target class (using K-means with $K = 2$), we remove the one with the smaller mass (the same criterion as used in [118]).

For CI, training set cleansing is performed simultaneously with attack detection – all training images from a GMM component are removed if its "cluster impurity" measure exceeds the detection threshold. Since for all 21 attacks, CI estimates only one component for the true target class if the wide DNN architecture is used by the defender, even if the true target class is known to the defender, CI cannot identify or remove any backdoor training images. Hence, we do not show the failing results of CI for the wide DNN

[3]If the target class is incorrectly inferred, backdoor training images will not be removed. Moreover, clean training images will be falsely removed.

Table 4.6: Training set cleansing true positive rate (TPR) and false positive rate (FPR) of SS, AC, CI, and our DT-RED (represented in TPR/FPR form), for the 21 attacks, for (a) the wide DNN architecture, and (b) the compact DNN architecture. TPRs $\geq 90\%$ and FPRs $\leq 10\%$ are in bold.

pattern		A	B	C	D	E	F	G
1SC	SS	**98.0/5.2**	**100/5.0**	44.2/10.6	**97.4/5.3**	56.0/**9.4**	76.8/**7.3**	86.0/**6.4**
	AC	88.4/**0**	**98.8/0.0**	87.2/**0.5**	**95.2/0**	70.4/27.7	**93.6/0**	85.2/**0.1**
	RE	**94.8/8.4**	**97.2/0.3**	**95.6/8.6**	**92.8/2.8**	83.0/**0.2**	**98.6**/10.8	87.6/**0.3**
3SC	SS	**99.5/6.1**	**100/6.0**	66.2/10.1	**99.7/6.0**	**92.2/6.9**	84.8/**7.8**	79.2/**8.5**
	AC	**97.5/0**	**98.8/0**	**97.0/0**	**97.2/0**	**94.5/0**	**95.5/0.1**	87.7/**0.7**
	RE	**98.0**/12.9	**98.7/1.6**	**99.2/6.2**	**98.2/0**	**90.3/0**	**98.5/2.8**	**92.7/0.1**
9SC	SS	**98.3/5.6**	**100/5.4**	89.1/**6.6**	**96.1/5.8**	**97.8/5.6**	**90.9/6.4**	**94.8/6.0**
	AC	**97.0/0**	**98.9/0**	**96.5/0**	**91.1/0.1**	**96.7/0**	**95.9/0**	89.6/**0**
	RE	**96.1/4.2**	**98.7/7.9**	**99.3/0.4**	**94.1/0**	88.7/**0**	**99.1/5.1**	**94.3/0**

(a) wide DNN architecture

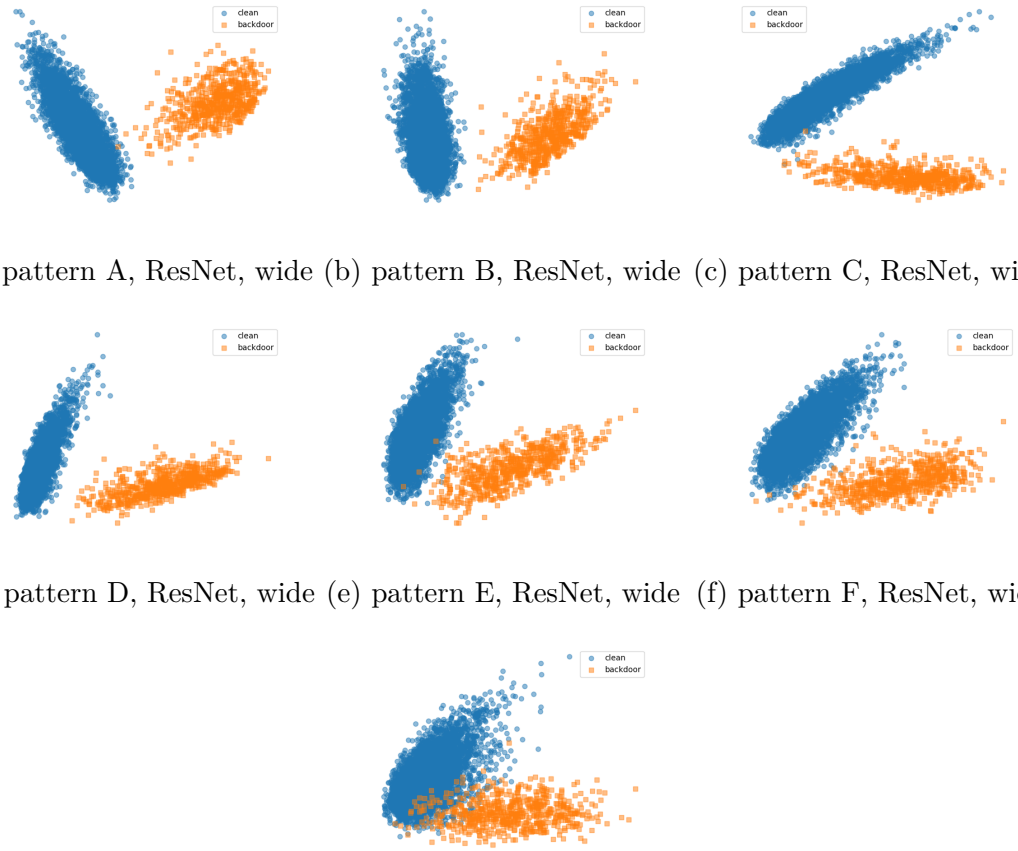
pattern		A	B	C	D	E	F	G
1SC	SS	23.6/12.6	**96.8/5.3**	39.6/11.0	69.2/**8.1**	21.2/12.9	49.2/10.1	18.2/13.2
	AC	36.4/47.0	82.6/38.8	72.2/40.5	**92.6**/25.9	36.4/41.5	**95.4**/40.9	**93.0**/37.5
	CI	55.8/**7.5**	**99.8/0**	**93.8**/13.1	n.a./n.a.	**96.2**/55.6	**100**/8.4	n.a./n.a.
	RE	**93.6**/20.9	**100/9.5**	**91.0**/10.6	**98.8**/12.2	87.8/**5.1**	**98.4**/16.7	**94.4**/14.7
3SC	SS	35.5/13.7	29.5/14.5	14.7/16.2	85/**7.8**	53.8/11.5	56.5/11.2	55.5/11.3
	AC	56.0/38.2	19.3/52.0	84.5/38.1	80.7/34.5	74.5/38.2	**97.0**/39.4	**91.5/0.8**
	CI	89.7/**1.9**	**97.0/0.1**	**98.7/0**	**99.0/0**	**97.5**/54.6	**97.7/2.2**	**94.2/1.1**
	RE	**94.8**/11.3	**99.2**/12.6	**99.0/4.5**	**95.8/0.1**	**90.2/2.8**	**99.0/2.4**	**91.3**/12.6
9SC	SS	11.7/14.9	15.4/14.5	19.3/14.1	42.8/11.6	53.0/10.5	70.9/**8.5**	68.0/**8.9**
	AC	**90.9**/42.8	58.5/38.0	8.0/52.2	78.9/40.8	65.2/38.7	**93.1/0.2**	60.7/40.1
	CI	**96.5/0**	**95.7/0**	**96.7/0**	**95.9/1.1**	**98.3**/43.9	**99.1/0.2**	**95.0/0.3**
	RE	**98.3/9.5**	**94.6**/10.2	**97.6/1.0**	**92.2/1.0**	**91.5/2.0**	**98.5/4.6**	**91.3/0.4**

(b) compact DNN architecture

architecture for brevity.

For each attack and each DNN architecture, we use true positive rate (TPR) and false positive rate (FPR) as the metric for evaluating the training set cleansing performance of each defense. TPR is defined as the percentage of backdoor training images correctly identified and removed. FPR is defined as the percentage of false removals (detected samples that are actually clean) from the *true backdoor target class*.

As shown in Tab. 4.6, for the wide DNN architecture, both SS and AC achieve very high TPR and low FPR for most of the attacks. The high TPRs of SS are consistent with the results reported in Tab. 2 of [117]. However, the low FPR of SS is only guaranteed by the (supervised) setting (at 1.5ϵ) of the number of training images to be removed from the true backdoor target class, where ϵ is the true number of backdoor images. In reality, in practice, both the true target class and the number of backdoor-poisoned images is unknown to the defender. Moreover, based on its design, which removes samples from



(a) pattern A, ResNet, wide (b) pattern B, ResNet, wide (c) pattern C, ResNet, wide

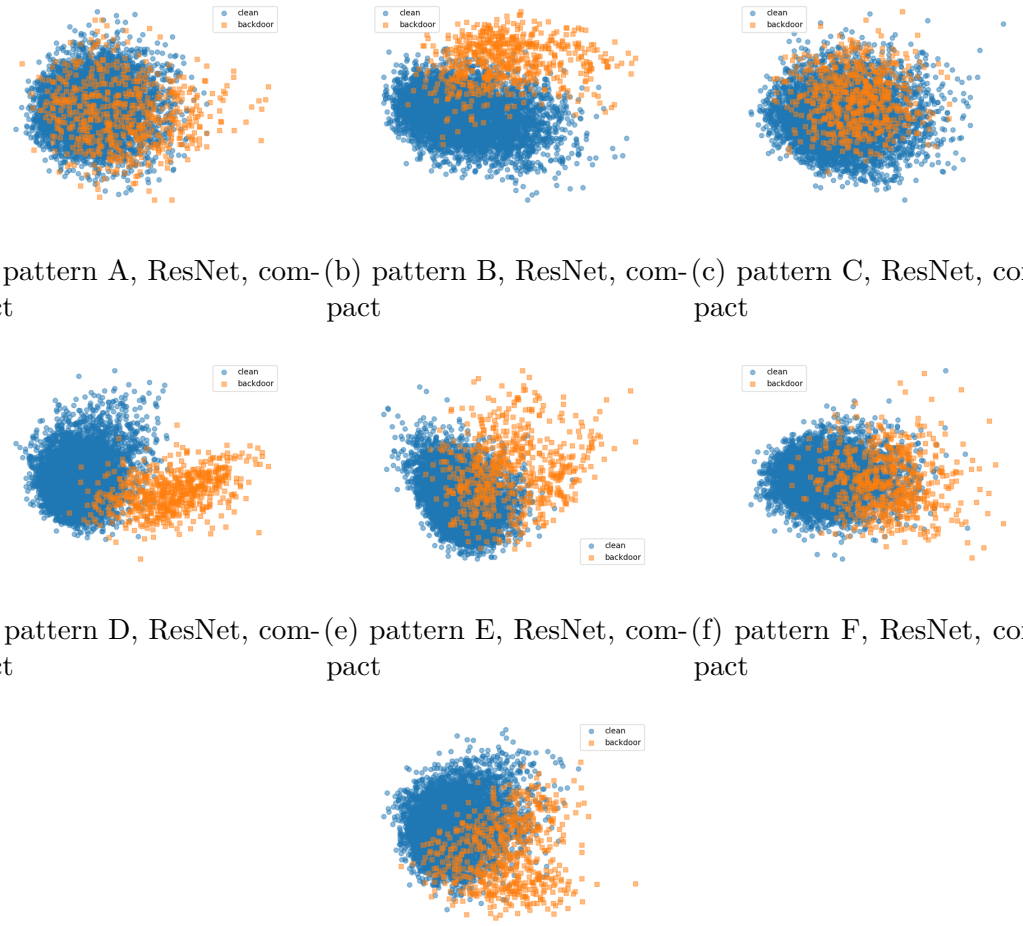
(d) pattern D, ResNet, wide (e) pattern E, ResNet, wide (f) pattern F, ResNet, wide

(g) pattern G, ResNet, wide

Figure 4.8: Distribution of the internal layer activations of the backdoor training images (orange squares) and the internal layer activations of the clean training images labeled to the true target class (blue circles) in 2D, for the 3SC attacks for pattern A-G, when the wide ResNet architecture is used by the defender.

all classes, SS will inevitably falsely remove a significant number of clean images from all classes other than the true backdoor target class. While this is not reflected in the results in Tab. 4.6 since we evaluate the FPR for SS considering only images labeled to the true backdoor class, SS's false removal of samples from other classes may result in degradation in the clean test accuracy of the retrained classifier, especially when the training samples are not adequate.

For the compact DNN architecture, both SS and AC are not effective against most of the attacks. This is because that the activations for the backdoor training images and the clean training images from the true backdoor target class are not separable in the low dimensional space. As an illustration, in Fig. 4.8, the backdoor training



(a) pattern A, ResNet, compact

(b) pattern B, ResNet, compact

(c) pattern C, ResNet, compact

(d) pattern D, ResNet, compact

(e) pattern E, ResNet, compact

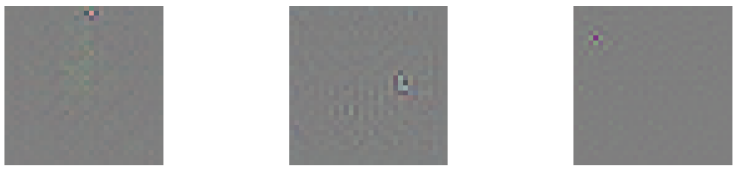
(f) pattern F, ResNet, compact

(g) pattern F, ResNet, compact

Figure 4.9: Distribution of the internal layer activations of the backdoor training images (orange squares) and the internal layer activations of the clean training images labeled to the true target class (blue circles) in 2D, for the 3SC attacks for pattern A-G, when the compact ResNet architecture is used by the defender.

images are clearly separable from the clean training images labeled to the target class when the activations are projected onto a 2D space, when the wide DNN architecture is used. However, as shown in Fig. 4.9, when the compact DNN architecture is used, the backdoor training images and the clean training images labeled to the target class are not clearly separable in the 2D space. We conjecture that such difference is due to the difference in the "learning capability" of the two DNN architectures – for the wide DNN architecture, the backdoor training images are learned as a "subclass", with the internal layer activations clearly separated from the activations for the clean training

(a) pattern A (b) pattern B (c) pattern C (d) pattern D



(e) pattern E (f) pattern F (g) pattern G

Figure 4.10: Pattern estimated (with a 127/255 offset to manifest negative perturbations) by DT-RED for the seven 1SC attacks for the wide DNN architecture.

images labeled to the target class. Readers may notice from both the above illustrations and the TPR/FPRs in Tab. 4.6b that training set cleansing performance for SS and AC is correlated with the separation between backdoor training images and clean training images in low-dimensional projected internal feature space.

By contrast, though CI fails to detect any attacks for the wide DNN architecture, it works well (for both detection and training set cleansing) for most of the attacks if the defender chooses to use the compact DNN architecture (with a lower dimension for the internal layer features).

In practice, an application may have its preferred DNN architecture. In comparison with SS, AC, and CI, DT-RED is uniformly effective, regardless of which DNN architecture is used for defense – it outperforms its competitors (with their optimistic settings) in both **detection** and **training set cleansing**.

Backdoor pattern estimation: In Fig. 4.10, we show the pattern estimated by DT-RED for the seven 1SC attacks for the wide DNN architecture. These attacks use the backdoor patterns in Fig. 4.6. We accurately recover most of these backdoor patterns (with, e.g., 2×10^{-5} squared error per pixel for pattern A). For pattern E, we recover the topmost pixel, which is the actual backdoor pattern learned by the classifier. By embedding this estimated pattern (essentially the single pixel perturbation) into the clean test images from the source class of the 1SC attack, we achieve 99.0% ASR on defenseless DNN with the wide ResNet architecture. Example backdoor training images with the estimated backdoor pattern removed (*i.e.*, cleansed images) are shown in Fig.

(a) pattern A (b) pattern B (c) pattern C (d) pattern D

(e) pattern E (f) pattern F (g) pattern G

Figure 4.11: Example backdoor training images with the backdoor pattern embedded (top), with the backdoor pattern estimated by DT-RED removed (middle), and the original clean images (down) for 1SC attacks for pattern A-G.

Table 4.7: Attack success rate (ASR) and clean test accuracy (ACC), jointly represented by ASR/ACC, of the retrained classifiers for the 21 attacks when DT-RED is applied, for both wide and compact DNN architectures.

pattern		A	B	C	D	E	F	G
wide DNN	1SC	4.9/91.9	2.7/92.1	2.2/91.8	1.1/92.7	3.5/92.6	0.8/91.6	2.2/92.7
	3SC	0.8/92.3	0.1/92.9	0/92.0	0.7/92.3	1.6/92.7	0.3/91.6	1.3/92.6
	9SC	0.6/92.5	0.2/92.9	0.3/92.9	1.4/91.7	0.8/92.1	0.8/91.9	0.4/92.4
compact DNN	1SC	1.9/90.2	0/90.2	2.6/90.4	0.3/90.0	3.1/89.9	0.6/90.5	2.2/89.8
	3SC	1.1/90.1	0.4/90.3	0.1/90.5	0.9/90.5	1.8/90.3	0.5/89.7	2.0/90.1
	9SC	0.9/89.9	0.9/90.0	0.7/90.2	2.2/90.4	0.7/90.1	0.9/90.6	1.4/90.0

Table 4.8: Attack success rate (ASR) and clean test accuracy (ACC), jointly represented by ASR/ACC, of the retrained classifiers for the 21 attacks when SS is applied (with the assumption that the existence of attack and the true target class are known to the defender), for both wide and compact DNN architectures.

pattern		A	B	C	D	E	F	G
wide DNN	1SC	1.7/91.7	0/92.8	98.6/92.5	0.3/91.6	97.3/91.5	76.6/92.1	1.7/91.5
	3SC	0.7/91.6	0.2/92.0	99.3/91.8	0.5/91.8	1.4/92.5	60.9/92.4	1.6/92.1
	9SC	0.2/91.9	0.4/91.9	91.7/91.8	1.6/91.8	0.7/92.0	7.1/91.6	0.6/92.6
compact DNN	1SC	99.1/90.6	0.3/90.5	98.3/90.5	68.5/90.1	91.5/90.8	83.7/90.8	5.6/89.6
	3SC	98.2/89.8	98.1/90.6	98.1/90.3	1.3/90.1	88.6/90.4	86.3/90.2	15.4/90.1
	9SC	90.2/90.0	96.4/90.5	96.1/90.1	5.1/89.9	4.2/90.1	65.6/90.2	1.2/89.6

4.11. These images are easily classified to the source class by the classifier trained on the poisoned training set. In practice, rather than remove these images, the trainer can instead retain these "cleaned" images in the training set (with the class label for a cleaned image chosen as the class decided by the (poisoned) classifier on the cleaned image).

Retraining: In Tab. 4.7, we show the ASR and the ACC of the classifiers retrained on the cleaned training set, for DT-RED following removal of the suspicious training images. The retrained classifier is the final product that will be provided to users/consumers. The configurations of retraining are determined by the training authority. As an illustration, here we simply use the same configurations as for training the defenseless classifiers in Sec. 4.4.6.1, for both wide and compact DNN architectures. With the protection of our RE, *the maximum ASR of the 21 attacks is merely 4.9% on CIFAR-10*. Also, we observe no degradation in ACC of the retrained classifiers due to the low FPR of DT-RED in training set cleansing.

In comparison, many of the retrained classifiers for SS, AC, and CI still have a

Table 4.9: Attack success rate (ASR) and clean test accuracy (ACC), jointly represented by ASR/ACC, of the retrained classifiers for the 21 attacks when AC is applied (with the assumption that the existence of attack and the true target class are known to the defender), for both wide and compact DNN architectures.

pattern		A	B	C	D	E	F	G
wide DNN	1SC	17.9/91.8	0/91.8	93.3/92.1	0.3/92.2	5.5/91.7	1.2/92.6	1.5/91.9
	3SC	0.7/91.7	0.1/91.7	0.1/91.8	0.9/91.8	0.9/92.1	0.6/91.8	2.0/92.5
	9SC	0.4/92.2	0.3/91.7	0.4/91.8	1.7/92.6	0.6/91.8	1.1/92.5	0.6/92.2
compact DNN	1SC	98.7/90.2	25.9/89.9	72.7/90.4	1.1/90.2	74.5/90.0	0.4/89.9	0.9/89.4
	3SC	90.6/90.1	98.7/89.9	92.4/90.3	1.2/90.1	35.6/90.1	0.4/90.2	2.2/90.6
	9SC	0.9/90.1	77.3/89.9	96.3/90.0	2.6/89.3	1.8/89.8	1.0/90.3	1.3/89.8

Table 4.10: Attack success rate (ASR) and clean test accuracy (ACC), jointly represented by ASR/ACC, of the retrained classifiers for the 21 attacks when CI is applied with the compact DNN architecture.

	A	B	C	D	E	F	G
1SC	98.5/90.6	0.1/90.9	3.2/90.5	n.a/n.a	3.0/89.9	0.4/90.0	n.a/n.a
3SC	2.2/90.6	0.5/90.1	0.3/90.2	0.9/90.1	0.9/89.3	0.5/90.0	2.5/90.3
9SC	0.6/90.3	0.8/90.3	0.8/89.9	2.3/90.5	0.6/89.6	0.9/90.1	1.6/90.0

high ASR. These comparison results are shown in Tab. 4.8, Tab. 4.9 and Tab. 4.10, respectively.

4.4.6.3 Ablation Study

In this section, we discuss how the choices of π and λ will possibly affect the detection and the training set cleansing performance of DT-RED. We also show that empirically there is a large range of choices for π and λ ensuring good performance of DT-RED against backdoor attacks on CIFAR-10 – both parameters can be easily and comprehensively determined with sufficient liberty. Moreover, as will be seen in Sec. 4.4.6.4, the same choices of π and λ made for the experiments on CIFAR-10 work generally well on other datasets.

Reasonable choices of π and λ: As we have mentioned in Sec. 4.4.3, π should be large, such that *only* for the backdoor class pairs (s, t^*) with $s \in \mathcal{S}^*$, π fraction of group misclassification from class s to t^* can be achieved with a relatively small perturbation. If π is chosen too small, for non-backdoor class pairs, there could be a small perturbation (with norm similar to the true backdoor pattern) that induces a (small) π fraction of images from the source class to be misclassified to the target class. Then the detection

will likely fail. However, if π is chosen too large (e.g. 1.0), it may be hard for all class pairs, including the true backdoor class pair, to achieve the π fraction of misclassifications from the source class to the target class. Hence, intuitively, π should be chosen large, but not too close to 1.

Our default choice of $\lambda = 1$ is straightforwardly based on the design of DT-RED. By setting $\lambda = 1$, for pattern estimation for all (s, t) pairs, a balance is kept between: 1) searching a pattern to induce a high fraction of group misclassification from class s to t; and 2) searching a pattern that causes class decision change when it is "removed" from the training images labeled to class t. If λ is chosen too large, for a backdoor class pair (s, t^*) with $s \in \mathcal{S}^*$, the estimated pattern will likely be effective in inducing high misclassifications from s to t^*, but will not induce misclassification when "removed" from the backdoor training images labeled to t^*, causing both TPR and FPR for training set cleansing to be low. If λ is chosen too small, for a backdoor class pair (s, t^*) with $s \in \mathcal{S}^*$, pattern estimation will focus on inducing misclassifications to classes other than t^* for all images labeled to t^* (including both backdoor training images and clean training images), such that π fraction misclassification will not be easily achieved with a small perturbation, causing a failure in detection. Hence, intuitively, λ should be close to 1.

Evaluation of DT-RED robustness to choices of π and λ: We test the detection and training set cleansing performance of DT-RED for a range of π and a range of λ, respectively. We consider two 3SC attacks, with one attack using pattern A, and the other using pattern C. We use the wide ResNet DNN architecture and the same configurations described in Sec. 4.4.6.2 to train the classifier (for defense purposes, without augmentation to training samples) on the poisoned training set. We also use the same configurations as in Sec. 4.4.6.2 for DT-RED, except that either π or λ is varied. Previous discussion indicated reasonable choices of π and λ. Here, we experimentally evaluate DT-RED for a much wider range of π and λ to see if performance drops for either detection or training set cleansing. Hence, we consider π for a range from 0.98 to as low as 0.82, and λ for a range from 1/4 to 4.

For detection, both attacks are successfully detected for $\pi \in \{0.82, 0.84, \ldots, 0.96, 0.98\}$ with the target class correctly inferred. The maximum order statistic p-values (over the range of π, paired with $\lambda = 1$) for the attack with pattern A and the attack with pattern C are "underflow" (a positive number less than 10^{-323}) and 0.018, respectively – clearly below the detection threshold $\theta = 0.05$. For $\lambda \in \{1/4, 1/3, 1/2, 1, 2, 3, 4\}$, paired with $\pi = 0.96$, the maximum order statistic p-values for the attack with pattern A and the attack with pattern C are 10^{-16} and 4×10^{-4}, respectively – far less than $\theta = 0.05$. Also,

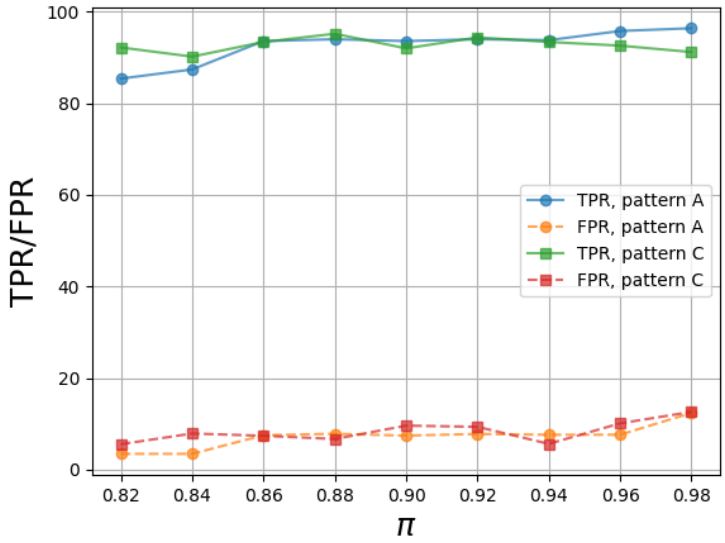


Figure 4.12: True positive rate (TPR) and false positive rate (FPR) of training set cleansing by DT-RED against the attack with pattern A and the attack with pattern C, respectively, for a wide range of π, with $\lambda = 1$.

the target class is correctly inferred for both attacks.

In Fig. 4.12, we show the training set cleansing performance of DT-RED against the two attacks for $\pi \in \{0.82, 0.84, \ldots, 0.96, 0.98\}$, paired with $\lambda = 1$. For both attacks, DT-RED achieves high true positive rate (TPR) and low false positive rate (FPR) for a large range of π. For the attack with pattern A, there is a slight drop in TPR when π is small (below 0.86), which is consistent with our previous discussion. Also, we observe a slight increase in FPR for both attacks when π is too close to 1. However, this experiment still empirically validates a large range of reasonable choices for π (e.g. from 0.86 to 0.96). In Fig. 4.13, we show the training set cleansing performance of DT-RED against the two attacks for $\lambda \in \{1/4, 1/3, 1/2, 1, 2, 3, 4\}$, paired with $\pi = 0.96$. For both attacks, the FPR increases as λ decreases. For the attack with pattern C, the TPR decreases as λ increases. Although for the attack with pattern A, the TPR stays high even for $\lambda = 4$, we expect it to decrease as λ further increases. Again, the observations match with our previous discussion – λ should be close to 1 (not exceeding the interval $[1/2, 2]$ if necessary) to ensure a high TPR and a low FPR in training set cleansing.

4.4.6.4 Experiments on Other Datasets

Datasets: In this section, we evaluate DT-RED defense in comparison with competitors

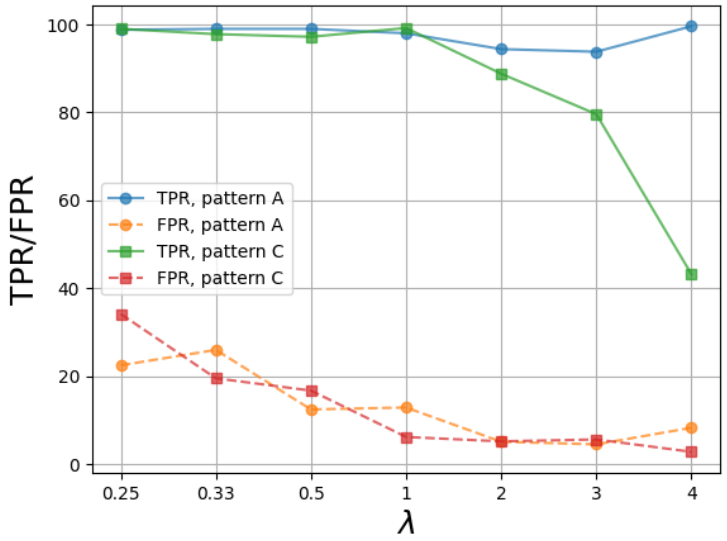


Figure 4.13: True positive rate (TPR) and false positive rate (FPR) of training set cleansing of DT-RED against the attack with pattern A and the attack with pattern C, respectively, for a wide range of λ, with $\pi = 0.96$.

Table 4.11: Training configurations for each of MNIST, F-MNIST, GTSRB, and CIFAR-100; and the resulting ACC when there is no attack.

	MNIST	F-MNIST	GTSRB	CIFAR-100
DNN architecture	LeNet-5 [15]	VGG-9	DNN in [109]	ResNet-34 [14]
epoch	60	60	100	120
batch size	256	256	64	32
learning rate	10^{-2}	10^{-2}	10^{-3}	10^{-3}
data augmentation	no	no	no	yes
ACC	99.1%	90.8%	98.7%	72.8%

(i.e. SS, AC, and CI) on datasets including MNIST [15], Fashion-MNIST (F-MNIST) [177], GTSRB [178], and CIFAR-100 [174]. The details of these datasets can be found in previous chapters.

Training configurations: In Tab. 4.11, we summarize the training configurations for each dataset, including the choice of DNN architecture, the number of training epochs, batch size, learning rate, and whether training data augmentation is used. In particular, the "VGG-9" architecture for training on F-MNIST is customized by removing the last

Table 4.12: Source class(es), target class, and the number of backdoor training images per source class for the attacks on MNIST, F-MNIST, GTSRB, and CIFAR-100.

	MNIST	F-MNIST	GTSRB	CIFAR-100
source class(es)	5, 7, 10	1	14, ..., 18 25, ..., 31	1, ..., 20
target class	9	3	9	21
backdoor training images per source class	500	900	30	30

Table 4.13: Attack success rate (ASR) and clean test accuracy (ACC) of the classifiers trained on the poisoned training set of MNIST, F-MNIST, GTSRB, and CIFAR-100, when there is no defense.

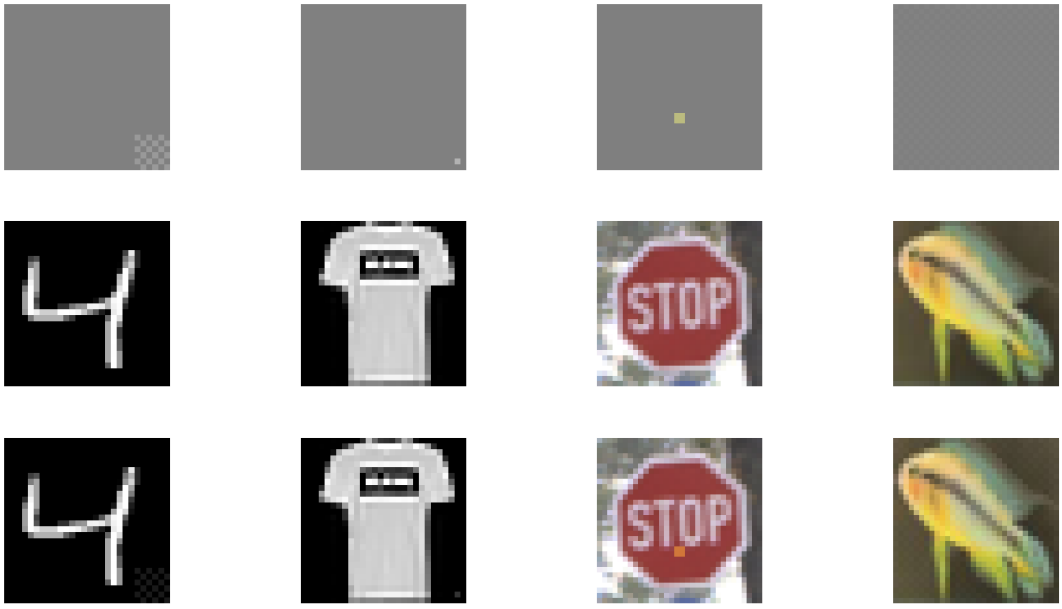
	MNIST	F-MNIST	GTSRB	CIFAR-100
ASR	96.3	91.7	87.8	98.0
ACC	98.9	91.0	98.1	72.4

two convolutional layers of VGG-11 [16] and using 16, 32, 64, 64, 128, 128 filters in the six remaining convolutional layers, respectively. For GTSRB, we use the same DNN architecture used in [109]. For CIFAR-100, the learning rate is decayed by 90% at each of epochs 60, 100, 110. The training data augmentation involves random cropping and random horizontal flipping. For all four datasets, we use cross entropy loss as the training loss function. For MNIST and F-MNIST, we use stochastic gradient descent (SGD) optimizer with momentum 0.9. For GTSRB and CIFAR-100, we use Adam optimizer with decay rate 0.9 and 0.999 for the first and second moment, respectively. The (clean) test accuracy for each dataset (in absence of attacks) is also reported in Tab. 4.11.

Attack crafting: We create one attack on each of the four datasets. The source class(es), target class, and the number of backdoor training images per source class are summarized in Tab. 4.12. For MNIST, F-MNIST, and CIFAR-100, the class indices used here (starting from 1) follow the label assignment specified in the document "torchvision.dataset"[4]. For GTSRB, the class labels are specified online[5]. Especially for the GTSRB dataset of trafic signs, we set the source classes to be those traffic signs related to "using cautious", e.g. a stop sign, a yield sign, etc., and the target class is

[4]`https://pytorch.org/docs/stable/torchvision/datasets.html`

[5]`http://benchmark.ini.rub.de/?section=gtsrb\&subsection=news`



(a) MNIST (b) F-MNIST (c) GTSRB (d) CIFAR-100

Figure 4.14: Backdoor pattern (with 0.5 offset) (top), example backdoor training image (down), and originally clean image (middle), for MNIST, F-MNIST, GTSRB, and CIFAR-100.

chosen as a speed limit sign of 120km/h. If an autonomous vehicle with a traffic sign recognizer/classifier undergoes such an attack, during testing, it will likely recognize a, e.g. stop sign with the backdoor pattern (that will be shown next), as a speed limit sign of 120km/h; and it will speed up instead stop, which may cause a catastrophe.

In Fig. 4.14, we show the backdoor patterns (with 0.5 offset) used to attack the four datasets, with example backdoor training images and the original clean images. The backdoor pattern for MNIST is a small patch (6×6) of "chessboard" located at the bottom right corner, with perturbation size 25/255. The backdoor pattern for F-MNIST is a pixel perturbation located at the bottom right corner with perturbation size 50/255. The backdoor pattern for GTSRB is a yellow square which mimics a sticker that can be physically attached to a traffic sign. To digitally embed such a pattern to a traffic sign image, we perturb in both the 'red' and the 'green' channels with perturbation size 60/255. Lastly, for CIFAR-100, we use the image-wide "chessboard" pattern, i.e. pattern A, with perturbation size 3/255.

In Table 4.13, we show the ASR and the clean test ACC of the classifiers trained on the four poisoned training sets, when there is no defense. All the attacks are successful,

Table 4.14: Maximum detection statistics (Silhouette score for AC and cluster impurity for CI) for the classifier trained on the poisoned training set and the classifier trained on the clean classifier, respectively, for MNIST, F-MNIST, GTSRB, and CIFAR-100.

	MNIST		F-MNIST		GTSRB		CIFAR-100	
	attack	clean	attack	clean	attack	clean	attack	clean
AC	0.1945	0.1100	0.4878	0.2220	0.2307	0.0883	0.7901	0.7790
CI	0.2604	0.0515	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.

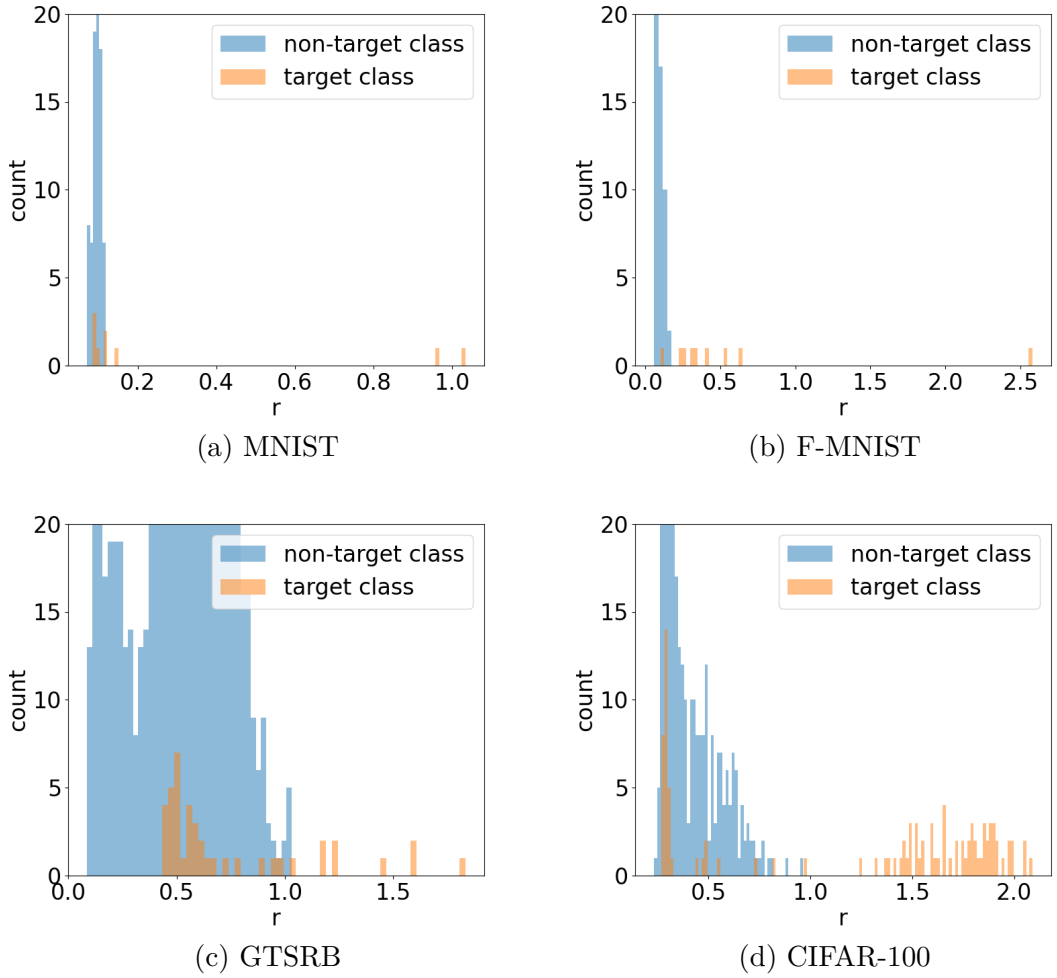
with high ASR and no clear degradation in ACC.

Defense performance evaluation: For each attack, we train a classifier on the poisoned training set using the the same DNN architecture and training configurations as specified in Tab. 4.11; except for CIFAR-100, we do not use data augmentation here. For each attack, we also train a benchmark classifier on the clean training set using the same training configuration.

For detection, again, we do not consider SS because it does not make inferences about presence/absence of backdoor poisoning. AC and CI do detect if there is poisoning, but the detection threshold depends on the data domain and may need sufficient number of benchmark DNNs trained on the clean, unpoisoned training set for its supervised setting. For each data domain, we have trained only one classifier on the poisoned training set and one classifier on the clean training set. We report the maximum detection statistics, the Silhouette score for AC and cluster impurity for CI, associated with each of these two classifiers in Tab. 4.14. Note that for AC, we use the same reduced feature dimension, 10, as in [118].

For AC, the maximum Silhouette score (taken over all classes as putative backdoor target classes) associated with the classifier trained on the poisoned training set is clearly larger than for the classifier trained on the clean training set for MNIST, F-MNIST, and GTSRB. Especially for MNIST, our results are consistent with the results on MNIST reported in Tab. 3 of [118]. However, only for MNIST and GTSRB, but not for F-MNIST, the true backdoor target class has the largest Silhouette score for the classifier trained on the poisoned training set for these two datasets. For CIFAR-100, no matter whether there is poisoning, for most classes, the Silhouette score is close to 0.1. But for a few classes, not necessarily the true backdoor target class when there is poisoning, the Silhouette score is very high, even higher than the Silhouette score for the true backdoor target class. Thus, AC cannot make reliable detections for F-MNIST and CIFAR-100.

For CI, the maximum cluster impurity for the classifier trained on the poisoned training set is larger than for the classifier trained on the clean training set for MNIST.



(a) MNIST (b) F-MNIST

(c) GTSRB (d) CIFAR-100

Figure 4.15: Histogram of the reciprocal statistics when DT-RED is applied to the attacks on MNIST, F-MNIST, GTSRB, and CIFAR-100, respectively.

But CI yields only one component for every class, regardless of the existence of backdoor poisoning, for F-MNIST, GTSRB, and CIFAR-100. This is because the penultimate layer feature dimension for the DNN architectures for these three datasets is overly large (1024, 512, and 512 respectively), with the BIC model complexity penalty too high when more than one mixture component is used.

Lastly, we apply DT-RED with the *same* (default) configurations as specified earlier, i.e. $\delta = 10^{-4}$, $\pi = 0.95$, and $\lambda = 1$. Compared with AC and CI, DT-RED uses a statistical confidence threshold that does not require supervised setting, unlike AC. DT-RED successfully detects all four attacks and infers no attack for the four clean datasets. Moreover, in Fig. 4.15, for each attack, we show the histogram of the reciprocal statistics r_{st} for the $K(K-1)$ (s,t) class pairs. Clearly, for all four attacks, when the

Table 4.15: True positive rate (TPR) and false positive rate (FPR) of training set cleansing (jointly represented in TPR/FPR form) when applying SS, AC, CI, and DT-RED to the attacks on MNIST, F-MNIST, GTSRB, and CIFAR-100, respectively. TPRs $\geq 90\%$ and FPRs $\leq 10\%$ are in bold.

	MNIST	F-MNIST	GTSRB	CIFAR-100
SS	**100/9.3**	**99.3/7.6**	**29.7/37.7**	80.3/23.6
AC	**100/0.1**	**97.6/0.1**	62.0/24.8	**94.0/0.8**
CI	**99.5/0**	n.a./n.a.	n.a./n.a.	n.a./n.a.
DT-RED	**97.3/1.4**	**95.4/5.8**	**99.7/4.8**	**93.5/5.2**

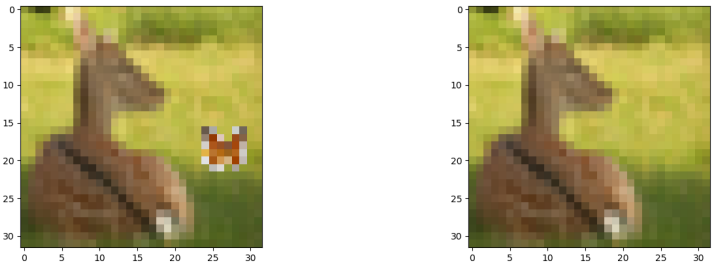
Table 4.16: Attack success rate (ASR) and clean test accuracy (ACC) of retrained classifiers after DT-RED training set cleansing, for attacks against MNIST, F-MNIST, GTSRB, and CIFAR-100, respectively.

	MNIST	F-MNIST	GTSRB	CIFAR-100
ASR	0.1	1.0	0.0	0.15
ACC	99.1	90.8	98.6	71.8

training set is backdoor poisoned, there is at least one reciprocal statistic corresponding to the *true backdoor* target class that is an outlier to the null model – the success of DT-RED detection is thus visualized.

In Table 4.15, we show the TPR and FPR of training set cleansing when SS, AC, CI, and DT-RED are applied to the four attacks, respectively. Again, we use the same settings as in [117] for SS, wherein SS is endowed with the knowledge of both whether there is an attack and the target class of an attack (even those are in fact unknown). We give the same (needed) information to AC. Note that DT-RED and CI do not require this information. Compared with SS, AC, and CI (with SS and AC benefitting from knowledge of the attack, which is not given to DT-RED), which have poor TPR or FPR for at least one data set, DT-RED achieves consistently high TPRs and low FPRs across all four data sets.

Retraining: After removing the detected training images, we retrain a classifier for each dataset using the same DNN architecture and training configuration specified in Tab. 4.11. Here, we only perform retraining for DT-RED for brevity. In Tab. 4.16, we show that all the retrained classifiers have both low ASR and almost no degradation in clean test ACC compared with the classifier trained on the clean, unpoisoned training set (see the last row of Table 4.11). *Our DT-RED is thus effective in protecting these datasets from being backdoor poisoned.*



(a) with “butterfly” (b) clean

Figure 4.16: Example images (a) with a patch replacement pattern, a “butterfly”, and (b) the original clean image.

4.5 Before/During Training Detection of Patch Replacement Backdoor Patterns

In this section, we briefly discuss before/during training backdoor detection for attacks with other types of backdoor patterns. In particular, we focus on the popular patch replacement pattern. We show that, compared with the additive perturbation pattern, patch replacement patterns (with the patch in the same fixed position for all poisoned training and test images) are much easier to detect during the DNN’s training phase.

We illustrate a simple idea for detecting patch replacement patterns during training on CIFAR-10 using the same Attack A from Sec. 3.5.6.1. An example backdoor training image (with a “butterfly” as the backdoor pattern) and the original clean image are shown in Fig. 4.16. For simplicity, we focus on the true backdoor target class, which contains 4500 clean training images and 500 backdoor training images. The images considered here are colored, and with image size 32×32. We apply a 3×3 convolution filter with 3 input channels (for color images) and 1 output channel to each of the 5000 training images labeled to this target class, with no padding at the boundary of the images. The weights associated with this convolution filter are independently randomly sampled from a standard normal distribution. We flatten the resulting $5000 \times 30 \times 30$ convolution output for these 5000 images into a 4.5×10^6 dimensional vector.

Note that the same backdoor pattern, the “butterfly”, has been embedded in 500 different backdoor training images. Thus, for each of these images, for any 3×3 subregion of the “butterfly”, applying the same convolution filter will yield the same output – i.e. the same convolution output will appear 500 times in the 4.5×10^6 dimensional vector. It is unusual for a pattern to appear this many times for an unpoisoned training set. Even

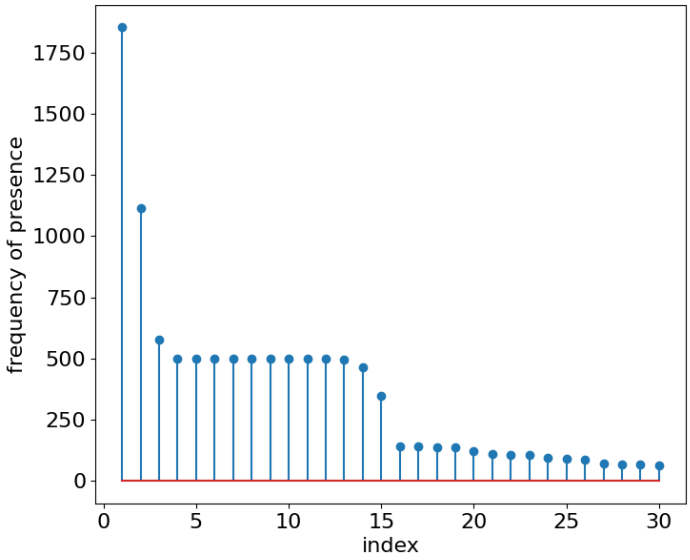


Figure 4.17: Top 30 frequencies of occurrence of unique convolution outputs, when applying the convolution filter to the target class training images.

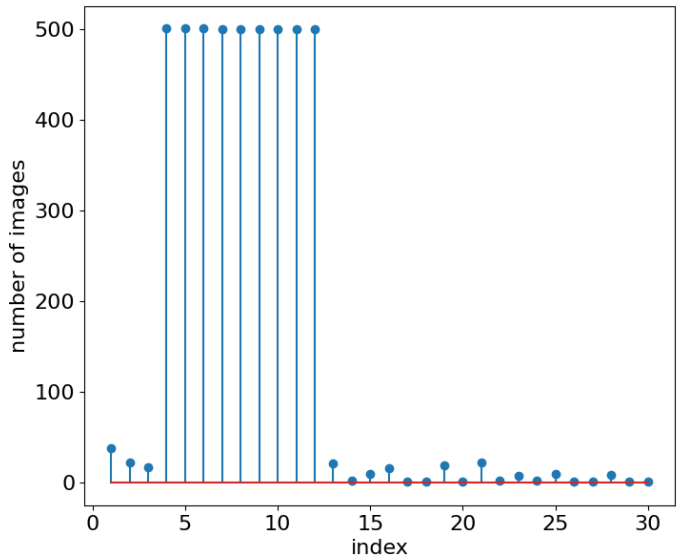


Figure 4.18: Number of training images involved in producing each of the 30 convolution outputs in Figure 4.17.

if a subset of training images contain the same object, this object will likely not have exactly the same pixel values in all these images due to the variance of the background environment during image capture.

In Fig. 4.17, we show the 30 unique convolution outputs with the top frequency of presence in the 4.5×10^6 dimensional vector. There are several convolution outputs that appear exactly 500 times. However, this observation cannot be directly used for developing a before/during training defense against patch replacement backdoors. First, there is no evidence these unique convolution outputs that appear exactly 500 times are associated with a backdoor pattern. Second, there are other unique convolution outputs that appear even many more times. Thus, for each of these 30 unique convolution outputs, we count the number of training images that are involved in producing each of the convolution outputs. As illustrated in Fig. 4.18, *only* for the outputs that appear exactly 500 times, the number of training images involved is 500. This can be the basis for building a detector that detects the presence of backdoor poisoning and identifies the backdoor training images. The other outputs that appear an abnormally large number of times (as shown in Fig. 4.17) are actually caused by over/under exposure of the background scene (e.g. the sky), such that a large number of pixels in a *single* image have the same intensity value.

However, this easily detectable "signature" for a patch replacement backdoor pattern does not exist for additive perturbation backdoor patterns. This is because the intensity value of a pixel being perturbed also depends on its original intensity value. Still, we can develop a before/during training defense against patch replacement backdoors based on the above idea. Such a defense can be developed in parallel with the defense against additive perturbation backdoors to cover both attack types of interest.

Chapter 5
Backdoor Attacks and Defenses for 3D Point Cloud Classifiers

5.1 Overview

In Sec. 2.2.5, we mentioned that backdoor attacks and defenses have been extended to many domains other than images, including videos, text and speech. In this chapter, we consider backdoor attacks and defenses for 3D point cloud (PC) data[1]. We will first introduce PC data, related applications and processing tools. Then we discuss the recent development in adversarial attacks against PC classifiers. Our main contributions in this chapter include:

- In Sec. 5.3, we propose a backdoor attack against 3D PC classifiers. In particular, we propose a backdoor pattern customized for PC data dubbed "backdoor points". Backdoor points contains a small set of points inserted in each PC sample, with optimized spatial location and local geometry. We show that effectiveness of the backdoor attack mostly depends on the spatial location of the backdoor points, while careful design of their local geometry helps the backdoor attack evade the state-of-the-art PC anomaly detectors (which sanitizes the PC to remove outlier points).

- In Sec. 5.4, we propose a post-training defense (without access to the training set) named "PC-RED", which is inspired by RED for image backdoor attacks, to detect whether a PC classifier is attacked by the backdoor attack above. We propose a novel reverse-engineering problem specific to backdoor points. Different with image

[1]In this section, when saying PC, we refer to 3D PC. PC data with other point dimensions will be specified as "general PC".

REDs, PC-RED reverse-engineers a putative backdoor pattern for each *source class* and simultaneously estimates a target class. This different design is to address the generally strong robustness of PC classifiers against adversarial perturbations. Moreover, for some putative source classes, there exists a spatial location *close to* most source class PCs, such that a single inserted point can cause most of these PCs to be misclassified to a *common* target class, *irrespective of the existence of backdoor attack*. Such an "*intrinsic backdoor*" can easily cause false detections when there is actually no attack. While distinguishing intrinsic backdoors from those caused by attack is still an open problem even for image backdoor defense, we propose a novel combined detection statistic to address this challenging problem.

5.2 Related Works

5.2.1 Point Cloud Data

In general, a point cloud (PC) can be denoted as

$$\mathbb{X} = \{\mathbf{x}_i \in \mathbb{R}^l | i = 1, \cdots, n\} \in \mathcal{X}, \tag{5.1}$$

where $\mathbf{x}_i$ is the i-th point represented by a l-dimensional vector and n is the total number of points. 3D PC (with $l = 3$) is the most common type of PC data, which is widely considered in applications such as autonomous driving, industrial robotics, and augmented reality [187, 188]. In practice, a 3D PC is commonly captured by 3D sensors including radio detection and ranging (RADAR) [189], light detection and ranging (LiDAR) [190], and ultrasonic sensors [191], with each point represented by its (x, y, z) coordinates in space. General PCs with higher point dimension can also be constructed from 3D PCs by involving additional features, e.g. point intensity, beyond the 3D coordinates for each point [192].

5.2.2 Point Cloud Classifiers

Like for images, PC data are involved in tasks including classification, object detection, and segmentation. For example, for autonomous driving, it is important for an autonomous vehicle on the road to effectively obtained the real-time information of its surroundings to avoid collision. Thus, the autonomous vehicle needs to detect objects such as vehicles, pedestrians, bicycles, etc., from the real-time 3D point cloud captured

by scanning its surroundings [193].

As the fundamental to most PC learning tasks, PC classification has drawn a lot of attention with rapid development in related techniques. Early approaches first represents 3D PCs as three-mode binary tensors using a series of voxels. Any element in the tensor is denoted as 1 if there are points located in the corresponding voxel; and denoted as 0 if the corresponding voxel is empty. Then a 3D convolutional neural networks, e.g. VoxNet [23], can be applied to the three-mode tensors to classify the voxelized PC. Here, we mention a commonly used voxelization method (especially for autonomous driving), where 3D PCs are converted using 2D voxels with the height being omitted. The resulting voxelized data is called a "bird eye view" (BEV) map, which can then be fed to convolution-based image DNNs. Recently, DNNs directly consuming PC data have been developed. PointNet [194] is the pioneering method in this stream. It processes each point in the PC independently using a common "encoder" based on convolution layers. The obtained features for all the points are then aggregated using a symmetric function – max pooling – to achieves permutation invariance of the points. Due to the simplicity and strong representation capability of PointNet, it is used as the backbone of many 3D learning modules [187], and is also the basis for many subsequent methods, e.g. [192, 195–197]. Note that in practice, the set of points being captured may be different when the sensor is placed at different locations. Thus, PC classification based on multi-view has been developed [198, 199]. More recently, PC classification has been discussed under multi-agent scenarios, where each agent has its own view of the scene and the learning task (including classification and detection) is accomplished by the collaboration between agents (e.g. exchanging intermediate features obtained by the agents' local DNN) [200, 201].

5.2.3 Adversarial Attacks against Point Cloud Classifiers

Existing adversarial attacks against PC classifiers are mostly TTE attacks (introduced in Sec. 1.2.2 for the image domain). These attacks "fool" a victim classifier by adding points to a test PC, perturbing its points, or removing some of its points [202–205] – these operations are the analogue, for PCs, of TTE perturbations applied to 2D images. However, PC TTE attacks do not transfer nearly as well as image TTE attacks. Even for two classifiers trained on the same dataset, with the same architecture but different parameter initializations, test PCs generated using one classifier do not reliably "fool" the other [202, 204]. Such poor transferability may be due to the larger discrepancy of the decision boundaries between two PC classifiers. Especially for PointNet and its variants, the "critical points" selected from a PC by max pooling may be very different for two

classifiers; thus perturbing a critical point selected by classifier A cannot "fool" classifier B if it is "dropped out" (i.e. not selected by max pooling) by classifier B. Hamdi et al. [206] improve the transferability of PC TTE attacks; but the success rate to "fool" the victim classifier using their transferred attack is still less than 65%, even with the victim classifier's training set exploited by the attacker. In Sec. 5.3.3.4, we will show experimentally the poor transferability of PC TTE attacks on a benchmark dataset for PointNet. Thus, the effectiveness of existing PC TTE attacks largely relies on knowledge of the victim classifier, which is usually not available in practice.

By contrast, backdoor attacks do not require the knowledge of the victim classifier, nor do they need access to the clean training set. In parallel with our PCBA that will be presented in Sec. 5.3, in a concurrent work [207], a PC backdoor attack is proposed, with the backdoor pattern being a rotation to all points in the PC around the origin (assuming that the PCs are aligned) with a common, prescribed degree. However, this attack may not be effective when the training process involves augmentation to the training PCs that includes rotation of points. Also in [207], a clean-label backdoor attack based on a point perturbation pattern is proposed. In another concurrent work [208], a MorphNet is trained to embed a sample-specific backdoor pattern to each PC.

5.3 PCBA: A Backdoor Attack against 3D Point Cloud Classifiers

5.3.1 Key Ideas of PCBA

Extending backdoor attacks from the image domain to the PC domain is not trivial, with the following challenges:

- Challenge 1: Existing backdoor patterns for image backdoor attacks, including the additive perturbation patterns and the patch replacement patterns are defined in the "pixel domain". These backdoor patterns are not applicable to 3D PCs, for which "pixels" are undefined.

- Challenge 2: Designing a backdoor pattern learnable by 3D PC classifiers is difficult since they extract different features than image classifiers, especially convolutional neural networks like [14].

- Challenge 3: The backdoor pattern should be robust to test-time preprocessing of

3D PCs like random sampling, should be evasive of anomaly detectors (ADs), and should be scene-plausible.

To address challenge 1, we propose to insert a small cluster of points as the backdoor pattern, dubbed "backdoor points", which can be implemented either digitally (to mimic, e.g., spurious points caused by vehicle exhaust), or physically using an object (e.g. a ball) captured along with the scene by the 3D sensor. To address challenge 2, the spatial location of the backdoor cluster is optimized by making use of a surrogate classifier that is *independently* trained by the attacker, using its own (separate) data set. Such optimization is necessary to ensure that the victim classifier learns the backdoor pattern during its training. To address challenge 3, the local geometry of the actual backdoor points embedded in each PC sample is also optimized, such that these points have similar local density as the original points in the PC. Then, the probability that all inserted points are removed by a point AD will be largely reduced.

We consider a common practical scenario, wherein a training authority learns a 3D PC classifier using a dataset collected from the public. Unfortunately, among the data donors, there is an attacker who aims to embed a backdoor mapping in the classifier. The key assumptions for this practical scenario are as follows:

- The attacker has *no access* to the training process, including knowledge of the victim classifier's architecture, the loss function, and other training configurations.
- The attacker has *no access* to the clean training data collected by the trainer from other (benign) sources.
- The attacker is able to collect data independently (to train a surrogate classifier and create backdoor training samples). This collected data is assumed i.i.d. with the clean training samples collected by the trainer.
- The attacker has the capability to contribute its data to the training set of the victim classifier.

The first two assumptions are consistent with the role of a backdoor attacker as detailed in Sec. 2.2.1, who is merely one of the data donors. These two assumptions make our backdoor attack more practical than PC TTE attacks, which rely on knowledge of the victim classifier. The third and the fourth assumptions are the basis of image backdoor attacks and traditional poisoning attacks aiming to degrade the classifier's accuracy – the classifier can be more adequately trained by collecting data from as many sources as possible, among which there may be an attacker.

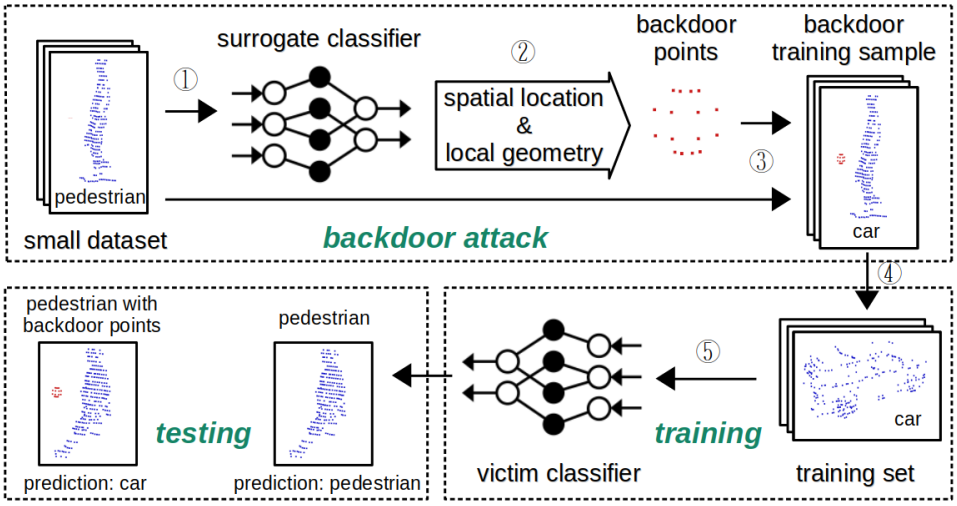


Figure 5.1: Outline of our PCBA. The attacker collects a small dataset to train a surrogate classifier (①). The backdoor points is generated using the surrogate classifier with optimized spatial location (Sec. 5.3.2.2) and local geometry (Sec. 5.3.2.1) (②). The backdoor points is embedded in clean PCs from a source class, e.g. "pedestrian" (③), to generate backdoor training samples labeled to a target class, e.g. "car". These samples are used to poison the training set possessed by the trainer (④), on which the victim classifier is trained (⑤). During testing, the victim classifier is supposed to classify source class PCs embedded with the backdoor points to the target class, while correctly classifying backdoor-free test PCs.

Following these assumptions, and based on the key ideas regarding the challenges mentioned above, the outline of our PCBA is summarized in Fig. 5.1. In the next section, we present the details regarding the design of backdoor points, including the optimization of the spatial location and local geometry of the backdoor points.

5.3.2 Backdoor Points

The key to our PCBA is the design of the backdoor pattern and the associated embedding function. Due to the irregularity of 3D PCs, and inspired by PC TTE attacks [202, 209], candidate backdoor embedding mechanisms include adding points, dropping points, and perturbing points. Here, we choose to add/insert a small cluster of points as the backdoor pattern, for two reasons. First, in practice, a set of inserted points can potentially be implemented physically by placing an object, e.g. a ball, in the scene, captured by a 3D sensor; or, these points can be digitally inserted into a PC to *mimic* an object or a cluster of spurious points (which are usually caused by vehicle exhaust in the context of autonomous driving). Second, an ideal backdoor pattern is a *common* pattern; but

point dropping and point perturbations are a function of the original points – it is thus difficult to create a common backdoor pattern using these mechanisms. Formally, the *backdoor embedding function* is defined as:

$$m_{\text{insert}}(\mathbb{X}; \mathbb{V}) = \mathbb{X} \cup \mathbb{V}, \tag{5.2}$$

where the backdoor pattern $\mathbb{V}$, dubbed *"backdoor points"*, is defined as:

$$\mathbb{V} = \{\mathbf{u}_i + \mathbf{c} | \mathbf{u}_i \in \mathbb{R}^3, \mathbf{c} \in \mathbb{R}^3, i = 1, \cdots, n'\}. \tag{5.3}$$

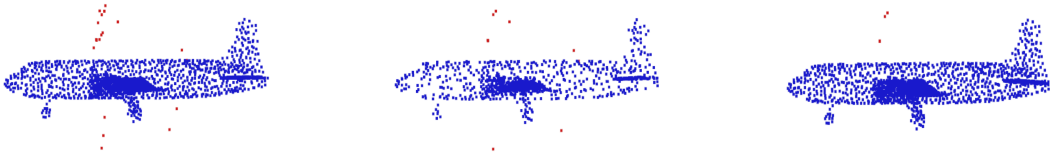
Note that we consider 3D PCs here, where each point in $\mathbb{X}$ follows the representation in Eq. 5.1 with point dimension $l = 3$. Also note that $\mathbb{V}$ is jointly determined by its *local geometry* $\mathbb{U} = \{\mathbf{u}_i \in \mathbb{R}^3 | i = 1, \cdots, n'\}$ and its *spatial location* $\mathbf{c}$ – how to specify these two elements is discussed next.

5.3.2.1 Local Geometry of Backdoor Points

Ideally, the embedded backdoor points should have the same local geometry for all backdoor training/test PCs. However, this is not feasible for backdoor attacks physically implemented using even the same object – the actual points associated with the object captured by a 3D sensor are likely different from PC to PC. Fortunately, local geometry of backdoor points is less critical than its spatial location for the victim classifier to learn the backdoor mapping, as will be empirically shown by our substantial experiments in Sec. 5.3.3. Here, we allow backdoor points embedded in each PC to have slightly *different* local geometry.

For practical consideration, the design of backdoor points' local geometry mainly addresses its *robustness* to possible PC preprocessing, e.g. point sub-sampling, and PC anomaly detectors (ADs) deployed during testing. As shown in Fig. 5.2, point sub-sampling keeps a subset of points for classification; thus, part of the inserted backdoor points will be inevitably removed. A PC AD, e.g. [4], removes outlier points with abnormal local density. Accordingly, the backdoor points should: a) contain a sufficient number of points; b) have a similar local point density as the PC into which they are embedded. For backdoor attacks implemented physically using an object, criterion a) can be achieved if the object is sufficiently large. Criteria b) is automatically achieved due to the usually stable scanning frequency of 3D sensors.

For digitally implemented backdoor attacks, backdoor points' local geometry $\mathbf{U}$ can be specified by the attacker by defining a suitable stochastic point generator. For example,



(a) prior-processing (b) random sampling (c) outlier removal

Figure 5.2: Preprocessing and anomaly detection of test PCs. (a) A PC with randomly inserted points (in red). (b) PC undergoes random sampling (with half the points removed). (c) PC undergoes the point AD in [4] which removes outlier points – most of the inserted points are removed.

in one of our experiments, to mimic a physically implemented backdoor attack using a ball, we generate backdoor points randomly located on a sphere with some radius r using the random generator:

$$\mathbf{g}(r, \boldsymbol{\theta}, \boldsymbol{\phi}) = [r \sin \boldsymbol{\theta} \cos \boldsymbol{\phi}, r \sin \boldsymbol{\theta} \sin \boldsymbol{\phi}, r \cos \boldsymbol{\theta}]^T, \tag{5.4}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are random variables uniformly distributed in $[0, \pi]$ and $[0, 2\pi]$ respectively and r is a parameter to be specified. Regardless of the generator's form, criterion a) above can be achieved by generating a sufficient number of points. For criterion b), we propose to optimize (over the parameters of the generator) the distribution of the local density of all points in $\mathbf{U}$ by a novel approach based on median absolute deviation (MAD), a robust measure of variability [167]. Inspired by [4], we measure the local density of a point using its kNN distance. Then, the median kNN distance of a PC $\mathbb{X} \in \mathcal{X}$ for backdoor point embedding is:

$$D_{\text{knn}}(\mathbb{X}) = \underset{i \in \{1, \cdots, n\}}{\text{median}} \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{S}(\mathbf{x}_i, k)} ||\mathbf{x}_i - \mathbf{x}_j||_2, \tag{5.5}$$

where $\mathcal{S}(\mathbf{x}_i, k)$ contains k nearest neighbors of $\mathbf{x}_i$ in the PC $\mathbb{X}$. For the same example of generating random points on a sphere with radius r, for each PC $\mathbb{X}$ for embedding, we find the optimal radius r by solving:

$$\begin{aligned} \min_{r>0} \quad & \mathbb{E}_{\boldsymbol{\theta}, \boldsymbol{\phi}} \Big[\underset{i \in \{1, \cdots, n'\}}{\text{median}} \Big| D_{\text{knn}}(\mathbb{X}) - \frac{1}{k} \sum_{\mathbf{u}_j \in \mathcal{S}(\mathbf{u}_i, k)} ||\mathbf{u}_i - \mathbf{u}_j||_2 \Big| \Big] \\ \text{s. t.} \quad & \mathbf{u}_i = \mathbf{g}(r, \boldsymbol{\theta}, \boldsymbol{\phi}), \quad \forall i \in \{1, \cdots, n'\}, \end{aligned} \tag{5.6}$$

We practically solve (5.6) via a grid search. Note that for other geometries e.g. a cube, a cluster of points mimicking spurious points, etc., a different generator function would be chosen, possibly with different parameters to be optimized.

5.3.2.2 Spatial Location of Backdoor Points

Given the local geometry $\mathbb{U}$ fixed, the spatial location $\mathbf{c}$ should be specified following two criteria. **C1**: The backdoor mapping should be well learned by the victim classifier. **C2**: The backdoor points should be spatially *close* to the PC into which they are embedded, so that, in practice, the inserted backdoor object can be captured along with the object associated with the PC (i.e., in the same bounding box) by a 3D sensor.

Empirically, a backdoor attack with randomly located backdoor points is not guaranteed to be successful, as will be shown in Sec. 5.3.3.5. Thus, our attacker *optimizes* the spatial location $\mathbf{c}$ using a surrogate classifier $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ independently trained on a small dataset $\mathcal{D}_{\text{small}}$. If there is no attack, the landscape of the posterior probability function associated with the target class will likely be similar between the (unattacked) victim classifier (trained on the clean training set $\mathcal{D}_{\text{clean}}$ only) and the surrogate classifier. This is due to the fact that $\mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\text{small}}$ are generated i.i.d. according to the same distribution. In other words, for both classifiers, the target class posterior probability will likely be large for a *typical* target class PC, and be small for a *typical* source class PC. However, there is no guarantee for the two classifiers to have the same (or a very similar) decision boundary between the source class and the target class. This intuition is jointly illustrated in Fig. 5.3a and Fig. 5.3b.

The purpose of backdoor poisoning is to have the victim classifier learn to classify backdoor training samples to the target class (i.e. **C1**) – target class posterior probability for these PCs should also be large after the victim classifier's training on the poisoned training set. Thus, we optimize the spatial location $\mathbf{c}$ such that the embedding of backdoor points "pushes" the backdoor training PCs toward typical target class PCs. A simple illustration of the *expected* landscape of the target class posterior probability function (and the learned decision boundary) for the victim classifier being attacked is shown in Fig. 5.3c. For this classifier, a typical source class test PC embedded with similar backdoor points will also have a large target class posterior probability.

Formally, we denote the *surrogate classifier's* posterior probability function for the target class $t \in \mathcal{Y}$ as $p(t|\cdot) : \mathcal{X} \to [0, 1]$. For a non-target class PC, $p(t|\cdot)$ is supposed to increase when it is "pushed" towards the target class t. Thus, considering also **C2**, we find the minimum average distance from $\mathbf{c}$ to the source class PCs, such that any point[2] inserted at spatial location $\mathbf{c}$ induces these source class PCs to have at least a certain

[2]One can append $\mathbb{V}$ with arbitrary local geometry or even a single point at $\mathbf{c}$ to $\mathbb{X}$ when solving (5.7).

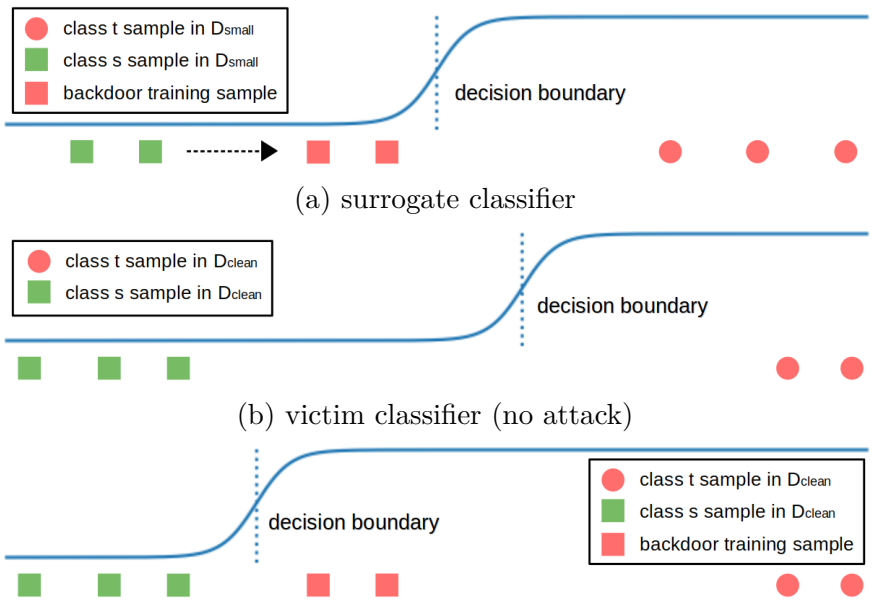


Figure 5.3: An intuitive illustration of class t posterior probability for: (a) The surrogate classifier trained on $\mathcal{D}_{\text{small}}$. Clean samples from class s are "pushed" towards class t with backdoor points embedding, and are labeled to class t. (b) The victim classifier without backdoor poisoning (trained on $\mathcal{D}_{\text{clean}}$). (c) The victim classifier trained on the backdoor poisoned training set $\mathcal{D}_{\text{train}}$ – the backdoor training samples influence the learned decision boundary.

level of average posterior probability for class t, i.e.:

$$\begin{aligned} \min_{\mathbf{c}\in\mathbb{R}^3} \quad & \frac{1}{|\mathcal{D}_s|}\sum_{(\mathbb{X},y)\in\mathcal{D}_s} d(\mathbf{c},\mathbb{X}) \\ \text{s. t.} \quad & \frac{1}{|\mathcal{D}_s|}\sum_{(\mathbb{X},y)\in\mathcal{D}_s} p(t|m_{\text{insert}}(\mathbb{X};\mathbb{V})) \geq \epsilon_0 + \epsilon, \end{aligned} \tag{5.7}$$

where $\mathcal{D}_s \subset \mathcal{D}_{\text{small}}$ is the subset of samples from the source class $s \in \mathcal{Y}$ possessed by the attacker. $d(\mathbf{c},\mathbb{X})$ measures the distance from point $\mathbf{c}$ to PC $\mathbb{X} \in \mathcal{X}$. In our experiments, we use $d(\mathbf{c},\mathbb{X}) = \min_{\mathbf{x}\in\mathbb{X}} ||\mathbf{c}-\mathbf{x}||_2$ for its simplicity and piece-wise differentiability in $\mathbf{c}$. $\epsilon_0 = \frac{1}{|\mathcal{D}_s|}\sum_{(\mathbb{X},y)\in\mathcal{D}_s} p(t|\mathbb{X})$ is the initial "soft" class confusion from class s to class t, which is usually close to zero due to the inevitable over-fitting on $\mathcal{D}_{\text{small}}$ during the surrogate classifier's training. Finally, ϵ is a small positive number describing how close the source class PCs should be "pushed" toward class t by inserting points at $\mathbf{c}$. Unlike the image domain, where a small, common perturbation can induce a group of images from one class to be misclassified to another class [172], the feasible set of (5.7) for even a moderately

Algorithm 5 Optimal spatial location for backdoor points.

Inputs: source class s, target class t, data subset $\mathcal{D}_s$, surrogate classifier f, ϵ and ϵ_0, step size δ, maximum iteration count $\tau_{\max}$, scaling factor α.
Initialization: $\mathbf{c}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\lambda^{(0)}$ set to a small positive number (e.g. 10^{-5}), $\mathbf{c}^* = \boldsymbol{\infty}$, $\rho^{(0)} = 0$.
for $\tau = 0 : \tau_{\max} - 1$ **do**
    $\mathbf{c}^{(\tau+1)} = \mathbf{c}^{(\tau)} - \delta \nabla_{\mathbf{c}} L_{\text{PCBA}}(\mathbf{c}^{(\tau)}, \lambda^{(\tau)})$
    $\rho^{(\tau+1)} = \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbb{X},y)\in\mathcal{D}_s} p(t|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}^{(\tau+1)}\}))$
    if $\rho^{(\tau+1)} \geq \epsilon_0 + \epsilon$ **then**
        $\lambda^{(\tau+1)} = \lambda^{(\tau)} \cdot \alpha$
        if $\sum_{(\mathbb{X},y)\in\mathcal{D}_s} \left[d(\mathbf{c}^{(\tau+1)}, \mathbb{X}) - d(\mathbf{c}^*, \mathbb{X})\right] < 0$ **then**
            $\mathbf{c}^* = \mathbf{c}^{(\tau+1)}$
    else
        $\lambda^{(\tau+1)} = \lambda^{(\tau)} / \alpha$
Outputs: $\mathbf{c}^*$

large ϵ may contain only spatial locations far apart from the original PCs in $\mathcal{D}_s$, which violates **C2**. Thus, in practice, ϵ is chosen to ensure that there is at least one solution with sufficiently small objective value for (5.7) (e.g. $\epsilon = 0.02$ in our experiments).

We solve (5.7) using Alg. 5, where

$$L_{\text{PCBA}}(\mathbf{c}, \lambda) = \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbb{X},y)\in\mathcal{D}_s} \left[\lambda \cdot d(\mathbf{c}, \mathbb{X}) - \log p(t|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}\}))\right] \tag{5.8}$$

is the Lagrangian of (5.7), with a single point inserted at $\mathbf{c}$, and with the logarithm used for better smoothness. λ is updated automatically (using a scaling factor $\alpha > 1$) to constrain the optimization variables in the feasible set (as an alternative to projection which is hard to realize here). $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard normal distribution used to initialize $\mathbf{c}$ – the PCs are usually aligned to the origin for classification [194]. To avoid poor local optima, one can perform Alg. 5 multiple times, with different initialization, and pick the best solution to (5.7).

5.3.3 Experiments

5.3.3.1 Datasets

Like existing PC TTE attacks [202, 204, 206], we use the aligned benchmark dataset ModelNet40 [210] for our experiment. ModelNet40 contains 12311 CAD models (2048 points for each PC) from 40 common object categories. Following the original train-test

split of ModelNet40, 2468 PCs are used for testing. From the remaining 9843 PCs, we randomly choose 1000 PCs as the "small dataset" ($\mathcal{D}_{\text{small}}$) possessed by the attacker. The remaining 8843 PCs are possessed by the trainer ($\mathcal{D}_{\text{clean}}$) and are not accessible to the attacker. Additionally, we consider a practical street view LiDAR dataset KITTI [211]. From each scene, we extract PCs corresponding to labeled objects inside their bounding boxes provided with the dataset and align them. Due to high class imbalance of the original KITTI dataset, we construct two (super) classes: a "vehicle" class consists of "car", "van", and "truck" from the original dataset; a "human" class consists of "pedestrian" and "cyclist" from the original dataset. We consider PCs with no less than 256 points and randomly keep 256 points for each PC. Also, we keep a subset of PCs for the "vehicle" class such that the two classes have equal number of samples. Consequently, we obtain 2662 PCs evenly distributed in the two classes – 200 are possessed by the attacker, 1800 are possessed by the trainer, and 662 are used for testing.

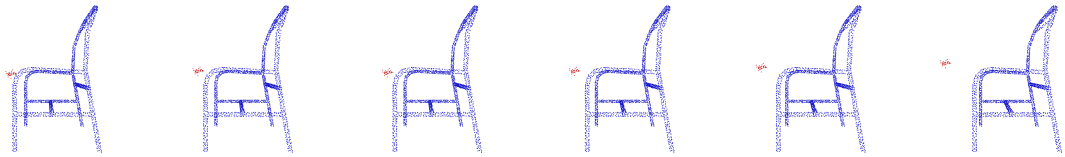
5.3.3.2 Attack Implementation

We implemented 36 attacks involving 9 (source, target) class pairs in total for the two datasets – for each class pair, we create 4 attacks with different types of local geometry for the embedded backdoor points.

Specify source and target classes: For ModelNet40, we arbitrarily chose 7 (source, target) class pairs, which are: (chair, toilet), (vase, curtain), (laptop, chair), (nigh stand, table), (sofa, monitor), (cone, lamp), (airplane, wardrobe). For KITTI, we consider the only two ordered class pairs: (human, vehicle) and (vehicle, human). We name these 9 class pairs as $\mathbf{P_1}$, $\mathbf{P_2}$, ..., $\mathbf{P_9}$ respectively for brevity.

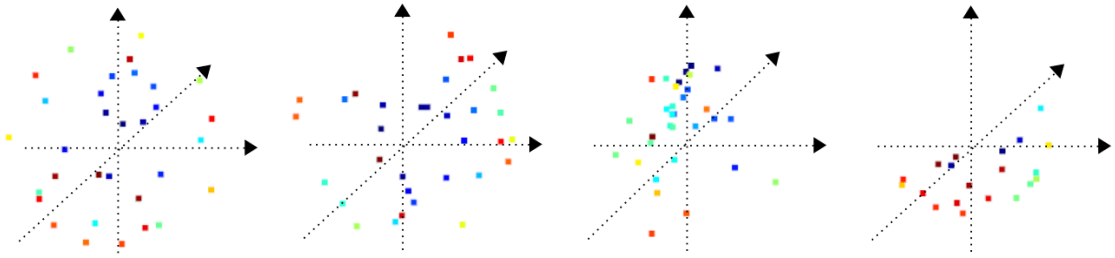
Train a surrogate classifier: For each dataset, we trained a PointNet with the same architecture in [194] on the PCs possessed by the attacker. Training was performed for 250 epochs with batch size 32 and learning rate 10^{-3} (with 0.5 decay per 20 epochs). 2048 points and 256 points per PC are used for ModelNet40 and KITTI, respectively.

Specify the spatial location of backdoor points: For the four attacks associated with each (source, target) class pair, we specified one *common* spatial location for backdoor point embedding using Alg. 5 and the surrogate classifier trained on its associated dataset. The parameters for the attacker's optimization were set to $\epsilon = 0.02$, $\delta = 0.01$, $\tau_{\max} = 3000$, $\alpha = 1.5$. In particular, although ϵ is numerically small, there is already a moderate distance between the optimal spatial location (solution to (5.7)) and the PCs used for backdoor embedding, as shown in Fig. 5.4. Larger ϵ may cause the embedded backdoor points to be too far from the PC to be captured in the same bounding box by a 3D



(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.025$ (d) $\epsilon = 0.05$ (e) $\epsilon = 0.1$ (f) $\epsilon = 0.2$

Figure 5.4: Example PCs with backdoor points embedded at the optimal spatial location obtained for $\epsilon \in \{0.005, 0.01, 0.025, 0.05, 0.1, 0.2\}$. The larger the ϵ, the further the backdoor points (in red) are apart from the object of the PC, i.e. the chair (in blue).



(a) GS (b) RS (c) RP (d) HS

Figure 5.5: Illustration of the four types of local geometry. GS is a non-optimized geometry; while RS, RP, and HS are optimized geometries with stochastic generators.

sensor. The choices of the other three parameters are not critical to the performance of our PCBA.

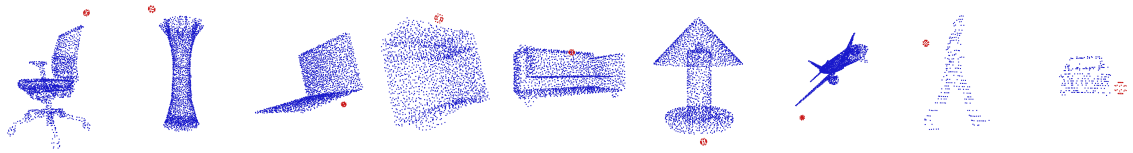
Specifying the local geometry of backdoor points: For each class pair, we created four attacks with the following four different types of local geometry respectively. We set $k = 4$ in Eq. (5.5) and (5.6) for local geometry optimization. Examples of these local geometries are shown in Fig. 5.5.

1) **GS**: 32 points uniformly spaced on a sphere, generated by Eq. (5.4) with deterministic $\theta \in \{\frac{1}{8}\pi, \frac{3}{8}\pi, \frac{5}{8}\pi, \frac{7}{8}\pi\}$, and $\phi \in \{\frac{1}{8}\pi, \frac{3}{8}\pi, \cdots, \frac{15}{8}\pi\}$. Radius is manually set to $r = 0.04$ for scene-plausibility, but *without* optimizing (5.6).

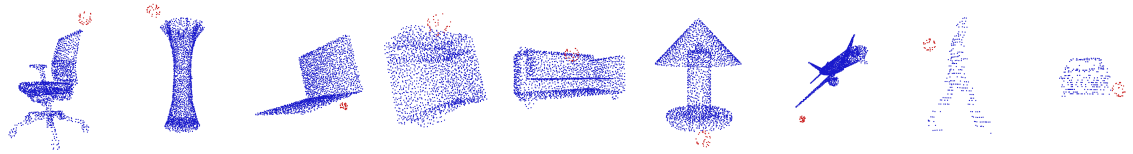
2) **RS**: 32 points randomly distributed on a sphere generated by Eq. (5.4), with $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ uniformly sampled from $[0, \pi]$ and $[0, 2\pi]$ respectively. Radius r is obtained by solving (5.6).

3) **RP**: 32 points randomly distributed in a ball generated in the same way as RS, except that r is now a random variable uniformly distributed in $[0, r_{\max}]$, where $r_{\max}$ is optimized instead of r in (5.6).

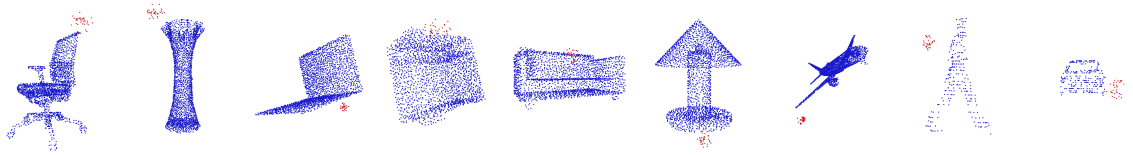
4) **HS**: Points randomly distributed on a half sphere with random orientation (to mimic a surface of a ball facing a 3D scanner) – generated from RS by keeping points having



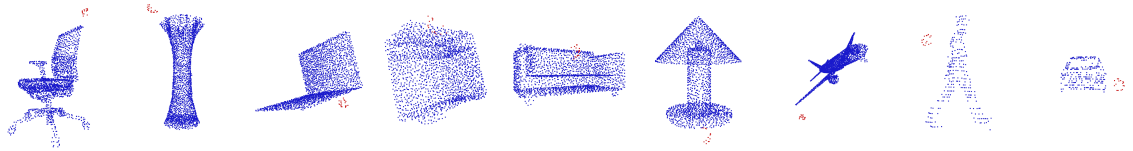
(a) Example backdoor training samples for local geometry GS, for class pairs P1-P9.



(b) Example backdoor training samples for local geometry RS, for class pairs P1-P9.



(c) Example backdoor training samples for local geometry RP, for class pairs P1-P9.



(d) Example backdoor training samples for local geometry HS, for class pairs P1-P9.

Figure 5.6: Example backdoor training samples for the 36 attacks.

positive inner product with a random vector.

Create backdoor training samples: For each attack, with the specified spatial location and local geometry, we generated backdoor training samples using a subset of clean PCs possessed by the attacker from the source class, following Eq. (5.3) and (5.2). Example backdoor training samples are shown in Fig. 5.6. For ModelNet40 and KITTI, 15 and 30 backdoor training samples are generated for poisoning the training set, respectively.

5.3.3.3 Training

Learning the victim classifier is performed by the trainer, on the poisoned training set $\mathcal{D}_{\text{train}}$. Based on the assumptions in Sec. 5.3.1, the entire training process is not accessible to the attacker. Like PC TTE attacks [202, 203], we consider three DNN architectures for the victim classifier – PointNet [194], PointNet++ [195], and DGCNN [196]. We use the same DNN architecture and training protocol for these models as described in their original papers. Notably, for ModelNet40, each PC is *preprocessed* by randomly sampling

		ModeNet40				KITTI			
		ASR (avg)	ASR (min)	ACC (avg)	ACC (min)	ASR (avg)	ASR (min)	ACC (avg)	ACC (min)
PointNet [194]	GS	94.0	91.9	88.7	88.2	92.8	89.1	99.3	99.2
	RS	96.0	93.0	88.7	88.2	93.4	87.3	99.4	99.4
	RP	94.9	90.0	88.6	87.8	94.0	90.9	99.4	99.1
	HS	96.0	93.0	88.6	88.2	91.2	91.2	99.5	99.5
PointNet++ [195]	GS	94.6	89.5	91.4	91.0	95.9	92.7	99.5	99.5
	RS	96.9	92.0	91.0	90.2	93.1	87.6	99.4	99.4
	RP	96.9	95.0	91.0	90.2	93.5	89.7	99.7	99.5
	HS	93.7	88.0	91.4	91.1	88.6	87.6	99.5	99.5
DGCNN [196]	GS	93.2	90.0	92.9	90.8	96.7	95.5	99.5	99.5
	RS	93.9	87.0	91.1	90.7	95.0	91.5	99.8	99.7
	RP	96.1	90.0	91.0	90.6	96.4	93.1	99.6	99.4
	HS	93.7	87.0	91.0	90.8	92.8	90.6	99.5	99.4

Table 5.1: Average and minimum ASR and ACC (in %), respectively, over the 9 attacks (for class pairs P1, P2, ..., P9), for the 4 local geometries (GS, RS, RP, and HS), the 2 datasets (ModelNet40 and KITTI), and the three victim classifier architectures (PointNet, PointNet++, and DGCNN). All attacks are successful with ASR $\geq 87\%$.

1024 points before feeding to the classifier *(both during training and test)*. Similarly, 128 points are randomly chosen to remain for each PC for KITTI. As a benchmark, without poisoning, the test accuracy of the trained PointNet, PointNet++, and DGCNN are 88.5%, 91.5%, and 91.4% for ModeNet40; 99.5%, 99.7%, and 99.7% for KITTI.

5.3.3.4 Attack Performance Evaluation

Again, the performance of a backdoor attack is jointly evaluated by attack success rate (ASR) and clean test accuracy (ACC). A successful backdoor attack should have a high ASR and negligible degradation in ACC compared with the clean benchmarks in Sec. 5.3.3.3. Thus, *all 36 attacks are successful* (with all ASRs $\geq 87\%$) regardless of the victim classifier's architecture, as shown in Tab. 5.1. Apart from that, we observe that for each class pair, *with the same optimal spatial location, the choice of the local geometry does not significantly affect the learning of the backdoor mapping.* Especially for attacks with geometry RP, the backdoor points inserted to each PC have high randomness; but the ASR for these attacks are still uniformly high. For physically implemented backdoor attacks, this property allows more freedom in choosing the geometry of the inserted object to achieve scene-plausibility. Also, since high ASRs are achieved when each test PC is sub-sampled to 1024 points – nearly half of the points are removed – *our PCBA is robust to test-time sub-sampling.* Moreover, in Fig. 5.7, we show ASR curves for the

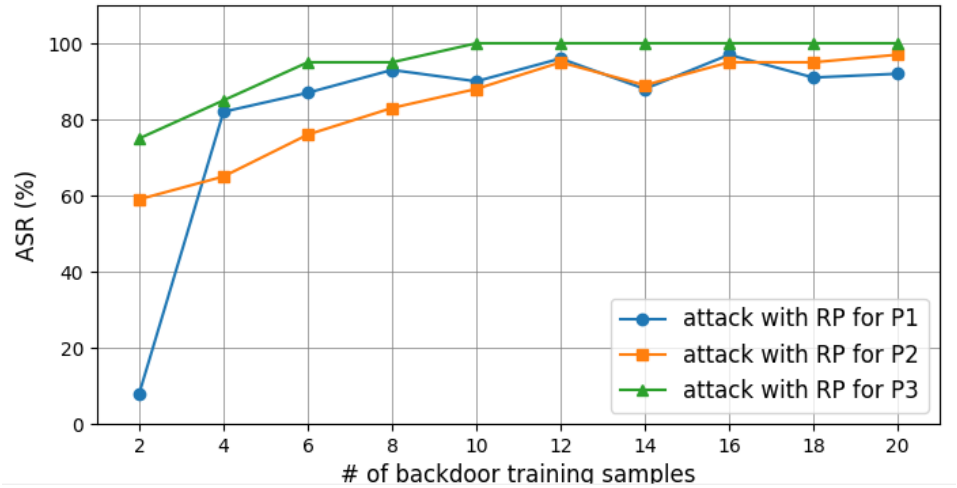


Figure 5.7: ASR versus number of backdoor training samples for attacks with local geometry RP associated with class pairs P1, P2, and P3 (for example). With merely 8 backdoor training samples, all three attacks achieve ASR $> 80\%$.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
PointNet	11.4	32.0	10.5	25.0	44.7	45.5	39.4	18.2	28.5
PointNet++	0	45.0	0	16.7	1.1	18.2	0	1.3	0
DGCNN	12.4	40.5	38.9	20.8	7.2	27.3	11.6	0	0.3

Table 5.2: Success rate of targeted PC TTE attacks (for class pairs P1-P9) transferred from the surrogate classifier, for victim classifier architectures PointNet, PointNet++, and DGCNN – PC TTE attacks transfer poorly.

three attacks with local geometry RP for class pairs P1, P2, and P3, over a range of number of backdoor training samples used for poisoning the victim classifier's training set. Our PCBA is effective, with only a few backdoor training samples inserted in the training set containing 8843 clean PCs; thus it is also very stealthy.

Additionally, we compare our PCBA with PC TTE attacks implemented by point addition in the same scenario described in Sec. 5.3.1. Following [202], for each of the 9 class pairs, we created adversarial PCs by inserting 32 points to test PCs from the source class. The locations for the inserted points are optimized using the *surrogate classifier* such that the adversarial PCs are classified to the target class by the surrogate classifier. As shown in Tab. 5.2, these adversarial PCs cannot reliably "fool" the victim classifier trained on clean PCs possessed by the trainer – PC TTEs transfer poorly; thus, they are less threatening than our PCBA in cases where the victim classifier is not accessible to the attacker.

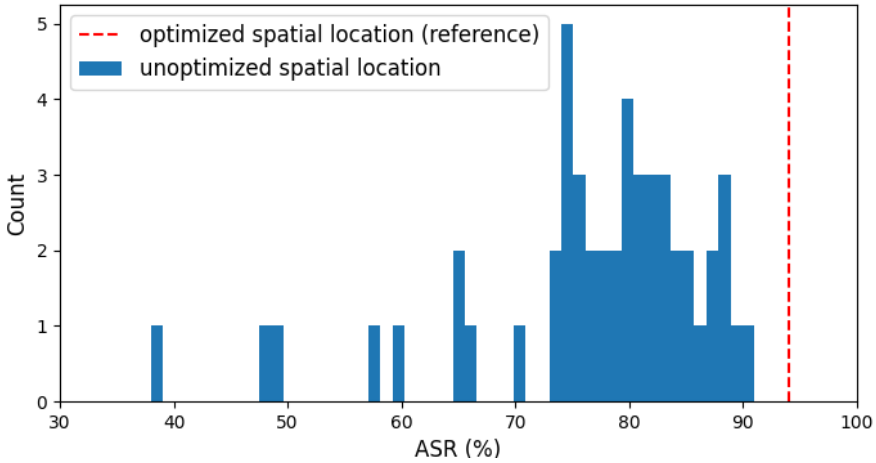


Figure 5.8: Histogram of ASR for 50 attacks created without spatial location optimization – most of them are clearly outperformed by our BA with optimized spatial location.

	GS	RS	RP	HS
P1	49.0 (94.0)	90.0 (93.0)	88.0 (94.0)	81.0 (93.0)
P2	9.0 (93.0)	97.0 (98.0)	96.0 (96.0)	87.0 (97.0)
P3	51.0 (95.0)	100 (100)	100 (100)	90.0 (100)
P4	8.1 (91.9)	94.0 (95.0)	95.3 (96.5)	81.4 (95.3)
P5	2.0 (95.0)	90.0 (95.0)	87.0 (90.0)	84.0 (93.0)
P6	35.0 (95.0)	90.0 (95.0)	90.0 (90.0)	95.0 (100)
P7	63.0 (94.0)	94.0 (96.0)	98.0 (98.0)	88.0 (94.0)
P8	97.0 (96.4)	99.7 (99.4)	98.8 (97.0)	92.4 (91.2)
P9	87.9 (89.1)	85.2 (87.3)	90.9 (90.9)	90.6 (91.2)

Table 5.3: ASR (in %) for the 36 attacks for victim classifier architecture PointNet, when the PC AD in [4] is deployed during test. ASRs (in %) without AD deployed are shown in parenthesis for reference.

5.3.3.5 Backdoor Points with Random Spatial Location

Here, we show the *necessity of spatial location optimization* for our PCBA. For class pair P1 and local geometry GS, we created 50 attacks in the same way as described in Sec. 5.3.3.2, but *without* spatial location optimization. In particular, for each attack, we pick a random spatial location $\mathbf{c} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and scale it such that the average distance from the scaled $\mathbf{c}$ to the source class PCs (i.e. objective of (5.7)) is the same as for the optimized spatial location obtained for the attack associated with P1 and GS. As shown in Fig. 5.8, all 50 attacks (with maximum ASR 91.0%) have smaller ASR than the attack with the optimized spatial location (with ASR 94.0%). Moreover, some of the 50 attacks are not reliable, with low ASR.

5.3.3.6 BA against PC Anomaly Detectors (ADs)

Detectors for image BAs, e.g. our basic RED, highly depend on the format of the backdoor pattern; hence they are not applicable to our PCBA. Still, we consider the state-of-the-art defense against PC TTE attacks – a PC AD in [4], which aims to remove points inserted/perturbed by a TTE attacker. It measures the kNN distance (with $k = 2$) for each point in a PC and removes points with abnormally high or low kNN distance (falling outside of ± 1.1 standard deviation interval around the average). In Tab. 5.3, we show ASR of the 36 attacks for victim classifier being a PointNet, when the above PC AD is deployed during testing. For the non-optimized geometry GS, most attacks are no longer reliable because the backdoor points embedded in many test PCs are entirely removed. For the three optimized geometries (RS, RP, and HS), the PC AD only causes limited degradation in ASR compared with the no detector case. There is still a 81.0% minimum ASR for the 27 attacks for these three local geometries.

5.4 PC-RED: Reverse-Engineering-Based Detection of PCBA without Access to The Training Set

5.4.1 Key Ideas of PC-RED

Our PC-RED is inspired by the basic RED in Sec. 3.3, following the same set of assumptions for the post-training defense scenario introduced in Sec. 2.3.2.2. PC-RED detects the presence of any backdoor attacks, infer the source class and the target class of the attack, and estimate the common spatial location for the attacker-specified backdoor points. Like REDs for images, PC-RED consists of a backdoor pattern reverse-engineering step (to address the assumption that there is no access to the training set) and an unsupervised detection inference step (to address the assumption that there is no clean classifiers for reference). However, extending REDs for images to detect PCBA is not trivial. First, PCBA uses a special backdoor pattern – backdoor points – customized for PCs. Second, PC classifiers exhibits relatively strong robustness to modifications of the input PC [202], such that typical backdoor pattern reverse-engineering problems formulation for image RED may not successfully "push" PCs from one class to another (across the decision boundary between the two classes) for some class pairs. Third, also for some classes, there may exist an "intrinsic backdoor" – there exists a spatial location close to most source class PCs, such that a single inserted point can cause most of

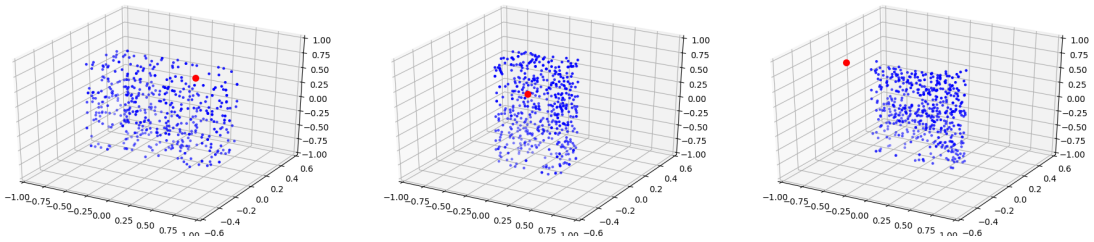


Figure 5.9: Example of intrinsic backdoor from one of our experiments (P_6-PN in Tab. 5.4): for PCs from the same source class, spatial locations estimated *sample-wise* (in red) are all close to these PCs (in blue), but are different from PC to PC.

these PCs to be misclassified to a common target class, irrespective of the existence of a backdoor attack.

Considering the above challenges, with the properties of the backdoor points used by PCBA, PC-RED is designed based on the following intuitions. **I1**: For *most* non-backdoor class pairs, a *common* set of inserted points that induces high group misclassification from source class to target class will be spatially *far* from the points of source class PCs; but for backdoor class pair denoted by (s^*, t^*), the existence of the *backdoor mapping* guarantees the existence of a *common* spatial location close to the source class PCs (likely near the true spacial location $\mathbf{c}^*$ of the inserted backdoor points), where a set of inserted points can induce most source class PCs to be misclassified to the target class. **I2**: A few non-backdoor class pairs may be associated with an *"intrinsic backdoor"*. For these class pairs, like a true backdoor class pair, there exists a spatial location close to most PCs from the source class, such that a common set of inserted points there will induce most of these PCs to be misclassified to the target class. However, such a spatial location, different from $\mathbf{c}^*$ (specified by the attacker) for the true backdoor, will likely be *close to* the points of most *target class* PCs. **I3**: Unlike backdoor mappings caused by attack with a *single common* spatial location $\mathbf{c}^*$, an intrinsic backdoor is likely due to the source and target classes being *"semantically"* similar, such that there may exist *several* intrinsic backdoor points for a given non-backdoor class pair, with each one close to source class PCs (Fig. 5.9). In this case (for a source class with an intrinsic backdoor), the *closest* (*sample-wise*) spatial location for a set of inserted points to induce a (*sample-wise*) misclassification to the target class can be different for different PCs from the same source class.

5.4.2 Pattern Estimation of PC-RED

To find backdoor class pairs if there are any, based on I1, we need to first find, for each class pair, the *common* spatial location *closest* to the source class PCs such that a set of points inserted there induces most of these PCs to be misclassified to the target class, i.e., we need to reverse-engineer the backdoor pattern. According to the results in Sec. 5.3.3.4, the backdoor mapping mostly relies on the spatial location $\mathbf{c}^*$ but not the local geometry $\mathbb{U}^*$ of the inserted points. Thus, we can focus on reverse-engineering the spatial location of the backdoor points with arbitrary local geometry – for simplicity, we insert a *single* point at the spatial location. Formally, we aim to solve the following problem for each class pair $(s,t) \in \mathcal{Y} \times \mathcal{Y}$:

$$\begin{aligned} \min_{\mathbf{c}\in\mathbb{R}^3} \quad & \sum_{\mathbb{X}\in\mathcal{D}_s} d(\mathbf{c},\mathbb{X}) \\ \text{s. t.} \quad & \frac{1}{|\mathcal{D}_s|}\sum_{\mathbb{X}\in\mathcal{D}_s} \mathbb{1}\big(f(m_{\text{insert}}(\mathbb{X};\{\mathbf{c}\})) = t\big) \geq \pi. \end{aligned} \tag{5.9}$$

Note that this formulation is similar to problem (5.7), with the conditional posterior probability replaced by the "hard" class label assignment. However, instead of estimating a spatial location for a designated backdoor class pair on a surrogate classifier to devise an attack, here, $f : \mathcal{X} \to \mathcal{Y}$ is the pre-trained classifier to be inspected, and we estimate a spatial location for each class pair. Moreover, $\mathcal{D}_s$ here represents the small set of samples from class s used by the defender for detection, which is different from the small set of samples possessed by the attacker for creating the attack. $d(\mathbf{c},\mathbb{X})$ measures the distance from $\mathbf{c}$ to $\mathbb{X}$, where the same definition below (5.7) is used here. Finally, like in backdoor pattern reverse-engineering problems formulated for image REDs, $\pi \in [0,1]$ here is a group misclassification fraction which is typically set large (e.g. $\pi = 0.9$).

However, problem (5.9) is difficult to solve in practice. First, the indicator function in the constraint is not differentiable. Second, unlike image backdoor patterns (e.g. an additive perturbation) typically constrained by some range of valid pixel values, the search space for a point spacial location is unlimited. Also, due to the generally strong adversarial robustness of recent PC classifiers [194, 202], for many class pairs, there may not even exist a spatial location reasonably close to the source class PCs, where an inserted point can induce high (e.g. at least π) group misclassification to the target class. For these class pairs, even finding a solution just to satisfy the constraint of (5.9) may be infeasible in practice. To address the two challenges above, we perform backdoor

Algorithm 6 Spatial location estimation for class $s \in \mathcal{Y}$.

Inputs: data subset $\mathcal{D}_s$ for detection, classifier f to be inspected, target misclassification fraction π, step size δ, maximum iteration count $\tau_{\max}$, scaling factor α.
Initialization: $\mathbf{c}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\lambda^{(0)}$ set to a small positive number (e.g. 10^{-5}), $\hat{\mathbf{c}}(s) = \boldsymbol{\infty}$, $\rho^{(0)} = 0$.
for $\tau = 0 : \tau_{\max} - 1$ **do**
    $\mathbf{c}^{(\tau+1)} = \mathbf{c}^{(\tau)} - \delta \nabla_{\mathbf{c}} L_{\text{PC-RED}}(\mathbf{c}; \mathcal{D}_s, \lambda^{(\tau)})|_{\mathbf{c}=\mathbf{c}^{(\tau)}}$
    $\rho^{(\tau+1)} = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbb{X} \in \mathcal{D}_s} \mathbb{1}\big(f(m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}^{(\tau+1)}\})) \neq s\big)$
    if $\rho^{(\tau+1)} \geq \pi$ **then**
        $\lambda^{(\tau+1)} = \lambda^{(\tau)} \cdot \alpha$
        if $\sum_{\mathbb{X} \in \mathcal{D}_s} \big[d(\mathbf{c}^{(\tau+1)}, \mathbb{X}) - d(\hat{\mathbf{c}}(s), \mathbb{X})\big] < 0$ **then**
            $\hat{\mathbf{c}}(s) = \mathbf{c}^{(\tau+1)}$
        else $\lambda^{(\tau+1)} = \lambda^{(\tau)} / \alpha$
Outputs: $\hat{\mathbf{c}}(s)$

pattern estimation for each putative *source class* by minimizing a *differentiable surrogate* objective. In particular, for each source class $s \in \mathcal{Y}$, we search for the closest spatial location to PCs from class s, such that a point inserted there causes at lease π fraction of these PCs to be misclassified to any class *other than* s (untargeted misclassifications). Formally, we minimize loss:

$$L_{\text{PC-RED}}(\mathbf{c}; \mathcal{D}_s, \lambda) = \sum_{\mathbb{X} \in \mathcal{D}_s} \big[h(s|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}\})) - \max_{k \neq s} h(k|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}\}))\big] + \lambda \sum_{\mathbb{X} \in \mathcal{D}_s} d(\mathbf{c}, \mathbb{X}) \tag{5.10}$$

over $\mathbf{c}$ using Alg. 6. The first sum in Eq. (5.10) is inspired by the *untargeted* CW loss in [43], where $h(k|\mathbb{X})$ is the output of $\mathbb{X} \in \mathcal{X}$ for class $k \in \mathcal{Y}$ directly prior to the softmax activation, which is smoother for optimization than the class posterior constrained in interval $[0, 1]$. Using such untargeted loss, we let the source class PCs "vote" for a target class by:

$$\hat{t}(s) = \underset{k \neq s}{\text{argmax}} \sum_{\mathbb{X} \in \mathcal{D}_s} \mathbb{1}\big(f(m_{\text{insert}}(\mathbb{X}; \{\hat{\mathbf{c}}(s)\})) = k\big) \tag{5.11}$$

where $\hat{\mathbf{c}}(s)$ is the spatial location estimated for class s. Compared with our basic RED that estimates a backdoor pattern for each class pair, our PC-RED here performs $\mathcal{O}(K)$ backdoor pattern estimations instead of $\mathcal{O}(K^2)$ (where $K = |\mathcal{Y}|$ is the number of classes); thus it is more *efficient* for large K (according to our discussion on the efficiency of RED in Sec. 3.6). The second sum in Eq. (5.10) is a regularizer which constrains the distance

of $\mathbf{c}$ to the points of the source class PCs. The coefficient λ is automatically adjusted according to line 6-10 of Alg. 6.

In addition to the *group* backdoor pattern (i.e. a *common* spatial location) estimation above, based on intuition I3, for each putative source class $s \in \mathcal{Y}$, we also need to estimate a *sample-wise* spatial location for each $\mathbb{X} \in \mathcal{D}_s$, given the estimated target class $\hat{t}(s)$. Formally, we minimize the following loss using the same Alg. 6:

$$L_{\text{PC}-\text{RED}-\text{SW}}(\mathbf{c}; \mathbb{X}, \lambda) = h(s|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}\})) - h(\hat{t}(s)|m_{\text{insert}}(\mathbb{X}; \{\mathbf{c}\})) + \lambda d(\mathbf{c}, \mathbb{X}) \quad (5.12)$$

with $\mathcal{D}_s$ replaced by a set of a single PC $\{\mathbb{X}\}$, and with the loss in line 4 replaced by Eq. (5.12). We denote the estimated sample-wise (SW) spatial location for $\mathbb{X} \in \mathcal{D}_s$ as $\hat{\mathbf{c}}_{\text{sw}}(s, \mathbb{X})$.

5.4.3 Detection Inference of PC-RED

Slightly different with our basic RED, we derive a detection statistic for each putative source class instead of for each class pair. The detection statistic is composed of three basic statistics (obtained from the backdoor pattern estimation step above) corresponding to the three intuitions in Sec. 5.4.1 respectively. For each $s \in \mathcal{Y}$, we get: 1) the average distance from the estimated spatial location to points of source class PCs:

$$r_{\text{s}}(s) = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbb{X} \in \mathcal{D}_s} d(\hat{\mathbf{c}}(s), \mathbb{X}); \quad (5.13)$$

2) the average distance from the estimated spatial location to points of PCs from the estimated target class $\hat{t}(s)$:

$$r_{\text{t}}(s) = \frac{1}{|\mathcal{D}_{\hat{t}(s)}|} \sum_{\mathbb{X} \in \mathcal{D}_{\hat{t}(s)}} d(\hat{\mathbf{c}}(s), \mathbb{X}); \quad (5.14)$$

and 3) a normalized similarity score:

$$w(s) = \frac{z(s) - \min\limits_{k \in \mathcal{Y}} z(k)}{\max\limits_{k \in \mathcal{Y}} z(k) - \min\limits_{k \in \mathcal{Y}} z(k)}, \quad (5.15)$$

where

$$z(k) = \frac{1}{|\mathcal{D}_k|} \sum_{\mathbb{X} \in \mathcal{D}_k} \frac{\hat{\mathbf{c}}(k) \cdot \hat{\mathbf{c}}_{\text{sw}}(k, \mathbb{X})}{|\hat{\mathbf{c}}(k)| \, |\hat{\mathbf{c}}_{\text{sw}}(k, \mathbb{X})|} \quad (5.16)$$

is the average cosine similarity[3] between the estimated sample-wise spatial location for each $\mathbb{X} \in \mathcal{D}_k$ and the estimated group spatial location for class $k \in \mathcal{Y}$. The normalization limits the similarity score in interval $[0, 1]$ for generalization to different domains.

According to intuition I1, $r_{\mathrm{s}}(s)$ will likely be large if $(s, \hat{t}(s))$ is not a backdoor class pair; otherwise, $r_{\mathrm{s}}(s)$ will likely be small. If for some class s, $(s, \hat{t}(s))$ is associated with an intrinsic backdoor mapping, such that $r_{\mathrm{s}}(s)$ is abnormally small, based on I2 and I3, $r_{\mathrm{t}}(s)$ or $w(s)$ (or both) will likely be abnormally small. Thus, for each putative source class $s \in \mathcal{Y}$, we compute the combined detection statistic:

$$r(s) = w(s)\frac{r_{\mathrm{t}}(s)}{r_{\mathrm{s}}(s)}, \tag{5.17}$$

which will be abnormally large only if $(s, \hat{t}(s))$ is a backdoor class pair.

Our inference is based on an *unsupervised* anomaly detection. We check among the statistics for all $s \in \mathcal{Y}$ if there exists an abnormally large one. Ideally, like image REDs, e.g. our basic RED, we can fit a null distribution using all statistics excluding the *largest one* and evaluate its *atypicality* under the null using the maximum *order statistic* p-value, which is generally insensitive to the number of statistics used for detection. However, like image backdoor attacks, PCBA may cause *collateral damage*: recall from Sec. 3.3.6.3 that a backdoor mapping with the same backdoor pattern may be learned for some class pair (s, t^*) with $s \in \mathcal{Y} \setminus \{s^*, t^*\}$. In such case, there may be *multiple* abnormally large statistics associated with *the same target class* t^* but different source classes; thus, the null distribution estimated with only the largest statistic being excluded may be biased. To solve this problem, we exclude statistics for all $s \in \mathcal{Y}$ such that $\hat{t}(s) = \hat{t}(s_{\max})$ when estimating the null, where $s_{\max} = \arg\max_{k \in \mathcal{Y}} r(k)$. Given all statistics in $[0, \infty)$, similar to our basic RED, we choose a single-tailed parametric density form (e.g. Gamma distribution) for our null distribution, with cdf G, such that any abnormally large statistics (likely corresponding to backdoor class pairs) will likely appear in the tail (e.g. Fig. 5.10). Then the *maximum order statistic p-value* is:

$$\mathrm{pv} = 1 - G(r(s_{\max}))^{K-J}, \tag{5.18}$$

where J is the number of statistics being excluded when estimating the null distribution. A detection threshold θ is chosen (e.g. the classical $\theta = 0.05$), such that a backdoor attack is detected with confidence $1 - \theta$ if $\mathrm{pv} < \theta$. When a backdoor attack is detected,

[3]PCs are usually aligned to the origin for classification.

	$1/r_s$	r_t/r_s	w/r_s	$r = w \cdot r_t/r_s$
P_1-PN	(**6.2e^{-3}**, **0.36**)	(**4.5e^{-2}**, 9.2e^{-6})	(**1,7e^{-7}**, **0.19**)	(**3.3e^{-3}**, **0.38**)
P_2-PN	(**3.8^{-3}**, **0.16**)	(**u.f.**, **0.32**)	(**3.5e^{-3}**, **0.26**)	(**u.f.**, **0.19**)
P_3-PN	(**4.3e^{-15}**, **0.33**)	(**6.1e^{-6}**, 9.8e^{-2})	(**u.f.**, **0.27**)	(**u.f.**, **0.20**)
P_4-PN	(**2.2e^{-7}**, 2.6e^{-2})	(**2.8e^{-3}**, **0.58**)	(**5.6e^{-9}**, 9.2e^{-3})	(**u.f.**, **0.22**)
P_5-PN	(0.24, **0.11**)	(0.12, **0.19**)	(**1.4e^{-2}**, 6.1e^{-2})	(**5.4e^{-2}**, **0.27**)
P_6-PN	(0.24, 1.6e^{-2})	(0.21, **0.60**)	(**1.4e^{-2}**, 2.6e^{-2})	(**7.6e^{-4}**, **0.33**)
P_7-PN	(**4.3e^{-3}**, 9.7e^{-2})	(**6.7e^{-5}**, 9.0e^{-3})	(**5.5e^{-9}**, 7.0e^{-3})	(**u.f.**, 9.3e^{-2})
P_1-PN++	(**u.f.**, 8.2e^{-6})	(**u.f.**, **0.99**)	(**u.f.**, **0.94**)	(**5.5e^{-13}**, **0.99**)
P_1-DGCNN	(**4.4e^{-5}**, 4.3e^{-2})	(0.10, **0.59**)	(0.22, 2.9e^{-2})	(**1.9e^{-3}**, **0.18**)

Table 5.4: Order statistic p-value (pv), in form of (attack pv, clean pv), for nine pairs of classifiers being attacked with the associated clean classifier, for the statistic r used by our detector, and for three alternative statistics ($1/r_s$, r_t/r_s, and w/r_s). Attacks are associated with class pairs $P_1, ..., P_7$ in Sec. 5.3.3.2; classifier architectures include PointNet (PN), PointNet++ (PN++), and DGCNN. "u.f." represents "underflow" for numbers in range $(0, 10^{-323})$. Successful detections (with $\phi = 0.05$) are in bold.

$\hat{t}(s_{\max})$ is inferred as the target class.

5.4.4 Experiments

5.4.4.1 Devising Backdoor Attacks

We evaluate PC-RED on the benchmark dataset ModelNet40 [210]. In particular, we consider the same attacks of "$P_1, ..., P_7$" in Sec. 5.3.3.2 with the local geometry "RS" shown in Fig. 5.5. Other attack configurations and training configurations are the same as in Sec. 5.3.3.2. To evaluate false detections of our PC-RED, for each classifier being attacked, we also train a classifier with no backdoor attack, using the same training configurations.

5.4.4.2 Defense Performance Evaluation

Detector configurations: Following the general assumptions for post-training backdoor defense, for each class, we randomly select 10 clean PCs that are correctly classified from the ModelNet40 data set, to form the clean set used for detection – these PCs are not used for training. For backdoor pattern estimation (both group-wise and sample-wise) using Alg. 6, we set $\pi = 0.9$, $\delta = 0.1$, $\tau_{\max} = 3$k, and $\alpha = 1.5$. These choices are not critical to the performance of our detector, and can be easily chosen to minimize the loss value at convergence. We also perform 10 random initializations and pick the best

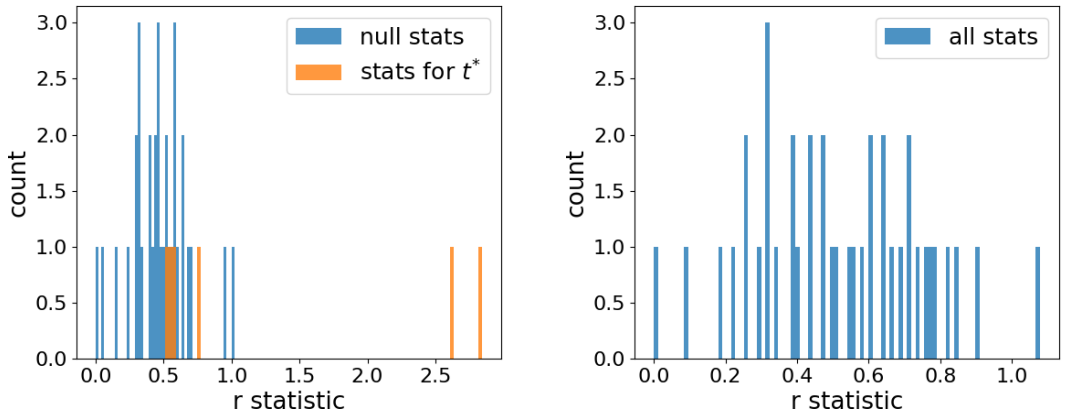


Figure 5.10: Histogram of r statistics for the classifier with backdoor attack (left) and its associated clean classifier without backdoor attack (right). For the attack case, statistics for two source classes "voting" to the backdoor target class t^* are abnormally large.

solution, which is a common practice for solving highly non-convex problems.

Detection performance: For each classifier being attacked and its associated clean classifier, we report the order statistic p-values obtained using our statistic $r = w \cdot r_{\mathrm{t}}/r_{\mathrm{s}}$ (Eq. (5.17)) in Tab. 5.4 (last column). In general, the p-values are large for most clean classifiers and are small when there is a PCBA, as we expect. Applying the classical $\phi = 0.05$ threshold to these p-values, *we only missed one attack* (P_5-PN, and barely, since the p-value is 0.054) *with zero false detections*. For each detected attack, the backdoor target class is also correctly inferred.

Ablation study: As the *first* defense designed for PCBA, we do not have a competitor to compare with. Still, we show detection performance using simplified statistics instead of the combined one used by our detector. In Tab. 5.4, for these simplified statistics, the p-values for some clean classifiers (e.g. P_4-PN, P_6-PN) are small due to the existence of *intrinsic backdoors*, which easily causes false detections. Even for some classifiers being attacked (e.g. P_5-PN, P_6-PN), using these simplified statistics, the null estimation will be affected by intrinsic backdoors, such that the resulting p-value will not be small enough to trigger a detection. These results highlight the importance of all three components (each strongly motivated by the intuition in Sec. 5.4.1) of our detection statistic.

Visualization of detection: In Fig. 5.10, we show the distribution of our statistic for both the classifier being attacked and the clean classifier for P_4-PN, for example. When there is a BA, two statistics associated with the true BA target class clearly appear in the "tail" of the null distribution, triggering a detection with correct inference of the BA target class.

Chapter 6
Conclusions and Future Work

In this thesis, we presented a thorough view of backdoor defenses, mainly focusing on image DNN classifiers, with extension to other domains such as point clouds. We provided a timely review of a broad range of topics in the field of adversarial learning, with the main focus on backdoor attacks. Our review covers a variety of design choices of backdoor attacks addressing the requirements for a broad range of practical attack scenarios.

In the main body of this thesis, we identified three major backdoor defense scenarios, which are before/during training scenario, post-training scenario, and in-flight scenario. Each scenario has its corresponding role for the defender, with a unique set of assumptions. For the post-training scenario, we proposed a reverse-engineering-based defense that detects if a pre-trained classifier has been backdoor attacked. To address the assumption that a post-training defender has no access to the training set, we reverse-engineer a putative backdoor pattern for each class pair, and proposed a purely unsupervised anomaly detector to jointly infer the presence of the attack and the backdoor class pair. Based on this approach, we further proposed several other defenses to address a variety of types of image backdoor patterns, to improve the efficiency of the defense, and to consider a very challenging 2-class, multiple-attack scenario. For the before/during training scenario, we proposed a clustering-based defense that infers if the training set is poisoned, and identifies the backdoor training images inserted by the attacker if there is an attack. The method adopts a cluster impurity statistic for detection, which requires setting a detection threshold. Beyond this, we proposed a reverse-engineering-based defense, inspired by our approach for the post-training scenario, which is implemented with a purely unsupervised anomaly detector. Finally, we proposed a backdoor attack against point cloud classifiers, with a post-training defense, again, extended from our reverse-engineering-based post-training defense for image backdoor attacks, which successfully detects our own attack. In summary, we addressed backdoor defenses from multiple

aspects in terms of defense scenarios and data domains.

The approaches discussed in this thesis also lead to potential future works. For example, our reverse-engineering-based backdoor detectors explicitly include the backdoor pattern embedding mechanism in the backdoor pattern reverse-engineering problems. Although multiple detectors, with each addressing a specific type of backdoor pattern, can be deployed in parallel, we are developing an approach that can reverse-engineer "universal" backdoor patterns. For another example, we can further study the theory behind the collateral damage effect introduced in Sec. 3.3.6.3 and the spatial invariance property of backdoor mappings introduced in Sec. 3.5.2.

Appendix A
Derivations and Proofs

A.1 Closed Form Solution to Problem (3.28)

Here we neglect the iteration indices for simplicity. With $\mathbf{v}$ fixed and T scheduled (using Eq. (3.29)) based on the current iterations' weighted misclassification fraction $Q_t(\mathbf{v}, \mathbf{w})$, solving (3.28) over $\mathbf{w}$ yields a closed-form solution, which is given as (3.30). Here, we show the details.

We first cast (3.28) into the following Lagrangian:

$$L(\mathbf{w}, \nu) = \sum_{s \neq t} w_s q_{st}(\mathbf{v}) - T \sum_{s \neq t} w_s \log w_s + \nu(\sum_{s \neq t} w_s - 1). \tag{A.1}$$

For all $s \neq t$, we obtain the following partial derivative:

$$\frac{\partial L}{\partial w_s} = q_{st}(\mathbf{v}) - T(\log w_s + 1) + \nu. \tag{A.2}$$

By setting the above partial derivative to 0, we obtain, for all $s \neq t$:

$$w_s = \exp(\frac{q_{st}(\mathbf{v}) + \nu - T}{T}) \tag{A.3}$$

By plugging the above into the constraint $\sum_{s \neq t} w_s = 1$, we get the closed-form solution (3.30).

A.2 Proof of Theorems for ET-RED

A.2.1 Proof of Theorem 3.7.1

For random variables $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following distribution P_i, for any realization $\mathbf{x}$ of $\mathbf{X}$, we have

$$\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{x})) = \mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}) | \mathbf{X} = \mathbf{x}) = \mathbb{E}\big[\mathbb{1}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}))\big|\mathbf{X} = \mathbf{x}\big] \tag{A.4}$$

Thus, ET, the left hand side of Eq. (3.36), can be written as

$$\mathbb{E}\big[\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})|\mathbf{X})\big] = \mathbb{E}\big[\mathbb{1}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X}))\big] = \mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})) \tag{A.5}$$

Based on the inclusion-exclusion rule

$$\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})) + \mathrm{P}(\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})) = 1 + \mathrm{P}_{\mathrm{MT},i} - \mathrm{P}_{\mathrm{NT},i} \tag{A.6}$$

Since, $\mathbf{X}$ and $\mathbf{Y}$ are i.i.d. random variables, $\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})) = \mathrm{P}(\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y}))$; thus, we get Eq. (3.36).

A.2.2 Proof of Theorem 3.7.2

Step 1: We show that for any $i \in \mathcal{Y}$ and $\mathbf{X}$, $\mathbf{Y}$ i.i.d. following distribution P_i, $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$ *if and only if* $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset$.

(a) If $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset$, there exists $\mathbf{v}$ such that $\mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{X})$ and $\mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{Y})$. By Definition 3.7.1,

$$\mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{X}) \Rightarrow f(\mathbf{X} + \mathbf{v}) \neq f(\mathbf{X}) \tag{A.7}$$

$$\mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{Y}) \Rightarrow f(\mathbf{Y} + \mathbf{v}) \neq f(\mathbf{Y}) \tag{A.8}$$

Then, by Definition 3.7.2, the existence of such $\mathbf{v}$ yields $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$.

(b) We prove the "only if" part by contradiction. Given $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$, suppose $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) = \emptyset$. By Definition 3.7.2, if $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$, there exists $\mathbf{v} \in \mathcal{V}_\epsilon(\mathbf{X})$ such that $f(\mathbf{Y} + \mathbf{v}) \neq f(\mathbf{Y})$. Since $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) = \emptyset$, $\mathbf{v} \notin \mathcal{V}_\epsilon(\mathbf{Y})$; hence, by Definition 3.7.1

$$||\mathbf{v}||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2 > \epsilon. \tag{A.9}$$

Similarly, there exists $\mathbf{v}' \in \mathcal{V}_\epsilon(\mathbf{Y})$ such that $f(\mathbf{X}+\mathbf{v}') \neq f(\mathbf{X})$ and $\mathbf{v}' \notin \mathcal{V}_\epsilon(\mathbf{X})$; hence

$$||\mathbf{v}'||_2 - ||\mathbf{v}^*(\mathbf{X})||_2 > \epsilon. \tag{A.10}$$

By Definition 3.7.1, for all $\mathbf{u} \in \mathcal{V}_\epsilon(\mathbf{Y})$, $||\mathbf{u}||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2 < \epsilon$. Thus, we have

$$||\mathbf{v}||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2 > ||\mathbf{u}||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2 \tag{A.11}$$

and accordingly $||\mathbf{v}||_2 > ||\mathbf{u}||_2$ for all $\mathbf{u} \in \mathcal{V}_\epsilon(\mathbf{Y})$. Similarly, for all $\mathbf{u}' \in \mathcal{V}_\epsilon(\mathbf{X})$, we have

$$||\mathbf{v}'||_2 - ||\mathbf{v}^*(\mathbf{X})||_2 > ||\mathbf{u}'||_2 - ||\mathbf{v}^*(\mathbf{X})||_2 \tag{A.12}$$

and thus, $||\mathbf{v}'||_2 > ||\mathbf{u}'||_2$ for all $\mathbf{u}' \in \mathcal{V}_\epsilon(\mathbf{X})$. Clearly, there is a contradiction since there cannot exist an element in one set having a larger norm than all elements in another set while vise versa; and therefore, $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset$.

Step 2: We show that the upper bound for $\mathrm{P}_{\mathrm{MT},i}$ goes to 0 when $\epsilon \to 0$. Note that ϵ is the "quality gap" between the practical solution and the optimal solution.

Note that by Definition 3.7.1, $\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset$ *only if* $\big|\,||\mathbf{v}^*(\mathbf{X})||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2\big| \leq \epsilon$; also based on Step 1:

$$\mathrm{P}_{\mathrm{MT},i} = \mathrm{P}(\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset) \leq \mathrm{P}(\big|\,||\mathbf{v}^*(\mathbf{X})||_2 - ||\mathbf{v}^*(\mathbf{Y})||_2\big| \leq \epsilon). \tag{A.13}$$

For $\mathbf{X}$ following continuous distribution P_i, $||\mathbf{v}^*(\mathbf{X})||_2$ also follows some continuous distribution[1] on $(0, +\infty)$. Since $\mathbf{X}$ and $\mathbf{Y}$ are i.i.d., the right hand side of Eq. (A.13) goes to 0 as $\epsilon \to 0$.

A.2.3 Proof of Theorem 3.7.3

For any $\mathbf{X} \sim P_i$, since $\mathbf{v}_0$ satisfies $f(\mathbf{X}+\mathbf{v}_0) \neq f(\mathbf{X})$ and

$$||\mathbf{v}_0||_2 - ||\mathbf{v}^*(\mathbf{X})||_2 < ||\mathbf{v}_0||_2 \leq \epsilon, \tag{A.14}$$

[1]One can construct very extreme cases such that $\mathrm{P}(||\mathbf{v}^*(\mathbf{X})||_2 = d) > 0$ for some constant $d > 0$. In other words, there is a set of samples from class i with non-negligible probability that are *equal distant* to the decision boundary. However, the probability for these cases is zero for practical domains and highly non-linear classifiers.

$\mathbf{v}_0 \in \mathcal{V}_\epsilon(\mathbf{X})$ by Definition 3.7.1. Thus, $\mathbf{v}_0 \in \mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y})$ for $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following P_i; and

$$P(\mathcal{V}_\epsilon(\mathbf{X}) \cap \mathcal{V}_\epsilon(\mathbf{Y}) \neq \emptyset) = 1. \tag{A.15}$$

Thus, based on Step 1 (and Eq. (A.13)) in Apdx A.2.2, we have $\mathrm{P_{MT}} = 1$. Since $\mathrm{P_{MT}} + \mathrm{P_{NT}} \leq 1$, Eq. (3.36) can be written as

$$\mathrm{ET}_{i,\epsilon} \geq \frac{1}{2} + \frac{1}{2}(\mathrm{P_{MT}} - 1 + \mathrm{P_{MT}}) = \mathrm{P_{MT}} \tag{A.16}$$

Then we have

$$\mathrm{ET}_{i,\epsilon} = 1. \tag{A.17}$$

A.2.4 Proof of Theorem 3.7.4

First, we derive the solution to problem (3.41). Note that $\mathbf{v} \in \mathbb{R}^n$ can be decomposed (using the orthonormal basis specified by $\mathbf{A}$ and $\mathbf{B}$) as $\mathbf{v} = \mathbf{A}\mathbf{v}_a + \mathbf{B}\mathbf{v}_b$ with $\mathbf{v}_a \in \mathbb{R}^d$ and $\mathbf{v}_b \in \mathbb{R}^{n-d}$. We substitute this decomposition into the constraint of problem (3.41). In words, the constraint means that $\mathbf{v}$ induces $\mathbf{Ac}$ to be (mis)classified to class 1 from class 0; thus, according to the expression of the classifier in Eq. (3.39), the constraint can be written as

$$(\mathbf{Ac} + \mathbf{A}\mathbf{v}_a + \mathbf{B}\mathbf{v}_b)^T(\mathbf{A}\mathbf{A}^T - \mathbf{B}\mathbf{B}^T)(\mathbf{Ac} + \mathbf{A}\mathbf{v}_a + \mathbf{B}\mathbf{v}_b) \leq 0. \tag{A.18}$$

Expanding the left hand side of the above inequality and using the fact that $\mathbf{A}^T\mathbf{B} = 0$ for simplification, we get a much simpler expression of the constraint:

$$||\mathbf{v}_a + \mathbf{c}||_2 \leq ||\mathbf{v}_b||_2. \tag{A.19}$$

Thus, a lower bound of the (square of the) objective to be minimized in problem (3.41) can be derived as

$$||\mathbf{v}||_2^2 = ||\mathbf{A}\mathbf{v}_a + \mathbf{B}\mathbf{v}_b||_2^2 = ||\mathbf{v}_a||_2^2 + ||\mathbf{v}_b||_2^2 \geq ||\mathbf{v}_a||_2^2 + ||\mathbf{v}_a + \mathbf{c}||_2^2, \tag{A.20}$$

with equality holds if and only if $||\mathbf{v}_b||_2 = ||\mathbf{v}_a + \mathbf{c}||$, i.e. $\mathbf{Ac} + \mathbf{v}$ on the decision boundary of the classifier. Note that the right hand side of the inequality above is minimized when $\mathbf{v}_a = -\frac{\mathbf{c}}{2}$. Then, the optimal solution $\mathbf{v}^*(\mathbf{c}) = \mathbf{A}\mathbf{v}_a^*(\mathbf{c}) + \mathbf{B}\mathbf{v}_\mathbf{b}^*(\mathbf{c})$ to problem (3.41)

satisfies:

$$\begin{cases} \mathbf{v}_a^*(\mathbf{c}) = -\frac{\mathbf{c}}{\mathbf{2}}, \\ ||\mathbf{v}_b^*(\mathbf{c})||_2 = \frac{||\mathbf{c}||_2}{2}. \end{cases} \tag{A.21}$$

Next, for any $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^d$, we derive the condition for $\mathbf{c}' \in \mathcal{T}(\mathbf{c})$. Since $f(\mathbf{A}\mathbf{c}') = f(\mathbf{A}\mathbf{c})$ is already satisfied, by Def. 3.7.4, $\mathbf{c}' \in \mathcal{T}(\mathbf{c})$ if and only if $f(\mathbf{A}\mathbf{c}' + \mathbf{v}^*(\mathbf{c})) \neq f(\mathbf{A}\mathbf{c}')$, which is equivalent to (by Eq. (3.39))

$$(\mathbf{A}\mathbf{c}' + \mathbf{v}^*(\mathbf{c}))^T(\mathbf{A}\mathbf{A}^T - \mathbf{B}\mathbf{B}^T)(\mathbf{A}\mathbf{c}' + \mathbf{v}^*(\mathbf{c})) \leq 0. \tag{A.22}$$

Using the decomposition of $\mathbf{v}^*(\mathbf{c})$, expanding and rearranging terms on the left hand side of the above, we obtain

$$\mathbf{c}'^T\mathbf{c}' + \mathbf{v}_a^*(\mathbf{c})^T\mathbf{v}_a^*(\mathbf{c}) - \mathbf{v}_b^*(\mathbf{c})^T\mathbf{v}_b^*(\mathbf{c}) + 2\mathbf{c}'^T\mathbf{v}_a^*(\mathbf{c}) \leq 0. \tag{A.23}$$

With the optimal solution in Eq. (A.21) substituted in, we get

$$\mathbf{c}'^T(\mathbf{c}' - \mathbf{c}) \leq 0, \tag{A.24}$$

or, equivalently

$$||\mathbf{c}' - \mathbf{c}/2||_2 \leq ||\mathbf{c}/2||_2. \tag{A.25}$$

A.2.5 Proof of Lemma 3.7.1

Based on Thm. 3.7.4, for latent space dimension $d = 1$ and two scalar[2] i.i.d. random samples C and C' with continuous distribution G, $C' \in \mathcal{T}(C)$ if and only if $|C' - C/2| \leq$

[2]We do not use bold $\mathbf{C}$ but C instead, since it is given as a scalar random variable.

$|C/2|$. Thus, by Def. 3.7.5, we have (for one-dimensional space)

$$\begin{aligned}
\mathrm{ET}_{(d=1)} &= \mathbb{E}_{C\sim G}[G(\frac{C}{2}+\frac{|C|}{2}) - G(\frac{C}{2}-\frac{|C|}{2})] \\
&= \int_{-\infty}^{\infty}[G(\frac{c}{2}+\frac{|c|}{2}) - G(\frac{c}{2}-\frac{|c|}{2})]g(c)dc \\
&= \int_{-\infty}^{0}[G(0)-G(c)]g(c)dc + \int_{0}^{\infty}[G(c)-G(0)]g(c)dc \\
&= \int_{0}^{G(0)}[G(0)-G(c)]dG(c) + \int_{G(0)}^{1}[G(c)-G(0)]dG(c) \\
&= G(0)^2 - \frac{1}{2}G(0)^2 + \frac{1}{2}[1-G(0)^2] - G(0)[1-G(0)] \\
&= \frac{1}{2} - G(0) + G(0)^2,
\end{aligned} \tag{A.26}$$

where g is the density function of distribution G. Note that the last line of Eq. (A.26) is strictly in the interval $[\frac{1}{4}, \frac{1}{2}]$ for G in range $[0, 1]$. The upper bound of ET when $d = 1$ is $\frac{1}{2}$, which is achieved if and only if $G(0) = 0$ or $G(0) = 1$.

A.2.6 Proof of Theorem 3.7.5

By Lem. 3.7.1, there exists $d = 1$ and $G(0) = 0$ or $G(0) = 1$, such that $\mathrm{ET} = \frac{1}{2}$. Here, we only need to show that $\mathrm{ET} \leq \frac{1}{2}$ for any d and G. Again, by the definition of ET (Def. 3.7.5) and Thm. 3.7.4, we can write ET as

$$\mathrm{ET} = \mathbb{E}_{\mathbf{C}\sim G}\Big[\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2)|\mathbf{C}\Big], \tag{A.27}$$

for $\mathbf{C}$ and $\mathbf{C}'$ i.i.d. following distribution G. Similar to our proof of Thm. 3.7.1

$$\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{c}}{2}||_2 \leq ||\frac{\mathbf{c}}{2}||_2) = \mathbb{E}_{\mathbf{C}'\sim G}\Big[\mathbb{1}\{||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2\}\Big|\mathbf{C} = \mathbf{c}\Big]. \tag{A.28}$$

Thus, based on Eq. (A.27) and (A.28), ET can be written as

$$\mathrm{ET} = \mathbb{E}_{\mathbf{C},\mathbf{C}'\sim G}\Big[\mathbb{1}\{||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2)\}\Big] = \mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2) \tag{A.29}$$

Since $\mathbf{C}$ and $\mathbf{C}'$ are i.i.d,

$$\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2) = \mathrm{P}(||\mathbf{C} - \frac{\mathbf{C}'}{2}||_2 \leq ||\frac{\mathbf{C}'}{2}||_2) \tag{A.30}$$

Also note that for continuous distribution

$$\begin{aligned}
&\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2, ||\mathbf{C} - \frac{\mathbf{C}'}{2}||_2 \leq ||\frac{\mathbf{C}'}{2}||_2) \\
\leq &\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2^2 + ||\mathbf{C} - \frac{\mathbf{C}'}{2}||_2^2 \leq ||\frac{\mathbf{C}}{2}||_2^2 + ||\frac{\mathbf{C}'}{2}||_2^2) \\
= &\mathrm{P}(||\mathbf{C} - \mathbf{C}'||_2^2 \leq 0) \\
= &0
\end{aligned} \tag{A.31}$$

Then, by the inclusion-exclusion rule

$$\begin{aligned}
&\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2) + \mathrm{P}(||\mathbf{C} - \frac{\mathbf{C}'}{2}||_2 \leq ||\frac{\mathbf{C}'}{2}||_2) \\
&- \mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2, ||\mathbf{C} - \frac{\mathbf{C}'}{2}||_2 \leq ||\frac{\mathbf{C}'}{2}||_2) \leq 1,
\end{aligned} \tag{A.32}$$

Substituting Eq. (A.30) and (A.31) into Eq. (A.32), we have

$$\mathrm{P}(||\mathbf{C}' - \frac{\mathbf{C}}{2}||_2 \leq ||\frac{\mathbf{C}}{2}||_2) \leq \frac{1}{2}, \tag{A.33}$$

thus, by Eq. (A.29)

$$\mathrm{ET} \leq \frac{1}{2} \tag{A.34}$$

which finishes the proof.

A.3 Using ET to Detect Backdoor Attacks with Patch Replacement Pattern

The detection framework of ET-RED is *not* specific to any particular backdoor embedding mechanism. Here, we repeat our derivation and analysis in Sec. 3.7 by considering patch replacement backdoor patterns embedded by Eq. (2.2). Basically, the definitions, theorems and algorithms presented in this section are matched with those in Sec. 3.7 – they are under the same framework which is generally independent of the backdoor pattern embedding mechanism.

Like the organization of our analysis for additive perturbation pattern in Sec. 3.7, we first introduce several definitions related to ET but for patch replacement patterns – these definitions are similar to those in Sec. 3.7 and are customized to patch replacement patterns. Especially, the ET statistic is defined in the same fashion as in Def. 3.7.3.

Then, we show that the same constant detection threshold $\frac{1}{2}$ on the ET statistic for detecting backdoor attacks with with additive perturbation patterns can be used for detecting backdoor attacks with patch replacement patterns as well. Finally, we present the detailed procedure of our detection for patch replacement patterns (as the counterpart of Alg. 3 in Sec. 3.7.5). Again, this section can be viewed as an independent section, where *the notations are self-contained.*

A.3.1 Definition of ET for Patch Replacement patterns

Consider the same classifier $f : \mathcal{X} \to \mathcal{Y}$ to be inspected (as in Sec. 3.7) with the same label space $\mathcal{Y} = \{0, 1\}$ and continuous sample distribution P_i on $\mathcal{X}$ for class $i \in \mathcal{Y}$. For any sample $\mathbf{x}$ from any class, the optimal solution to

$$\underset{\mathbf{s}=\{\mathbf{m},\mathbf{u}\}}{\text{minimize}} \, ||\mathbf{m}||_1 \qquad \text{subject to } f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{x}) \tag{A.35}$$

is defined as $\mathbf{s}^*(\mathbf{x}) = \{\mathbf{m}^*(\mathbf{x}), \mathbf{u}^*(\mathbf{x})\}$. Again, $\boldsymbol{m} \in \mathcal{M}$ is the binary mask and $\mathbf{u} \in \mathcal{X}$ is the patch for replacement. The practical solutions are usually sub-optimal. Thus, similar to Def. 3.7.1, we present the following definition in adaption to patch replacement patterns.

Definition A.3.1. (ϵ-solution set for patch replacement patterns) For any sample $\mathbf{x}$ from any class, regardless of the method being used, the ϵ-solution set to problem (A.35), is defined by

$$\mathcal{S}_\epsilon(\mathbf{x}) \triangleq \{\{\mathbf{m}, \mathbf{u}\} \big| \, ||\mathbf{m}||_1 - ||\mathbf{m}^*(\mathbf{x})||_1 \leq \epsilon, f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{x})\}, \tag{A.36}$$

where $\epsilon > 0$ is the "quality" bound of the solutions, which is usually small for existing methods.

Similar to Def. 3.7.2, we present the following definition (with abused notation) for the transferable set for patch replacement patterns.

Definition A.3.2. (Transferable set for patch replacement patterns) The transferable set for any sample $\mathbf{x}$ and $\epsilon > 0$ is defined by

$$\mathcal{T}_\epsilon(\mathbf{x}) \triangleq \{\mathbf{y} \in \mathcal{X} \big| f(\mathbf{y}) = f(\mathbf{x}), \exists \{\mathbf{m}, \mathbf{u}\} \in \mathcal{S}_\epsilon(\mathbf{x}) \, \text{s.t.} \, f(m_{\text{patch}}(\mathbf{x}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{y})\}. \tag{A.37}$$

Finally, we present the following definition of the ET statistic for patch replacement patterns, which looks *exactly the same* as Def. 3.7.3 in Sec. 3.7.2. However, here, the

definition of the transferable set has been customized for patch replacement patterns. Even though, the similarity between these definitions and their counterparts in Sec. 3.7.2 has already highlighted the generalization capability of the detection framework of ET-RED.

Definition A.3.3. (ET statistic for patch replacement patterns) For any class $i \in \mathcal{Y} = \{0, 1\}$ and $\epsilon > 0$, considering i.i.d. random samples $\mathbf{X}, \mathbf{Y} \sim P_i$, the ET statistic for class i is defined by $\mathrm{ET}_{i,\epsilon} \triangleq \mathbb{E}\big[\mathrm{P}(\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})|\mathbf{X})\big]$.

A.3.2 Detecting Backdoor Attacks with Patch Replacement Patterns Using ET

For the ET statistic for patch replacement patterns defined above, the same constant detection threshold $\frac{1}{2}$ can be used for distinguish backdoor target classes from non-target classes. The connection between the ET statistic for patch replacement patterns and the constant threshold $\frac{1}{2}$ is established by the same Thm. 3.7.1 in Sec. 3.7.3 with the same proof. Thus, these details are not included here for brevity. Note that $\mathrm{P}_{\mathrm{MT},i}$ and $\mathrm{P}_{\mathrm{NT},i}$ for class i are defined in the same way as in Thm. 3.7.1, though the transferable set $\mathcal{T}_\epsilon(\mathbf{x})$ is customized for patch replacement patterns in the current section – this is the main reason why we abuse the notation for the transferable set for patch replacement patterns in Def. A.3.2.

In the following, like in Sec. 3.7.3, we discuss the non-attack case and the attack case respectively. Readers should notice that the theorems (and the associated proofs) and discussions are similar to those in Sec. 3.7.3. Such similarity further highlight the generalization capability of ET-RED.

Non-attack case. Property 3.7.1 from Sec. 3.7.3 is also applicable here. That is, if class $(1 - i)$ where $i \in \mathcal{Y} = \{0, 1\}$ is not a backdoor target class, $\mathrm{P}_{\mathrm{NT},i}$ for class i will likely be larger than $\frac{1}{2}$.

As for $\mathrm{P}_{\mathrm{MT},i}$, again, Thm. 3.7.2 is also applicable here, but with a slightly different proof (shown below) in adaption to the modifications to the definitions for the patch replacement patterns in Apdx. A.3.1.

Proof. Step 1: Similar to the proof in Apdx. A.2.2, We show that for any $i \in \mathcal{Y}$ and $\mathbf{X}$, $\mathbf{Y}$ i.i.d. following distribution P_i, $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$ *if and only if* $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset$.

(a) If $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset$, there exists $\mathbf{s}$ such that $\mathbf{s} \in \mathcal{S}_\epsilon(\mathbf{X})$ and $\mathbf{s} \in \mathcal{S}_\epsilon(\mathbf{Y})$. By Def. A.3.1,

$$\mathbf{s} \in \mathcal{S}_\epsilon(\mathbf{X}) \Rightarrow f(m_{\text{patch}}(\mathbf{X}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{X}) \tag{A.38}$$

$$\mathbf{s} \in \mathcal{S}_\epsilon(\mathbf{Y}) \Rightarrow f(m_{\text{patch}}(\mathbf{Y}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{Y}) \tag{A.39}$$

Then, by Definition A.3.2, the existence of such $\mathbf{s}$ yields $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$.

(b) We prove the "only if" part also by contradiction. Given $\mathbf{X} \in \mathcal{T}_\epsilon(\mathbf{Y})$ and $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$, suppose $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) = \emptyset$. By Definition A.3.2, if $\mathbf{Y} \in \mathcal{T}_\epsilon(\mathbf{X})$, there exists $\mathbf{s} = \{\mathbf{m}, \mathbf{u}\} \in \mathcal{S}_\epsilon(\mathbf{X})$ such that $f(m_{\text{patch}}(\mathbf{Y}; \mathbf{u}, \mathbf{m})) \neq f(\mathbf{Y})$. Since $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) = \emptyset$, such $\mathbf{s} \notin \mathcal{S}_\epsilon(\mathbf{Y})$; hence, by Definition A.3.1

$$||\mathbf{m}||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1 > \epsilon. \tag{A.40}$$

Similarly, there exists $\mathbf{s}' = \{\mathbf{m}', \mathbf{u}'\} \in \mathcal{S}_\epsilon(\mathbf{Y})$ such that $f(m_{\text{patch}}(\mathbf{X}; \mathbf{u}', \mathbf{m}')) \neq f(\mathbf{X})$ and $\mathbf{s}' \notin \mathcal{S}_\epsilon(\mathbf{X})$ (since it is assumed that $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) = \emptyset$); hence

$$||\mathbf{m}'||_1 - ||\mathbf{m}^*(\mathbf{X})||_1 > \epsilon. \tag{A.41}$$

By Definition A.3.1, for all $\tilde{\mathbf{s}} = \{\tilde{\mathbf{m}}, \tilde{\mathbf{u}}\} \in \mathcal{S}_\epsilon(\mathbf{Y})$, $||\tilde{\mathbf{m}}||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1 < \epsilon$. Thus, we have

$$||\mathbf{m}||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1 > ||\tilde{\mathbf{m}}||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1 \tag{A.42}$$

and accordingly $||\mathbf{m}||_1 > ||\tilde{\mathbf{m}}||_1$ for all $\tilde{\mathbf{s}} \in \mathcal{S}_\epsilon(\mathbf{Y})$. Similarly, for all $\tilde{\mathbf{s}}' = \{\tilde{\mathbf{m}}', \tilde{\mathbf{u}}'\} \in \mathcal{S}_\epsilon(\mathbf{X})$, we have

$$||\mathbf{m}'||_1 - ||\mathbf{m}^*(\mathbf{X})||_1 > ||\tilde{\mathbf{m}}'||_1 - ||\mathbf{m}^*(\mathbf{X})||_1 \tag{A.43}$$

and thus, $||\mathbf{m}'||_1 > ||\tilde{\mathbf{m}}'||_1$ for all $\tilde{\mathbf{s}}' \in \mathcal{S}_\epsilon(\mathbf{X})$. Clearly, there is a contradiction since there cannot exist an element in one set having a larger norm than all elements in another set while vise versa; and therefore, $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset$.

Step 2: We show that the upper bound for $\mathrm{P}_{\mathrm{MT},i}$ goes to 0 when $\epsilon \to 0$.

Note that by Definition A.3.1, $\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset$ *only if* $\big|||\mathbf{m}^*(\mathbf{X})||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1\big| \leq \epsilon$; also based on Step 1:

$$\mathrm{P}_{\mathrm{MT},i} = \mathrm{P}(\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset) \leq \mathrm{P}(\big|||\mathbf{m}^*(\mathbf{X})||_1 - ||\mathbf{m}^*(\mathbf{Y})||_1\big| \leq \epsilon). \tag{A.44}$$

For $\mathbf{X}$ following continuous distribution P_i, $\mathbf{m}^*(\mathbf{X})$ also follows some continuous distribu-

tion on $(0, +\infty)$. Since $\mathbf{X}$ and $\mathbf{Y}$ are i.i.d., the right hand side of Eq. (A.44) goes to 0 as $\epsilon \to 0$. □

Based on the above, we have reached the same conclusions for patch replacement patterns as in Sec. 3.7.3. That is, for the non-attack case, we will likely have $\mathrm{P}_{\mathrm{NT},i} \geq \mathrm{P}_{\mathrm{MT},i}$, and consequently, $\mathrm{ET}_{i,\epsilon} \leq \frac{1}{2}$ based on Thm. 3.7.1.

Attack case. For class $i \in \mathcal{Y} = \{0, 1\}$, we consider a successful backdoor attack with target class $(1-i)$. The backdoor used by the attacker is specified by $\mathbf{s}_0 = \{\mathbf{m}_0, \mathbf{u}_0\}$ with mask $\mathbf{m}_0$ and patch $\mathbf{u}_0$. Thus, for any $\mathbf{X} \sim P_i$, due to the success of the backdoor attack, $f(\mathbf{X}) = i$ and $f(m_{\mathrm{patch}}(\mathbf{X}; \mathbf{u}_0, \mathbf{m}_0)) \neq f(\mathbf{X})$. Similar to our discussion in Sec. 3.7.3, $\mathbf{s}_0$ will likely be a common element in both $\mathcal{S}_\epsilon(\mathbf{X})$ and $\mathcal{S}_\epsilon(\mathbf{Y})$ for $\mathbf{X}$, $\mathbf{Y}$ i.i.d. following P_i. For the same reason, in this case, the ET statistic will likely be greater than $\frac{1}{2}$.

Similar to additive perturbation patterns, for patch replacement patterns, we also have a guarantee for a large ET statistic ($\mathrm{ET}_{i,\epsilon} = 1$) when the mask size of the pattern used by the attacker is sufficiently small. Such property is summarized in the theorem below.

Theorem A.3.1. If class $(1-i)$ it the target class of a backdoor attack with patch replacement pattern $\mathbf{s}_0 = \{\mathbf{m}_0, \mathbf{u}_0\}$ such that $||\mathbf{m}_0||_1 \leq \epsilon$, we will have $P(\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset) = 1$ for $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following P_i; and furthermore, $\mathrm{ET}_{i,\epsilon} = 1$.

Proof. For any $\mathbf{X} \sim P_i$, since $\mathbf{s}_0 = \{\mathbf{m}_0, \mathbf{u}_0\}$ satisfies $f(m_{\mathrm{patch}}(\mathbf{X}; \mathbf{u}_0, \mathbf{m}_0)) \neq f(\mathbf{X})$, and also because

$$||\mathbf{m}_0||_1 - ||\mathbf{m}^*(\mathbf{X})||_1 < ||\mathbf{m}_0||_1 \leq \epsilon, \tag{A.45}$$

by Definition A.3.1, we have $\mathbf{s}_0 \in \mathcal{S}_\epsilon(\mathbf{X})$. Thus, $\mathbf{s}_0 \in \mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y})$ for $\mathbf{X}$ and $\mathbf{Y}$ i.i.d. following P_i; and

$$P(\mathcal{S}_\epsilon(\mathbf{X}) \cap \mathcal{S}_\epsilon(\mathbf{Y}) \neq \emptyset) = 1. \tag{A.46}$$

The rest of the proof (showing that $\mathrm{ET}_{i,\epsilon} = 1$ for this attack scenario) is exactly the same as the proof of Thm. 3.7.3 shown in Apdx. A.2.3, thus is neglected here. □

A.3.3 Detection Procedure of ET-RED for Patch Replacement Patterns

The same procedure, i.e. Alg. 3 in Sec. 3.7.5, can be used for detecting backdoor attacks with patch replacement pattern, with only the following two modifications. We only need to first replace line 8 of Alg. 3 by:

"Obtain an empirical solution $\hat{\mathbf{s}}(\mathbf{x}_n^{(i)}) = \{\hat{\mathbf{m}}(\mathbf{x}_n^{(i)}), \hat{\mathbf{u}}(\mathbf{x}_n^{(i)})\}$ to problem (A.35) using random initialization."
and then change line 9 of Alg. 3 to:
"$\widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)}) \leftarrow \widehat{\mathcal{T}}_\epsilon(\mathbf{x}_n^{(i)}) \cup \{\mathbf{x}_k^{(i)} | k \in \{1, \cdots, N_i\} \backslash n, f(m_{\text{patch}}(\mathbf{x}_k^{(i)}; \hat{\mathbf{u}}(\mathbf{x}_n^{(i)}), \hat{\mathbf{m}}(\mathbf{x}_n^{(i)}))) \neq f(\mathbf{x}_k^{(i)})\}$".
Such a simple "module-based" modification allows ET-RED to be applicable to a variety of backdoor pattern embedding mechanisms, which again, shows the generalization capability of the ET-RED framework.

Bibliography

[1] Goodfellow, I., J. Shlens, and C. Szegedy (2015) "Explaining and harnessing adversarial examples," in *Proc. ICLR*.

[2] Chen, X., C. Liu, B. Li, K. Lu, and D. Song (2017), "Targeted backdoor attacks on deep learning systems using data poisoning," https://arxiv.org/abs/1712.05526v1.

[3] A. Saha, H. P., A. Subramanya (2020) "Hidden trigger backdoor attacks," in *AAAI*.

[4] Zhou, H., K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu (2019) "DUP-Net: Denoiser and Upsampler Network for 3D Adversarial Point Clouds Defense," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1961–1970.

[5] D. P. Kingma, J. B. (2015) "Adam: A Method for Stochastic Optimization," in *ICLR*.

[6] Bishop, C. M. (2006) *Pattern Recognition and Machine Learning*, Springer.

[7] Rabiner, L. R. and B.-H. Juang (1993) *Fundamentals of Speech Recognition*, Englewood Cliffs, NJ: Prentice Hall.

[8] Nigam, K., A. K. McCallum, S. Thrun, and T. M. Mitchell (2000) "Text Classification from Labeled and Unlabeled Documents using EM," *Machine Learning*, **39**, pp. 103–134.

[9] Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016) "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788.

[10] Hearst, M., S. Dumais, E. Osuna, J. Platt, and B. Scholkopf (1998) "Support vector machines," *IEEE Intelligent Systems and their Applications*, pp. 18–28.

[11] Patle, A. and D. S. Chouhan (2013) "SVM kernel functions for classification," in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pp. 1–9.

[12] Hastie, T., R. Tibshirani, and J. Friedman (2009) *The elements of statistical learning: data mining, inference and prediction*, Springer.

[13] Goodfellow, I. J., Y. Bengio, and A. Courville (2016) *Deep Learning*, MIT Press.

[14] He, K., X. Zhang, S. Ren, and J. Sun (2016) "Deep residual learning for image recognition," in *Proc. CVPR*.

[15] Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998) "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, **86**(11), pp. 2278–2324.

[16] Simonyan, K. and A. Zisserman (2015) "Very deep convolutional networks for large-scale image recognition," in *ICLR*.

[17] Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012) "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25.

[18] Girshick, R., J. Donahue, T. Darrell, and J. Malik (2014) "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

[19] Girshick, R. (2015) "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*.

[20] Ren, S., K. He, R. Girshick, and J. Sun (2017) "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(6), pp. 1137–1149.

[21] He, K., G. Gkioxari, P. Dollár, and R. Girshick (2017) "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988.

[22] Carreira, J. and A. Zisserman (2017) "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733.

[23] Maturana, D. and S. Scherer (2015) "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928.

[24] Sak, H., A. Senior, and F. Beaufays (2014) "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 338–342.

[25] Zhu, X., P. Sobihani, and H. Guo (2015) "Long Short-Term Memory Over Recursive Structures," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1604–1612.

[26] Tai, K. S., R. Socher, and C. D. Manning (2015) "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566.

[27] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017) "Attention is All you Need," in *Advances in Neural Information Processing Systems*.

[28] Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy (2016) "Hierarchical Attention Networks for Document Classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489.

[29] Zhou, X., X. Wan, and J. Xiao (2016) "Attention-based LSTM Network for Cross-Lingual Sentiment Classification," in *EMNLP*.

[30] Devlin, J., M. Chang, K. Lee, and K. Toutanova (2019) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

[31] Minaee, S., N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao (2021) "Deep Learning–Based Text Classification: A Comprehensive Review," *ACM Comput. Surv.*, **54**(3).

[32] Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2021) "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*.

[33] Liu, Z., Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo (2021) "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[34] Bellman, R. (1957) *Dynamic Programming*, Dover Publications.

[35] Roh, Y., G. Heo, and S. E. Whang (2021) "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering*, **33**(4), pp. 1328–1347.

[36] "Kaggle," `https://www.kaggle.com/`.

[37] "Ckan," `http://ckan.org`.

[38] "Quandl," `https://www.quandl.com`.

[39] "Datamarket," `https://datamarket.com`.

[40] Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009) "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

[41] Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020) "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, pp. 1877–1901.

[42] Breier, J., X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu (2018) "Practical Fault Attack on Deep Neural Networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, p. 2204–2206.

[43] Carlini, N. and D. Wagner (2017) "Towards Evaluating the Robustness of Neural Networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57.

[44] Jagielski, M., A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li (2018) "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35.

[45] Li, X., G. Kesidis, D. J. Miller, and V. Lucic (2021) "Backdoor Attack and Defense for Deep Regression," .
URL `https://arxiv.org/abs/2109.02381`

[46] Biggio, B. and F. Roli (2018) "Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning," *Pattern Recognition*, **84**, pp. 317–331.

[47] Huang, L., A. Joseph, B. Nelson, B. Rubinstein, and J. Tygar (2011) "Adversarial machine learning," in *Proc. 4th ACM Workshop on Artificial Intelligence and Security (AISec)*.

[48] Miller, D. J., X. Hu, Z. Qiu, and G. Kesidis (2017) "Adversarial learning: A critical review and active learning study," *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6.

[49] Biggio, B., B. Nelson, and P. Laskov (2012) "Poisoning Attacks against Support Vector Machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, p. 1467–1474.

[50] ——— (2011) "Support Vector Machines Under Adversarial Label Noise," in *Proceedings of the Asian Conference on Machine Learning*, pp. 97–112.

[51] Xiao, H., B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli (July 2015) "Support vector machines under adversarial label contamination," *Neurocomputing*, **160**(C), pp. 53–62.

[52] Nelson, B., M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia (2008) "Exploiting Machine Learning to Subvert Your Spam Filter," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*.

[53] Muñoz-González, L., B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli (2017) "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*.

[54] Yang, C., Q. Wu, H. H. Li, and Y. Chen (2017) "Generative Poisoning Attack Method Against Neural Networks," *ArXiv*, **abs/1703.01340**.

[55] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014) "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*.

[56] Shafahi, A., W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein (2018) "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 6106–6116.

[57] Wang, Y. and K. Chaudhuri (2018) "Data Poisoning Attacks against Online Learning," *ArXiv*, **abs/1808.08994**.

[58] Duda, R., P. Hart, and D. Stork (2001) *Pattern classification*, Wiley.

[59] Shen, Y. and S. Sanghavi (2019) "Learning with Bad Training Data via Iterative Trimmed Loss Minimization," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 5739–5748.

[60] Carnerero-Cano, J., L. Muñoz-González, P. Spencer, and E. C. Lupu (2020) "Regularisation Can Mitigate Poisoning Attacks: A Novel Analysis Based on Multiobjective Bilevel Optimisation," *ArXiv*, **abs/2003.00040**.

[61] Pope, P., C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein (2021) "The Intrinsic Dimension of Images and Its Impact on Learning," in *International Conference on Learning Representations*.

[62] Wang, W., Y. Huang, Y. Wang, and L. Wang (2014) "Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 496–503.

[63] Roweis, S. T. and L. K. Saul (2000) "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, **290**, pp. 2323–2326.

[64] Tenenbaum, J. B., V. de Silva, and J. C. Langford (2000) "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, **290**, p. 2319.

[65] Fefferman, C., S. Mitter, and H. Narayanan (2016) "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, **29**(4), pp. 983–1049.

[66] Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014) "Intriguing properties of neural networks," in *Proc. ICLR*.

[67] Papernot, N., P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami (2017) "Practical Black-Box Attacks against Machine Learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, p. 506–519.

[68] Papernot, N., P. D. McDaniel, and I. J. Goodfellow (2016) "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples," .
URL `http://arxiv.org/abs/1605.07277`

[69] Weng, L., P.-Y. Chen, L. Nguyen, M. Squillante, A. Boopathy, I. Oseledets, and L. Daniel (2019) "PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 6727–6736.

[70] Chen, P.-Y., H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh (2017) "ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, p. 15–26.

[71] Papernot, N., P. McDaniel, S. Jha, M. Fredrikson, Z. Celik, and A. Swami (2016) "The Limitations of Deep Learning in Adversarial Settings," in *Proc. 1st IEEE European Symp. on Security and Privacy*.

[72] Kurakin, A., I. J. Goodfellow, and S. Bengio (2017) "Adversarial Machine Learning at Scale," in *ICLR*.

[73] Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2018) "Towards Deep Learning Models Resistant to Adversarial Attacks," in *Proc. ICLR*.

[74] M.-Dezfooli, S.-M., A. Fawzi, and P. Frossard (2016) "DeepFool: a simple and accurate method to fool deep neural networks," in *Proc. CVPR*.

[75] Chen, P.-Y., Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh (2018) "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples," in *AAAI*.

[76] Papernot, N., P. McDaniel, X. Wu, S. Jha, and A. Swami (2016) "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597.

[77] Hinton, G., O. Vinyals, and J. Dean (2015) "Distilling the Knowledge in a Neural Network," in *NIPS Deep Learning and Representation Learning Workshop*.

[78] Xu, W., D. Evans, and Y. Qi (2018) "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in *NDSS*.

[79] Lin, J., C. Gan, and S. Han (2019) "Defensive Quantization: When Efficiency Meets Robustness," in *International Conference on Learning Representations*.

[80] Buckman, J., A. Roy, C. Raffel, and I. Goodfellow (2018) "Thermometer Encoding: One Hot Way To Resist Adversarial Examples," in *International Conference on Learning Representations*.

[81] Dhillon, G. S., K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, and A. Anandkumar (2018) "Stochastic activation pruning for robust adversarial defense," in *International Conference on Learning Representations*.

[82] Athalye, A., N. Carlini, and D. A. Wagner (2018) "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in *ICML*, pp. 274–283.

[83] Grosse, K., P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel (2017) "On the (Statistical) Detection of Adversarial Examples," .
URL `http://arxiv.org/abs/1702.06280`

[84] Li, X. and F. Li (2017) "Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5775–5783.

[85] Metzen, J. H., T. Genewein, V. Fischer, and B. Bischoff (2017) "On Detecting Adversarial Perturbations," in *ICLR*.

[86] Feinman, R., R. R. Curtin, S. Shintre, and A. B. Gardner (2017), "Detecting Adversarial Samples from Artifacts," .

[87] Miller, D., Y. Wang, and G. Kesidis (2019) "When Not to Classify: Anomaly Detection of Attacks (ADA) on DNN Classifiers at Test Time," *Neural Computation*, **31**(8), pp. 1624–1670.

[88] Cisse, M., P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunierr (2017) "Parseval Networks: Improving Robustness to Adversarial Examples," in *Proc. ICML*.

[89] Tsuzuku, Y., I. Sato, and M. Sugiyama (2018) "Lipschitz-margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks," in *Proc NIPS*.

[90] Fazlyab, M., A. Robey, H. Hassani, M. Morari, and G. Pappas (2019) "Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks," in *Advances in Neural Information Processing Systems*.

[91] Tramèr, F., A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel (2018) "Ensemble Adversarial Training: Attacks and Defenses," in *International Conference on Learning Representations*.

[92] Dong, Y., Z. Deng, T. Pang, J. Zhu, and H. Su (2020) "Adversarial Distributional Training for Robust Deep Learning," in *NeurIPS*.

[93] Tsipras, D., S. Santurkar, L. Engstrom, A. Turner, and A. Madry (2019) "Robustness May Be at Odds with Accuracy," in *International Conference on Learning Representations*.

[94] Rice, L., E. Wong, and J. Z. Kolter (2020) "Overfitting in adversarially robust deep learning," in *ICML*.

[95] Shokri, R., M. Stronati, C. Sogn, and V. Shmatikov (2017) "Membership Inference Attacks Against Machine Learning Models," in *Proc. IEEE Symposium on Security and Privacy*.

[96] Salem, A., Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes (2019) "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models." in *NDSS*.

[97] Tramer, F., F. Zhang, A. Juels, M. Reiter, and T. Ristenpart (2016) "Stealing Machine Learning Models via Prediction APIs," in *Proc. USENIX Security Symposium*.

[98] Papernot, N., P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami (2017) "Practical black box attacks against machine learning," in *Proc. Asia CCS*.

[99] Orekondy, T., B. Schiele, and M. Fritz (2019) "Knockoff Nets: Stealing Functionality of Black-Box Models," in *CVPR*.

[100] Chandrasekaran, V., K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan (2020) "Exploring Connections Between Active Learning and Model Extraction," in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1309–1326.

[101] Lee, T., B. Edwards, I. Molloy, and D. Su (2019) "Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations," in *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 43–49.

[102] Kesarwani, M., B. Mukhoty, V. Arya, and S. Mehta (2018) "Model Extraction Warning in MLaaS Paradigm," .

[103] Juuti, M., S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan (2019) "PRADA: Protecting Against DNN Model Stealing Attacks," *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 512–527.

[104] Wang, Y., D. J. Miller, and G. Kesidis (2019) "When Not to Classify: Detection of Reverse Engineering Attacks on DNN Image Classifiers," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8063–8066.

[105] Zhu, L., Y. Li, X. Jia, Y. Jiang, S.-T. Xia, and X. Cao (2021) "Defending against Model Stealing via Verifying Embedded External Features," in *ICML 2021 Workshop on Adversarial Machine Learning*.

[106] Gu, T., K. Liu, B. Dolan-Gavitt, and S. Garg (2019) "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks," *IEEE Access*, **7**, pp. 47230–47244.

[107] Liu, Y., S. Ma, Y. Aafer, W.-C. Lee, and J. Zhai (2018) "Trojaning attack on neural networks," in *Proc. NDSS*, San Diego, CA.

[108] Nelson, B. and B. B. et al. (2009) "Misleading learners: Co-opting your spam filter," in *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*.

[109] Wang, B., Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Zhao (2019) "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symposium on Security and Privacy*.

[110] Xiang, Z., D. J. Miller, and G. Kesidis (2020) "Detection of Backdoors in Trained Classifiers Without Access to the Training Set," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15.

[111] Miller, D. J. and G. Kesidis (filed by Anomalee Inc. on May 27, 2020; provisional patent filed May 29, 2019) "Post-Training Detection and Identification of Human-Imperceptible Backdoor-Poisoning Attacks on Neural Network Classifiers," *United States Patent Pending*.

[112] Guo, W., L. Wang, X. Xing, M. Du, and D. Song (2019), "TABOR: A highly accurate approach to inspecting and restoring Trojan backdoors in AI systems," https://arxiv.org/abs/1908.01763.

[113] Wang, R., G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang (2020) "Practical Detection of Trojan Neural Networks: Data-Limited and Data-Free Cases," in *Proc. ECCV*.

[114] Chou, E., F. Tramèr, G. Pellegrino, and D. Boneh (2018), "SentiNet: Detecting Physical Attacks Against Deep Learning Systems," .
URL `http://arxiv.org/abs/1812.00292`

[115] Gao, Y., C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal (2019) "STRIP: A Defence Against Trojan Attacks on Deep Neural Networks," in *Proc. ACSAC*.

[116] Doan, B. G., E. Abbasnejad, and D. C. Ranasinghe (2020) "Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems," in *Annual Computer Security Applications Conference*, p. 897–912.

[117] Tran, B., J. Li, and A. Madry (2018) "Spectral signatures in backdoor attacks," in *Proc. NIPS*.

[118] Chen, B., W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava (2018), "Detecting backdoor attacks on deep neural networks by activation clustering," http://arxiv.org/abs/1811.03728.

[119] Miller, D. J., Z. Xiang, and G. Kesidis (2020) "Adversarial Learning in Statistical Classification: A Comprehensive Review of Defenses Against Attacks," *Proceedings of the IEEE*, **108**, pp. 402–433.

[120] Xiang, Z., D. J. Miller, H. Wang, and G. Kesidis (2021) "Detecting Scene-Plausible Perceptible Backdoors in Trained DNNs Without Access to the Training Set," *Neural Computation*, **33**(5), pp. 1329–1371.

[121] Xiang, Z., D. J. Miller, and G. Kesidis (2021) "Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing," *Computers and Security*, **106**.

[122] Xiang, Z., D. Miller, and G. Kesidis (2019) "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense," in *Proc. IEEE MLSP*, Pittsburgh.

[123] Xiang, Z., D. Miller, H. Wang, and G. Kesidis (Oct. 2020) "Revealing Perceptible Backdoors in DNNs, Without the Training Set, via the Maximum Achievable Misclassification Fraction Statistic," in *Proc. IEEE MLSP*.

[124] Xiang, Z., D. J. Miller, and G. Kesidis (2020) "Revealing Backdoors, Post-Training, in DNN Classifiers via Novel Inference on Optimized Perturbations Inducing Group Misclassification," in *Proc. IEEE ICASSP*, pp. 3827–3831.

[125] Xiang, Z., D. J. Miller, and G. Kesidis (2021) "L-Red: Efficient Post-Training Detection of Imperceptible Backdoor Attacks Without Access to the Training Set," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3745–3749.

[126] Xiang, Z., D. J. Miller, S. Chen, X. Li, and G. Kesidis (2022) "Detecting Backdoor Attacks Against Point Cloud Classifiers," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[127] ——— (2021) "A Backdoor Attack against 3D Point Cloud Classifiers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[128] Xiang, Z., D. Miller, and G. Kesidis (2022) "Post-Training Detection of Backdoor Attacks for Two-Class and Multi-Attack Scenarios," in *International Conference on Learning Representations*.

[129] Zhong, H., H. Zhong, A. Squicciarini, S. Zhu, and D. Miller (2020) "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proc. CODASPY*.

[130] Barni, M., K. Kallas, and B. Tondi (2019) "A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning," *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105.

[131] Li, Y., B. Wu, Y. Jiang, Z. Li, and S.-T. Xia (2020), "Backdoor learning: A survey," .
URL `https://arxiv.org/pdf/2004.04692.pdf`

[132] Rakin, A. S., Z. He, and D. Fan (2020) "TBT: Targeted Neural Network Attack With Bit Trojan," pp. 13195–13204.

[133] Bai, J., B. Wu, Y. Zhang, Y. Li, Z. Li, and S.-T. Xia (2021) "Targeted Attack against Deep Neural Networks via Flipping Limited Weight Bits," in *International Conference on Learning Representations*.
URL `https://openreview.net/forum?id=iKQAk8a2kM0`

[134] Qi, X., J. Zhu, C. Xie, and Y. Yang (2021) "Subnet Replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting," in *ICLR Workshop on Security and Safety in Machine Learning Systems*.

[135] Tang, R., M. Du, N. Liu, F. Yang, and X. Hu (2020) "An embarrassingly simple approach for trojan attack in deep neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 218–228.

[136] Yunfei Liu, J. B., Xingjun Ma and F. Lu (2020) "Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks," in *ECCV*.

[137] Nguyen, T. A. and A. T. Tran (2021) "WaNet - Imperceptible Warping-based Backdoor Attack," in *International Conference on Learning Representations*. URL `https://openreview.net/forum?id=eEn8KTtJOx`

[138] Wang, T., Y. Yao, F. Xu, S. An, H. Tong, and T. Wang (2021), "Backdoor Attack through Frequency Domain," `2111.10991`.

[139] Ahmed, N., T. Natarajan, and K. Rao (1974) "Discrete Cosine Transform," *IEEE Transactions on Computers*, **C-23**(1), pp. 90–93.

[140] Zeng, Y., W. Park, Z. M. Mao, and R. Jia (2022), "Rethinking the Backdoor Attacks' Triggers: A Frequency Perspective," `2104.03413`.

[141] Nguyen, A. and A. Tran (2020) "Input-Aware Dynamic Backdoor Attack," in *Proceedings of Advances in Neural Information Processing Systems*.

[142] Li, Y., Y. Li, B. Wu, L. Li, R. He, and S. Lyu (2021) "Invisible Backdoor Attack with Sample-Specific Triggers," in *IEEE International Conference on Computer Vision (ICCV)*.

[143] Turner, A., D. Tsipras, and A. Madry (2019), "Clean-Label Backdoor Attacks," https://people.csail.mit.edu/madry/lab/cleanlabel.pdf.

[144] Li, X., G. Kesidis, D. J. Miller, and V. Lucic (2021), "Backdoor Attack and Defense for Deep Regression," .

[145] Yang, Z., N. Iyer, J. Reimann, and N. Virani (2019), "Design of intentional backdoors in sequential models," `1902.09972`.

[146] Wang, L., Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song (2021) "BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning," in *IJCAI*, pp. 3699–3705.

[147] Pan, S. J. and Q. Yang (2010) "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), pp. 1345–1359.

[148] Zhuang, F., Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He (2021) "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE*, **109**(1), pp. 43–76.

[149] Wang, S., S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen (2020) "Backdoor Attacks against Transfer Learning with Pre-trained Deep Learning Models," *IEEE Transactions on Services Computing*, pp. 1–1.

[150] Konečný, J., H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon (2016) "Federated Learning: Strategies for Improving Communication Efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*.

[151] Bagdasaryan, E., A. Veit, Y. Hua, D. Estrin, and V. Shmatikov (2019), "How To Backdoor Federated Learning," `1807.00459`.

[152] Xie, C., K. Huang, P.-Y. Chen, and B. Li (2020) "DBA: Distributed Backdoor Attacks against Federated Learning," in *International Conference on Learning Representations*.
URL `https://openreview.net/forum?id=rkgyS0VFvr`

[153] Xie, C., M. Chen, P.-Y. Chen, and B. Li (2021) "CRFL: Certifiably Robust Federated Learning against Backdoor Attacks," in *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 11372–11382.

[154] Dai, J., C. Chen, and Y. Li (2019) "A Backdoor Attack Against LSTM-Based Text Classification Systems," *IEEE Access*, **7**, pp. 138872–138878.

[155] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[156] Chen, X., A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang (2021) *BadNL: Backdoor Attacks against NLP Models with Semantic-Preserving Improvements*, p. 554–569.

[157] Zhao, S., X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang (2020) "Clean-Label Backdoor Attacks on Video Recognition Models," pp. 14431–14440.

[158] Zhai, T., Y. Li, Z. Zhang, B. Wu, Y. Jiang, and S.-T. Xia (2021) "Backdoor Attack Against Speaker Verification," in *2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2560–2564.

[159] Li, X., Z. Chen, Y. Zhao, Z. Tong, Y. Zhao, A. Lim, and J. T. Zhou (2021) "PointBA: Towards Backdoor Attacks in 3D Point Cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[160] Liu, K., B. Doan-Gavitt, and S. Garg (2018) "Fine-pruning: Defending against backdoor attacks on deep neural networks," in *Proc. RAID*.

[161] Chen, H., C. Fu, J. Zhao, and F. Koushanfar (2019) "DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4658–4664.

[162] Xu, X., Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li (2019), "Detecting AI trojans using meta neural analysis," https://arxiv.org/abs/1910.03137.

[163] Kolouri, S., A. Saha, H. Pirsiavash, and H. Hoffmann (2020) "Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 298–307.

[164] Li, X., Z. Xiang, D. J. Miller, and G. Kesidis (2022) "Test-Time Detection of Backdoor Triggers for Poisoned Deep Neural Networks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[165] Li, Y., X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma (2021) "Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks," in *Proc. ICLR*.

[166] Zeng, Y., S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia (2022) "Adversarial Unlearning of Backdoors via Implicit Hypergradient," in *International Conference on Learning Representations*.

[167] Hampel, F. R. (1974) "The influence curve and its role in robust estimation," *Journal of the American Statistical Association*, **69**.

[168] Mirza, M. and S. Osindero (2014), "Conditional Generative Adversarial Nets," .
URL `http://arxiv.org/abs/1411.1784`

[169] Fredrikson, M., S. Jha, and T. Ristenpart (2015) "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, p. 1322–1333.

[170] Dong, Y., X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, and J. Zhu (2021) "Black-box Detection of Backdoor Attacks with Limited Information and Data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[171] Liu, Y., W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang (2019) "ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, p. 1265–1282.

[172] Moosavi-Dezfooli, S.-M., A. Fawzi, and P. Frossard (2017) "Universal adversarial perturbations," in *Proc. CVPR*.

[173] Juang, B.-H. and S. Katigiri (1992) "Discriminative learning for minimum error classification," *IEEE Trans. on Signal Processing*.

[174] Krizhevsky, A. (2012) "Learning Multiple Layers of Features from Tiny Images," *University of Toronto*.

[175] Sermanet, P., S. Chintala, and Y. LeCun (2012) "Convolutional neural networks applied to house numbers digit classification," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*.

[176] Albanie, S., "Pretrained VGG-face model," http://www.robots.ox.ac.uk/ albanie/pytorch-models.html.

[177] Xiao, H., K. Rasul, and R. Vollgraf (2017), "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," .
URL `http://arxiv.org/abs/1708.07747`

[178] J. Stallkamp, J. S., M. Schlipsing and C. Igel (2012) "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." *Neural Networks*, **32**, pp. 323–332.

[179] Gao, Z., A. Feng, X. Song, and X. Wu (2019) "Target-Dependent Sentiment Classification With BERT," *IEEE Access*, **7**, pp. 154290–154299.

[180] Li, R., W. Zhang, H.-I. Suk, L. Wang, J. Li, D. Shen, and S. Ji (2014) "Deep Learning Based Imaging Data Completion for Improved Brain Disease Diagnosis," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, pp. 305–312.

[181] Coates, A., H. Lee, and A. Y. Ng (2011) "An Analysis of Single Layer Networks in Unsupervised Feature Learning," in *AISTATS*.

[182] Li, Y., X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma (2021) "Anti-Backdoor Learning: Training Clean Models on Poisoned Data," in *Advances in Neural Information Processing Systems*.

[183] Huang, K., Y. Li, B. Wu, Z. Qin, and K. Ren (2022) "Backdoor Defense via Decoupling the Training Process," in *International Conference on Learning Representations*.

[184] Chen, T., S. Kornblith, M. Norouzi, and G. Hinton (2020) "A Simple Framework for Contrastive Learning of Visual Representations," in *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607.

[185] Graham, M. and D. Miller (2006) "Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection," *IEEE Transactions on Signal Processing*, **54**(4), pp. 1289–1303.

[186] Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011) "Reading Digits in Natural Images with Unsupervised Feature Learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

[187] Chen, S., B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington (2021) "3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception," *IEEE Signal Processing Magazine*, **38**(1), pp. 68–86.

[188] Guo, J., P. V. K. Borges, C. Park, and A. Gawel (2019) "Local Descriptor for Robust Place Recognition Using LiDAR Intensity," *IEEE Robotics and Automation Letters*, **4**(2), pp. 1470–1477.

[189] Schumann, O., M. Hahn, J. Dickmann, and C. Wöhler (2018) "Semantic Segmentation on Radar Point Clouds," in *2018 21st International Conference on Information Fusion (FUSION)*, pp. 2179–2186.

[190] Yue, X., B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli (2018) "A LiDAR Point Cloud Generator: From a Virtual World to Autonomous Driving," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, p. 458–464.

[191] Korres, G. and M. Eid (2016) "Haptogram: Ultrasonic Point-Cloud Tactile Stimulation," *IEEE Access*, **4**, pp. 7758–7769.

[192] Zhao, H., L. Jiang, C. Fu, and J. Jia (2019) "PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[193] Yang, B., W. Luo, and R. Urtasun (2018) "PIXOR: Real-time 3D Object Detection from Point Clouds," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7652–7660.

[194] Charles, R. Q., H. Su, M. Kaichun, and L. J. Guibas (2017) "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85.

[195] Qi, C. R., L. Yi, H. Su, and L. J. Guibas (2017) "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5099–5108.

[196] Wang, Y., Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon (2019) "Dynamic Graph CNN for Learning on Point Clouds," *ACM Trans. Graph.*, **38**(5).

[197] Yang, J., Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian (2019) "Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3318–3327.

[198] Su, H., S. Maji, E. Kalogerakis, and E. G. Learned-Miller (2015) "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*.

[199] Su, J.-C., M. Gadelha, R. Wang, and S. Maji (2019) "A Deeper Look at 3D Shape Classifiers," in *Computer Vision – ECCV 2018 Workshops* (L. Leal-Taixé and S. Roth, eds.).

[200] Li, Y., Z. An, Z. Wang, Y. Zhong, S. Chen, and C. Feng (2021) "V2X-Sim: A Virtual Collaborative Perception Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*.

[201] Li, Y., S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang (2021) "Learning Distilled Collaboration Graph for Multi-Agent Perception," in *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)*.

[202] Xiang, C., C. R. Qi, and B. Li (2019) "Generating 3D Adversarial Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[203] Wen, Y., J. Lin, K. Chen, C. L. P. Chen, and K. Jia (2020) "Geometry-Aware Generation of Adversarial Point Clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1.

[204] Liu, D., R. Yu, and H. Su (2019) "Extending Adversarial Attacks and Defenses to Deep 3D Point Cloud Classifiers," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2279–2283.

[205] Tu, J., M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun (2020) "Physically Realizable Adversarial Examples for LiDAR Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[206] Hamdi, A., S. Rojas, A. Thabet, and B. Ghanem (2020) "AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds," in *ECCV 2020*, pp. 241–257.

[207] Li, X., Z. Chen, Y. Zhao, Z. Tong, Y. Zhao, A. Lim, and J. T. Zhou (2021) "PointBA: Towards Backdoor Attacks in 3D Point Cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[208] Tian, G., W. Jiang, W. Liu, and Y. Mu (2021), "Poisoning MorphNet for Clean-Label Backdoor Attack to Point Clouds," `2105.04839`.

[209] Zheng, T., C. Chen, J. Yuan, B. Li, and K. Ren (2019) "PointCloud Saliency Maps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[210] Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao (2015) "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[211] Menze, M. and A. Geiger (2015) "Object Scene Flow for Autonomous Vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vita

Zhen Xiang

Upon the completion of this dissertation, Zhen Xiang is a fifth-year PhD student in the Department of Electrical Engineering at Pennsylvania State University (PSU), advised by Prof David J. Miller and Prof George Kesidis. Before that, he received his B.Sc. degree in Electronics and Computer Engineering from Hong Kong University of Science and Technology with an outstanding student award in 2014, and his M.Sc. degree in Electrical Engineering from University of Pennsylvania in 2016. Zhen Xiang's research interests include adversarial machine learning and statistical signal processing. During his PhD study at PSU, Zhen Xiang was intensively involved in defeating adversarial attacks against machine learning systems, including backdoor attacks and evasion attacks. His research works were published on top-tier journals and conferences including IEEE TNNLS, Proc. IEEE, ICLR, ICCV, etc. He also served as reviewers for journals including IEEE TNNLS, Computers & Security, and IEEE SPM, and conferences including ICASSP, MLSP. He was a student member of IEEE since 2012.