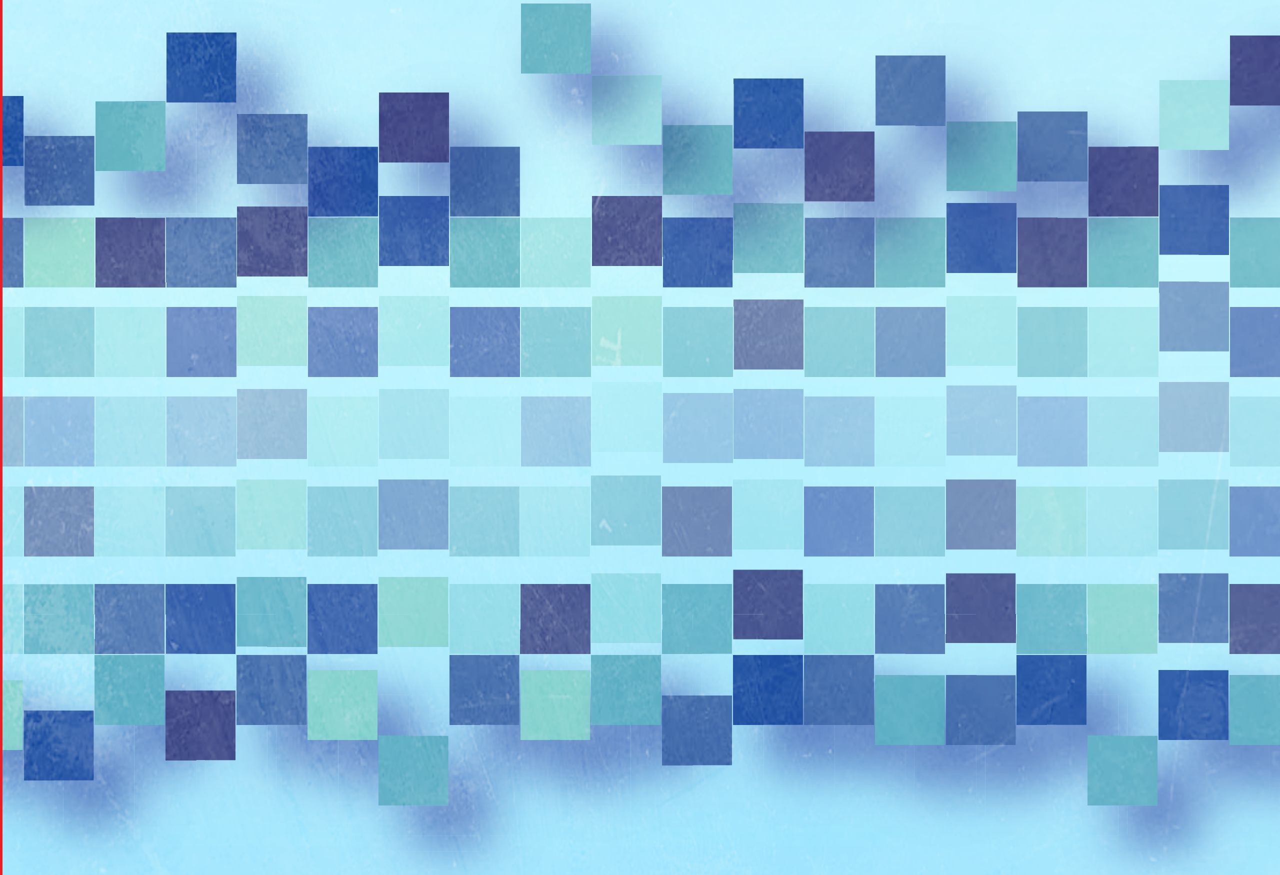
No Time to Spare:
Adversarial Machine Learning at Training and Inference Time



Institute for Computing and
Information Sciences

Xiaoyun Xu

RADBOUD
UNIVERSITY
PRESS

Radboud
Dissertation
Series

No Time to Spare: Adversarial Machine Learning at Training and Inference Time

Xiaoyun Xu

No Time to Spare: Adversarial Machine Learning at Training and Inference Time
Xiaoyun Xu

Radboud Dissertation Series
ISSN: 2950-2772 (Online); 2950-2780 (Print)

Published by RADBOUD UNIVERSITY PRESS
Postbus 9100, 6500 HA Nijmegen, The Netherlands
www.radbouduniversitypress.nl

Design: Xiaoyun Xu
Cover: Proefschrift AIO | Guntra Laivacuma
Printing: DPN Rikken/Pumbo

ISBN: 9789465152103
DOI: 10.54195/9789465152103
Free download at: https://doi.org/10.54195/9789465152103

RADBOUD
UNIVERSITY
PRESS

No Time to Spare: Adversarial Machine Learning at Training and Inference Time

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.M. Sanders,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

maandag 12 januari 2026
om 12:30 uur precies

door

Xiaoyun Xu

geboren op 5 januari 1995
te Sichuan, China

Promotor:

Prof. dr. L. Batina

Copromotor:

Dr. S. Picek

Manuscriptcommissie:

Prof. dr. M. Loog (voorzitter)
Prof. dr. ing. D. Jakobovic (Sveučilište u Zagrebu, Kroatië)
Dr. L. Mariot (Universiteit Twente)
Prof. L.Y. Chen (Université de Neuchâtel, Zwitserland)
Prof. Z. Zhao (Xi'an Jiaotong University, China)

No Time to Spare: Adversarial Machine Learning at Training and Inference Time

Dissertation to obtain the degree of doctor
from Radboud University Nijmegen,
on the authority of the Rector Magnificus prof. dr. J.M. Sanders,
according to the decision of the Doctorate Board
to be defended in public on

Monday, January 12, 2026
at 12:30 pm

by

Xiaoyun Xu

born on January 5, 1995
in Sichuan, China

Supervisor:

Prof. dr. L. Batina

Co-supervisor:

Dr. S. Picek

Manuscript Committee:

Prof. dr. M. Loog (chair)
Prof. dr. ing. D. Jakobovic (University of Zagreb, Croatia)
Dr. L. Mariot (University of Twente)
Prof. L.Y. Chen (University of Neuchâtel, Switzerland)
Prof. Z. Zhao (Xi'an Jiaotong University, China)

Contents

Chapter 1

Introduction

Machine Learning (ML) has revolutionized the way we approach complex problems, enabling breakthroughs in areas such as autonomous vehicles [1], financial analysis [2], medical analytics [3], and ChatGPT [4]. Its ability to learn patterns from data and make accurate predictions has made it an indispensable tool in modern technology. However, despite its remarkable capabilities, ML has vulnerabilities. One of the most pressing challenges in the field is its susceptibility to adversarial ML attacks, where malicious actors exploit weaknesses in ML models to manipulate their behavior.

Adversarial ML aims to explore the vulnerabilities of ML technologies. Those attacks can take various forms, but two of the most prominent are evasion attacks [5, 6, 7] and backdoor attacks [8, 9, 10, 11]. In evasion attacks, an adversary crafts carefully designed inputs, often imperceptible to humans, to deceive a trained model during inference, causing it to make incorrect predictions. For example, adding subtle noise to an image might cause a facial recognition system to misidentify a person. On the other hand, backdoor attacks involve poisoning the training data or model to embed a hidden trigger. Once activated, the model behaves as intended, but when the trigger is present, it produces malicious outputs. These attacks highlight the fragility of ML systems and the need for robust defenses.

To mitigate these threats, researchers have developed a range of common defenses. These include adversarial training (AT) [12], where models are trained on adversarial examples to improve their resilience, and defensive distillation [13], which involves training a model to be less sensitive to small input perturbations. Other approaches include input preprocessing [14], anomaly detection [15], and formal verification methods [16] to ensure model robustness. Despite these efforts, adversarial ML remains an ongoing arms race, as attackers continually devise new strategies to bypass defenses.

This thesis investigates the fundamental principles underlying evasion and backdoor attacks, focusing on developing robust defense mechanisms to counteract these threats. By integrating insights from both evasion and backdoor attack domains, we demonstrate the significant overlaps between these two areas of adversarial ML. These overlaps serve as a foundation for proposing effective and efficient defense strategies that address vulnerabilities across multiple attack vectors. In this introduction, we present the essential background and motivation for our research, highlighting its critical importance in advancing the field of ML security. Following this, we provide an overview of the structure and contributions of this thesis, outlining the key themes and methodologies explored in subsequent chapters.

1.1 Background

1.1.1 Overview of Machine Learning and Its Security Risks

ML is a branch of artificial intelligence (AI) that focuses on developing algorithms capable of learning patterns from data and making data-driven decisions without explicit programming [17]. ML algorithms can be broadly categorized into *supervised* and *unsupervised* learning paradigms based on data availability and learning objectives.

Architectures of Machine Learning. Classical ML has its roots in statistics and early AI research. In 1805, Legendre introduced the method of least squares, later forming the basis of linear regression [18]. In the 1950s, Alan Turing raised the question of machine intelligence [19], Arthur Samuel built a self-learning checkers program [20], and Rosenblatt proposed the perceptron, an early neural network [21]. After periods of slow progress, breakthroughs like backpropagation in the 1980s [22], support vector machines in the 1990s [23].

Recently, Deep Learning (DL) architectures have gotten more attention due to their promising performance on various tasks, such as Convolutional Neural Networks (CNNs) [24, 25, 26, 27], Recurrent Neural Networks (RNNs) [28, 29, 30], and Transformer Architectures [31, 32, 33]. The basis of these structures is the Feedforward Neural Networks (FNNs), also known as multilayer perceptrons (MLPs), consisting of input, hidden, and output layers [22]. MLPs introduced the concept of learning hierarchical representations using non-linear activation functions. An MLP with L layers can be mathematically represented as:

$$\mathbf{h}^{(0)} = \mathbf{x},$$

$$\mathbf{h}^{(l)} = f^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad \text{for } l = 1, 2, \ldots, L$$

Where:

- $\mathbf{x}$ is the input vector.
- $\mathbf{W}^{(l)}$ is the weight matrix of layer l.
- $\mathbf{b}^{(l)}$ is the bias vector of layer l.
- $f^{(l)}$ is the activation function (e.g., ReLU, sigmoid).
- $\mathbf{h}^{(l)}$ is the output of layer l.

The final output is:

$$\mathbf{y} = \mathbf{h}^{(L)}$$

Designed specifically for structured grid data, CNNs introduced local connectivity, weight sharing, and pooling operations [24], making them highly effective for visual tasks. Landmark architectures such as AlexNet [25], VGG [26], ResNet [27], and ConvNext [34] demonstrated the scalability and power of CNNs for large-scale image recognition tasks.

For sequence modeling, RNNs [28] introduced temporal recurrence to capture dependencies over time. However, issues of vanishing gradients were mitigated by advanced architectures such as Long Short-Term Memory (LSTM) [29] and Gated Recurrent Units (GRUs) [30], which introduced gating mechanisms to preserve long-term information.

The Transformer model [35] revolutionized sequence modeling by replacing recurrence with attention mechanisms, enabling parallelization and capturing long-range dependencies more effectively. Since its inception, variants such as BERT [31], GPT [36], and ViT [32] have established new benchmarks in natural language processing and computer vision.

Learning Paradigms of Machine Learning. Supervised ML refers to the algorithm that learns from labeled training data to make predictions or decisions.

The goal is to train a model that can generalize patterns from known examples to accurately predict outcomes for new, unseen data. Notable approaches include handwritten digit recognition [24], ImageNet classification [25, 27], etc.

Unsupervised ML discovers hidden patterns or structures in unlabeled data without predefined outcomes. Unlike supervised learning, there are no target labels. The algorithm explores the data on its own to find meaningful insights. Early unsupervised ML technologies explored clustering (e.g., K-Means [37], Hierarchical Clustering [38], and Density-Based Clustering [39]) and dimensionality reduction (e.g., Principal Components Analysis [40] and t-SNE [41]). Recent works focus on unsupervised ML for pre-training, such as Contrastive Learning [42], Masked Language Modeling [31], and Masked Image Modeling [43].

The Stages of Machine Learning. The ML workflow consists of two primary phases: *training* (model learning) and *inference* (model prediction). In the training phase, the goal is to teach the model to recognize patterns in data by optimizing its parameters. The model's parameters are tuned while training to align the model's output with the ground truth as much as possible. This is achieved by minimizing the loss function, which quantifies the difference between the model's output and the ground truth. More specifically, the optimization of the model's parameters is guided by the gradient of the loss function, so that the loss is minimized in the direction where the loss value decreases fastest. This process is commonly conducted on a small random subset (batch) of training data, i.e., Mini-Batch Gradient Descent [44]. However, training a well-performing model suffers from difficulties such as the gradient explosion [45] and overfitting [46]. These problems can be alleviated to a certain extent with, e.g., batch normalization [47] for gradient explosion, dropout [46] for overfitting.

Once the model is trained, it is expected to perform deterministically at the inference stage and then can be deployed to different tasks according to users' requirements. A necessary step is to stop operations that introduce uncertainty. For example, the dropout at the training stage stops the activation value of a neuron with a certain probability, which helps to train a more generalizable model [46]. In the inference time, dropout must be closed to maintain the same output for the same input.

Security Risks From Adversarial Machine Learning. The boom in ML has also attracted attacks from malicious adversaries. According to adversaries' capabilities, attacks can occur in two stages (training and inference) of ML.

In the training stage, adversaries attempt to introduce abnormality into the model's parameters, which results in malicious behaviors of the victim model. These attacks can be divided into backdoor and poisoning attacks according to the adversaries' goal. The **backdoor attack** [8, 10, 11] refers to injecting a secret functionality into the victim model that is activated through malicious inputs that contain the trigger. The **poisoning attack** [48, 49, 50] refers to introducing shortcuts in the training data such that the model learns the shortcuts instead of patterns of the data. These attacks are usually achieved by introducing poisoned data points in the training data, but adversaries can also inject a backdoor by directly editing the model's weights [51].

In the inference stage, adversaries attempt to mislead the well-trained model (evasion attack [5, 7]) or extract privacy information from the model (membership inference [52, 53] and model stealing attacks [54, 55]). The **evasion attack** refers to manipulating input data at inference time to cause an ML model to make incorrect predictions, while keeping the input visually/functionally similar to benign data. The **membership inference attack** refers to determining whether a data point was used to train a target model or not. **Model stealing attacks** (also called model extraction attacks) occur when an adversary reconstructs or approximates a target ML model by querying its predictions [54] or using side-channel information [55]. Unlike backdoor and poisoning attacks (which corrupt training data), attacks at inference time exploit vulnerabilities in deployed models without altering their parameters.

Among these security risks raised by malicious adversaries, evasion and backdoor attacks are considered the most prominent and practical problems, as other attack methods are limited by significant drawbacks [56, 57, 58]. For threats during the training time, the poisoning attacks require poisoning the whole training set (100% of poisoning rate). A poisoning rate below 100% (even 90%) will result in a significant decrease in attack effectiveness for the poisoning attack [56]. In contrast, backdoor attacks can perform well even at a very low poisoning rate [57]. For threats at the inference time, membership inference attacks cannot give reliable evidence of data usage in the training [58]. Model stealing attacks may require physical access to the devices where the model is deployed [55]. In contrast, evasion attacks can be achieved by slightly perturbing the input of the model at both black-box and white-box settings. Therefore, this thesis focuses on the backdoor attacks for training time and evasion attacks for inference time.

1.1.2 Evasion Attacks

Evasion attacks [7, 6, 5], commonly referred to as adversarial attacks, involve the application of imperceptible adversarial perturbations to the original input of a target ML model, specifically a neural network in this context. These perturbations generate adversarial examples designed to deceive the victim model.

Consider an L-layer neural network F_θ tasked with classification in a d_Y-dimensional space, and a training dataset $D = \{(x_i, y_i)\}_{i=1}^n$ in a d_X-dimensional space. The primary objectives of adversarial attacks can be delineated as follows:

- **Misleading the Network**: The generated perturbation δ should effectively mislead the network by maximizing the cross-entropy loss $\mathcal{L}_{CE}$. This can be formalized as:

  $$\max_{\delta \in S} \mathcal{L}_{CE}(\theta, x_i + \delta, y_i), \tag{1.1}$$

  where $x_i \in \mathbb{R}^{d_X}$ and $y_i \in \{0, 1\}^{d_Y}$, θ represents the parameters of the neural network, and $\mathcal{L}_{CE}$ denotes the standard cross-entropy loss. The perturbation aims to reduce the model's confidence in the ground truth labels while increasing confidence in incorrect labels, thereby amplifying the loss between the adversarial outputs and the true labels.

- **Preserving Similarity**: The adversarial examples should remain as similar as possible to the original clean examples. This is achieved by constraining δ to a small domain, defined as:

  $$S = B(x_i, r) = \{\delta \in \mathbb{R}^{d_X} : \|\delta\|_\infty \leq r\}, \tag{1.2}$$

  where S represents the l_∞-ball of radius r centered at x_i in $\mathbb{R}^{d_X}$. The similarity between x_i and $x_i + \delta$ can be quantified using various norms, such as l_0, l_2, and l_∞.

When employing these attacks to generate adversarial examples for training purposes, the learning objective is formulated as a min-max optimization problem:

$$\min_{\theta} \max_{\delta \in S} \mathcal{L}_{CE}(\theta, x_i + \delta, y_i). \tag{1.3}$$

This objective seeks to minimize the worst-case loss over the set of possible perturbations, thereby enhancing the robustness of the model against adversarial examples.

White-box Evasion Attacks. The concept of evasion attacks first appeared in [6], which proposed a formal framework and algorithms against the adversarial spam detection domain. Then, the evasion attacks were popularized by Biggio et al. [5] and Szegedy et al. [7] in image classification. The key to such an attack is to find the perturbation that can mislead the victim model, i.e., Eq. (1.1).

The adversary has full access to the model and training data in the white-box setting. A straightforward way is then to search for the adversarial perturbation by the gradient of the victim model, i.e., *gradient-based methods.* Fast Gradient Sign Method (FGSM) [59] is one of the most representative works that use the gradient of image pixels to create adversarial perturbations. More specifically, FGSM computes the gradient of Eq. (1.1) for one step and linearizes the gradient (i.e., the sign of the gradient) to obtain an optimal max-norm constrained perturbation. The perturbation is constrained by a constant for imperceptibility. To find more general perturbations for AT, Projected Gradient Descent (PGD) [12] extends FGSM to multiple iterations, i.e., calculating the gradient multiple times with a smaller step size than FGSM. PGD also introduced a random initialization of the perturbation, further exploring a larger search space. In addition to the above iterative gradient-based methods, the adversary can design a more specific objective to find the perturbation via an optimization process [7, 60] or generate adversarial perturbations with generative adversarial networks [61, 62, 63] and diffusion models [64, 65, 66].

Black-box Evasion Attacks. The requirements of white-box access to the model and training data may not always be available, and black-box access is more practical for conducting successful attacks. Black-box attacks consist of *query-based* and *transfer-based* attacks.

Query-based attacks involve repeatedly querying the target model to generate adversarial perturbations based on the model's responses, which may include either the final decision (referred to as hard-label attacks) or the confidence scores (referred to as soft-label attacks). In hard-label attacks, the adversary typically employs random search methods, which involve iteratively updating the perturbation direction through random exploration or heuristic adjustments. The critical challenge lies in determining the optimal search direction, which can be addressed using random walk [67], evolutionary algorithms [68], estimating the normal vector to the decision boundary [69], or discrete optimization methods [70]. Alternatively, the adversary may estimate the gradient based on the model's decisions, enabling the use of gradient-free methods to solve the perturbation optimization problem [71, 72]. In soft-label attacks, similar random search [73, 74, 75] and gradient estimation methods [76, 77, 78] can be applied. However, soft-label attacks are generally more likely to succeed than hard-label attacks, as the availability of confidence scores provides additional information to locate the decision boundary more effectively. Furthermore, the adversary can integrate query-based feedback with the surrogate model to reduce the number of queries required [79, 80], thereby increasing the attack's efficiency.

Transfer-based attacks, on the other hand, aim to deceive the target model by leveraging perturbations generated from a locally accessible surrogate model to which the adversary has white-box access. The primary challenge in these attacks is to prevent the perturbation from overfitting to the surrogate model, ensuring its transferability to other unknown models. To enhance transferability, the adversary can utilize benign samples to mitigate overfitting [81, 82] and employ gradient-based techniques [83, 84] to escape suboptimal local maxima. Additionally, feature-level attacks [85, 86], which target the intermediate feature representations of the surrogate model, can further improve transferability. Another strategy involves creating an ensemble of surrogate models to generate more transferable perturbations [87, 88]. These approaches collectively highlight the sophisticated methodologies employed in transfer-based attacks to achieve adversarial success across diverse models.

1.1.3 Defenses Against Evasion Attacks

Defenses against evasion attacks can be broadly categorized into three approaches: (1) training robust models, (2) purifying potential adversarial perturbations during the pre-processing stage, and (3) enhancing robustness through architectural modifications. However, pre-processing purification methods often significantly degrade image quality, leading to reduced performance on clean data [89, 14, 90]. Similarly, architectural modifications can introduce obfuscated gradients, which may inadvertently create new vulnerabilities [91, 92, 93]. Given these limitations, this thesis focuses on developing robust models through training strategies. Among these, one of the most straightforward and effective methods is to leverage adversarial examples as feedback during training, a technique commonly referred to as Adversarial Training (AT). This approach aims to improve model robustness by explicitly incorporating adversarial examples into the training process, thereby enabling the model to learn to resist such attacks.

Effectiveness of Adversarial Training (AT). AT was initially developed to enhance the robustness of ML models during the learning process. In the context of supervised classification, for instance, the defender generates adversarial examples and incorporates them as supplementary training data with their corresponding correct labels. A critical aspect of this approach is the generation of generalized adversarial examples, which, when used effectively alongside clean data, can improve the model's robustness against adversarial attacks. Goodfellow et al. [59] introduced a method that integrates the Cross-Entropy (CE) loss computed on FGSM adversarial examples and their correct labels into the objective function, i.e., FGSM-based AT. While FGSM-based AT optimizes the model using both adversarial and clean examples, its one-step generation process is insufficient to defend against more sophisticated iterative attacks, such as the PGD attack. PGD AT [12] addresses this limitation by generating more generalized adversarial examples through multiple iterations and random initialization. However, AT often leads to a notable decline in the model's performance on clean data and is prone to overfitting. To address this, TRADES [94] proposes a theoretically grounded upper bound on the discrepancy between adversarial and clean error rates, enabling the design of an AT method that balances robustness and accuracy. Additionally, to mitigate overfitting, defenders can leverage externally generated data, such as that produced by GANs or diffusion models, to enhance both robustness and accuracy [95, 96, 97]. Recent advancements [98] demonstrate that improved generative models further augment the effectiveness of AT.

Efficiency of Adversarial Training. Another critical consideration for defenders is the efficiency of AT, as generating adversarial examples and incorporating additional loss terms are computationally intensive processes. Free-AT [99] addresses this challenge by reusing gradient information computed during model parameter updates, thereby eliminating the overhead associated with adversarial example generation. Fast-AT [100] demonstrates that employing a one-step FGSM approach with random initialization of perturbations and a larger step size can achieve performance comparable to PGD-based AT while significantly reducing computational time. Furthermore, Haizhong et al. [101] observed that adversarial examples generated in one training epoch often remain effective in subsequent epochs. This insight allows for the reuse of adversarial examples across multiple training epochs, substantially reducing the overall training time. These advancements collectively contribute to making AT more practical and scalable for real-world applications.

1.1.4 Backdoor Attacks

Backdoor attacks compromise the integrity of ML models by ensuring that the model behaves normally on benign inputs but misclassifies inputs containing a specific trigger into a target class. The trigger can manifest as a visible pattern inserted into the input space or as a property that alters the feature representation of the input in the feature space. Ultimately, regardless of the specific attack method, the parameters of the backdoored model in the parameter space are modified. To implant a backdoor, attackers typically assume control over a small portion of the training data in the poison training scenario [8, 9, 102]. In the supply-chain setting, where backdoored models are provided to users, attackers also control the training process [103, 11, 104, 10, 105]. Additionally, backdoors can be introduced by directly modifying the model's weights [106, 51, 107, 108].

Input-space attacks. Traditional backdoor attacks often employ simple patterns as triggers. For instance, BadNets [8] uses a fixed patch, while Blend [9] incorporates a Hello Kitty pattern into images. These non-stealthy triggers introduce abnormal data into the training set, making them susceptible to detection by human inspectors or defensive mechanisms [109, 110]. To enhance stealthiness, more sophisticated triggers have been developed to achieve invisibility in the input space. For example, IAD [11] introduces dynamic triggers that vary across inputs, while WaNet [10] proposes warping-based triggers that evade human inspection. BppAttack (Bpp) [105] leverages image quantization and dithering to create imperceptible changes. Although these methods successfully bypass traditional defenses [110], they still introduce separable features that can be detected by feature-space defenses [111, 112].

Feature-space attacks. Recognizing the vulnerability of input-space attacks to feature-space defenses, backdoor attacks have evolved to achieve greater stealth in the feature space. These attacks often assume additional control over the training process. For instance, [103, 113, 114, 115] design new loss functions to minimize the disparity between the backdoor and benign features. Beyond loss function modifications, TACT [116] and SSDT [117] highlight that source-specific attacks - where only specified source classes are poisoned - help obscure differences between benign and backdoor features. Additionally, [118] proposes Adaptive-Blend (Adap-Blend) and Adaptive-Patch (Adap-Patch), which obscure feature differences by (1) including poisoned samples with correct labels, (2) using asymmetric triggers (stronger triggers at inference), and (3) employing trigger diversification during training.

Supply-chain attacks. Supply-chain attacks have garnered significant attention due to their applicability in real-world scenarios where backdoored models are distributed as final products. In these attacks, adversaries control both the training data and the training process. Notably, feature-space attacks [103, 119, 115, 113, 120, 121, 114, 122, 117] that assume control over the training process are a subset of supply-chain attacks, as their output is a backdoored model. Beyond training control, another category of supply-chain attacks involves directly modifying the model's weights in the parameter space, known as parameter-space attacks. Techniques such as T-BFA [123], TBT [124], and ProFlip [125] explore altering susceptible bits of DNN parameters stored in memory (e.g., DRAM) to inject backdoors. SRA [107] and handcrafted backdoors [51] directly modify subsets of model parameters to increase the logits of the target class. However, these methods typically require a local benign dataset to guide the selection of parameters for modification. Data-free backdoor attacks [126] eliminate the need for benign data by using substitute data unrelated to the main task for fine-tuning. DFBA [108] further advances this approach by proposing a retraining-free, data-free backdoor attack that injects a backdoor path - a single neuron from each layer except the output layer - into the victim model.

1.1.5 Defenses Against Backdoor Attacks

Backdoor defenses can be categorized into two primary strategies: detection and mitigation. Detection involves identifying whether a model has been compromised (model detection) [110, 127, 113, 128, 112]. Model detection techniques analyze the intrinsic properties or behavioral anomalies of the model to uncover hidden backdoors. Mitigation aims to neutralize the backdoor effect in compromised models. This can be achieved through (1) pruning-based defenses, which remove backdoor-related neurons or channels [129, 130, 131, 132], or (2) fine-tuning-based defenses, which erase the backdoor trigger by retraining the model on sanitized data or incorporating adversarial unlearning objectives [133, 134, 135, 112]. Recent advancements propose proactive defenses [136, 137, 138], leveraging the defender's home-field advantage* to preemptively detect or suppress potential backdoors during model development.

Backdoor detection. Backdoor trigger reverse engineering, also referred to as trigger inversion, is widely regarded as one of the most practical defenses for backdoor detection due to its applicability in both poisoning training and supply-chain

*The defender retains full control over the system and can monitor or modify the training process.

scenarios [111, 128, 112, 139]. As a post-training method, trigger inversion operates by identifying potential backdoor triggers within a specific model. If a trigger is successfully identified, the model is deemed backdoored, and the discovered trigger can subsequently be used to unlearn the backdoor. This process is implemented as an optimization procedure that leverages the model and a local benign dataset. For instance, Neural Cleanse (NC) [110] pioneered trigger inversion by optimizing a mask and pattern in the input space that could mislead the model into predicting a target class. This optimization is repeated across all classes, and the model is flagged as backdoored if an outlier trigger significantly smaller than those for other classes is detected. While NC-like methods are effective against fixed-patch trigger attacks, such as BadNets [8] and Blend [9], they often fail against input-stealthy attacks like WaNet [10]. To overcome this challenge, FeatureRE [111] shifted the trigger inversion process from the input space to the feature space. Unicorn [128] further advanced this approach by introducing a transformation function capable of handling attacks in other spaces, such as numerical space [105].

Backdoor Mitigation. Backdoor mitigation techniques primarily encompass fine-tuning and pruning, both of which have demonstrated efficacy without requiring prior knowledge of backdoor triggers. Pruning-based approaches focus on identifying and eliminating neurons associated with backdoor functionality. Fine-Pruning (FP) [129] removes inactive neurons when processing benign inputs and subsequently fine-tunes the pruned network. Adversarial Neuron Pruning (ANP) [130] identifies backdoor-related neurons by introducing adversarial noise to neuron weights, thereby activating the backdoor. Recovery-based Neuron Pruning (RNP) [132] employs an unlearning and recovery process on benign data, leveraging the recovery phase to suppress backdoor neurons while preserving the model's performance on benign tasks. In contrast to these approaches, which rely on benign data, Channel Lipschitz Pruning (CLP) [131] directly analyzes the Channel Lipschitzness Constant of the network and removes channels with high Lipschitz constants in a data-free manner.

Traditional fine-tuning as a defense mechanism typically relies on trigger inversion methods to reconstruct the trigger and subsequently unlearn it. For example, BTI-DBF(U) [112] fine-tunes backdoor models using triggers reconstructed through their inversion algorithm. However, there is no assurance that the reconstructed trigger accurately represents the true backdoor trigger. Recent studies have explored fine-tuning without explicit trigger information, instead leveraging prior human knowledge. For instance, FT-SAM [133] identifies a positive correlation between the weight norms of neurons and backdoor-related neurons. Consequently, they propose a fine-tuning approach that adjusts the large outliers in weight norms using Sharpness-Aware Minimization (SAM). I-BAU [134] adopts a min-max fine-tuning framework akin to adversarial training, where the inner maximization seeks perturbations that mislead the model, and the outer minimization maintains the model's performance on benign data.

1.2 Motivation

The motivation for this study stems from two interconnected research imperatives. First, the protection of trained models against evasion attacks is a pressing concern in the field of ML security. Evasion attacks, which involve carefully crafted adversarial perturbations to mislead models, undermine the reliability and robustness of deployed systems. Safeguarding models against such attacks is critical to ensuring their trustworthiness, particularly in high-stakes applications such as cybersecurity, autonomous systems, and healthcare. Developing effective countermeasures to mitigate evasion threats is, therefore, a fundamental objective of this work. Adversarial training (AT) is considered one of the most promising strategies against evasion attacks. Existing works about AT focus on empirically improving robustness (such as the better combination of hyper-parameters [140, 141] or data augmentation [98]) or new end-to-end AT methods to solve existing problems [142, 143]. However, these methods remain computationally intensive and often struggle to generalize across diverse settings, particularly when applied to state-of-the-art architectures. In **Part I** (Chapters 2 and 3), this thesis aims to explore the better generalizability and time-consumption of AT from the perspective of information theory and self-supervised training. By leveraging theoretical insights, the aim is to develop more scalable and adaptable AT methods that can effectively enhance model robustness across various scenarios.

Second, the representation of adversarial perturbations provides valuable insights for enhancing backdoor defenses. Adversarial perturbations often uncover subtle patterns and vulnerabilities within a model's decision boundaries, which can be leveraged to identify and neutralize backdoor triggers. By utilizing these representations, it becomes possible to design more precise and effective defense mechanisms that specifically target backdoor-related features, thereby improving the model's resilience against such threats. This approach not only strengthens the security of ML systems but also advances our understanding of the intricate relationship between adversarial robustness and backdoor vulnerabilities. Existing research primarily focuses on defenses operating in the input space [110, 144] or feature space [111, 128], ignoring the potential relevance of adversarial perturbations. In **Part II** (Chapters 4, 5, and 6), this thesis investigates the development of effective and efficient defenses by leveraging adversarial perturbations, aiming to bridge this gap and provide a novel perspective on backdoor defense. In addition, Chapter 6 explores the dual role of adversarial representations in enhancing backdoor attacks and analyzing their underlying mechanisms. Existing backdoor attacks focus on input invisibility or inseparability in feature space to improve the stealthiness of backdoor attacks, neglecting the fact that backdoor behaviors are ultimately embedded within the parameters of the backdoored model. By strategically incorporating adversarial perturbations and parameter space techniques into backdoor attacks, we aim to uncover the fundamental reasons for their success and identify potential weaknesses in existing defenses.

This thesis provides a comprehensive perspective on the relationship between adversarial and backdoor threats, enabling the development of more robust models capable of withstanding both types of attacks. Ultimately, this work contributes to advancing the theoretical and practical understanding of ML security, paving the way for more resilient and trustworthy systems.

1.3 Thesis Contributions and Outline

This thesis focuses on adversarial ML and makes several significant contributions in two directions, which are outlined in Figure 1.1:

- In **Part I**, we explore how information bottleneck can improve adversarial training in both supervised and self-supervised training paradigms. We propose novel training methods for adversarial robustness with explainable theoretical proofs. We also provided SOTA empirical performance in the experiment and obtained the Top-1 result on the most famous benchmark, RobustBench [145], in the field of adversarial robustness.

- In **Part II**, we explore the connection between adversarial perturbation and backdoor attacks. We propose two backdoor detection methods, including applying adversarial perturbation in the input space to improve backdoor detection performance and detecting backdoors according to adversarial noise in the parameter space for more effective and efficient detection. With parameter space defenses, we show that correct backdoor attacks designed to be stealthy

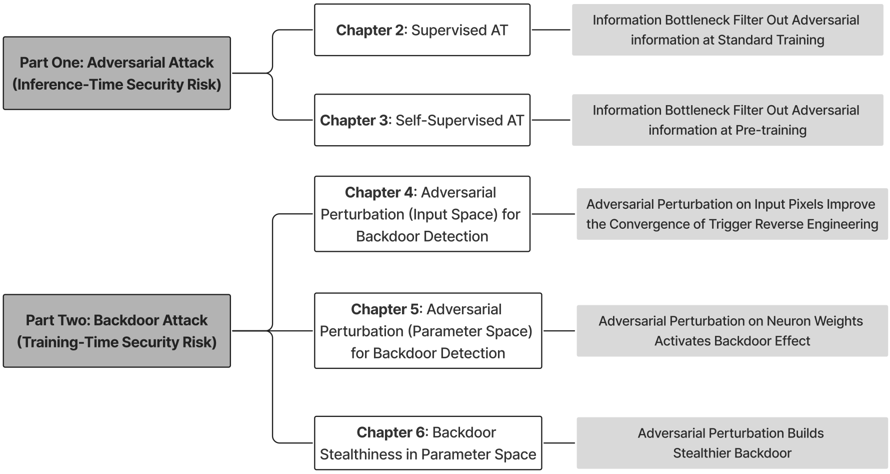

Part One: Adversarial Attack (Inference-Time Security Risk)
Chapter 2: Supervised AT
Information Bottleneck Filter Out Adversarial information at Standard Training
Chapter 3: Self-Supervised AT
Information Bottleneck Filter Out Adversarial information at Pre-training
Part Two: Backdoor Attack (Training-Time Security Risk)
Chapter 4: Adversarial Perturbation (Input Space) for Backdoor Detection
Adversarial Perturbation on Input Pixels Improve the Convergence of Trigger Reverse Engineering
Chapter 5: Adversarial Perturbation (Parameter Space) for Backdoor Detection
Adversarial Perturbation on Neuron Weights Activates Backdoor Effect
Chapter 6: Backdoor Stealthiness in Parameter Space
Adversarial Perturbation Builds Stealthier Backdoor

Figure 1.1: The structure of the thesis consists of two parts about inference-time and training-time risk (introduction and discussion chapters are omitted). The first part (**Part I**) contains two chapters (**Chapter 2 and 3**) about how we introduce information bottleneck into traditional supervised AT for standard training and self-supervised AT for pre-training. The second part (**Part II**) contains three chapters (**Chapter 4, 5, and 6**) about how we use our experience in adversarial attacks for more effective and efficient backdoor detection, and improving the stealthiness of backdoor.

can be easily spotted. Therefore, we propose a novel backdoor attack method to explore achieving comprehensive stealthiness against backdoor defenses.

In **Chapter 2**, we investigate the relationship between adversarial perturbations and their learned representations in the feature space from the perspective of the information bottleneck theory. Our analysis reveals that compressing redundant information in the input space enhances the robustness of deep learning models. Building on this insight, we propose a novel AT method, namely IB-RAR, to defend against evasion attacks. IB-RAR improves the accuracy by an average of 2.66% against five adversarial attacks on three benchmark datasets (CIFAR-10, CIFAR-100, and Tiny-ImageNet) compared to baseline methods. In addition, IB-RAR also reaches an accuracy of 35.86% against PGD without adversarial training, while standard-trained models have almost 0% accuracy. This chapter is based on the following paper:

- **Xiaoyun Xu**, Guilherme Perin, Stjepan Picek. IB-RAR: Information Bottleneck as Regularizer for Adversarial Robustness. IEEE/IFIP International Conference on

Dependable Systems and Networks Workshops (DSN-W), 2023.

> The author's contribution: The author of this thesis contributed to the formulation and conception of this work, writing and engineering work, including implementing the code and running the experiments. Guilherme Perin and Stjepan Picek contributed to the interpretation of research data and writing.

In **Chapter 3**, we further extend the information bottleneck to a self-supervised paradigm. We propose a theoretically grounded adversarial pre-training method, MIMIR, in self-supervised form and demonstrate improved performance on clean and adversarial examples. Our method also provides a foundation for future research aimed at developing more robust and interpretable machine-learning models. Specifically, we applied the self-supervised method called mask image modeling to adversarial training. We provide theoretical justification with the upper and lower bounds of mutual information, which intuitively explains why MIMIR achieves improved performance. We conducted extensive experiments on three benchmark datasets (CIFAR-10, Tiny-ImageNet, and ImageNet-1K), and MIMIR achieves 3.98% higher robustness on ViT-B and ImageNet-1K compared to the current SOTA method. In particular, MIMIR achieved the Top-1 robustness in the Robustbench. In addition, we also proposed two adaptive attacks against MIMIR, i.e., when the attacker knows the design of MIMIR. MIMIR shows stable robustness against adaptive attacks. This chapter is based on the following paper:

- **Xiaoyun Xu**, Shujian Yu, Zhuoran Liu, Stjepan Picek. MIMIR: Masked Image Modeling for Mutual Information-based Adversarial Robustness. arXiv preprint.

> The author's contribution: The author of this thesis contributed to the formulation and conception of this work, writing and engineering work, including implementing the code and running the experiments. Stjepan Picek helped design the MIMIR AT methods, and Zhuoran Liu helped design the two adaptive attacks. Regarding the theoretical justification, the author of this thesis derives the lower bound, and the upper bound is derived by Shujian Yu. All co-authors contributed to the interpretation of research data and writing.

In **Chapter 4**, we explore the sensitivity of backdoored models to adversarial examples. Our findings indicate that adversarial examples can exploit the covert functionality injected by backdoor attacks to generate more subtle and effective perturbations. Specifically, we propose a backdoor detection method, USB, that utilizes adversarial examples to boost the reverse engineering of backdoor triggers. USB determines that a trained model is backdoored if there is an outlier among the reversed triggers. USB can detect stronger backdoor triggers for both patch-based (BadNet and Latent) and non-patch-based (InputAware Dynamic) backdoors. We conducted experiments on 240 models to assess our approach. We compared USB with NC and TABOR methods, and USB provides competitive performance on various datasets. This chapter is based on the following paper:

- **Xiaoyun Xu**, Oguzhan Ersoy, Behrad Tajalli, Stjepan Picek. Universal soldier: Using universal adversarial perturbations for detecting backdoor attacks. IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2024.

> The author's contribution: The author of this thesis contributed to the formulation and conception of this work, writing and engineering work, including implementing the code and running the experiments. Behrad Tajalli helped design the experiment for an advanced attack (WaNet). Oguzhan Ersoy and Stjepan Picek contributed to designing the USB method, the interpretation of research data, and writing.

In **Chapter 5**, we investigate the impact of adversarial perturbations on neuron weights, which directly activate the backdoor functionality without the need for trigger inversion. Specifically, the proposed method, BAN, is 1.37× (on CIFAR-10) and 5.11× (on ImageNet200) more efficient with an average 9.99% higher detect success rate than the state-of-the-art defense BTI-DBF against five representative attacks, including BadNets, Blend, WaNet, IAD, and Bpp. We also exploit the neuron noise to further design a simple yet effective fine-tuning defense for removing the backdoor, such that we build a workable framework. Our work highlights the utility of analyzing neuron weights in the parameter space for understanding backdoor behavior and underscores the importance of the parameter space in developing advanced defense methodologies. This chapter is based on the following paper:

- **Xiaoyun Xu**, Zhuoran Liu, Stefanos Koffas, Shujian Yu, Stjepan Picek. BAN: Detecting Backdoors Activated by Adversarial Neuron Noise. Advances in Neural Information Processing Systems (NeurIPS), 2024.

> The author's contribution: The author of this thesis contributed to the formulation and conception of this work, writing and engineering work, including implementing the code and running the experiments. Stjepan Picek helped design the BAN detection and fine-tuning. Zhuoran Liu and Stefanos Koffas helped visualize the experimental results and demonstrate the method structure. All co-authors contributed to the interpretation of research data and writing.

In **Chapter 6**, we conduct a systematic analysis of 12 representative backdoor attacks and 17 defenses. We identify a critical blind spot: current backdoor attacks, despite being designed to be stealthy against backdoor defenses, often fail when confronted with diverse practical defense mechanisms. This vulnerability arises because the injected backdoor inevitably introduces prominent backdoor-related neurons, which are detectable by advanced defenses. To address this limitation, we propose a novel backdoor attack incorporating an adversarial backdoor injection module inspired by AT. This module ensures stealthiness across the input, feature, and parameter spaces, enabling the attack to maintain robustness against a wide range of representative defense methods. All 12 baseline attacks failed against at least a part of the 17 defenses on the three benchmark datasets. Our method is the only one that can sustain against all the defenses. We further validate the effectiveness of the adversarial backdoor injection module by integrating it with the other 10 attacks. We also propose an adaptive defense that knows the design of our method to verify its robustness. This chapter is based on the following paper:

- **Xiaoyun Xu**, Zhuoran Liu, Stefanos Koffas, Stjepan Picek. Towards Backdoor Stealthiness in Model Parameter Space. ACM Conference on Computer and Communications Security (CCS), 2025.

> The author's contribution: The author of this thesis contributed to the formulation and conception of this work, writing and engineering work, including implementing the code and running the experiments. Zhuoran Liu and Stefanos Koffas contributed to the analysis of adaptive defense and supply chain attacks. All co-authors contributed to the interpretation of research data and writing.

In **Chapter 7**, we conclude our findings in adversarial ML. We hope our research raises awareness of risks in ML applications. We also emphasize the key points of proposing effective and efficient defenses in future work.

1.4 List of Publications

This thesis is built on several publications co-authored by the author during the Ph.D study, as indicated.

Xiaoyun Xu, Guilherme Perin, Stjepan Picek. IB-RAR: Information Bottleneck as Regularizer for Adversarial Robustness. IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2023. [**Chapter 2**]

Xiaoyun Xu, Shujian Yu, Zhuoran Liu, Stjepan Picek. MIMIR: Masked Image Modeling for Mutual Information-based Adversarial Robustness. arXiv preprint. [**Chapter 3**]

Xiaoyun Xu, Oguzhan Ersoy, Behrad Tajalli, Stjepan Picek. Universal soldier: Using universal adversarial perturbations for detecting backdoor attacks. IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2024. [**Chapter 4**]

Xiaoyun Xu, Zhuoran Liu, Stefanos Koffas, Shujian Yu, Stjepan Picek. BAN: Detecting Backdoors Activated by Adversarial Neuron Noise. Advances in Neural Information Processing Systems (NeurIPS), 2024. [**Chapter 5**]

Xiaoyun Xu, Zhuoran Liu, Stefanos Koffas, Stjepan Picek. Towards Backdoor Stealthiness in Model Parameter Space. ACM Conference on Computer and Communications Security (CCS), 2025. [**Chapter 6**]

Other Publications during the Ph.D Study

Xiaoyun Xu, Stjepan Picek. Poster: Boosting Adversarial Robustness by Adversarial Pre-training. ACM Conference on Computer and Communications Security (CCS), 2023.

Zhuoran Liu, Senna van Hoek, Péter Horváth, Dirk Lauret, **Xiaoyun Xu**, Lejla Batina. Real-world Edge Neural Network Implementations Leak Private Interactions Through Physical Side Channel. arXiv preprint.

Part I

Inference-Time Adversarial Machine Learning

Chapter 2

Information Bottleneck in Adversarial Training

This chapter proposes a novel method, IB-RAR, which uses Information Bottleneck (IB) to strengthen adversarial robustness for both adversarial training and non-adversarial-trained methods. We first use the IB theory to build regularizers as learning objectives in the loss function. Then we filter out unnecessary features of intermediate representation according to their Mutual Information (MI) with labels, as the network trained with IB provides easily distinguishable MI for its features. Experimental results show that IB-RAR can be naturally combined with adversarial training and provides consistently better accuracy on new adversarial examples. The IB-RAR method improves the accuracy by an average of 2.66% against five adversarial attacks for ResNet-18, wide ResNet-28-10, and VGG-16, trained with three adversarial training benchmarks and the CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. In addition, IB-RAR also provides good robustness for undefended methods, such as training with cross-entropy loss only. Finally, without adversarial training, the VGG-16 network trained using IB-RAR on the CIFAR-10 dataset reaches an accuracy of 35.86% against PGD examples, while using all layers reaches 25.61% accuracy.

2.1 Introduction

Deep learning networks are vulnerable to adversarial attacks [7, 5]. Neural network predictions can be easily fooled by subtle adversarial perturbations, while the input remains visually imperceptible to humans. Such perturbations can be generated by specific algorithms, such as Fast Gradient Sign Method (FGSM) [59], projected gradient descent (PGD) [12], and Carlini & Wagner (CW) [60]. The main goal of these algorithms is to find as small perturbations as possible that mislead the prediction model. This potential vulnerability raises concerns about the reliability of practical deep learning applications, especially in security-sensitive fields, such as vulnerability detection [146], drug discovery [147], and financial market predictions [148].

Previous works have proposed many possible causes for successful adversarial attacks. Goodfellow et al. [59] argued that the adversarial examples are generated by the excessive linearity behavior of DNNs in high-dimensional spaces. Ilyas et al. [149] have demonstrated that adversarial attacks can arise from features (can be well-generalized) instead of bugs (do not generalize due to effects of poor statistical concentration). The features may be robust or not robust. Non-robust features can be easily manipulated by imperceptible noise, while robust features will not. Still, the community needs to reach a consensus on the underlying reason for the prevalence of adversarial examples.

To further analyze the impact of adversarial examples, IB is used as a learning objective to improve adversarial robustness [150, 151]. The IB is supposed to find the optimal trade-off between compression of input X and prediction of Y by MI ($I(\cdot)$) [152]. IB provides both performance and adversarial robustness when embedded into the learning objective. Intuitively, this is because X is mapped to Y through intermediate representation T (outputs of hidden layers). Compression of X ($I(X, T)$) naturally removes the noise in X and makes it difficult to transfer small perturbations via the bottleneck. However, computing mutual information is difficult in practice, especially when dealing with high-dimensional data. To address this problem, Alemi et al. [150] proposed Variational Information Bottleneck (VIB). They used the internal representation of a certain intermediate layer as a stochastic encoding T of the input data X. They aimed to learn the most informative representation T about the target Y, measured by the mutual information between their corresponding encoding values. Their experiments also showed that VIB is robust to overfitting and adversarial attacks. Ma et al. [153] proposed the HSIC Bottleneck and replaced mutual information with Hilbert Schmidt Independence Criterion (HSIC). They used the HSIC bottleneck as a learning objective for every layer of the network, which is an alternative to the conventional CE loss and back-propagation. Wang et al. [151] proposed HBaR, which combined HSIC Bottleneck of all hidden layers and back-propagation to improve both adversarial robustness and accuracy of clean data.

These IB-related methods use one layer (VIB) or all layers (HBaR) by default to build IB in their learning objectives, but we find that the IB of each layer has a different impact on robustness (see Table 2.3). Deeper layers usually provide more robustness with IB. The reason is that shallower layers usually generate representations with noise that are not informative enough to be distinguishable (see Figure 2.1). Therefore, MI computed from shallower layers is less meaningful than from deeper layers. We aim to compute mutual information (MI) between intermediate layers and their inputs or between intermediate layers and their targets. Then the MI is embedded into the loss function according to IB. We summarize the following two questions about applying IB as a learning objective:

- Which intermediate layers do we need to use? To address question 1), we propose using only robust layers to compute MI that is then used to apply IB in the loss function. We refer to robust layers as layers providing obviously higher accuracy than the network trained with only CE loss under PGD attack (since PGD provides good robustness against various attacks, as discussed later). To reduce the impact of adversarial training, we evaluate the performance of robust layers without adversarial training. This chapter empirically shows that compared to training with cross-entropy (CE) only, each layer of the network provides different degrees of robustness when applying IB as a learning objective. Then, using robust layers for IB objective upgrades robustness to adversarial attacks.
- Can the representation of the non-robust layers be further improved? To address question 2), we compute a mask to remove unnecessary feature channels of convolutional layers, as the outputs of non-robust layers are extracted by subsequent convolutional layers. When the network is trained with the loss function with IB and learns more informative features, some features are not relevant to the classification or target as they are not informative enough.

Our evaluation consists of two parts: (1) Combining our method with state-of-the-art adversarial training methods, e.g., PGD [12], TRADES [94], and MART [154]. Experimental results show that our method can improve the robustness of adversarial training methods. (2) Combining without adversarial training methods. Experimental results show that our method provides robustness compared to other IB-related techniques and plain CE. We also find that the robustness of VGG-16 mainly comes from the last convolutional block and the first two fully connected layers when using our method. When using these three layers to compute MI, the IB-RAR method provides higher accuracy on adversarial examples than using all layers. Ablation study results reveal which part of our method provides robustness and the connection among them. In addition, applying IB as a learning objective also accelerates training convergence according to experimental results. Our implementation is available at `https://github.com/xiaoyunxxy/IB-RAR/`.
Our main contributions are:

- We apply IB as a regularizer to improve robustness and natural accuracy (on clean data), and we remove unnecessary features based on the regularized network.
- We show that our method can be naturally embedded into state-of-the-art adversarial training methods. Our method improves accuracy on adversarial examples by an average of 2.66% against five adversarial attacks for ResNet-18, wide ResNet-28-10, and VGG-16 in Tables 2.1 and 2.2.
- We show that our method can improve robustness for weaker methods, like plain stochastic gradient descent (SGD) trained with the CE loss function only. With-

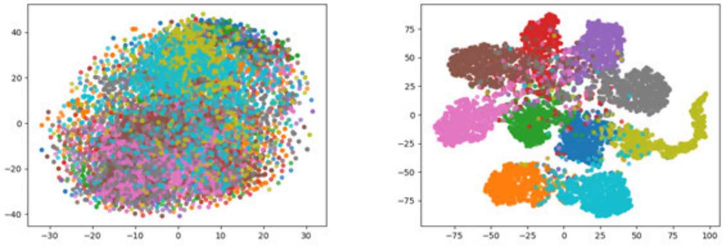


Figure 2.1: t-SNE [155] visualization for the first convolutional block (left) and the penultimate fully connected layer (right) of CIFAR-10 with VGG-16. Different colors refer to different classes.

out adversarial training, the VGG-16 network trained using our method reaches an accuracy of 35.86% against PGD examples, while using all layers reaches 25.61%.

2.2 Related Work

The IB is supposed to find the optimal trade-off between compression of input X and prediction of Y [152]. Empirically, IB provides both performance and adversarial robustness when embedded into the learning objective. Alemi et al. [150] proposed Variational Information Bottleneck (VIB). They use the internal representation of a certain intermediate layer as a stochastic encoding Z of the input data x. They aim to learn the most informative representation Z about the target Y, measured by the mutual information between their corresponding encoding values. Their experiments also showed that VIB is robust to overfitting and adversarial attacks. Ma et al. [153] proposed the HSIC Bottleneck and replaced mutual information with Hilbert Schmidt Independence Criterion (HSIC). They used the HSIC bottleneck as a learning objective for every layer of the network, an alternative to the conventional CE loss and back-propagation. Wang et al. [151] proposed HBaR, which combined HSIC Bottleneck of all hidden layers and back-propagation to improve both adversarial robustness and accuracy of clean data. Previous works tend to use one layer (such as VIB) or all layers (such as HSIC Bottleneck and HBaR) for variants of IB, but we find that the IB of each layer has a different impact on robustness. As such, using partial layers will increase the robustness against adversarial attacks, as discussed in Section 2.3.

2.3 Methodology

We first propose using IB as a regularizer for the learning objective, which also helps the network to learn better generalization of training data [152]. Specifically, we embed mutual information from intermediate representations to inputs X and targets Y. However, the outputs of convolutional layers contain independent feature channels. Then, we find that

IB-regularized feature channels generalize better than not using IB. Thus, we propose filtering out low correlation feature channels among well-generalized features according to their mutual information to their label. Note that the IB regularizer is the foundation of the filtering process. Figure 2.2 shows the structure of our method.

2.3.1 Threat Model

Adversary Goals. This chapter focuses on adversarial attacks on image classifiers. The adversary aims to create imperceptible perturbations for the input image, so the network misclassifies perturbed images. The experiments investigate untargeted individual adversarial attacks (the input can be misled to any label).

Adversary Knowledge. Our evaluation is conducted under white-box accessibility. The adversary has complete knowledge of the target network and its parameters. The adversary also has full access to the training data of the target network. The adversary knows the defense method, so IB-RAR is also evaluated by a specifically designed adaptive attack discussed in Appendix Section 2.6.2.

Adversary Capabilities. The adversary can perturb the input to well-trained networks when the perturbation is imperceptible. Following previous research, *imperceptibility* is defined as the distance from the perturbed image to its original copy. The distance is formalized as l_n-norm, i.e., $||\delta||_n \leq r$.

2.3.2 Mutual Information Loss

Problem Setup: We consider an L-layer neural network F_θ for classification in d_Y-dimensional space, and training dataset $D = \{(x_i, y_i)\}_{i=0}^{n-1}$ in d_X-dimensional space, where $x_i \in \mathbb{R}^{d_X}$ and $y_i \in \{0,1\}^{d_Y}$. The x_i refers to an example from training data, and we use X to indicate a batch of training data in this section. A network assigns to x_i one element in $\{0,1\}^{d_Y}$. The training aims to minimize the difference between predicted results and the ground truth, which is quantified by the standard CE loss: $\mathcal{L}_{CE}(\theta, F_\theta(x_i), y_i)$. T_l indicates the output of l_{th} layer to describe the intermediate representation of a network. The IB is embedded as a learning objective by the following loss function:

$$\underset{\theta}{min}\ \mathcal{L} = \mathcal{L}_{CE} + \alpha \sum_{l=1}^{L} I(X, T_l) - \beta \sum_{l=1}^{L} I(Y, T_l), \tag{2.1}$$

where α and β are two real numbers to control the trade-off between compression of input X and prediction of Y [152].

Specifically, the second term $\alpha \sum_l I(X, T_l)$ minimizes the relevance between inputs and intermediate features as the loss is supposed to decrease. Decreasing $I(X, T_l)$ compresses

Algorithm 2.1 Training with loss based on IB

```
Input: training data $D$, network $F_\theta$ with parameters $\theta$, batch size $m$, learning rate $a$, loss $\mathcal{L}_{CE}$, optimizer SGD
Output: optimized weights $\theta$
  while Maximum epoch not reached do
    Sample $X$, a batch of data from $D$
    Forward: Calculate $F_\theta(X)$ and $T = \{T_l | 0 \leq l < L\}$
    $T_{last} = T_{last} * mask$
    Calculate $\sum_{l=1}^{L} I(X, T_l)$ and $\sum_{l=1}^{L} I(Y, T_l)$
    Calculate loss as in Eq. (2.1)
    Backward: update $\theta$ by : $\theta \leftarrow \theta - a\nabla\mathcal{L}$
  end while
```

input to an efficient representation, which naturally removes noise and irrelevant information from X in T_l [152]. The term X refers to a batch of inputs at each iteration while training. The third term $\beta \sum_l I(Y, T_l)$ maximizes the relevance to the ground truth. The term Y refers to a batch of labels corresponding to X. All hidden layer outputs are embedded in the loss through summations. Because the compression measured with $I(X, T_l)$ also indicates a loss of useful information about Y (as this information is indiscriminate to all content in input X), $I(Y, T_l)$ becomes necessary to guarantee the accuracy on clean data. Algorithm 2.1 describes how to train a network with our proposed loss from Eq. (2.1). The *mask* and T_{last} are discussed in Section 2.3.3. As the computation of mutual information is difficult, we use HSIC [156] as an alternative plan for $I(\cdot)$.

Combination with adversarial training: In addition to training with clean data, the IB loss from Eq. (2.1) can also be easily combined with adversarial training as follows:

$$\begin{aligned} \mathcal{L}_{adv} &= \underset{\delta \in S}{max}\, \mathcal{L}_{CE}(\theta, F_\theta(X+\delta), X) \\ \underset{\theta}{min}\;\; \mathcal{L}_{adv} &+ \alpha \sum_{l=1}^{L} I(X, T_l) - \beta \sum_{l=1}^{L} I(Y, T_l). \end{aligned} \tag{2.2}$$

This way, it is easy to perform adversarial training by replacing the loss in Algorithm 2.1 with Eq. (2.2). Here, the adversarial perturbation is generated with the PGD algorithm, as it has been shown robust to various attacks [12].

Selection of Robust Layers: While other IB-related methods use one layer (such as VIB [150]) or all layers (such as HSIC Bottleneck [153] and HBaR [151]), we find that deploying IB to different hidden layers will provide different robustness (see Table 2.3). Using a part of hidden layers for IB can get higher adversarial robustness than all layers or a single layer. We refer to this part of the layers as robust layers. To distinguish robust layers from others, we apply IB to each hidden layer and train an independent network

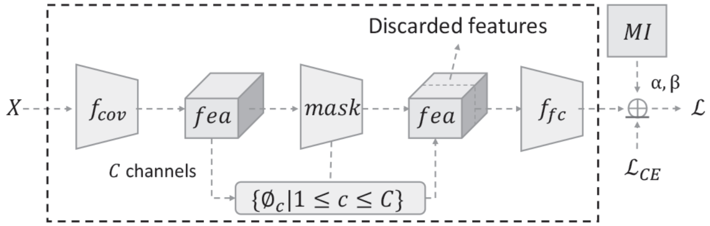


Figure 2.2: The structure of IB-RAR. First, we use IB as a regularization combined with CE. Then, feature channels extracted from the convolutional layer are filtered by ϕ_c.

for each layer. If a network shows higher accuracy on adversarial examples compared to the network trained with CE only, the corresponding layer is considered a robust layer.

Table 2.1: Top-1 Natural accuracy (in %) on clean examples and adversarial accuracy (in %) on adversarial examples. The adversarial examples are generated with PGD, CW, FGSM, FAB, and NIFGSM on CIFAR-10 and Tiny ImageNet. In the methods part, PGD, TRADES, and MART are benchmark methods. The method "(IB-RAR)" refers to benchmarks combined with our method. Each result is the average of three runs.

	CIFAR-10 with VGG-16					
Inputs Methods	Natural	PGD	CW	FGSM	FAB	NIFGSM
PGD	75.02	42.45	37.80	47.32	41.03	47.59
PGD (IB-RAR)	**76.22**	**45.09**	**41.83**	**50.53**	**46.22**	**51.93**
TRADES	73.44	43.92	38.28	47.94	41.64	48.41
TRADES (IB-RAR)	**80.63**	**44.13**	**41.81**	**51.45**	**43.63**	**51.69**
MART	73.52	44.64	37.58	48.73	40.56	48.95
MART (IB-RAR)	**80.54**	44.34	**41.45**	**52.19**	**44.72**	**51.93**
	Tiny ImageNet with VGG-16					
PGD	37.54	17.73	13.77	19.46	13.76	22.14
PGD (IB-RAR)	**40.25**	**18.30**	**14.08**	**20.07**	**14.29**	**22.62**
TRADES	36.80	18.13	13.73	19.57	14.01	22.16
TRADES (IB-RAR)	**39.10**	**18.45**	**14.19**	**20.22**	**14.49**	**22.87**
MART	34.94	17.49	13.06	18.88	13.68	21.23
MART (IB-RAR)	**36.68**	**18.05**	**13.36**	**19.33**	**13.81**	**22.02**

2.3.3 Removing Unnecessary Features

After training the network with MI loss from Eq. (2.1), the network is supposed to learn a more informative generalization of training data. The output of convolutional layers will have a closer connection to their targets, quantified by MI in this chapter. Specifically, feature channels containing more noise will be irrelevant to their label. The network provides the most informative representations after extracting all convolutional layers. Otherwise, those convolutional layers are not necessary for the network structure. Therefore, we evaluate feature channels given by the last convolutional layer according to their MI to the target Y. Still, there is another reason for choosing the last convolutional layer. According to results in [157], distilled non-robust features from the last convolutional layer significantly decrease the accuracy on clean and adversarial examples. It means that non-robust

Table 2.2: Top-1 Natural accuracy (in %) on clean examples and adversarial accuracy (in %) on adversarial examples. The adversarial examples are generated with PGD, CW, FGSM, FAB, and NIFGSM on CIFAR-10 and CIFAR-100. In the methods part, PGD, TRADES, and MART are benchmark methods. The "method (IB-RAR)" refers to benchmarks combined with our method. Each result is the average of three runs.

	CIFAR-10 with ResNet-18					
Inputs Methods	Natural	PGD	CW	FGSM	FAB	NIFGSM
PGD	75.05	45.21	74.09	48.60	42.26	49.71
PGD (IB-RAR)	**75.10**	**45.55**	**74.10**	**48.83**	**42.74**	**50.03**
TRADES	73.04	45.91	72.16	48.51	42.59	49.92
TRADES (IB-RAR)	**73.07**	**46.13**	72.16	**48.85**	**42.74**	**50.09**
MART	72.96	46.17	72.00	49.19	41.62	50.34
MART (IB-RAR)	**76.85**	**48.92**	**75.78**	**52.52**	**45.01**	**54.72**
	CIFAR-100 with WRN-28-10					
PGD	39.88	9.74	13.66	16.85	10.28	14.53
PGD (IB-RAR)	37.68	**16.60**	**15.98**	**19.44**	**14.85**	**19.48**
TRADES	39.38	10.44	14.69	17.60	10.42	15.38
TRADES (IB-RAR)	36.41	**19.18**	**16.67**	**20.69**	**16.61**	**21.95**
MART	39.91	12.30	14.29	17.85	11.73	16.57
MART (IB-RAR)	**40.65**	**23.44**	**17.96**	**24.46**	**19.24**	**26.41**

(unnecessary) features have a large negative effect on the last convolutional layer, which should be discarded.

Structurally, a network F_θ trained with our MI loss is given by the concatenation of L hidden layers outputs F_{θ_l}. The network's output is the output of the last layer in the network, i.e., F_{θ_L}. Each layer uses the output of the previous layer as input. We consider F_θ has C kernels to extract multiple feature channels at the last convolution layer:

$$F_{\theta_{last}}(x) = T_{last} = \{f_c | 1 \leq c \leq C\}.$$

Following, we compute a mask based on the MI values of each feature channel:

$$\begin{aligned} T_{last} &= T_{last} * mask \\ mask &= \{\phi_c | 1 \leq c \leq C\} \\ \phi_c &= \begin{cases} 1, \; I(f_c, Y) \geq thr \\ 0, \; otherwise. \end{cases} \end{aligned} \tag{2.3}$$

Then the mask is used to filter the feature channels. Channels with MI less than the threshold *thr* are removed. The threshold is decided according to the sorted MI values of these feature channels. Empirically, we use a small threshold to eliminate 5% of all feature channels. In other words, the MI values of that 5% of feature channels are lower than the MI values of all other channels. The threshold is the maximum of MI values of that 5% of feature channels. The application of the mask is shown in Algorithm 2.1. Note that removing unnecessary features is built on our MI loss, as it requires non-robust features to be more distinct from other features concerning MI values. Experimental evidence can be found in the ablation study, row (5) of Table 2.5.

2.4 Experimental Evaluation

Following prior literature, experiments are conducted with four standard datasets: CIFAR-10 [158], SVHN [159], CIFAR-100 [158], and Tiny ImageNet [160]. We use VGG-16 [26] for CIFAR-10, SVHN, and Tiny ImageNet. We use ResNet-18 [27] for CIFAR-10, SVHN. We use WideResNet-28-10 [161] for CIFAR-100. The implementation is built with PyTorch and Torchattacks [162] frameworks.

Algorithms: We evaluate our method with the following adversarial learning algorithms: Projected Gradient Descent (PGD) [12], TRADES [94], and MART [154]. Clean examples are not used for PGD adversarial training but are used in TRADES and MART for evaluation following previous works. We combine our method with these algorithms and compare them against the performance of the original algorithms. In addition, we also compare our method to non-adversarial training algorithms: Cross-Entropy, HSIC Bottleneck as Regularizer (HBaR) [151], and Variational Information Bottleneck (VIB) [150].

Metrics: We evaluate accuracy on natural inputs (Test Acc., i.e., accuracy on clean data) and adversarial examples (Adv. Acc.) for all algorithms. The adversarial examples are generated by: (1) PGD$_n$ [12], the PGD attack with n steps in optimization; (2) FGSM [59]; (3) CW [60]; (4) FAB [163]; (5) NIFGSM [164]. We set parameters for attacks (implemented with Torchattacks) following the prior literature: step size = 2/255 (alpha), r = 8/255 (eps, the limitation for perturbation δ), default steps= 10, and CW steps = 200. We use the following hyperparameters for all training:

- StepLR: lr = 0.01, step_size=20, gamma=0.2.
- Optimizer: SGD, weight_decay=1e-2.
- Maximum epoch: 60.
- Batch size: 100.

Adaptive Evaluation. To demonstrate the effectiveness of IB-RAR as a defense, we provide two levels of adaptive evaluation: (1) To demonstrate that the success of IB-RAR is not limited to a few cases and that the attack algorithms converged, we use multiple attacks and iteration steps. The results of adversarial robustness are shown in

Table 2.1, Table 2.2, and Figure 2.3. (2) We assume that the adversary designs a new attack specifically targeted to IB-RAR, as the adversary has full knowledge of IB-RAR and white-box access to the network, which is discussed in Appendix Section 2.6.2.

2.4.1 Adversarial Robustness Results with Adversarial Training

We show that our method reaches better adversarial robustness along with state-of-the-art adversarial training benchmarks. Different regularizers (α and β in $\mathcal{L}$) are evaluated to find the optimal hyperparameters. We also find that our method can boost the convergence of the network.

Accuracy on Adversarial Examples. Tables 2.1 and 2.2 show test accuracy and adversarial accuracy results on CIFAR-10, CIFAR-100, and Tiny ImageNet, respectively. PGD refers to adversarial training with PGD examples. Results for SVHN are in Appendix due to space limitation. TRADES and MART are baseline methods mentioned in the experimental setting. The "method (IB-RAR)" refers to using our method to improve the baseline method, i.e., using the mutual information loss in Eq. (2.2) and using the mask in Eq. (2.3) to remove unnecessary feature channels.

Combined with all benchmark methods, our method improves adversarial robustness compared to baselines. In Table 2.1, our method also improves the test accuracy on clean examples, especially for TRADES and MART. Note that we use clean examples to compute MI in Eq. (2.2). Suppose we use adversarial examples to compute MI, i.e., using $I(X+\delta, T_l)$ to build the IB objective in the loss. In that case, the performance increases when defending against the PGD attack (or keeping almost the same performance) but decreases the performance against other attacks.

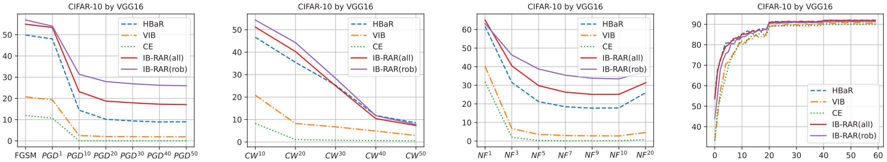


Figure 2.3: CIFAR-10 with VGG-16: comparison of our method and IB-based baselines. The performance is evaluated under different optimization steps of (a) PGD attacks, (b) CW attacks, (c) NIFGSM attacks, and (d) clean data. The IB-RAR(rob) refers to our method, which uses IB regularizers for only robust hidden layers. The IB-RAR(all) refers to using IB regularizers for all hidden layers. The accuracy on clean data at the last epoch is IB-RAR(rob) 91.33%, IB-RAR(all) 91.97%, HBaR 91.93%, VIB 90.52%, CE only 89.88%. Each result is the average of three runs.

2.4.2 Robustness Without Adversarial Training

Using Partial Layers is Better We empirically show that using partial layers to compute IB loss is better, as shown by results provided in Table 2.3. We deploy MI loss (Eq. (2.1)) to every layer of VGG-16 and use CIFAR-10 for training, as there are both convolutional and fully connected layers in the VGG structure. Each row in Table 2.3 shows the performance of a network, which is trained by computing IB loss (Eq. (2.1)) for a single block of VGG-16.

Clearly, the robustness mainly comes from Conv Block 5 (the fifth convolutional block in VGG-16), FullyC 1 (the first fully connected layer in VGG-16), and FullyC 2. We refer to Conv Block 5, FullyC 1, and FullyC 2 as robust layers, as they provide obvious robustness compared to other layers. Using all layers to build the two regularizers for MI loss degrades its robustness compared to using robust layers. Based on MI loss, we further improve its robustness on other convolutional blocks by filtering out unnecessary feature channels, see row (2) and row (6) in Table 2.5 for comparison. Compared to using all layers or other single layers, our method provides the best accuracy (35.86%) under the PGD$_{10}$ (10 iteration steps for the PGD algorithm) attack. This is achieved by the defender without any prior knowledge of adversarial examples.

This phenomenon also occurs in other networks trained with other datasets, i.e., every single hidden layer can provide a different degree of adversarial robustness when computing the MI of the intermediate representation for the IB objective. Their behaviors are similar but not the same. For example, when VGG-16 is trained with SVHN, the last four layers provide adversarial accuracy, i.e., Conv Block 4 (6.44%), Conv Block 5 (16.83%), FullyC 1 (8.97%), and FullyC 2 (9.98%). When training ResNet-18 with SVHN, the last layer provides higher adversarial accuracy (6.13%) than other layers. The accuracy of trained VGG-16 and ResNet-18 with CE only and the SVHN dataset is lower than 1%. Based on our observations, the robust layers will be the last few layers of the network.

Comparison with Other IB-related Methods Figure 2.3 shows that our method achieves the best robustness compared to other IB-based baselines when training without adversarial examples. Specifically, we compare our method with CE, HBaR [151], and VIB [150] under the same conditions. CE refers to training with only cross-entropy loss function, i.e., no defense on this baseline.

We obtain improved accuracy on clean data compared to VIB and CE only. Our method achieves the natural accuracy for IB-RAR(rob) of 91.33%, IB-RAR(all) of 91.97%, while HBaR, VIB, and CE only achieve 91.93%, 90.52%, and 89.88%, respectively.

We use a progressively increasing number of steps to make sure the attacks are converged. Under PGD, CW, and NIFGSM attacks, our method continuously shows better accuracy

Layer	Adv. acc.	Test acc.
Conv Block 1	0.04	89.32
Conv Block 2	0.05	90.17
Conv Block 3	0.02	90.53
Conv Block 4	0.01	89.66
Conv Block 5	8.25	89.58
FullyC 1	9.85	91.04
FullyC 2	3.27	90.97
All Layers	25.61	91.96
Rob. Layers	35.86	90.97

Table 2.3: The Adv. Acc. and Test Acc. of using a single layer to compute MI in Eq. (2.1) for our method. The Adv. Acc. is evaluated under our default PGD attack. The network is VGG-16 trained with CIFAR-10. Rob (robust) Layers refers to using outputs of Block 5, FullyC 1, and FullyC 2.

compared to baselines.

2.4.3 Discussion and Future Work

One possible explanation for why our method works is that there are shared features among different classes of training data. Shared features refer to the similar characteristics of objects in two classes of data. For example, cats and dogs are very similar, while cats and airplanes are not so similar in terms of shape. Clusters in Figure 2.4(a) also show the similarity between classes in terms of distance. The MI loss and mask reduce that shared feature and increase the distance among classes in Figure 2.4.

In addition, to check whether similar classes tend to be classified as each other, we evaluate the number of times the network predicted the adversarial example as a specific class (top-4 classes) The test set of CIFAR-10 is used to generate the adversarial examples, containing 1000 images for each class. In Table 2.4, cars are thought to be the truck 681 times. The highest number of classifications of the truck class is also the car class, i.e., 427 times. Such a bidirectional tendency also exists in other classes. The network learned the most often shared features from these pairs compared to other classes. It is easier for the adversary to find imperceptible perturbations of similar pairs, as their distance in classification should be close to each other.

This intuitive idea can be a starting point for future work that could investigate, for instance, the following aspect. Currently, our method builds the IB objective by using inputs, outputs, and intermediate network representations. It is not specifically designed for adversarial perturbation or shared features. The straightforward next step is distilling shared features for every class, since the shared features could help adversarial attack algorithms find small enough perturbations. Then, according to distilled features, the network can learn well-generalized features but discard shared features. As discarding shared features may also result in the loss of useful information for the class, a key problem might be controlling the trade-off between discarding shared features and retaining enough information for generalization.

2.5 Conclusions

This chapter proposes an improved IB-based loss function to improve adversarial robustness and a feature channel mask to remove unnecessary features. We first use IB as a regularizer to improve robustness on fully connected layers and learn better generalization of input data. Based on a well-generalized network, we remove less relevant feature channels of convolutional layers according to the MI between these channels and the true labels. We also discuss that using partial layers for MI loss improves robustness against adversarial attacks. Our experimental results show that our method consistently improves the adversarial robustness of state-of-the-art adversarial training technologies. The IB-RAR method improves accuracy by an average of 2.66% against five adversarial attacks for all networks in Tables 2.1 and 2.2, compared to three adversarial training benchmarks. IB-RAR can also provide modest robustness to weaker methods without prior knowledge of adversarial examples. Our findings also show that our method increases the accuracy of clean data, as the noise in input data is removed. Finally, our experimental evaluation results demonstrate that our method can enhance the robustness against various adversarial attacks.

Target class	**Predicted results**			
plane :	bird-352	ship-247	deer-156	truck-110
car :	truck-681	ship-166	plane-55	frog-24
bird :	deer-260	frog-259	dog-141	plane-120
cat :	dog-415	deer-173	bird-144	frog-134
deer :	bird-285	frog-196	cat-169	horse-147
dog :	cat-299	frog-208	bird-169	horse-143
frog :	cat-411	bird-240	deer-187	dog-63
horse :	dog-335	deer-335	truck-82	bird-75
ship :	plane-280	bird-196	truck-181	deer-116
truck :	car-427	ship-192	horse-135	plane-101

Table 2.4: The adversarial example classification tendency table of CIFAR-10 trained with VGG-16. The target class column is the ground truth. The rest of each row is the prediction results and the class count. Class count refers to the number of times an input of the target class was classified as that class.

Inputs Methods	CIFAR-10 with VGG-16				CIFAR-10 with ResNet-18			
	Natural	PGD	NIFGSM	FGSM	Natural	PGD	NIFGSM	FGSM
(1) $\mathcal{L}_{CE}$	89.99	0.10	0.18	11.80	92.19	0.00	0.00	5.22
(2) $\mathcal{L}$	92.03	12.39	13.90	43.49	93.32	3.85	4.71	40.46
(3) $\mathcal{L}_{CE} + \alpha \sum_{l=1}^{L} I(X, T_l)$	41.69	0.16	0.20	9.98	10.00	10.00	10.00	10.00
(4) $\mathcal{L}_{CE} - \beta \sum_{l=1}^{L} I(Y, T_l)$	91.50	0.06	0.99	31.66	92.75	0.00	0.00	8.90
(5) $\mathcal{L}_{CE} + FC$	89.41	0.16	0.14	12.89	92.41	0.00	0.01	4.26
(6) $\mathcal{L} + FC$ (IB-RAR)	91.50	35.86	37.44	55.92	93.13	5.37	6.09	39.34

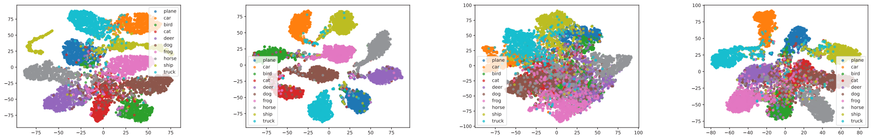
Table 2.5: Ablation study for our method. We remove a part of our method one by one in each row. We evaluate the their natural test accuracy (in%) and adversarial robustness (in%) against PGD_{10}, NIFGSM_{10}, and FGSM.

2.6 Appendix

2.6.1 Ablation Study

We conduct an ablation study to verify the effectiveness of the proposed mutual information loss and the mask to remove the unnecessary feature channels. Here, we refer to them as $\mathcal{L}$ and FC. The results are shown in Table 2.5. The network in a row (1) of Table 2.5 is trained with CE loss function only. It gets almost zero accuracies on PGD and CW attacks and very low accuracy on FGSM, as training with CE only does is vulnerable. The network trained with mutual information loss, row (2) of $\mathcal{L}$, gains modest accuracy under adversarial attack, but it is lower than our method, i.e., the last row $((L) + FC)$. We also evaluate the regularizer terms ($I(X, T_l)$ and $I(Y, T_l)$) in mutual information loss. Removing the enhancement of $I(Y, T_l)$, row (3), dramatically degrades the accuracy of clean data because decreasing only $I(X, T_l)$ removes both useful information and noise in inputs. Removing the penalty of $I(X, T_l)$, row (4), gets a good network and slightly higher accuracy on clean data and adversarial examples compared to training with only CE loss in row (1). The reason is that increasing $I(Y, T_l)$ will increase the relevance between intermediate results (outputs of hidden layers) and their labels. The $\mathcal{L}_{CE} + FC$ in a row (5) does not improve the robustness because the network trained with CE loss only does not learn a well-generalized representation in the sense of mutual information.

To further support our experiments, we illustrate the correlation between the features generated with our method and baselines by using a 2D t-SNE plot [155]. In the case of clean examples as shown in Figures 2.4(a) and 2.4(b), which are trained with CE loss only (row (1) in Table 2.5) and our method (row (6) in Table 2.5). The accuracy (PGD) of the network in Figure 2.4(b) only increases by 2.04% (see specific values in Table 2.5) compared to the network in Figure 2.4(a). However, it sustains better-clustered results, and a more obvious distance is maintained between clusters because the two regularizers in Eq. (2.1) remove noise in input and unnecessary features in the output of hidden layers. Figure 2.4(c)



(a) Plain (b) IB-RAR (c) TRADES (d) TRADES(IB-RAR)

Figure 2.4: The results of t-SNE [155] for CIFAR-10 with VGG-16. Each cluster indicates a 2-dimensional feature representation. The feature representation is extracted from the VGG-16 network.

Method	PGD-AD$_{10}$	PGD$_{10}$	PGD-AD$_{100}$	PGD$_{100}$
plain (IB-RAR)	15.38	35.86	22.64	31.37
AT	45.06	42.26	44.71	42.01
AT (IB-RAR)	45.97	45.03	45.60	44.60

Table 2.6: Results of IB-RAR against adaptive white-box attacks (10 steps or 100 steps) on CIFAR-10 with VGG-16. PGD-AD refers to the adaptive attack.

shows interacted clusters. Figure 2.4(d) for our method shows better-clustered results compared to clusters in Figure 2.4(c).

2.6.2 Adaptive Attack Evaluation

Since IB-RAR provides a new loss function, we examine customized attacks where the adversary takes advantage of the knowledge of IB-RAR. As the learning objective of IB-RAR is to minimize $+\alpha \sum_{l=1}^{L} I(X, T_l) - \beta \sum_{l=1}^{L} I(Y, T_l)$ along with CE loss in Eq. (2.1) and Eq. (2.2), a natural idea is using PGD to maximize it. We propose a white-box attack: the adversary uses $\mathcal{L}$, i.e., Eq. (2.1), as a loss function to implement the PGD algorithm. To ensure that the attack converges, we use 10 and 100 steps for PGD attacks. Table 2.6 shows the attack results. The adaptive attack is effective because it decreases the accuracy of the network without adversarial training, i.e., plain (IB-RAR), but the network still retains better robustness compared to training with CE only. When combined with adversarial training, the adaptive attack does not decrease the accuracy because its robustness comes from adversarial training and IB-RAR. If the adversary attack is specifically designed to target IB-RAR, it will weaken the attack performance of attacking adversarial training. On the contrary, an adaptive attack targeting adversarial training will weaken the ability to attack IB-RAR. The results demonstrate that IB-RAR is robust to the adaptive white-box attack.

Chapter 3

Information Bottleneck in Adversarial Pre-training

Vision Transformers (ViTs) have emerged as a fundamental architecture and serve as the backbone of modern vision-language models. Despite their impressive performance, ViTs exhibit notable vulnerability to evasion attacks, necessitating the development of specialized Adversarial Training (AT) strategies tailored to their unique architecture. While a direct solution might involve applying existing AT methods to ViTs, our analysis reveals significant incompatibilities, particularly with state-of-the-art (SOTA) approaches such as Generalist [165] (CVPR 2023) and DBAT [142] (USENIX Security 2024). This chapter presents a systematic investigation of adversarial robustness in ViTs and provides a novel theoretical Mutual Information (MI) analysis in its autoencoder-based self-supervised pre-training. Specifically, we show that MI between the adversarial example and its latent representation in ViT-based autoencoders should be constrained via derived MI bounds. Building on this insight, we propose a self-supervised AT method, MIMIR, that employs an MI penalty to facilitate adversarial pre-training by masked image modeling with autoencoders. Extensive experiments on CIFAR-10, Tiny-ImageNet, and ImageNet-1K show that MIMIR can consistently provide improved natural and robust accuracy, where MIMIR outperforms SOTA AT results on ImageNet-1K. Notably, MIMIR demonstrates superior robustness against unforeseen attacks and common corruption data and can also withstand adaptive attacks where the adversary possesses full knowledge of the defense mechanism. Our code and trained models are publicly available: `https://github.com/xiaoyunxxy/MIMIR`.

3.1 Introduction

ViTs [32] and their variants [33, 166] have achieved substantial progress and serve as foundational components in modern vision-language models. Prominent multimodal frameworks, including CLIP [167], BLIP [168], and Mini-GPT4 [169], typically employ ViTs as their image encoders. However, similar to convolutional neural networks (CNNs), attention-based models provide limited robustness against evasion attacks [170, 171, 172, 173, 174]. Evasion attacks [5, 7] (also known as adversarial attacks), where well-trained deep models are fooled by introducing human-imperceptible perturbations to inputs, remain a persistent challenge in deep learning security. In 2024, the National Institute of Standards and Technology (NIST) explicitly listed adversarial attacks as a significant threat to AI systems, and pointed out the importance of conducting robustness testing and mitigation, such as adversarial training and formal verification, when deploying AI tools [175]. Nevertheless, improving adversarial robustness remains a difficult task. SOTA methods, such as [140, 176, 177], achieve only marginal robustness gains, commonly below 2% compared with those before them.

So far, Adversarial Training (AT) is widely recognized as the most practically effective defense [173, 172, 141] against evasion attacks. AT operates by augmenting the training dataset with adversarially perturbed samples [12], yet introduces two key limitations: (1) substantial computational overhead due to the generation of adversarial examples during training [12], and (2) a potential degradation in natural accuracy [178]. Numerous methods have been proposed to mitigate these challenges. Techniques such as FreeAT [99] optimize efficiency by reusing gradient information during adversarial example generation, while FastAT [100] employs an improved Fast Gradient Sign Method (FGSM) to accelerate training. TRADES [94], SCORE [143], Generalist [165], and DBAT [142] explore how to achieve the best trade-off between natural and robust accuracy. Additionally, pre-training strategies have also been leveraged to enhance the performance of AT [179, 140].

Applying existing AT methods to ViTs presents unique challenges due to the fundamental differences between attention-based architectures and CNNs. Unlike CNNs, ViTs lack inductive biases [32], including locality, two-dimensional neighborhood structure, and translation equivariance. These biases are inherent to CNNs as prior knowledge, enabling efficient learning with limited data, whereas ViTs typically require larger training datasets to achieve comparable generalization performance [32]. Consequently, AT for ViTs entails significantly higher computational costs. Initial research on AT for ViTs explored their unique attention mechanism. For instance, robustness can be improved by randomly dropping gradients according to attention [172] or improving training efficiency by dropping low-attention image embeddings [173]. The majority of recent works have focused on adapting CNN-based AT methodologies to ViTs, given AT's success in building robust CNNs. Unfortunately, standard CNN AT techniques are not fully transferable to ViTs. Empirical studies [170, 141] reveal that *strong data augmentations* (such as Randaugment [180], CutMix [181], and MixUp [182], which improve robustness in CNNs) often degrade AT performance for ViTs. To mitigate this, recent work [170] suggests progressively increasing augmentation intensity (e.g., distortion magnitudes in RandAugment or the sampling probability of MixUp/CutMix) during training. Furthermore, SOTA AT strategies, such as Generalist [165] and DBAT [142], are less effective when applied to ViTs (see Table 3.1), and there is a lack of evaluation on large datasets, such as ImageNet-1K, which further limits their generalizability.

Pre-training has emerged as a complementary approach to ViT AT, with studies showing that adversarial fine-tuning of naturally pre-trained models can enhance robustness [172, 140]. AdvXL [183] notably advanced this paradigm by developing efficient AT for web-scale datasets. However, the mechanisms underlying pre-training's effectiveness are not fully understood, and results are inconsistent across implementations. For instance, models pre-trained on ImageNet-21K using SimMIM [184] demonstrate comparable performance to scratch-trained counterparts, while CLIP [167] pre-training has been shown to degrade performance in some configurations [185].

While previous methods of ViT AT, such as [140, 141, 170], focus on searching for better combinations of training hyperparameters, they suffer from performance drops across different architectures and datasets. In contrast, we aim for a generalizable method via pre-training. Specifically, this work presents a systematic investigation of self-supervised pre-training for ViT robustness through the lens of Mutual Information (MI) and Information Bottleneck (IB) theory. IB introduces a joint objective of simultaneously minimizing the MI between inputs and latent features while maximizing the MI between labels and latent features to mitigate the impact of the adversarial noise in the inputs. Regarding the ViT AT, we develop a novel theoretical justification for self-supervised autoencoders, demonstrating that reducing MI between inputs and latent features enhances ViT robustness. Based on this finding, we propose a theoretically grounded adversarial pre-training method, **M**asked **I**mage **M**odeling for Mutual **I**nformation-based Adversarial **R**obustness (**MIMIR**[*]). Specifically, we convert Masked Image Modeling (MIM) into an effective and efficient adversarial pre-training method. The basic idea is to predict the masked content of inputs, which is a self-supervised learning task. The effectiveness of MIMIR is analyzed and guaranteed by our theoretical justification. The efficiency comes from discarding the masked content (75% of image patches are discarded in our experiments), which greatly reduces the computing requirements. Figure 6.1 provides an illustrative diagram of MIMIR.

We validate MIMIR's effectiveness through extensive experiments on CIFAR-10 [158], Tiny-ImageNet [186], and ImageNet-1K [187], showing consistent improvements in both natural and adversarial accuracy. In addition, we test the generalizability of MIMIR by combining MIMIR with three MIM methods for three representative architectures, including MAE [43] for ViTs, Group Window Attention [188] for hierarchical transformer (Swin [33]), and SparK [189] for CNN (ConvNext [34]), where MIMIR outperforms SOTA AT methods on ImageNet-1K.

[*] Mimir is a figure in Norse mythology, renowned for his knowledge and wisdom.

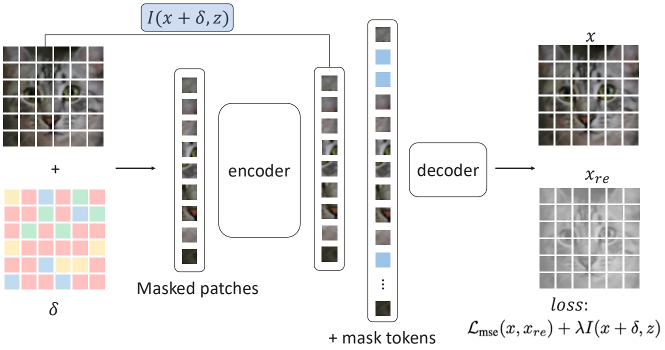


Figure 3.1: Diagram illustrating the working mechanism of MIMIR. In the pre-training, adversarial perturbations δ are generated and added to natural images x to create adversarial images $x+\delta$, as shown on the left. Patches of generated adversarial images are fed as training data to the ViT-based autoencoder, where the output of the decoder x_{re} and the natural input image x are used to calculate the loss. In particular, we propose using MI ($I(x+\delta,z)$) as an additional penalty to compose the pre-training loss of MIMIR, as shown in the bottom right. After pre-training, a trained encoder is combined with a randomly initialized classification layer as the final model that is further fine-tuned for classification tasks.

Our main contributions are:

- By revising the current ViT AT strategies, we point out that ViT adversarial pre-training methods compromise natural accuracy and lack a systematic study. To this end, we provide a theoretical analysis using adversarial examples and the Mutual Information penalty. The theoretically grounded MI bounds motivate us to decrease the MI between adversarial examples and the learned latent representation.
- Based on the findings of our analysis, we propose a self-supervised defense – MIMIR against adversarial attacks on ViTs. We evaluate MIMIR using multiple architectures on three datasets under various adversarial attacks, demonstrating its effectiveness. We also provide results with the latest CNN architecture, ConvNext [34], showcasing the performance of MIMIR even on architectures different from ViTs.
- We show MIMIR is resistant to two adaptive attacks where the adversary is aware of MIMIR's design. We build a PGD-fea attack that increases the distance between features extracted from natural and adversarial examples. Moreover, we build a PGD-MI attack that includes the MI penalty as the learning objective to find adversarial examples.

3.2 Related Work

3.2.1 Masked Image Modeling - MIM

MIM refers to a self-supervised pre-training framework that aims to reconstruct pre-defined targets, such as discrete tokens [190], raw RGB pixels [43, 191], or features [192]. The final goal is to use the pre-trained model as a starting point for downstream fine-tuning. The downstream tasks include, for instance, classification and object detection. More specifically, to build a high-performance ViT f_e without a classification layer, we consider f_e as an encoder to extract discriminative input features. Then, we design a lightweight decoder f_d, which uses the output of f_e as its input. The goal of f_d is to reconstruct the original inputs (let us consider MAE [43] as an example). The aim is to decrease the distance between the input x and $x_{re} = f_d \circ f_e(x)$. After the encoder f_e and decoder f_d are trained, we use f_e plus a manually initialized classification layer as the starting point of fine-tuning.

3.2.2 Mutual Information - MI

MI measures the mutual dependence between two random variables, X and Y [193, 194]. It quantifies the amount of information contained in one random variable about another random variable or the reduced uncertainty of a random variable when another random variable is known. It can be written as:

$$I(X,Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_{(X)}(x)P_{(Y)}(y)} \right), \tag{3.1}$$

where $P_{(X,Y)}$ is the joint probability density function of X and Y. $P_{(X)}$ and $P_{(Y)}$ are the marginal probability density function of X and Y. MI can be equivalently expressed as:

$$I(X,Y) = H(X) - H(X|Y). \tag{3.2}$$

Estimating the exact MI is not easy, as it is difficult to precisely estimate $P_{X,Y}$ or P_X and P_Y in high-dimensional space [195, 196]. In practice, DIB [197] suggested using the matrix-based Renyi's α-order entropy I_α [198, 199] to estimate MI, which avoids density estimation and variational approximation. An alternative way is the Hilbert-Schmidt Independence Criterion (HSIC) [156], which is a kernel-based dependence measure defined in a reproducing kernel Hilbert space (RKHS) and usually used as a surrogate of MI. Details about definitions and empirical estimators of I_α and HSIC are provided in Appendix 3.7.7. In this chapter, we evaluate both I_α and HSIC as our MI measurements.

3.2.3 Information Bottleneck - IB

The IB concepts were first proposed in [200] and further developed for deep learning in [201, 152]. It has been widely used to improve adversarial robustness [202, 150, 197, 151]. IB describes the generalization of a deep network in two phases: 1) empirical error minimization (ERM) and 2) representation compression [152]. For a network with input x and label y, there is an intermediate representation t_l for each layer l, i.e., the output of l-th layer. The IB principle aims to keep more relevant information in t_l about target y while decreasing the irrelevant information about input x. The information between intermediate representation t_l and input x or label y is quantified by MI, denoted by $I(\cdot)$. During neural network training, in the ERM phase, the model increases shared information between t_l with respect to both x and y. Afterward, in the compression phase, the model decreases information contained in t_l about x but preserves (or even increases) information about y. The reduction of $I(x, t_l)$ can be interpreted as a way of reducing noise or compressing irrelevant or redundant features in x. At the end of the training, the model strikes a trade-off that maximizes $I(y, t_l)$ and minimizes $I(x, t_l)$. Formally, the IB minimizes the following Lagrangian:

$$\mathcal{L} = I(x, t_l) - \beta I(y, t_l), \tag{3.3}$$

where β is a Lagrange multiplier that controls the trade-off between predicting y and compressing x.

3.2.4 Vision Transformer

The transformers [35] were first proposed in natural language processing (NLP). With the mechanism of global self-attention, transformers can effectively capture the non-local relationships among all text tokens [203, 204, 205]. A substantial effort is done to apply the transformer and self-attention mechanism in computer vision [32, 206, 166]. The pioneering work, ViT [32], demonstrated that the pure transformer architecture can achieve competitive performance on various tasks. ViT also reveals that transformers lack inductive biases [32]. For example, locality, two-dimensional neighborhood structure, and translation equivariance are inherent to CNNs but not applicable to ViTs [32]. Due to this shortcoming, ViTs usually require large-scale training to get competitive performance, such as pre-training on ImageNet-21K [187] and JFT-300M [207]. To alleviate the ViT's need for large datasets, DeiT [208] introduced a teacher-student strategy to distill knowledge from a teacher CNN for a student ViT. In the MIM field, MAE [43] uses a masked autoencoder with a lightweight decoder as a visual representation learner. Its learning objective is to reconstruct the original image by the decoder while using masked images as input to the autoencoder. The advantage is that MAE can randomly discard 75% image patches when pre-training under ImageNet-1K [187], which means more efficient training.

3.2.5 Adversarial Attacks on ViTs

The concept of adversarial attacks first appeared in [6], which proposed a formal framework and algorithms against the adversarial spam detection domain. Then, the adversarial attacks were popularized by Biggio et al. [5] and Szegedy et al. [7] in image classification. The generation of adversarial examples depends on the model's gradient or estimated gradient in the black box situation [209]. Therefore, adversarial attacks can be easily applied to transformers by using the gradient of attention blocks with respect to the inputs. This also raises the question of whether transformers are more robust than CNNs. Benz et al. [210] found that CNNs are less robust than ViTs due to their shift-invariant property. Bhojanapalli et al. [211] found that ResNet models are more robust than transformers at the same model size under FGSM attack, but under PGD [12] attack, transformer models show better robustness. As ViTs process the input image as a sequence of patches, Gu et al. [174] found that ViTs are more robust than CNNs to naturally corrupted patches because the attention mechanism is helpful in ignoring naturally corrupted image patches. The later work [170] revealed that CNNs could be as robust as ViTs against adversarial attacks if CNNs are trained with proper hyperparameters.

3.2.6 Adversarial Defense

PGD [12] adversarial training is considered as one of the most effective defenses for CNNs and can withstand adaptively designed attacks [91]. However, PGD AT is harmful to the accuracy of clean data. Generalist [165] solves this problem by formulating different training strategies for robust and natural generalization separately. DBAT [142] solves the decrease in natural accuracy by adding dummy classes [212] to the classification space.

Due to the difference between CNNs and ViTs, there have been some recent efforts to explore new adversarial training approaches for ViTs [172, 141, 173]. Mo et al. [172] presented a new adversarial training strategy based on the following observations: 1) pre-training with natural data can provide better robustness after adversarial fine-tuning, 2) gradient clipping is necessary for adversarial training, and 3) using SGD as the optimizer is better than Adam. Debenedetti et al. [141] also presented an improved training strategy for ViTs by evaluating different combinations of data augmentation policies. They found that weak data augmentation and large weight decay are better than previous canonical training approaches. As adversarial training is time-consuming, AGAT [173] leverages the attention score while training to discard non-critical image patches after every layer. Unlike previous works, we provide a different training paradigm by using MIM for adversarial pre-training. Our method is efficient as we discard 75% image patches while pre-training. Our method is effective as we eliminate the information of adversarial perturbations from two information sources of natural and adversarial inputs. We also provide theoretical proof that the information of adversarial perturbations is eliminated.

3.2.7 Self-Supervised Adversarial Pre-Training

Self-supervised learning [213, 214, 43, 215] refers to extracting meaningful representation from unlabeled data, which can be used for downstream recognition tasks. Self-supervised methods are beneficial for out-of-distribution detection on difficult, near-distribution outliers [179], which leads to using self-supervised training to improve adversarial robustness [216, 179, 217, 218, 219, 220]. The basic idea is to build a min-max learning object similar to traditional adversarial training in Eq. (1.3). With the development of self-supervision technologies, more advanced technologies, such as contrastive learning and MAE [43], are applied to adversarial pre-training. For example, Jiang et al. [217] considered using two adversarial samples or combining one adversarial sample and one natural sample to learn a consistent representation in contrastive learning. In more recent work, You et al. [221] proposed NIM De^3 to denoise adversarial perturbations. However, the motivation of these works relies on complex self-supervised pre-training technologies, making it more difficult to understand the inner mechanisms or provide theoretical results. MIMIR not only provides better performance but also provides intuitive insights with theoretical motivation.

3.3 MIMIR

3.3.1 Threat Model

Adversary's goal. The attacker aims to fool the trained model with both non-targeted and targeted attacks. The goal is to decrease the overall classification accuracy (non-targeted) or compel the model to recognize any inputs as a specific target (targeted). Meanwhile, the adversarial perturbations applied to the input should be invisible so that they will not be easily detected. During the training phase, the model optimizes its parameters to minimize the loss between predicted outputs and true labels, thereby enhancing classification accuracy. In contrast, the adversary's objective is to develop algorithms that generate perturbations capable of maximizing this loss. For a targeted attack, the attacker decreases the loss between the output and the specified target label. To maintain the imperceptibility of the perturbations, the magnitude of the adversarial modifications is constrained by the l_∞ norm, ensuring that the alterations to the input data remain within a visually indistinguishable range.

Adversary knowledge. The attacker has white-box access to the model, including training data, architectures, hyperparameters, and model weights. The attacker can implement iterative attacks and unlimited queries to update adversarial examples multiple times in white and black-box settings. Adversarial examples can be created according to model architectures, parameters, the gradients of loss function, and datasets. In addition, we also consider adaptive adversaries who are also aware of potential defenses. The adversary can design new attacks for a specific model according to the design details of the defense method.

Defender's goal. From the defender's perspective, the main goal is to train a robust model against potential adversarial attacks. The defender considers the following four objectives:

- The defender aims to prevent the performance of natural data from decreasing significantly but allows a slight drop of natural accuracy for a trade-off in exchange for robustness.
- The defense method should provide a notable improvement compared to models without defenses when subjected to various adversarial attacks, especially to adaptive attacks that are aware of the details of the defense method.
- The defense method should be efficient and scalable to large datasets like ImageNet-1K [187].

3.3.2 Design Intuition

MIM has been proven useful as a pre-training method for ViTs on various tasks [222, 43, 192, 190]. To train a powerful model using MIM, one can mask out a part of the foreground of inputs and reconstruct it using the model. Masking out the foreground instead of the background reduces discriminative information in visible information to the model. Reconstructing foreground parts is harder than background and helps the model learn more discriminative information [223]. Inspired by this phenomenon, we aim to build a more difficult task by adding adversarial perturbations to natural inputs. The adversarial perturbations increase the distance between the input and the reconstruction target. If we can reconstruct natural inputs from adversarial examples, it means the features learned by the model are robust against adversarial attacks. In other words, we want the encoder to learn a latent representation that does not carry information concerning adversarial perturbations while enabling the decoder to reconstruct the natural image. Note that simply masking out all foreground does not make sense, as it would render the reconstruction of meaningful content infeasible. From the perspective of IB, there is a bottleneck between the encoder and decoder. Our adversarial pre-training task contains two information sources: the natural data x and the adversarial perturbations δ. As the information flows through the bottleneck, adversarial information from δ is eliminated. The natural information from x is maintained because of the reconstruction of the target x.

3.3.3 Design Details

Autoencoder. MIMIR consists of an encoder f_e and a decoder f_d aligned with the general design of MAE [43]. As with other autoencoders, the encoder extracts discriminative features z from inputs x. The decoder reconstructs original inputs according to

Algorithm 3.1 MIMIR Pre-training

```
Input: training data $D$, number of epochs $E$, encoder $f_e$, decoder $f_d$, network parameters $\theta$, $\mathcal{L}_{\text{mse}}$, $\lambda$.
Output: optimized weights $\theta$
for $e = 0 \rightarrow E - 1$ do
    $x \leftarrow$ sample_batch($D$)
    $\delta \leftarrow$ random_initialization
    $x_{re} \leftarrow f_d \circ f_e(x + \delta)$
    $\delta \leftarrow \underset{\delta \in S}{max}\, \mathcal{L}_{\text{mse}}(x + \delta, x_{re})$
    Forward:
    $z \leftarrow f_e(x + \delta)$
    $x_{re} \leftarrow f_d(z)$
    $loss \leftarrow \mathcal{L}_{\text{mse}}(x, x_{re}) + \lambda I(x + \delta, z)$
    Backward:
    $\theta \leftarrow \theta - \alpha \nabla loss$
end for
```

the discriminative features. Following the design of ViT [32], the input x is separated into non-overlapping image patches. We randomly mask out a part of the patches and use the remaining patches as inputs for the following process in the encoder. This random masking process uses uniform distribution to prevent potential sampling bias, such as all foreground information being masked, as it becomes infeasible to find the reconstruction target. Thus, we aim to keep a part of the foreground as a hint for reconstruction. The information of masked content is recorded as mask tokens m, not used by the encoder but reserved for later use by the decoder. Each token is a learned vector indicating the presence of a masked patch to be predicted. The mask token is shared by all inputs of the decoder. Like unmasked patches, mask tokens are also assigned corresponding positional embeddings to be in the correct location in the reconstructed image. We emphasize that mask tokens are not used in the encoder part.

To train a ViT, we use the same transformer blocks as ViT to build the encoder. The encoder only processes the visible patches, making training much more efficient. When converting to other architectures, such as ConViT [166], we use corresponding transformer blocks to build the encoder. The decoder accepts the encoded visible image patches and mask tokens as inputs. The decoder is built using the same transformer blocks as the encoder instead of using ViT [32] transformer blocks for all. Then, the decoder is followed by a fully connected layer, which outputs the same number of patches as the original image.

Adversarial Pre-training Target. The training target is to extract discriminative features from visible image patches by the encoder and then reconstruct the invisible patches by the decoder. Therefore, we need a differentiable measurement to quantify the distance between the original image and the reconstructed results. Following the original MAE [43], this distance is measured by Mean Squared Error (MSE). To create a more difficult reconstruction task, we apply adversarial perturbations δ on the inputs of the encoder. Thus, the adversarial perturbations are also masked along with the image upon input into the encoder. The decoder reconstructs the original natural inputs by using the latent features z extracted from adversarial examples. The outputs of decoder x_{re} and x are used to calculate the MSE loss ($\mathcal{L}_{\text{mse}}$), which is further used to optimize the model. Note that the reconstruction differs from the original MAE; we do not use the encoder inputs as reconstruction targets. Formally, the pre-training process (described in Algorithm 3.1) can be written as follows:

$$\begin{gathered} z = f_e(x+\delta),\ x_{re} = f_d(z), \\ \mathcal{L}_{\text{mse}}(x, x_{re}) = (x - x_{re})^2. \end{gathered} \tag{3.4}$$

MI as Penalty. Inspired by IB, we show in Section 3.3.4 that MI between latent representation and adversarial examples decreases as the accuracy on adversarial examples, i.e., $I(x+\delta, z)$ is decreasing while training. Motivated by this finding, we directly use $I(x+\delta, z)$ as a penalty in our final loss function:

$$loss_{\text{mi}} = \mathcal{L}_{\text{mse}}(x, x_{re}) + \lambda I(x+\delta, z), \tag{3.5}$$

where λ is a regularizer for the MI penalty. We use $I(x+\delta, z)$ instead of $I(x, z)$ as a penalty. This is because $x \rightarrow x+\delta \rightarrow z$ follows the Markov chain since z is extracted from $x+\delta$. According to Data Processing Inequality (DPI) [224], $I(x, z) \leq I(x+\delta, z)$. $I(x+\delta, z)$ is closer to z on the Markov chain.

Generating Adversarial Examples. To conduct the adversarial pre-training, we need an attack that finds proper adversarial perturbations δ. As the autoencoder does not provide classification outputs, it is not possible to directly use existing adversarial attacks, such as PGD [12]. Nevertheless, it is feasible to design a new algorithm to find δ by maximizing $loss_{\text{mse}}$ in Eq. (3.4). As the feature z is extracted from only visible image patches, we only attack the visible patches. We do not add any perturbations to mask tokens since the outputs of the autoencoder are only impacted by visible patches. Then, the adversarial pre-training learning objective can be written as:

$$\begin{gathered} \mathcal{L}_{\text{adv}} = \max_{\delta \in S} \mathcal{L}_{\text{mse}}(f_d \circ f_e(x+\delta), x), \\ \min_{\theta} \mathcal{L}_{\text{adv}} + \lambda I(x+\delta, z), \end{gathered} \tag{3.6}$$

where θ are the autoencoder parameters. After the autoencoder is trained, we discard the decoder and initialize a classification layer for the encoder to build a complete model. Finally, the complete model is fine-tuned by AT methods.

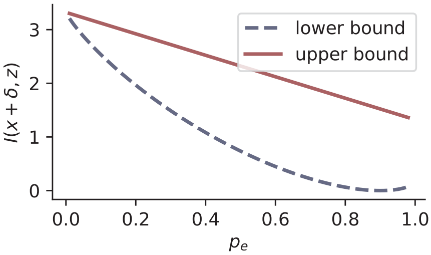


Figure 3.2: The example plots for the lower and upper bounds on the MI in Propositions 3.3.2 and 3.3.3. The entropy ($H(\cdot)$) is chosen uniformly at random from a set of 10 classes. The lower bound reaches its minimum at $p_e = 0.9$.

3.3.4 Theoretical Justification

Next, we provide theoretical justification showing that MI between the adversarial example and its latent representation, i.e., $I(x+\delta, z)$, should be constrained. Let F denote any classifier trained on natural samples with desirable prediction accuracy, which may suffer from adversarial attacks. We begin our analysis by first presenting Lemma 3.3.1.

Lemma 3.3.1 *Let $F(x+\delta)$ and $F(x_{re})$ denote, respectively, the predicted labels of adversarial sample $x+\delta$ and reconstructed sample x_{re}, we have:*

$$I(F(x+\delta), F(x_{re})) \leq I(F(x+\delta), x_{re}) \leq I(x+\delta, z). \tag{3.7}$$

Proof. There are two Markov chains:

$$\begin{aligned} &x+\delta \rightarrow F(x+\delta), \\ &z \rightarrow x_{re} \rightarrow F(x_{re}), \end{aligned} \tag{3.8}$$

which implies that $F(x+\delta)$ is an indirect observation of $x+\delta$, whereas both $F(x_{re})$ and x_{re} are indirect observations of z.

By the data processing inequality (DPI), we have

$$I(x+\delta, z) \geq I(F(x+\delta), z), \tag{3.9}$$

and

$$I(F(x+\delta), z) \geq I(F(x+\delta), x_{re}) \geq I(F(x+\delta), F(x_{re})). \tag{3.10}$$

□

Now, we define p_e as the probability that the predicted label of $x + \delta$ by F is not equal to that of x_{re}, i.e., $p_e = \mathbb{P}(F(x + \delta) \neq F(x_{re}))$. Intuitively, our autoencoder is trained to recover only natural sample x without any interference from δ. Hence, a relatively large value of p_e is expected. In the following, we establish the connection between p_e and $I(x + \delta, z)$ with both lower and upper bounds, showing that minimizing $I(x + \delta, z)$ also encourages a large value of p_e.

Proposition 3.3.2 *Let $H(\cdot)$ denote the information entropy and $H_b(p_e) = -p_e \log_2 p_e - (1 - p_e) \log_2(1 - p_e)$ be the binary entropy, we have:*

$$H(F(x + \delta)) - H_b(p_e) - p_e \log(|F(x + \delta)| - 1) \leq I(x + \delta, z), \tag{3.11}$$

where $|F(x + \delta)|$ is the total number of categories.[†]

Proof. By the chain rule of MI, we have

$$I(F(x + \delta), F(x_{re})) = H(F(x + \delta)) - H(F(x + \delta)|F(x_{re})). \tag{3.12}$$

By applying Fano's inequality [225, 226], we obtain:

$$H(F(x + \delta)|F(x_{re})) \leq H_b(p_e) + p_e \log(|F(x + \delta)| - 1). \tag{3.13}$$

Adding $I(F(x + \delta), F(x_{re}))$ to both sides of Eq. (3.13):

$$\begin{aligned} H(F(x + \delta)) - H_b(p_e) - p_e \log(|F(x + \delta)| - 1) \\ \leq I(F(x + \delta), F(x_{re})) \\ \leq I(x + \delta, z). \end{aligned} \tag{3.14}$$

The last line of Eq. (3.14) is by Lemma 3.3.1. □

Therefore, we obtain a lower bound of $I(x+\delta, z)$. If we use CIFAR-10 ($|F(x+\delta)| = 10$) and assume the predicted labels $F(x + \delta)$ follow a uniform distribution, we can visualize the lower bound as a function of p_e as shown in Figure 3.2, from which we observe an obvious monotonic inverse relationship between $I(x+\delta)$ and p_e in the range $p_e \in [0, 0.9]$. In fact, we can also obtain an upper bound, under the assumption that $I(F(x + \delta), x_{re}) \approx I(x + \delta, z)$, i.e., there is no information loss in the two Markov chains in Eq. (3.8).

[†] For instance, for CIFAR-10, $|F(x + \delta)| = 10$.

Proposition 3.3.3 *If $I(F(x+\delta), x_{re}) \approx I(x+\delta, z)$, we have:*

$$I(x+\delta, z) \lesssim H(F(x_{re})) - 2p_e, \tag{3.15}$$

in which the notation "$\lesssim$" refers to less than or approximately equal.

Proof. By the Hellman-Raviv inequality [227, 228], we have:

$$\begin{aligned} 2p_e &\leq H(F(x+\delta)|x_{re}) \\ &= H(F(x+\delta)) - I(F(x+\delta), x_{re}) \\ &\approx H(F(x+\delta)) - I(x+\delta, z). \end{aligned} \tag{3.16}$$

□

Similar to the lower bound, the upper bound also indicates $I(x+\delta, z)$ is inversely proportional to p_e as shown in Figure 3.2. In fact, apart from the above-mentioned lower and upper bounds, there exists an alternative and intuitive way to understand the mechanism of minimizing $I(x+\delta, z)$. For simplicity, let us assume the natural data x and adversarial perturbations δ are independent[‡], then:

$$I(x+\delta, z) = I(x, z) + I(\delta, z). \tag{3.17}$$

According to [230], minimizing the expected reconstruction error between natural sample x and corrupted input $x+\delta$ amounts to maximizing a lower bound of the mutual information $I(x, z)$, even though z is a function of the corrupted input. Therefore, by minimizing $I(x+\delta, z)$, the network is forced to minimize $I(\delta, z)$ (since $I(x, z)$ is maximized). In other words, only the adversarial information about δ has been removed from z when minimizing $I(x+\delta, z)$. This also explains the robustness of z.

3.4 Experiments

3.4.1 Experimental Setup

We evaluate MIMIR on three datasets: ImageNet-1K [187], Tiny-ImageNet [186], and CIFAR-10 [158], with three commonly used ViT architectures with multiple scales: ViT [32], ConViT [166] and CaiT [206]. In addition, we also evaluate MIMIR on the modern CNN architecture, ConvNext [34] in Appendix 3.7.4. Details of datasets are provided in Appendix 3.7.1. Hyperparameters of the decoder are included in Appendix 3.7.2.

[‡] This assumption is mild for certain scenarios, such as when considering universal or image-agnostic perturbations [229].

Training Setup. We train models from scratch for all experiments. Following MAE [43], we do pre-training by MIMIR for 800 epochs. Please note that we also compare our MIMIR + fine-tuning paradigm with the End2End paradigm. The End2End paradigm refers to the supervised training of a model from scratch without self-supervised pre-training. To compare between End2End and pre-training + fine-tuning, the standard training schedule is pre-training 800 epochs + fine-tuning 100 epochs versus End2End training 300 epochs in the existing works [43, 184, 231].

The number of warmup epochs is 40 for pre-training. We use AdamW [232] as an optimizer for both pre-training and fine-tuning. We apply the cosine decay as the learning rate schedule. At pre-training, MIMIR uses the 1-step PGD to generate adversarial examples for all three datasets. The perturbation budget is $\epsilon = 8, \alpha = 10$. For the fine-tuning stage, we use the 10-step PGD AT with perturbation bound $\epsilon = 8, \alpha = 2$ for CIFAR-10 and Tiny-ImageNet. For ImageNet-1K, we use 1-step PGD with random initialization for better efficiency. The perturbation bound is $\epsilon = 4$ for ImageNet-1K. We also use a 2-step APGD with a longer fine-tuning schedule to compare with SOTA works in Table 3.3. Details on training hyperparameters are provided in Appendix 3.7.3.

As discussed in Section 3.4.3, strong data augmentation is harmful to adversarial training at a common training schedule (50 or 100 epochs for fine-tuning) but helpful to a longer fine-tuning schedule. We only use weak augmentations, including random resized crops and random horizontal flips for 50 or 100 epochs of fine-tuning. For experiments in Table 3.3, we use strong augmentation for 300 epochs of fine-tuning, including weak augmentations, CutMix [181], MixUp [182], and Randaugment [180].

Evaluation Metrics. We use **natural accuracy** and **robust accuracy** as evaluation metrics. Natural accuracy refers to the accuracy of natural and unmodified inputs. The robust accuracy measures the accuracy under the AutoAttack (AA) [233]. AutoAttack is an ensemble of diverse parameter-free attacks, including white-box and black-box attacks. In our experiments, we use the standard version of AutoAttack that contains four attacks, including APGD-ce [233], APGD-t [233], FAB-t [163], and Square [234]. The perturbation budgets are $\epsilon = 8$ for CIFAR-10 and Tiny-ImageNet, $\epsilon = 4$ for ImageNet-1K.

Training stability. Due to the high computation cost, we cannot report the standard deviation for all experiments. To show that our method MIMIR has low variances, we train ViT-S on CIFAR-10 three times with different random seeds, running pre-training 400 epochs and fine-tuning 50 epochs. The natural accuracy is 86.07 ± 0.16 %. The adversarial accuracy under the 20-step PGD_{20} $(l_\infty, \epsilon = 8/255)$ is 47.24 ± 0.12 %.

3.4.2 Main Results

We first explore different AT methods and MIMIR for ViT on CIFAR-10, demonstrating the fundamental incompatibility between conventional AT approaches and ViT architectures. Following this baseline evaluation, we scale our investigation to the more challenging ImageNet-1K dataset, demonstrating the generalizability and scalability of our proposed MIMIR framework. The subsequent sections present comprehensive experimental results across three benchmark datasets: CIFAR-10, Tiny-ImageNet, and ImageNet-1K. This multi-scale evaluation strategy allows a thorough analysis of MIMIR's effectiveness under varying conditions, from smaller to large-scale visual recognition tasks.

CIFAR-10. Table 3.1 shows the performance of End2End adversarial training from scratch and Pre-training (MIMIR) + Fine-tuning on ViT-S trained on CIFAR-10. We provide the performance of 6 established or SOTA AT methods on CIFAR-10, indicating that traditional AT training strategies are not applicable to ViTs. Importantly, our experimental results also demonstrate that MIMIR can substantially improve all AT methods. The reason is that training ViTs from scratch is known to be difficult [32, 235] and even more difficult for adversarial training [172]. For example, robust accuracy is lower than 30% on ViT-B without pre-training [172]. In contrast, MIMIR provides a more straightforward methodology and avoids this difficulty by switching to pre-training with a theoretically grounded MIM learning task.

Table 3.1: Comparison between End2End AT and Pre-training (MIMIR) + Fine-tuning using ViT-S on CIFAR-10.

Training	AT Method	Natural	PGD	AA
End2End	AT [12]	75.36	32.84	26.17
	Fast AT [100]	76.81	32.57	21.41
	TRADES [94]	74.96	32.12	24.90
	MART [154]	72.42	24.47	23.45
	Generalist [165]	60.88	14.44	11.20
	DBAT [142]	68.32	18.83	5.25
MIMIR Pre-training	AT [12]	88.11↑12.75	56.63↑23.79	53.18↑27.01
	Fast AT [100]	87.22↑10.41	49.17↑16.6	35.89↑14.48
	TRADES [94]	88.19↑13.23	56.42↑24.3	51.70↑26.8
	MART [154]	80.55↑8.13	50.81↑26.34	39.92↑16.47
	Generalist [165]	88.81↑27.93	37.85↑23.41	33.67↑22.47
	DBAT [142]	88.56↑20.24	41.08↑22.25	24.59↑19.34

Time Consumption (End2End vs. Pre-training(PT)+Fine-tuning(FT)). Note that we follow the standard way to compare End2End and MIMIR (Pre-training + Fine-tuning) training methods by fixing the training schedule, following existing works [43, 184, 231, 236, 189], where we include pre-training 800 epochs + fine-tuning 100 epochs versus supervised End2End training of 300 epochs. The reason for having a larger number of pre-training epochs is that self-supervised pre-training is much more efficient (see Table 3.12 in Section 3.4.8) than End2End training, and the pre-trained backbone can be used multiple times for various fine-tuning tasks. For example, in Table 3.1, the 6 End2End AT methods cost $300 \times 6 = 1800$ fine-tuning epochs. MIMIR costs 800 pre-training epochs + 100×6 fine-tuning epochs, i.e., we only conduct the pre-training once for results in Table 3.1. MIMIR pre-training epoch is more efficient than a fine-tuning epoch by discarding 75% of image patches.

More specifically, Table 3.2 shows the total time consumption of End2End vs. Pre-training+Fine-tuning on three architectures with ImageNet-1K. Although MIMIR takes more training epochs, its pre-training+fine-tuning paradigm still costs less time than End2End adversarial training and gains much better performance on both natural and adversarial examples. Additional time consumption results with different datasets can be found in Table 3.12 in Section 3.4.8.

Table 3.2: Time consumption of End2End vs. Pre-training+Fine-tuning.

Arch	GPU	AT Method	Epochs	Hours
ViT-S	4 A6000	PGD_{10}	300	187.64
	4 A6000	MIMIR(PT)+PGD_{10}(FT)	800(PT)+100(FT)	123.76
ViT-B	4 A6000	PGD_{10}	300	451.39
	4 A6000	MIMIR(PT)+PGD_{10}(FT)	800(PT)+100(FT)	263.77
Swin-L	4 H100	PGD_{3}	300	180.14
	4 H100	MIMIR(PT)+PGD_{3}(FT)	800(PT)+100(FT)	168.20

ImageNet-1K. Table 3.3 compares MIMIR with previous works concerning adversarial robustness on ImageNet-1K ($l_\infty, \epsilon = 4/255$), which follows the evaluation of common standardized RobustBench [145]. Similar to other works [141, 185, 140], we consider simpler AT methods (i.e., PGD and APGD AT) instead of the latest AT methods, such as Generalist [165] and DBAT [142]. Indeed, since the latest methods introduce tailored components for CNNs to improve their adversarial robustness, they might not be effective for ViTs. The number of parameters, training epochs, steps in the inner maximization of AT, and clean and robust accuracy are reported to provide a more detailed understanding of the performance. The robust accuracy is evaluated by AutoAttack on the RobustBench [145] validation set (5,000 images). We divide the models into: *small* ($\approx$ 22M) and *large* ($\approx$ 86M) models, corresponding to ViT-S and ViT-B. Experimental results demonstrate that MIMIR outperforms all previous works across various training setups.

MIMIR with Various Architectures. In Table 3.4, we show that MIMIR can be applied to diverse architectures. Specifically, we use three representative options, including ViT+convolutional blocks (CVST) [140], the latest CNN architecture (ConvNext [34]), and a hierarchical vision transformer (Swin [33]). The ViT+CVST refers to using ConvStem [238] to replace the patch embedding in ViTs with a convolutional block. The ViT+CVST shows improved robustness compared to pure ViT models in [140].

Table 3.3: Comparison with SOTA results on ImageNet-1K under $\epsilon = 4/255$. The "Adv. Steps" refers to attack steps for generating adversarial examples for AT. The results of [183] are evaluated using 20-step PGD (AutoAttack is designed as a more powerful alternative to PGD), which is marked as † in the table. Although AdvXL only uses 20 epochs, it costs more time due to the huge datasets for pre-training and fine-tuning.

Architecture	Params (M)	FT	FT Epoch	Adv. Steps	Source	Natural	AA
ResNet-50	25.0	PGD	100	1	Aug warm-up [170]	67.44	35.54
DeiT-S	22.1	PGD	100	1	Aug warm-up [170]	66.62	36.56
DeiT-S	22.1	PGD	110	1	Light Recipe [141]	66.80	37.90
ViT-S	22.1	PGD	120	3	EasyRobust [237]	66.43	39.20
ViT-S	22.1	PGD	300	3	AT [185]	70.7	43.7
RobArch-S	26.1	PGD	110	3	RobArch [176]	70.17	44.14
ViT-S	22.1	APGD	300	2	Pre+AT [140]	69.22	44.04
ViT-S	22.1	PGD	100	3	MIMIR	68.08	41.88
ViT-S	22.1	PGD	300	3	MIMIR	71.52	45.90
ViT-S	22.1	APGD	300	2	MIMIR	71.00	46.10
ViT-S	22.1	APGD	300	3	MIMIR	70.96	46.16
ViT-B	86.6	ARD+PRM	10	5	ARD+PRM [172]	69.10	34.62
Swin-B	87.7	ARD+PRM	10	5	ARD+PRM [172]	74.36	38.61
ViT-B	86.6	PGD	120	3	EasyRobust [237]	70.64	43.04
Swin-B	87.7	PGD	120	3	EasyRobust [237]	75.05	47.42
RobArch-L	104	PGD	100	3	RobArch [176]	73.44	48.94
ViT-B	86.6	PGD	300	3	AT [185]	74.7	49.7
ViT-B	86.6	PGD	20	3	AdvXL [183]	73.4	53.0†
ViT-B	86.6	APGD	300	2	Pre+AT [140]	74.10	50.30
ViT-B	86.6	PGD	100	3	MIMIR	75.68	52.96
ViT-B	86.6	PGD	300	3	MIMIR	76.98	53.84
ViT-B	86.6	APGD	100	2	MIMIR	74.40	51.92
ViT-B	86.6	APGD	100	3	MIMIR	74.08	51.24
ViT-B	86.6	APGD	300	2	MIMIR	76.32	54.28
Swin-B	87.7	PGD	150	3	MIMIR	76.62	55.90

As CNN and hierarchical architecture cannot accept variable-length inputs, MIMIR is not directly compatible with ConvNext and Swin. To adapt MIMIR to the hierarchical Swin Transformer, we implemented Masked Image Modeling using Group Window Attention [188], which groups image patches within each local window of arbitrary size and performs masked self-attention in each group. To apply MIMIR to ConvNext, we use SparK [189] for CNN to handle irregular and randomly masked input images, which is achieved by sparse convolution. MIMIR achieves better or comparable results compared to SOTA results on RobustBench [145].

3.4.3 Ablation Study

Step by step ablation. Table 3.5 provides an ablation study to verify the design choices of MIMIR. The ablation uses 100 epochs of 1-step PGD (PGD_1) AT as the baseline. Then, we apply end-to-end clean ImageNet-1K pre-training (weights available in `timm` library[§]) as initialization of AT. After that, we replace the clean pre-training with MAE, adv MAE, and MIMIR step by step. The adv MAE refers to using adversarial examples but not using the MI $I(x + \delta, z)$ in the loss. The pre-training schedule is 800 epochs.

[§] `https://github.com/huggingface/pytorch-image-models/blob/main/timm/models/vision_transformer.py`

Table 3.4: Comparsion with SOTA ImageNet-1K results on RobustBench [145] with different architectures. †: The CVST modules are also pre-trained with MIMIR.

Architecture	Method	FT Epoch	Natural	AutoAttack
ViT-S+CVST	[140]	300	72.56	48.08
	MIMIR	300	72.72	**48.44**
	MIMIR†	300	**73.02**	48.09
ViT-B+CVST	[140]	250	76.30	54.66
	MIMIR	300	**76.72**	54.04
	MIMIR†	300	76.32	**55.08**
ConvNext-T	[140]	300	72.40	48.60
	MIMIR	300	**72.50**	**48.76**
Swin-B	[185]	300	76.16	**56.16**
	MIMIR	150	**76.62**	55.90
Swin-L	[185]	300	**78.92**	59.56
	MIMIR	100	78.62	**59.68**

We also use stronger adversarial fine-tuning for better performance, including 2-step PGD (PGD_2), APGD (APGD_2) FT, and a longer fine-tuning schedule (300 epochs). Please note that a longer training schedule does not guarantee better results [172]. Our results indicate that MIMIR outperforms baselines and can be further improved under the long training schedule. In addition, MIMIR is also efficient and increases less than 5% of time consumption on ImageNet-1K compared to MAE, which is shown in Table 3.12 in Section 3.4.8.

Table 3.5: Ablation of pre-training (PT) and fine-tuning (FT) methods on ImageNet-1K. (†: catastrophic over-fitting [100] due to 1-step AT when fine-tuning, which can be fixed by 2-step AT. The fixed natural and robust accuracy are 69.96 and 36.90, respectively.)

Architecture	Training Recipe	Natural	AA
ViT-S	PGD_1 FT w/o PT	66.02	31.40
	clean PT + PGD_1 FT	67.04	33.70
	MAE PT + PGD_1 FT	69.98	35.64
	adv MAE PT + PGD_1 FT	68.24	19.32†
	MIMIR PT + PGD_1 FT	71.02	37.22
	MIMIR PT + PGD_2 FT	70.78	38.16
	MIMIR PT + APGD_2 FT	68.78	42.86
	100 → 300 epochs FT	71.00	46.10

MI measure. In Section 3.3.4, we provide lower and upper bound (Eq. (3.11)) of $I(x+\delta, z)$. According to the two bounds, $I(x+\delta, z)$ is supposed to decrease while the autoencoder learns to reconstruct the natural image x. This motivates us to directly embed $I(x+\delta, z)$ as a minimizing learning objective. In this chapter, we use I_α [197] and HSIC [153] as estimators (detailed definitions in Appendix 3.7.7). Table 3.6 demonstrates the performance with different values of λ. According to the results, we use HSIC with $\lambda = 1e-05$ for all other experiments.

Table 3.6: Comparison between HSIC [156] and I_α [197] using ViT-T on CIFAR-10. Models are pre-trained 800 epochs and adversarially fine-tuned with 10-step PGD for 50 epochs.

Pre-train	λ	Estimator	Natural	PGD
MIMIR	0.001	HSIC	69.63	43.17
MIMIR	0.001	I_α	75.00	46.11
MIMIR	1e-05	HSIC	**76.30**	**47.60**
MIMIR	1e-05	I_α	75.53	46.75
MIMIR	1e-06	HSIC	74.90	46.19
MIMIR	1e-06	I_α	74.60	45.66

Table 3.7: Comparison between different pre-training settings. All models are pre-trained for 800 epochs and then fine-tuned with 10-step PGD for 50 epochs using ViT-T on CIFAR-10.

Pre-train	λ	Estimator	Natural	PGD
MAE	0.0	-	69.02	42.31
adv MAE (1-step)	0.0	-	74.69	46.28
adv MAE (10-step)	0.0	-	73.96	45.77
MIMIR	1e-05	HSIC	**76.30**	**47.60**

1-step is better than 10-step of AT in pre-training. We also show that MIMIR outperforms original MAE [43] and adv MAE with different PGD steps (to generate adversarial examples for training). MAE in Table 3.7 refers to using the original MAE for pre-training and then fine-tuning with 10-step PGD. The adv MAE refers to using adversarial examples without the MI $I(x+\delta, z)$ in loss. The adv MAE (10-steps) refers to using the 10-step PGD algorithm ($\epsilon = 8, \alpha = 2$) to generate adversarial examples at pre-training. The adv MAE provides higher accuracy than MAE, which supports our statement that using adversarial examples in Masked Image Modeling creates a more difficult reconstruction task. This more difficult task further improves the performance of downstream models (see also Table 3.10). We use the default learning rate (i.e., $5.0e-4$) of MAE, so there is a performance drop in experiments in Tables 3.6 and 3.7 since AT prefers larger learning rates for CIFAR-10 as shown in Table 3.17 in the appendix.

Data augmentation is not always harmful. As mentioned in previous works [170, 141], strong data augmentation makes the training samples too difficult to learn by ViTs while conducting adversarial training. However, we show that data augmentation is not harmful when the training schedule is extended. According to this finding, we apply strong data augmentation in our experiments with a longer fine-tuning schedule, such as our results in Table 3.3.

The strong data augmentation refers to the combination of Randaugment [180], CutMix [181], and MixUp [182]. In this section, we evaluate two different solutions to ease this problem. First, we only use simple data augmentation for adversarial training, including random crop (or random resize crop for ImageNet-1K) and random horizontal flip

Table 3.8: Data augmentation with longer fine-tuning schedule.

Arch	Epoch	Augmentation	Natural	PGD$_{20}$
ViT-B	800	Weak Augmentation	89.90	60.26
		+ CutMix [181],MixUp [182]	91.01	60.62
		+ Randaugment [180]	90.19	62.75

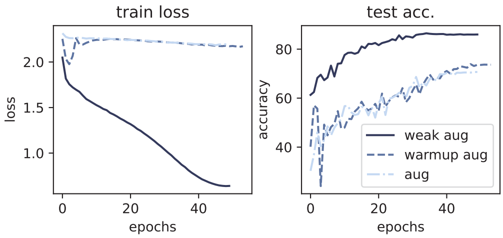


Figure 3.3: Training loss and natural accuracy of ViT-S with three different data augmentations on CIFAR-10.

("weak aug"). Second, we use a 10-epoch warmup procedure for strong data augmentation. The warmup of Randaugment is implemented by progressively increasing the distortion magnitude from 1 to 9 ("warmup aug"). For CutMix and MixUp, we warm up by increasing the mixup probability from 0.5 to 1.0. As shown in Figure 3.3, "weak aug" provides the best accuracy. The "warmup aug" shows a slightly improved accuracy compared to fusing strong augmentation. Therefore, we provide a different result from [170] on the smaller dataset CIFAR-10, i.e., we show that weak augmentation is better than warmup augmentation. Even data augmentation with reduced amplitude is still difficult to learn at the beginning of adversarial training. Although strong augmentation is harmful to a normal training schedule, we show in Table 3.8 that CutMix [181], MixUp [182], and Randaugment [180] increase the accuracy of adversarial training when training with a longer schedule, e.g., 800 epochs of fine-tuning. We conjecture that combining data and strong augmentations is helpful but difficult for adversarial training to learn. Thus, more epochs are needed to learn meaningful representation. Loss and accuracy curves can be found in Appendix 3.7.5. Due to the longer schedule, we use patch size 4 to reduce training time.

3.4.4 Training Epochs Evaluation Study

Pre-training epoch. Our experiments so far are based on 800-epoch pre-training. In Figure 3.4, we show the influence of different numbers of epochs. We use ViT-T and ViT-S as the models for CIFAR-10, ViT-S, and ViT-B for Tiny-ImageNet. We fine-tune each model for 50 epochs after MIMIR pre-training. Both adversarial and natural accuracy are improved with a longer training schedule. On CIFAR-10, the improvement of ViT-T is more significant than that of ViT-S. On Tiny-ImageNet, the increase is more obvious. Therefore, a longer pre-training schedule obviously increases performance.

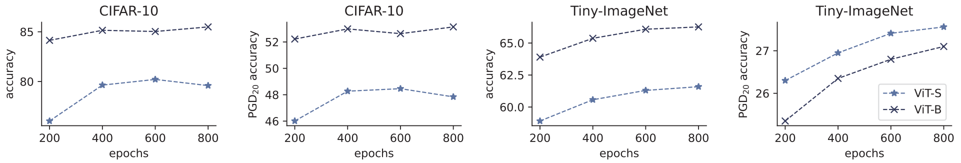


Figure 3.4: Natural and adversarial accuracy of ViT-S and ViT-B with different numbers of pre-train epochs under a 20-step PGD attack. The performance increases as the number of pre-train epochs increases.

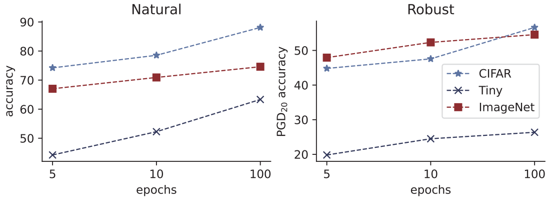


Figure 3.5: Natural and adversarial accuracy of ViT-S adversarially fine-tuned for 5 or 10 epochs on CIFAR-10, Tiny-ImageNet, and ImageNet-1K.

Fine-tuning epoch. To show that the performance of the final model is largely attributed to MIMIR pre-training, we do a very short fine-tuning for pre-trained models. This is to train the randomly initialized classification layer since we do not have the classification layer at pre-training. Therefore, we evaluate the pre-training performance. In Figure 3.5, we show that MIMIR pre-training plus 5 or 10 epochs of fine-tuning is enough to achieve similar performance compared to 100-epoch fine-tuning.

3.4.5 Adaptive Attacks

We evaluate MIMIR against adaptive adversaries following common practices [239]. Adaptive adversaries possess the capability to devise targeted attacks specifically tailored to exploit the mechanisms of MIMIR, particularly if they have prior knowledge of its architecture and defensive strategies. For example, the adversary may attack feature space [240, 241] since MIMIR trains the backbone to extract robust features. Here, the backbone refers to the ViT model without the classification layer, i.e., the encoder of MIMIR.

We provide two adaptive attacks specifically designed against MIMIR. First, we introduce the PGD Mutual Information attack (PGD-MI), which utilizes the MI $I(x+\delta, z)$ to generate adversarial examples, as $I(x+\delta, z)$ is used in MIMIR pre-training as a penalty in the loss. PGD-MI attacks the model by directly increasing the MI $I(x+\delta, z)$. Specifically, we add the MI loss into the PGD algorithm:

$$\max_{\delta \in S} \mathcal{L}_{CE}(x_i + \delta, y_i) + \lambda I(x + \delta, z). \tag{3.18}$$

Table 3.9: Adversarial accuracy by adaptive attacks. The models are pre-trained for 800 epochs by MIMIR and fine-tuned for 100 epochs by 1-step PGD AT.

Dataset	Model	PGD$_{20}$	PGD-MI$_{100}$	PGD-fea$_{100}$
CIFAR-10	ConViT-S	56.35	56.16	78.52
	ViT-S	56.63	56.31	78.41
	ViT-B	58.14	57.85	80.49
Tiny-ImageNet	ConViT-S	26.39	26.29	58.50
	ViT-S	26.37	26.18	57.36
	ViT-B	25.41	25.05	58.90
ImageNet-1K	ConViT-S	53.86	53.84	72.10
	ViT-S	54.56	54.55	72.27
	ViT-B	55.41	55.36	73.51

Table 3.10: Natural accuracy of MAE and MIMIR (800 epochs pre-training for both) that are fine-tuned on natural images.

Architecture	Pre-train	CIFAR-10	Tiny	ImageNet
ViT-B	MAE	96.79	73.38	82.92
	MIMIR	**96.91**	**75.43**	**83.20**
ConViT-S	MAE	94.95	69.03	78.37
	MIMIR	**95.38**	**70.40**	**79.21**
ViT-S	MAE	95.95	70.00	77.45
	MIMIR	95.95	**71.14**	**78.69**

where the value of λ in MIMIR pre-training is available to adversaries.

Second, we introduce a PGD feature attack (PGD-fea) that directly attacks the feature extracted by ViT backbones following [240]. In particular, we attack the feature extractor from the backbones after the adversarial fine-tuning. The PGD-fea attack increases the Euclidean distance between features extracted from natural and adversarial examples. We implement it using the PGD algorithm:

$$\max_{\delta \in S} \mathcal{L}_{\text{mse}}(f_e(x), f_e(x+\delta)). \tag{3.19}$$

Both PGD-MI and PGD-fea are optimized for 100 steps to ensure the attacking algorithm converges. The perturbation budget is the same as the previous evaluation, i.e., $\epsilon = 8/255$ for CIFAR-10 and Tiny-ImageNet, and $\epsilon = 4/255$ for ImageNet-1K. Table 3.9 demonstrates the adaptive evaluation results for PGD-MI and PGD-fea attacks, showing that MIMIR is robust against adaptive attacks on benchmark datasets.

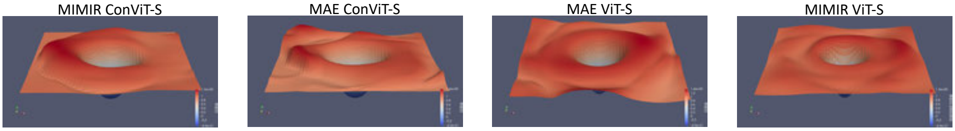


Figure 3.6: The loss landscapes of MIMIR and MAE pre-trained models.

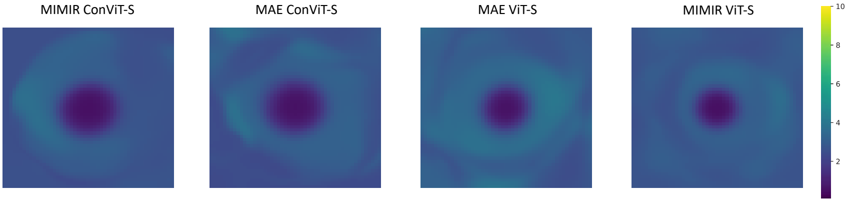


Figure 3.7: The loss heatmap of MIMIR and MAE pre-trained models.

3.4.6 Visualization of the Loss Landscape

To show that the robustness of MIMIR-trained models does not stem from gradient masking, we plot the loss landscape [242] in Figure 3.6. The loss landscape is the visualization of the loss function as parameters change. The basic idea is to plot the loss around the optimal parameters. Formally, we consider in the 2D case,

$$f_l(\alpha, \beta) = L(\theta^* + \alpha\theta_1 + \beta\theta_2), \tag{3.20}$$

where θ_1 and θ_2 are two direction vectors, α and β are two arguments of f_l. In practice, we use the parameters of trained models, i.e., θ^*. The landscapes of all models are smooth, i.e., the gradient at a certain point is clear and can also be easily estimated by local average gradients, which means the gradient is not masked or obfuscated. For completeness, we also provide the corresponding loss heatmap (Figure 3.7) and accuracy at every epoch (Figure 3.8).

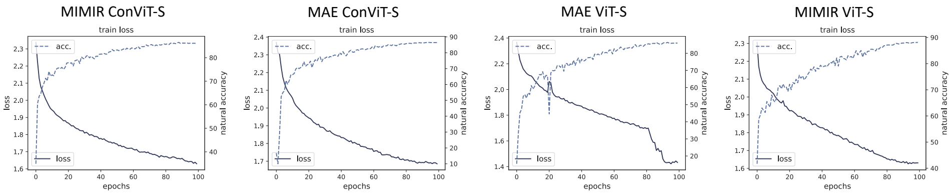


Figure 3.8: The loss and accuracy plots of MIMIR and MAE pre-trained models.

Table 3.11: Natural and adversarial accuracy of ViT-S that is fine-tuned for 5 or 50 epochs with natural images.

Fine-tune	Dataset	Pre-train	Natural	PGD
5 epochs	CIFAR-10	MIMIR	93.22	0.55
		MAE	93.16	0.01
	Tiny-ImageNet	MIMIR	63.65	0.00
		MAE	62.21	0.00
	ImageNet-1K	MIMIR	72.45	0.20
		MAE	69.47	0.02
50 epochs	CIFAR-10	MIMIR	95.95	0.28
		MAE	95.95	0.29
	Tiny-ImageNet	MIMIR	71.14	0.00
		MAE	70.00	0.00
	ImageNet-1K	MIMIR	78.69	0.18
		MAE	77.45	0.05

3.4.7 Fine-tuning with Natural Images

In Table 3.10, we show the results of MIMIR and the original MAE [43] with the same hyperparameters. We fine-tune for 50 epochs for CIFAR-10 and Tiny-ImageNet, 100 epochs for ImageNet-1K. The results in Table 3.10 are reported with 800 pre-training epochs. The blr used in Table 3.10 is 0.001. The fine-tuning batch size is 512 for CIFAR-10 and Tiny-ImageNet and 1024 for ImageNet-1K. We use weak data augmentation ("weak aug"), which includes random crop and random horizontal flip. Surprisingly, MIMIR outperforms MAE when fine-tuning with natural data.

According to Table 3.10 and 3.11, MIMIR consistently shows improved performance on natural data. Although the models in Table 3.11 show poor robustness due to fine-tuning on natural data, MIMIR pre-trained ones provide slightly better robustness. We want to clarify that poor robustness is expected when fine-tuning with natural data. First, it is known that standard training on natural data learns non-robust features [149], which hurts performance under adversarial attacks. Second, MIMIR pre-training is implemented using MSE loss plus an MI penalty between natural inputs and adversarial images. The adversarial perturbations and MI penalty help MIMIR create a more difficult and discriminative learning task to learn meaningful and robust features. This process does not include the classification layer of the final model. Therefore, MIMIR still needs a trivial fine-tuning process for superior performance on natural data and adversarial inputs. In other words, the superior performance of our experiments comes from the combination of MIMIR and the trivial fine-tuning process. In our case, we use the simplest PGD adversarial training and plain training on natural data.

Table 3.12: The average time consumption on 4 GPUs. The "mem." refers to GPU memory usage. The total time is estimated based on time consumption on a single epoch. The training schedule for PGD_{10} and FastAT is 300 epochs. The training schedule for MAE and MIMIR is 800 epochs.

		CIFAR-10 [158]		**Tiny-ImageNet** [186]		**ImageNet-1K** [187]	
Architecture	Method	time[H]	mem.[GB]	time[H]	mem.[GB]	time[H]	mem.[GB]
ViT-S	PGD_{10} AT	12.44	2.54×4	25.5	3.99×4	187.64	12.5×4
	FastAT	3.61	2.54×4	5.64	4.03×4	46.29	10.4×4
	MAE	3.58	3.24×4	7.33	3.27×4	59.91	11.1×4
	MIMIR	4.09	3.12×4	8.89	3.18×4	61.22	11.1×4
ViT-B	PGD_{10} AT	30.1	5.39×4	85.18	8.30×4	451.39	22.1×4
	FastAT	10.23	5.36×4	15.02	8.34×4	113.44	19.8×4
	MAE	11.78	5.95×4	23.67	5.95×4	109.09	17.0×4
	MIMIR	13.11	6.08×4	27.11	6.11×4	113.31	17.0×4
ConViT-S	PGD_{10} AT	36.88	6.64×4	74.75	12.19×4	552.21	32.5×4
	FastAT	8.88	5.86×4	15.27	10.62×4	119.27	26.4×4
	MAE	7.33	10.6×4	15.0	10.61×4	135.49	27.5×4
	MIMIR	10.0	10.4×4	20.0	10.54×4	135.8	28.3×4

3.4.8 Efficiency

We provide an analysis of the efficiency of MIMIR. Table 3.12 provides the total time consumption and memory usage of different adversarial training methods, which are evaluated on four A6000 GPUs. MIMIR is more efficient than 10-step PGD but slightly less efficient than FastAT, with higher robust accuracy than both 10-step PGD and FastAT. Note that FastAT could easily overfit [100], and our experiments on CIFAR-10 (see Table 3.1) also indicate that FastAT has converged and tends to overfit from around 300 epochs. Thus, we only provide the training time of FastAT with 300 epochs, which represents the best performance of FastAT. Further, we provide the training time of MAE in Table 3.12, which shows that the extra training time consumption introduced by the calculation MI between $x + \delta$ and z is small. In sum, MIMIR introduces limited computational overhead on the current most efficient method FastAT, while outperforming 10-step PGD and FastAT in robust accuracy.

3.5 Discussion and Limitations

Across our experiments, we have observed the promising performance of MIMIR. Following the principle of IB, we can intuitively consider a bottleneck between the encoder and decoder. As the reconstruction output is constrained by natural data x, the bottleneck will filter out information from adversarial perturbations δ. We provide a theoretical guarantee of this bottleneck in Section 3.3.4. In Eq. (3.5), we embed this bottleneck as a learning object to further improve the performance, which also confirms the correctness of our theoretical guarantee in Section 3.3.4. Table 3.7 shows that our method works better than related works even without embedding the bottleneck in Eq. (3.5). With the two information sources of x and δ, the model is trained to learn the robust features from x and forget the information of δ under the constraint of the reconstruction target.

While MIMIR shows better performance, there are still certain limitations. MIMIR is a pre-training method. An adversarial fine-tuning is necessary to build the final robust model. Thus, the shortcomings of traditional adversarial training cannot be completely avoided. In our experiments, we utilize the simple PGD algorithm for fine-tuning, but one can further improve MIMIR pre-trained models with more advanced approaches. In addition, MIMIR follows the design of MAE, and we also utilize the characteristic that ViTs can process variable-length inputs. Therefore, current MIMIR cannot directly handle pyramid-based ViTs and CNNs. While it is not trivial, we apply MIMIR to the latest CNN architecture (see Appendix 3.7.4) by sparse convolution from SparK [189]. However, the sparse convolution is not as efficient as dropping patch embeddings. We leave these limitations to future work.

3.6 Conclusions

This chapter provides a novel theoretical MI analysis on ViT adversarial training, showing that MI between the adversarial example and its latent representation in ViT-based autoencoders should be constrained by utilizing the MI bounds. Based on this finding, we propose MIMIR as a theoretically grounded pre-training method to improve adversarial robustness for ViTs. MIMIR uses adversarial examples as inputs and natural data as the reconstruction target. In this way, the information from the adversarial perturbations is decreased by the bottleneck, while the information from natural data is preserved while reconstructing the target. Our experimental results show that MIMIR substantially improves adversarial robustness compared to recent related works on various benchmark datasets.

3.7 Appendix

3.7.1 Datasets

We use three commonly used datasets to evaluate MIMIR: CIFAR-10 [158], TinyImageNet [186], and ImageNet-1K [187]. CIFAR-10 [158] comprises 50,000 images with size $3 \times 32 \times 32$ in 10 classes. ImageNet-1K [187] is the most commonly used dataset for the evaluation of ViTs and their variants, which is composed of more than 1.2 million high-resolution images in 1,000 classes. In our experiments, images from ImageNet-1K are resized to $3 \times 224 \times 224$. For completeness, we also include Tiny-ImageNet [186] as a medium size dataset between CIFAR-10 [158] and ImageNet-1K [187]. Tiny-ImageNet [186] contains 100,000 images with size $3 \times 64 \times 64$ in 200 classes.

3.7.2 Decoder Hyperparameters

We use transformer blocks but fewer layers as the backbone of the decoder. For CIFAR-10, we use the patch size of 2, 4 for Tiny-ImageNet, and 16 for ImageNet-1K. Table 3.13 shows the hyperparameters of decoder architectures. For different ViT architectures, we use the transformer blocks of the respective architectures to build the encoder.

Table 3.13: Model architectures of the encoder and decoder.

Model	Layers	Hidden size	MLP ratio	Heads
ViT-T (encoder)	12	192	4	3
decoder	2	128	4	16

3.7.3 Details of Training Hyperparameters

In Tables 3.14 and 3.15, we provide the default hyperparameters used in our experiments. We use different patch sizes for different datasets: patch size 2 for CIFAR-10, 4 for Tiny-ImageNet, and 16 for ImageNet-1K. Using smaller patch sizes increases the time consumption when calculating self-attention, but MIMIR pre-training discards 75% patches, making it still efficient. Due to the depth and comparatively small embedding size of CaiT, we use a different drop path and layer-wise decay when fine-tuning (for ImageNet-1K). For CaiT-XXS24, we use 0.95 and 0.15 as layer-wise decay and dropout, and 0.85 and 0.35 for CaiT-S36. We also apply the stochastic depth decay rule [243] to CaiT. CaiT-S36 models are only fine-tuned for 50 epochs due to time consumption, and it is sufficient to get superior results. The batch size to fine-tune CaiT is 512 due to the limitation of GPU memory. Other hyperparameters are consistent with Tables 3.14 and 3.15.

In addition, we use elucidating diffusion model (EDM) data as a data augmentation. Generative data is usually used to improve adversarial training [98, 244, 97, 96]. Specifically, we use 5 million generated CIFAR-10 data and 1 million Tiny-ImageNet data provided by [98]. The EDM data is applied to experiments with CIFAR-10 and Tiny-ImageNet but not to ImageNet-1K.

Table 3.14: Pre-training hyperparameters.

Config	Value
optimizer	AdamW
base learning rate	1.5e-4
weight decay	0.05
optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.95$
batch size	512(CIFAR-10, Tiny), 2,048 (ImageNet-1K)
learning rate schedule	cosine decay
warmup epochs	40
training epochs	800
augmentation	RandomResizedCrop, RandomHorizontalFlip

Table 3.15: Fine-tuning hyperparameters.

Config	Value
optimizer	AdamW
base learning rate	0.5e-2 (CIFAR-10), 1e-3 (ImageNet-1K, Tiny)
weight decay	0.05
optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
layer-wise lr decay	0.65
batch size	128 (CIFAR-10), 256 (Tiny), 1,024 (ImageNet-1K)
learning rate schedule	cosine decay
warmup epochs	10
training epochs	100
augmentation	RandomResizedCrop, RandomHorizontalFlip
drop path	0.1

3.7.4 Results on CNN

We also apply our MIMIR pre-training to CNN architecture. In Table 3.16, we report the performance of MIMIR pre-training (400 epochs) + APGD fine-tuning (300 epochs) on the modern CNN architecture, ConvNext [34]. Compared to the current SOTA work [141, 140], we achieve comparable or better performance.

Note that the original MIMIR is not compatible with CNN because MIMIR utilizes the feature that ViT accepts variable-length inputs to perform masked image modeling. However, CNNs only accept fixed-length inputs. To solve this problem, we use SparK [189] for CNN to handle irregular and randomly masked input images, which is achieved by sparse convolution.

Table 3.16: Results on ConvNext.

Architecture	Method	Adv. Step	Natural	AA
ConvNext-T	[141]	1	71.60	44.40
	[140]	2	72.40	48.60
	MIMIR	2	**72.50**	**48.76**

3.7.5 Data Augmentation Evaluation

Figure 3.9 demonstrates the loss and accuracy while training with different augmentations. “no mix” refers to using only weak augmentation, including RandomResizedCrop and RandomHorizontalFlip. “+mix” refers to using MixUp (0.8) and CutMix (1.0). “+aug” refers to using MixUp (0.8), CutMix (1.0), and Randaugment (rand-m9-mstd0.5-inc1).

3.7.6 Dropout is Important for Deeper Architecture

We also consider applying our method to deeper ViTs (i.e., CaiT [206]). CaiT is designed for deeper high-capacity transformers that benefit from depth. Simply increasing the depth may cause training failure of ViTs [206]. The CaiT solves this problem by LayerScale and class-attention. In adversarial training, training deeper networks is even more difficult.

Table 3.17: Different learning rates. Fine-tuned for 50 epochs.

Dataset	Models	LR	Natural	PGD_{10}
CIFAR-10	ViT-T	5.0e-4	76.30	47.60
		1.0e-3	80.69	49.56
		1.0e-2	85.62	48.78
		5.0e-2	85.12	50.30
		1.0e-1	84.51	50.40

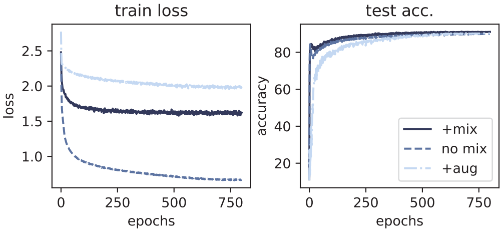


Figure 3.9: The training results of using different data augmentations with 800 epochs.

The dropout has a significant impact on robustness. In Table 3.18, we show the results using MIMIR with different dropout values.

3.7.7 Mutual Information and HSIC

MI measures the mutual dependence between two random variables, X and Y. It can be decomposed as:

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y), \\ &= H(Y) - H(Y|X), \\ &= H(X) + H(Y) - H(X,Y), \end{aligned} \tag{3.21}$$

where $H(X)$ and $H(Y)$ are the information entropies, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y.

Table 3.18: The experimental results on CaiT-S36. The “SDD” refers to the stochastic depth decay rule [243].

Dropout	Layer Decay LR	$\epsilon = 2/255$	
		Natural	PGD
0.1	0.65	72.68	35.60
0.1+SDD	0.65	73.06	38.88
0.2	0.75	75.41	43.99
0.25	0.75	71.46	48.86
0.25	0.85	76.43	53.49
0.25+SDD	0.85	75.92	56.53
0.35+SDD	0.85	76.05	56.78

Unfortunately, estimating MI in high-dimensional space is difficult since it involves a precise estimation of the underlying data distribution $P_{(X,Y)}$ or $P_{(X)}$ and $P_{(Y)}$. To address this issue, the deterministic information bottleneck (DIB) [197] uses the recently proposed matrix-based Rényi's α-entropy functional I_α [198, 199], which suggests similar quantities to $I(X,Y)$ in terms of the normalized eigenspectrum of the Hermitian matrix of the projected data in the reproducing kernel Hilbert space (RKHS), but avoids density estimation.

Specifically, given N pairs of samples $(x_i, y_i)_{i=1}^{N}$ (in our setup, N refers to the mini-batch size), we can obtain two Gram (or kernel) matrices K_x and K_y, for variables X and Y, respectively, with $(K_x)_{i,j} = \kappa_x(x_i, x_j)$, $(K_y)_{i,j} = \kappa_y(y_i, y_j)$, in which κ_x and κ_y are corresponding kernel functions. The information entropy of X can be expressed as:

$$\begin{aligned} H_\alpha(X) &= \frac{1}{1-\alpha}\log_2\left(\operatorname{tr}(\tilde{K_x}^{\alpha})\right) \\ &= \frac{1}{1-\alpha}\log_2\left(\sum_{i=1}^{N}\lambda_i(\tilde{K_x})^{\alpha}\right), \end{aligned} \tag{3.22}$$

where $\tilde{K}$ is the normalized version of K, i.e., $\tilde{K} = K/\operatorname{tr}(K)$, and $\lambda_i(\tilde{K})$ denotes the i-th eigenvalue of $\tilde{K}$.

Further, the joint entropy for X and Y can be expressed as:

$$H_\alpha(X,Y) = H_\alpha\left(\frac{K_x \circ K_y}{\operatorname{tr}(K_x \circ K_y)}\right), \tag{3.23}$$

where $K_x \circ K_y$ denotes the Hadamard product between the matrices K_x and K_y.

Given Eqs. (3.22) and (3.23), the matrix-based Rényi's α-order mutual information $I_\alpha(X;Y)$ in analogy of Shannon's MI is given by:

$$I_\alpha(X;Y) = H_\alpha(X) + H_\alpha(Y) - H_\alpha(X,Y). \tag{3.24}$$

Throughout this chapter, we use the radial basis function (RBF) kernel $\kappa(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ with kernel width σ to obtain the Gram matrices.

The Hilbert–Schmidt Independence Criterion (HSIC) [156] is also a kernel-based dependence measure and is usually used as a surrogate of MI. Formally, the HSIC is defined as the squared norm of the cross-covariance operator $||C_{XY}||^2$:

$$\begin{aligned}
&\mathrm{HSIC}_{P_{X,Y}}(X,Y) \\
&= ||C_{XY}||^2 \\
&= \mathbb{E}_{xyx'y'}[\kappa_x(x,x')\kappa_{y'}(y,y')] \\
&+ \mathbb{E}_{xx'}[\kappa_x(x,x')]E_{yy'}[\kappa_y(y,y')] \\
&- 2\mathbb{E}_{xy}[\mathbb{E}_{x'}[\kappa_x(x,x')]\mathbb{E}_{y'}[\kappa_y(y,y')]],
\end{aligned} \tag{3.25}$$

where κ_x and κ_y are kernel functions, $\mathbb{E}$ is the expectation, x' and y' are independent copies of x and y, respectively.

Given N pairs of samples $(x_i, y_i)_{i=1}^N$, the empirical estimator of HSIC is given by:

$$\widehat{\mathrm{HSIC}}_{P_{X,Y}}(X,Y) = \frac{1}{N^2}\mathrm{tr}\left(K_x H H_y H\right), \tag{3.26}$$

$(K_x)_{i,j} = \kappa_x(x_i, x_j)$, $(K_y)_{i,j} = \kappa_y(y_i, y_j)$, and $H = I - \frac{1}{N}\mathbb{1}\mathbb{1}^T$ is the centering matrix.

Part II

Training-Time Adversarial Machine Learning

Chapter 4

Adversarial Perturbation for Backdoor Detection

A deep learning model may be poisoned and still perform as expected when receiving a clean input, but will misclassify when receiving a backdoored input. This is similar to universal adversarial perturbations (UAP). Indeed, UAPs are input-agnostic perturbations capable of misleading a well-trained model. We observe an intuitive phenomenon: UAPs generated from backdoored models need fewer perturbations than UAPs from clean models for a successful attack. UAPs from backdoored models tend to exploit the shortcut from all classes to the target class, built by the backdoor. Based on this finding, this chapter propose a backdoor detection method called Universal Soldier for Backdoor Detection (USB). With it, we can reverse engineer potential backdoor triggers via UAPs. Experiments on 240 models show that USB effectively detects the injected backdoor and provides comparable or better results than state-of-the-art methods.

4.1 Introduction

Deep learning technologies are subject to security attacks like backdoor attacks [8, 9]. The backdoor attack commonly poisons a small part of the training data with a specific trigger to build a covert link between the trigger and the target label. The infected model behaves normally on clean inputs, but if the covert link is activated by an input with the trigger, the model will output an attacker-desired target label. The backdoor attack poses urgent security concerns when users outsource model training to third parties, such as Machine Learning as a Service (MLaaS) [245], BigML [246], or when users reuse pre-trained models from online platforms like Caffe Model Zoo [247] and Model Zoo [248].

There have been several proposals to detect backdoor attacks [110, 127, 144] by analyzing well-trained models. In particular, reverse engineering approaches, such as Neural Cleanse

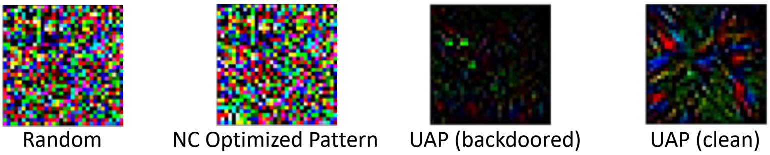


Figure 4.1: The random point is barely updated by NC.

(NC) [110], aim to reconstruct the trigger. However, such methods may capture the unique features of the target class instead of the trigger [127]. Both class features and triggers can lead a backdoored model to the target class. Reverse engineering decides whether there is a backdoor based on the size (L_1 norm) of the reconstructed triggers for every class. If the difference between the unique class features and the trigger is not particularly large concerning size, reverse engineering may not generate the trigger; see Fig. 4.4 as an example. Furthermore, these methods work well against patch-based triggers (e.g., a fixed square as a trigger), such as BadNet [8], but may fail under non-patch-based attacks (see Tab. 4.3), such as Input-Aware Dynamic backdoor attack (IAD) [11]. The reason is that reverse engineering usually starts from a random point, which is very different from triggers designed by more advanced attacks, and NC-style methods only optimize the mask (for details, see Sect. 4.2) without directly updating trigger patterns. In Fig. 4.1, we show the pattern updated by NC from random noise. Therefore, it is difficult for NC-style methods to generate attack-specific triggers. Moreover, NC-style methods also need a significant amount of data (the whole training set) to perform the optimization with a larger number of iterations [110].

This chapter presents a novel detection mechanism (USB) that does not suffer from the aforementioned issues. More specifically, we investigate an inference-time defense requiring only a small amount of clean data. To avoid using the class's unique feature as a trigger, we utilize the similarities between backdoor attacks and adversarial attacks, especially universal adversarial perturbations (UAP) [229]. The UAP effectively fools the victim model on any inputs because it captures the correlations among different regions of the decision boundary [229]. We conjecture that UAP can also capture the feature of backdoor neurons, resulting in smaller perturbations; see Fig. 4.1. This is because UAP utilizes the normals to the decision boundary in different regions of the decision space, i.e., the UAP finds the shortest path to cross the decision boundary. Backdoored models build shortcuts from all classes to the targeted class by the trigger. Therefore, UAPs from backdoored models are smaller than UAPs from clean models. USB requires less iterations and data, as we directly use UAP to capture the potential backdoor. As UAP can be generalized across different networks, we only need to generate UAP once for similar models, greatly reducing the time requirement.

We evaluated USB on 240 models (150 on CIFAR-10 [158] with ResNet-18 [27], 45 on

ImageNet [187] with Efficientnet-B0 [249], and 45 on CIFAR-10 [250] with VGG-16 [26]). As the results indicate, USB outperforms the latest detection techniques [110, 127]. Our contributions are summarized as follows:

- We propose a novel detection method, USB, that utilizes the similarities between backdoors and UAPs. We show that a UAP with the same target as a backdoor attack is smaller than UAPs with a different target from the backdoor attack regarding the L_1 norm.
- USB can detect stronger backdoor triggers for both patch-based (BadNet and Latent) and non-patch-based (Input-Aware Dynamic) backdoors. Existing methods tend to conduct reverse engineering from random noise, which may not work under advanced attacks. Our reversing process uses the target UAP as the starting point for initialization to avoid the local optimal triggers, as the UAP is closer to potential triggers.
- We conduct experiments on 240 models to assess our approach. We compare USB with the NC and TABOR methods and USB provides competitive performance on various datasets compared to state-of-the-art methods.

4.2 Related Work

Training-time Defenses. Training-time defenses refer to defenses that are conducted during the training of the model, including detecting poisoned data points in training data [109], reducing the impact of poisoned data on training the model by differential privacy [251], input pre-processing [252], and randomized smoothing [253]. These methods take advantage of the difference between clean and poisoned data concerning the victim model. For a clean sample that originally belongs to the target class, the model recognizes it as the target class because the sample contains the features of the target class. For a poisoned sample, the backdoored model extracts trigger features for classification. However, training-time defenses require access to training data, which may not be feasible in cases where the model is pre-trained by a third party.

Inference-time Defenses. Inference-time defenses refer to defenses with access to the pre-trained model and a certain amount of clean data, including detection by reverse engineering of the backdoor trigger [110, 127, 254], pruning [130], and machine unlearning to remove the backdoor [110]. The pruning and machine unlearning aim to remove the backdoor by directly modifying the victim model, while the reverse engineering conducts detection and reconstructs the backdoor trigger. Reverse engineering methods, such as NC [110] and TABOR [144], take advantage of the behavioral characteristics of the backdoor itself. The backdoor builds a shortcut from within regions of the space belonging to each label into the region belonging to the target. For backdoored models, transforming input features of any class into features of the target class requires less perturbation than

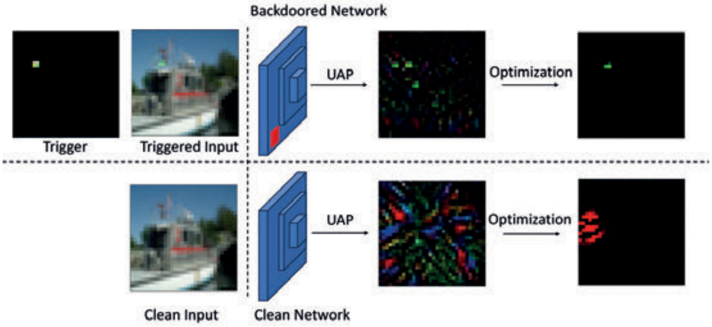


Figure 4.2: The structure of USB. First, we generate targeted UAP for the clean model and the backdoored model. Second, we optimize UAPs to reverse engineer the potential trigger. The pixel values of the two UAPs are scaled for visibility.

Algorithm 4.1 Computation of targeted UAP.

```
Input: Data points $X$, target class $t$, victim model $f$, desired $l_p$ norm of the perturbation $\delta$, desired error rate $e$
Output: Targeted UAP $v$
Initialize $v \leftarrow 0$
while Error$(X + v) \geq e$ do
    for $x_i$ in $X$ do
        if $f(x_i + v) \neq t$ then
            Minimal perturbation that send $x_i + v$ to class $t$:
            $\triangle v_i \leftarrow \underset{r}{arg\,min}\|r\|_2 \; s.t. f(x_i + v + r) = t$
            $v \leftarrow \triangle v_i$    ▷ Update the perturbation under limitation
        end if
    end for
end while
```

transforming into other classes, as the backdoor trigger must be small to remain stealthy. Reverse engineering searches for a trigger for each class of the model to be detected. Every searched trigger can mislead the model to output the corresponding class when applied to any inputs. If the model is backdoored, there will be an outlier trigger that is smaller than others.

4.3 Proposed Method

4.3.1 Threat Model

We consider defense against a backdoor attack under a white box scenario. The adversary (attacker) has white box access to the victim model and training data. The adversary injects backdoors through the dirty label attacks. The defender aims to detect these backdoors. In this chapter, we consider an all-to-one setting where the triggers mislead all inputs of backdoored models to a single target class.

Algorithm 4.2 Updating of targeted UAP.

```
Input: Data points $X$, target class $t$, victim model $f$, UAP $v$, Maximum iteration number $m$, learning rate $lr$
Output: Updated UAP $v' = pattern \times mask$
Initialize trigger by $v$ : $trigger = pattern \times mask = v$
for $i = 0$ to $m$ do
    $x \subseteq X$
    $x' = x \times (1 - mask) + pattern \times mask$
    $output = f(x')$
    $\mathcal{L} = \mathcal{L}(output, t) - SSIM(x, x') + norm_{L1}(mask)$
    Backward loss $\mathcal{L}$ to update $mask$ and $pattern$
    $mask : mask \leftarrow mask - lr \times \nabla mask$
    $pattern : pattern \leftarrow pattern - lr \times \nabla pattern$
    $v' = pattern \times mask$
end for
```

4.3.2 Defense Overview

Our method consists of two main processes to detect whether there is a backdoor in the targeted model. First, we generate a targeted UAP for the victim model. The UAP is supposed to capture special neurons that can easily lead to misclassification. Then, we use an optimization process to update the UAP so that it can focus on the most important part. This "most important part" refers to the part of UAP most likely to cause misclassification. We generate and optimize targeted UAPs for every class of the model. We check whether there is an outlier smaller than the others from these UAPs to decide if the model is backdoored or not. Fig. 4.2 illustrates the framework of our defense.

4.3.3 Targeted UAP

To work in the all-to-one setting, we modify the algorithm from [229] to generate targeted UAP, which misleads all inputs to the targeted class. Let us assume a well trained deep learning model f and K entries of training data $D = \{(x_i, y_i)\}_0^{K-1}$ where $x_i \in \mathbb{R}^{d_X}$ and $y_i \in \{0, 1\}^N$. N is the number of classes, and d_X is the input dimension. The targeted UAP algorithm aims to find a perturbation vector v that misleads the model f on most of the data points in D to a target class t. We use a very small number of data points X to work in a more realistic situation. Empirically, a size smaller than 1% of D can be enough for X. Then, the perturbation v should satisfy the following two constraints to ensure practicality. First, the generated perturbation v should successfully mislead the model f, i.e., the error rate should be larger than the desired threshold e:

$$Err(X+v) := \frac{1}{K}\sum_{i=0}^{K-1} r_i \geq e,$$
$$\text{where} \quad r_i = \begin{cases} 1, & f(x_i+v) \neq f(x_i) \\ 0, & f(x_i+v) = f(x_i). \end{cases}$$

Second, the perturbation v should be imperceptible. Specifically, the norm of v should be smaller than the limit δ:

$$P_{l_p,\delta}(v, \triangle v_i) = arg\min_{\triangle v_i} \|v + \triangle v_i\|_2, s.t. \|\triangle v_i\|_p \leq \delta.$$

In Alg. 4.1, we iteratively go through every data point in X to update UAP from scratch. At each iteration, the algorithm searches for the minimal perturbation that sends $x_i + v$ to the target class. Then, the error rate of inputting $X + v$ to f should be larger than the desired threshold e so that v is effective as a targeted UAP. This is feasible by solving the following optimization problem:

$$\triangle v_i \leftarrow arg\min_{r} \|r\|_2 \; s.t. f(x_i + v + r) = t.$$

Following the algorithm in [229], this search optimization is implemented by DeepFool [255].

4.3.4 UAP Optimization

UAP is a collection of normals* to the decision boundary in different regions [229], including but not only the regions where the trigger is located. Therefore, we further update the targeted UAP through an optimization phase. The optimization objective is formalized as a loss function:

$$\mathcal{L} = \mathcal{L}_{ce}(output, t) - SSIM(x, x') + norm_{L1}(mask), \tag{4.1}$$

where $\mathcal{L}_{ce}$ refers to the cross-entropy loss. The structural similarity index measure (SSIM) measures the similarity between images [256].

The details are provided in Alg. 4.2. The optimization achieves two goals: (1) it makes the targeted UAP focus on more important pixels, and (2) it ensures that the UAP can mislead the victim model. The first goal is embedded in the loss by a trigger and a mask, i.e., minimizing the mask by decreasing $norm_{L1}(mask)$. At the beginning of Alg. 4.2, the trigger and mask are initialized by the targeted UAP. The trigger is a copy of the UAP, and the mask has the same shape as the trigger. In every iteration, the algorithm takes a batch of data from X instead of using the whole X. The next iteration will use the

*Minimal distance from the region to the decision boundary.

Model	Acc.	ASR	Method	Reversed Trigger	Model Detection		Target Class Detection		
				L_1 norm	Clean	Backdoored	Correct	Correct Set	Wrong
Clean	85.38	N/A	NC	51.59	50	0	N/A	N/A	N/A
			TABOR	55.09	50	0	N/A	N/A	N/A
			USB	48.99	50	0	N/A	N/A	N/A
Backdoored (2×2 trigger)	83.43	95.04	NC	8.72	5	45	44	1	0
			TABOR	9.26	5	45	44	1	0
			USB	9.83	1	49	45	4	0
Backdoored (3×3 trigger)	83.59	97.57	NC	8.89	2	48	48	0	0
			TABOR	10.06	3	47	47	0	0
			USB	12.02	1	49	49	0	0

Table 4.1: Detection evaluation on CIFAR-10 where each case consists of 50 trained models.

data after x in order. Thus, all data in the X will be used. The purpose is to reduce the running time of each iteration. When initializing, elements in the mask are set to one, such that the first x' (line 4 in Alg. 4.2) equals $x + v$. Then, the trigger and mask will be updated according to the gradients generated on the trigger and mask during computation. Note that f will not be modified in Alg. 4.2. This optimization may introduce excessive perturbations, so we use SSIM to keep $x + v$ similar to the original image x. Finally, we decrease cross-entropy loss between output $(f(x'))$ and target class (t) for the second goal.

Detection. Based on the above discussion, we can generate targeted UAPs for all classes to detect whether a model has a backdoor. Given a model f that may have been injected with a backdoor, we generate N targeted UAPs corresponding to every class, i.e., $\{v_i\}_0^{N-1}$. Then, these UAPs are optimized by Alg. 4.2 to locate the position of the potential trigger. We use $\{v_i'\}_0^{N-1}$ to indicate optimized UAPs. As mentioned before, misleading a backdoored model to the target class needs a smaller perturbation compared to the untarget classes. Therefore, if f is backdoored on class t_b, the size of v_{t_b}' will be smaller than other UAPs in $\{v_i'\}_0^{N-1}$. The size of UAPs is quantified by the L_1 norm. Empirically, the L_1 norm of the targeted UAP for the backdoored class is more than one order of magnitude smaller than that of targeted UAPs for other classes without a backdoor. For example, for a ResNet-18 model with a BadNet backdoor on class 0, the L_1 norm v_0' generated by USB is 4.49, and the average L_1 norm of the other classes is 53.76.

4.4 Evaluation

We provide the experimental results for USB and compare them with NC [110] and TABOR [144], which are the typical state-of-the-art methods. Experiments are conducted with TrojanZoo [257]. We use different random seeds for every trained model.

4.4.1 Experimental Setup

Models, Datasets, and Backdoor. We use ResNet-18 [27] and VGG-16 [26] for CIFAR-10 [250], and Efficientnet-B0 [249] for ImageNet [187]. We use BadNet [8], Latent Back-

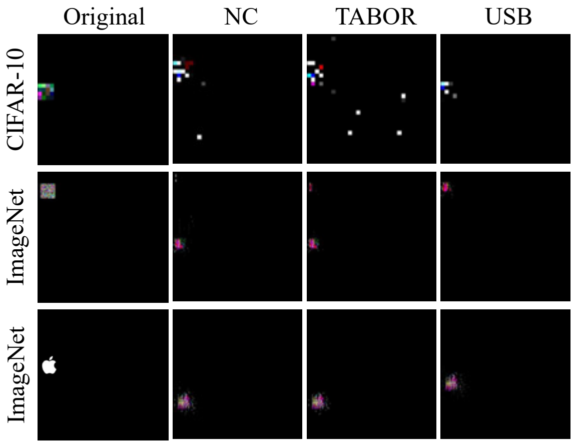
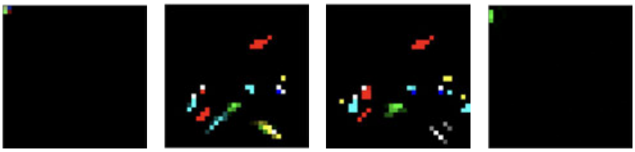


Figure 4.3: Examples of the original and reversed triggers by NC, TABOR, and USB for CIFAR-10 and ImageNet.



(a) 2×2 (b) NC (c) TABOR (d) USB

Figure 4.4: An example visualization of the original and reversed triggers by NC, TABOR, and USB for CIFAR-10.

door [258], and IAD [11] to inject backdoor into victim models. The triggers are generated in different positions and random colors.

Hyperparameters. For Alg. 4.1, we set the desired error rate to $e = 0.6$. X contains 300 data points. In our experiments, these are the minimum hyperparameters to obtain effective UAPs. The δ is set to 10, following experiments in [229] to ensure the UAP is imperceptible.

For Alg. 4.2, the maximum iteration number is $m = 500$. The learning rate (lr) is $lr = 0.1$, and the optimizer is Adam (for detection) with $beta = (0.5, 0.9)$. The hyperparameters to train clean and backdoored models are: batch size=96, lr=0.01, epoch=50, poison percent=0.01. Hyperparameters not mentioned are the default ones from TrojanZoo [257].

Evaluation. Following the previous work in [254], we designed two metrics to evaluate the defense performance: model detection and target class detection. We check whether a model is correctly identified as a clean or backdoored model. Then, for backdoored models,

Model	Acc.	ASR	Method	Reversed Trigger	Model Detection		Target Class Detection		
				L_1 norm	Clean	Backdoored	Correct	Correct Set	Wrong
Backdoored (20×20 trigger)	70.94	76.67	NC	276.78	0	15	14	1	0
			TABOR	271.83	0	15	12	2	1
			USB	461.32	0	15	14	1	0
Backdoored (25×25 trigger)	69.7	78.46	NC	347.48	0	15	13	2	0
			TABOR	341.47	2	13	13	0	0
			USB	547.56	0	15	15	0	0
Backdoored 	70.91	80.02	NC	396.72	1	14	14	0	0
			TABOR	406.1	3	12	12	0	0
			USB	621.0	1	14	14	0	0

Table 4.2: Detection evaluation on ImageNet where each case consists of 15 trained models. The apple is a backdoor trigger.

we check whether reverse engineering correctly identifies the target class. In Tab. 4.1, 4.2, and 4.3, *Clean* and *Backdoored* under *Model Detection* refer to the cases whether the detection identifies a model as clean or backdoored. For *Target Class Detection*, we have three categories: (i) *Correct* means the detection method identifies the true target class of a backdoored model, (ii) *Correct Set* refers to the case where the detection method identifies multiple backdoors on different classes, including the true target class, and (iii) *Wrong* refers to the case where the detection method successfully identifies a backdoored model but with wrong target class(es).

Model	Acc.	ASR	Method	Reversed Trigger	Model Detection		Target Class Detection		
				L_1 norm	Clean	Backdoored	Correct	Correct Set	Wrong
Clean	91.59	N/A	NC	40.78	15	0	N/A	N/A	N/A
			TABOR	48.53	14	1	0	0	1
			USB	47.53	15	0	N/A	N/A	N/A
Latent Backdoor (4×4 trigger)	87.20	99.66	NC	19.71	4	11	10	1	0
			TABOR	20.68	4	11	11	0	0
			USB	12.37	1	14	13	1	0
Input Aware Dynamic (32×32 trigger)	89.46	90.43	NC	0.0	15	0	N/A	N/A	N/A
			TABOR	1.8	15	0	N/A	N/A	N/A
			USB	0.13	0	15	15	0	0

Table 4.3: Detection evaluation by stronger backdoor attacks on VGG-16 trained with CIFAR-10.

4.4.2 Experimental Results

This section considers BadNets only as the attack method. Tab. 4.1 shows the detection results for CIFAR-10 (We also provide results on VGG architecture, Tab. 4.5, and GTSRB dataset, Tab. 4.6, in the appendix). For the backdoored models, USB achieves a higher accuracy (98%) for detecting backdoored models compared to NC (93%) and TABOR (92%). We believe that the misclassifications in NC and TABOR are caused by capturing the class's unique features rather than the trigger, illustrated in Fig. 4.4. We show more reversed triggers in Fig. 4.3 and Fig. 4.6 in the Appendix.

As ImageNet contains a large number of images, it is difficult to train many models on it. Thus, we use a subset of ImageNet, which contains ten classes. Each class has 1301 images. Tab. 4.2 shows the results for detecting backdoors for Efficientnet-B0 [249] trained with the subset of ImageNet [187]. Due to the larger image size and model architecture, we use 500 images for data points X in Alg. 4.1 and 4.2.

4.4.3 Stronger Backdoor Attacks

Tab. 4.3 shows detection results on Latent Backdoor [258] and IAD attack [11]. The trigger size for Latent Backdoor is $4 \times 4 \times 3$. Due to the IAD attack's characteristics, we use $32 \times 32 \times 3$ trigger size (the size of input images). The motivation is to show the generalization of USB under stronger attacks besides BadNets [8], especially since IAD is non-patch-based. IAD also generates different triggers according to different inputs.

According to Tab. 4.3, NC and TABOR show worse performance compared to detection results for BadNets, while USB still precisely detects most of the backdoored models. NC and TABOR do not work under the IAD attack, but USB detects such backdoors with the true target class. The reason is that NC-style methods do not directly optimize the pattern of the trigger. Indeed, they mainly optimize the mask that will be applied to the pattern. Moreover, IAD attacks design subtle triggers with specific features related to inputs, which is more difficult for an optimization procedure from random points.

4.4.4 Time Consumption

NC and TABOR require a large number of iterations to conduct detection. We evaluate the time consumption of NC, TABOR, and USB when conducting detection with Efficientnet-B0 on ImageNet. When detecting backdoored models with 20×20 triggers, the average time consumption (in seconds) for NC, TABOR, and USB are 1,154.02, 2,129.40, and 267.12, respectively. USB requires less time when reverse engineering the potential triggers. Although USB needs to generate targeted UAP, the UAP can be used for different models with similar architectures [229], as observed in our experiments. Thus, we only need to generate it once. Tab. 4.4 shows the details of time consumption compared to NC and TABOR.

4.4.5 Discussion

To explain USB, we analyze triggers reversed from every class using MNIST and a simple CNN architecture (see Appendix 4.7.3 for details) with two convolutional layers and two fully connected layers. We remove the constraint on the mask size to search for as powerful features as possible. We replace $\mathcal{L}$ in Alg. 4.2 by: $\mathcal{L} = \mathcal{L}_{ce}(output, t) - SSIM(x, x')$. Under this setting, we train a backdoored model with BadNet. Then, we conduct reverse engineering for all classes. According to results in Fig. 4.5, the optimization with the loss $\mathcal{L}$ tends to learn unique class features for the clean class and the trigger features for the

Model	Method	GPU Time [m:s] in every class									
		0	1	2	3	4	5	6	7	8	9
Backdoored (20×20 trigger)	NC	23:16	24:18	24:32	23:39	24:48	23:35	23:15	23:34	23:41	24:10
	TABOR	33:54	37:24	34:19	35:51	33:59	36:45	34:23	36:47	35:4	36:23
	USB	4:26	4:26	4:27	4:30	4:26	4:26	4:26	4:26	4:26	4:26
Backdoored (25×25 trigger)	NC	23:35	24:38	25:11	24:3	24:58	24:19	24:29	23:54	24:29	23:6
	TABOR	48:23	47:24	48:48	48:41	48:38	47:41	48:27	49:10	48:48	47:45
	USB	4:44	4:42	4:44	4:44	4:49	4:48	4:48	4:54	4:50	4:45
	NC	19:1	18:22	20:47	18:5	18:48	21:27	18:54	18:30	20:26	17:57
	TABOR	48:52	47:16	49:0	48:39	48:54	47:40	48:34	48:55	48:55	47:50
	USB	4:27	4:26	4:27	4:30	4:26	4:26	4:26	4:26	4:26	4:25

Table 4.4: Running time results of backdoor detection for Efficientnet-B0 [249]. Each result is the average of detection on 15 models.

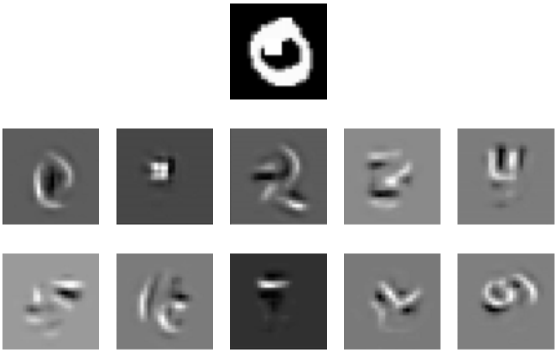


Figure 4.5: USB reverse engineering for 10 classes on MNIST. The true backdoor target is class 1. The first one is the clean image carrying the trigger. The rest are reversed triggers from class 0 to 9.

backdoor class. This is expected as we only have a backdoor on the target class, i.e., class 1. In this simplified situation, for clean classes without backdoors, only the unique class features allow the model to recognize that an input belongs to the class. For the class injected with the backdoor, the model will recognize the input as the backdoor target based on the unique feature of the target and the feature of the backdoor trigger.

Reverse engineering requires a choice between the unique class feature and the feature of the backdoor trigger. Regarding relatively simple features, the trigger feature is stronger than class features when training with poisoned data. Reverse engineering can find a small perturbation with strong features enough to mislead the model according to the learning objectives in the loss function. However, in scenarios such as training with GTSRB or larger datasets, there might be strong features that can generate perturbations with a similar size to backdoor triggers. This is why NC, Tabor, and USB provide more incorrect results when using GTSRB as training data in Tab. 4.6.

4.5 Limitations

Although USB works effectively in detecting backdoors, there are still limitations. First, the optimized triggers are directly related to the data X used by the optimization process. Generating these triggers relies on the gradient when inputting an image from X and trigger to the model. Therefore, if the data X is collected from a different distribution from the training data, existing detection methods, including USB, NC, and TABOR, may fail. This is also why we try to use less data in USB. Second, it might be difficult to generate targeted UAP for every class when the number of classes is high. For example, ImageNet contains 1,000 classes. It is more time-consuming to generate perturbation for inputs of all classes to the target class when the number of classes is large.

This limitation could be the starting point of future work. Currently, our algorithm (Alg. 4.1) only searches for small perturbations to generate targeted UAP. This can mislead the victim model. If we can directly search for targeted UAP according to the backdoored neurons in the model, the number of iterations for searching can be significantly reduced. In addition, reverse engineering naturally requires less data if it has knowledge about backdoor-related neurons, which helps solve the first limitation. A key problem for future work could be identifying those special neurons.

4.6 Conclusions and Future Work

This chapter proposes USB to detect potential backdoors. USB uses targeted UAP to capture sensitive features created by backdoors. We further optimize the UAP to generate the backdoor trigger. We run extensive experiments on several datasets to evaluate our method. Among the 175 backdoored models on several datasets, we successfully identified 171 backdoored ones and outperformed state-of-the-art methods. Further investigation is needed regarding optimizing UAP according to backdoored neurons in the victim model, which can significantly reduce the optimization time.

4.7 Appendix

4.7.1 Detection Results on VGG-16

In Tab. 4.5, we show the results of detecting backdoors for VGG-16 models trained with CIFAR-10. We use the same experimental settings as that in the experiment section. We also study Latent Backdoor [258] beside BadNet attack.

4.7.2 GTSRB

The results for GTSRB are shown in Tab. 4.6. On clean models, USB, NC, and TABOR all have incorrect results, as the number of classes in GTSRB is significantly larger than

Model	Acc.	ASR	Method	Reversed Trigger	Model Detection		Target Class Detection		
				L_1 norm	Clean	Backdoored	Correct	Correct Set	Wrong
Clean	91.59	N/A	NC	40.78	15	0	N/A	N/A	N/A
			TABOR	48.53	14	1	0	0	1
			USB	47.53	15	0	N/A	N/A	N/A
Backdoored (2×2 trigger)	88.28	99.39	NC	5.43	0	15	14	1	0
			TABOR	5.32	0	15	15	0	0
			USB	3.5	0	15	15	0	0
Backdoored (3×3 trigger)	88.30	99.77	NC	6.60	1	14	13	1	0
			TABOR	6.98	0	15	14	1	0
			USB	7.0	0	15	14	1	0

Table 4.5: Detection evaluation on VGG-16 trained with CIFAR-10 where each case consists of 15 trained models.

Model	Acc.	ASR	Method	Reversed Trigger	Model Detection		Target Class Detection		
				L_1 norm	Clean	Backdoored	Correct	Correct Set	Wrong
Clean	83.96	N/A	NC	181.17	12	3	N/A	N/A	N/A
			TABOR	185.21	13	2	N/A	N/A	N/A
			USB	39.8	12	3	N/A	N/A	N/A
Backdoored (2×2 trigger)	80.85	85.06	NC	13.36	0	15	13	2	0
			TABOR	37.02	0	15	13	2	0
			USB	10.86	3	12	12	0	0
Backdoored (3×3 trigger)	80.24	93.52	NC	14.78	0	15	13	2	0
			TABOR	15.11	0	15	13	2	0
			USB	12.02	2	13	13	0	0

Table 4.6: Detection evaluation on ResNet-18 trained with GTSRB where each case consists of 15 trained models.

that of CIFAR-10. Compared to the L_1 norm of NC and TABOR, USB provides a much smaller norm value because the reversed trigger is optimized from the targeted UAP.

4.7.3 Details of the Basic Model

To reduce the impact of complex features and many model parameters, we use MNIST and a basic CNN architecture with two convolutional layers (followed by the ReLU activation function and a 2D average pooling layer) and two fully connected layers. The input channel, output channel, and kernel size for the two convolutional layers are (1, 16, 5) and (16, 32, 5). The input and output channels for the two fully connected layers are (512, 512) and (512, 10). The model is trained using batch size=128, epochs=40, and poisoned rate=0.05.

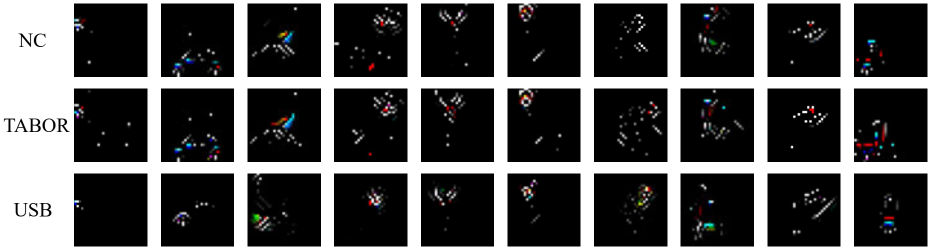


Figure 4.6: Reversed triggers from class 0 to 9.

Chapter 5

Adversarial Neuron Noise for Backdoor Detection

Backdoor attacks on deep learning represent a recent threat that has gained significant attention in the research community. Backdoor defenses are mainly based on backdoor inversion, which has been shown to be generic, model-agnostic, and applicable to practical threat scenarios. State-of-the-art backdoor inversion recovers a mask in the feature space to locate prominent backdoor features, where benign and backdoor features can be disentangled. However, it suffers from high computational overhead, and we also find that it overly relies on prominent backdoor features that are highly distinguishable from benign features. To tackle these shortcomings, this chapter improves backdoor feature inversion for backdoor detection by incorporating extra neuron activation information. In particular, we adversarially increase the loss of backdoored models with respect to weights to activate the backdoor effect, based on which we can easily differentiate backdoored and clean models. Experimental results demonstrate our defense, BAN, is 1.37× (on CIFAR-10) and 5.11× (on ImageNet200) more efficient with an average 9.99% higher detect success rate than the state-of-the-art defense BTI-DBF. Our code and trained models are publicly available at `https://github.com/xiaoyunxxy/ban`.

5.1 Introduction

Deep neural networks (DNNs) are known to be vulnerable to backdoor attacks, a setting where the attacker trains a malicious DNN to perform well on normal inputs but behave inconsistently when receiving malicious inputs that are stamped with triggers [8]. The malicious model is obtained by training with poisoned data [8, 9], by tampering with the training process [259, 104], or by directly altering the model's weights [51]. Backdoors are proposed for various domains, from computer vision [8, 259, 9, 10, 105, 11] to graph data [260] and neuromorphic data [261]. Still, backdoors in computer vision received most of the attention of the research community, which also means there is a significant variety of triggers. For instance, a trigger can be a pixel patch [8], an existing image [259], dynamic perturbation [11], or image warping [10]. Various backdoor attacks target different stages of the machine learning model pipeline, inducing several threat scenarios in which defenses are developed by exploiting different knowledge.

Trigger inversion is a principled method that makes minimal assumptions about the backdoor attacks in the threat model [111, 128, 112], and it has clear advantages to training-time [136] or run-time defenses [117, 262, 263]. *Input space* trigger inversion is first introduced by Neural Cleanse (NC) [110], where potential triggers for all target classes are reversed by minimizing the model loss of clean inputs. Median absolute deviations of reversed triggers from all classes are then calculated to detect triggers. More recently, input space trigger inversion methods have been shown to be ineffective against *feature space* backdoor attacks [10, 111]. To address this problem, FeatureRE [111] proposes a detection method using feature space triggers. The authors observe that features of backdoored and benign inputs are separable in the feature space by hyperplanes. Unicorn [128] formally defines and analyzes different spaces for trigger inversion problems. An invertible input space transformation function is proposed to map the input space to others, such as feature space, and to reconstruct the backdoor triggers. These trigger inversion methods can be considered as an optimization problem for the targeted class. They optimize the input images to mislead the model under various constraints and recover strong triggers in different spaces. The state-of-the-art trigger inversion method, BTI-DBF [112], increases the inversion efficiency by relaxing the dependency of trigger inversion on the target labels. BTI-DBF trains a trigger generator by minimizing the differences between benign samples and their generated poisoned version in decoupled benign features while maximizing the differences in remaining backdoor features. Feature space trigger inversion defenses are generic and effective against most backdoor attacks. However, we show that BTI-DBF may fail against BadNets, as its backdoor is not prominent in the feature space (see Section 5.3.4). In addition, existing works still suffer from huge computational overhead and overly rely on *prominent backdoor features*.

To resolve this shortcoming, we propose a backdoor defense called detecting **B**ackdoors activated by **A**dversarial neuron **N**oise (BAN). Our defense is inspired by the finding that backdoored models are more sensitive to adversarial noise than benign models [264, 265], and neuron noise can be adversarially manipulated to indicate backdoor activations [130]. Specifically, BAN generates adversarial neuron noise, where the weights of the victim model are adversarially perturbed to maximize the classification loss on a clean data set. Simultaneously, trigger inversion is conducted in the victim model's feature space to calculate the mask for benign and backdoor features. Clean inputs with masked feature maps are then fed to the adversarially perturbed model, based on the outputs of which backdoored models can be differentiated. Figure 5.1 presents a t-SNE visualization of the

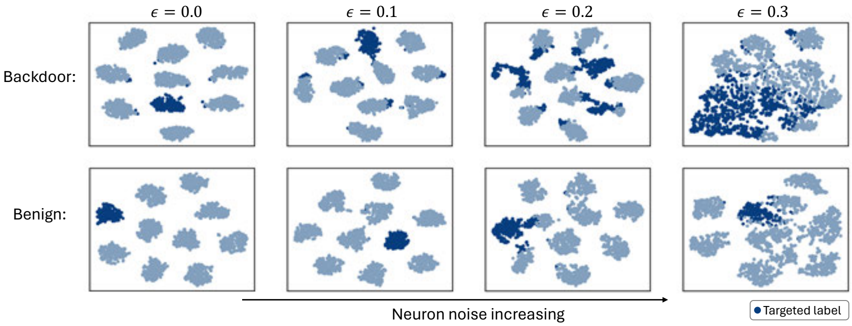


Figure 5.1: The feature plots of backdoor and benign models with neuron noise using ResNet18 on CIFAR-10. The darker blue represents the target label. As noise increases, the backdoor model identifies more inputs from each class as the target label. The clean model has fewer errors, and there is no significant increase in the number of misclassifications to the target class.

feature space for backdoored and clean models when they are perturbed with different levels of adversarial neuron noise. It can be observed that the backdoored model misclassifies parts of the data from all classes as the target class when the adversarial neuron noise increases, while the clean model has fewer misclassifications under the same level of noise. By leveraging the induced difference, BAN can successfully detect and mitigate backdoor attacks in both input and feature space.

We make the following contributions:

- We find a generalization shortcoming in current trigger inversion-based detection methods (FeatureRE [111], Unicorn [128], BTI-DBF [112]). In particular, feature space detections overly rely on highly distinguishable prominent backdoor features. We provide an in-depth analysis of trigger inversion-based backdoor defenses, showing that prominent backdoor features that are exploited by state-of-the-art defenses to distinguish feature space backdoors may not be suitable for the identification of input space backdoors.
- We propose detecting Backdoors activated by Adversarial neuron Noise (BAN) to mitigate this generalization shortcoming by introducing neuron noise into feature space trigger inversion. BAN includes an adversarial learning process to incorporate neuron activation information into the inversion-based backdoor detection. Experimental results demonstrate that BAN is 1.37× (on CIFAR-10) and 5.11× (on ImageNet200) more efficient with a 9.99% higher detect success rate than the state-of-the-art defense BTI-DBF [112].
- We also exploit the neuron noise to further design a simple yet effective defense for removing the backdoor, such that we build a workable framework.

5.2 Related Work

Attacks. Backdoor attacks [8, 259, 104, 10, 11, 105, 266, 267, 268, 269] refer to injecting a secret functionality into the victim model that is activated through malicious inputs that contain the trigger. To this end, substantial research has been proposed by poisoning a small percentage of training data using small and static triggers [8, 9, 259]. Early attacks generate backdoors in input space, where BadNets [8] is the first backdoor attack in DNNs. Blend [9] proposed three injection strategies to blend translucent images into inputs of DNNs as triggers. The authors controlled the transparency of the trigger to allow the trade-off between strength of attack and invisibility. Although these attacks work well, their triggers are still perceptible to humans and can be easily detected by backdoor defenses, such as Activation Clustering (AC) [109] and NC [110].

Dynamic and imperceptible triggers [10, 11, 105, 267], including feature space backdoor triggers [268, 11], are explored to bypass both human observers and input space defenses. IAD [11] designs input-specific triggers. To evaluate the uniqueness of dynamic triggers, the authors designed a cross-trigger test to determine whether the trigger of one input is reusable to others. WaNet [10] proposes warping-based triggers, which are unnoticeable and smooth in the input space. Bpp [105] exploits vulnerabilities in the human visual system and, by using image quantization and dithering, introduces invisible triggers that are stealthier than previous attacks.

Adaptive backdoor attacks [103, 118, 117] are built to systematically evaluate defenses, where attacks discourage the indistinguishability of latent representations of poisoned and benign inputs in the feature space. Adap-Blend [118] divides the trigger image into 16 pieces and randomly applies only 50% of the trigger during data poisoning. They use the full trigger image at inference time to mislead the poisoned model. Source-Specific and Dynamic-Triggers (SSDT) [117] considers the combination of source-specific and dynamic-trigger backdoors. Only the inputs from victim classes (source) with the dynamic triggers are classified to the target labels, which encourages the generalization of more diverse trigger patterns from different inputs. In this chapter, we evaluate BAN against various types of attacks, including input space, feature space, and adaptive attacks, to provide a systematic evaluation resembling practical threats.

Defenses. Trigger inversion-based backdoor defense is considered one of the most practical and general defenses against backdoors [111, 128, 112]. The recovered trigger is used to determine whether the model is backdoored. For example, NC [110] reverses input space triggers to detect backdoors by selecting significantly smaller triggers in size. Other methods, such as ABS [127] and FeatureRE [111], usually determine whether there is a backdoor based on the attack success rate of the trigger. More specifically, given a DNN model f and a small set of clean data $\mathcal{D}_c = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, NC recovers the potential trigger by solving the following objective:

$$\min_{\mathbf{m},\mathbf{t}} \mathcal{L}(f((1-\mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \mathbf{t}), y_t) + \lambda|\mathbf{m}|, \tag{5.1}$$

where $(\mathbf{x}, y) \in \mathcal{D}_c$ and $\mathbf{m}$ is the trigger mask, $\mathbf{t}$ is the trigger pattern, where $\odot$ is the element-wise product, and y_t is the target label. The mask $\mathbf{m}$ determines whether the pattern will replace the pixel. $\mathcal{L}$ is the cross-entropy loss function. Most prior works [144, 270, 127, 111, 128] follow this design to conduct trigger inversion for all possible target labels. For example, Tabor [144] adds more constraints to the NC optimization problem according to the overly large (large size triggers but no overlaps with the true trigger) and overlaying (overlap the true trigger but with redundant noise) problem of NC. Moving from input space to feature, FeatureRE [111] utilizes feature space constraint according to an observation that neuron activation values representing the backdoor behavior are orthogonal to others. Unicorn [128] proposes a transformation function that projects from the input space to other spaces, such as feature space [10] and numerical space [105]. Then, the authors conduct trigger inversion in different spaces.

Unlike previous NC-style methods, recent works [112, 271] explore different optimization objectives to avoid the time-consuming optimization for all possible target classes or to avoid the fixed mask-pattern design as advanced attacks utilize more complex and dynamic triggers. BTI-DBF [112] takes advantage of the prior knowledge that benign and backdoored features are decoupled in the feature space. It distinguishes them by the optimization objective, where benign features contribute to the correct predictions and backdoored features lead to wrong predictions. Based on the decoupled features, BTI-DBF trains a trigger generator by minimizing the difference between benign samples and their generated version according to the benign features and maximizing the difference according to the backdoored features. Feature space backdoor defenses are developed based on the fact that backdoor features are highly distinguishable from benign features. However, this finding is not consistently valid for input space attacks where the feature difference is small (See Sections 5.3.4).

5.3 BAN Method

5.3.1 The Pipeline of Training Backdoor Models

For brevity, consider an L-layer fully connected network f (similar principles apply to convolutional networks) that has $l = l_1 + l_2 + \cdots + l_L$ neurons. $\mathbf{x}_n \in \mathbb{R}^{d_x}$ and $y_n \in \{0, 1\}^{d_y}$ are the n_{th} image and its label in d_x and d_y dimensional spaces, respectively. The attacker creates a poisoned dataset $\mathcal{D}_p$ by poisoning generators G_X and G_Y for a subset of the whole training dataset, i.e., $\mathcal{D}_p = \mathcal{D}_c \cup \mathcal{D}_b$. $\mathcal{D}_c$ is the original clean data. $\mathcal{D}_b$ is the poisoned backdoor data, $\mathcal{D}_b = \{(\mathbf{x}', y')|\mathbf{x}' = G_X(\mathbf{x}), y' = G_Y(y), (\mathbf{x}, y) \in \mathcal{D} - \mathcal{D}_c\}$. In all-to-one attacks, $G_Y(y) = y_t$, y_t is the attacker-specified target class. In our experiments, we consider the dirty-label attack. In all-to-all attacks, ususally $G_Y(y) = (y+1)$ [112, 10, 105], which is also what we chose in our experiments. In the training stage, the backdoor is injected into the model by training with $\mathcal{D}_p$, i.e., minimizing the training loss on $\mathcal{D}_p$ to find the optimal weights and bias $(\mathbf{w}^*, \mathbf{b}^*)$:

$$\min_{\mathbf{w},\mathbf{b}} \mathcal{L}_{\mathcal{D}_p}(\mathbf{w}, \mathbf{b}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\in\mathcal{D}_p} \ell(f(\mathbf{x}; \mathbf{w}, \mathbf{b}), y), \tag{5.2}$$

where $\ell(\cdot, \cdot)$ is the cross-entropy. In the inference stage, the backdoored model predicts an unseen input $\hat{\mathbf{x}}$ as its true label $\hat{y}$ but predicts $G_X(\hat{\mathbf{x}})$ as $G_Y(\hat{y})$: $f(G_X(\hat{\mathbf{x}}); \mathbf{w}, \mathbf{b}) = G_Y(\hat{y})$.

5.3.2 Threat Model

Attacker's goal. We consider the attacker to be the pre-trained model's provider. The attacker aims to inject stealthy backdoors into the pre-trained models. The pre-trained models perform well on clean inputs but predict the attacker-chosen target label when receiving backdoor inputs.

Attacker knowledge. The attacker has white-box access to the model, including training data, architectures, hyperparameters, and model weights.

Defender's goal and knowledge. The main goal is to detect whether a given model is backdoored and then remove the potential backdoor according to the detection results. Following [111, 128, 112], we assume the defender has white-box access to the model and holds a few local clean samples. However, the defender does not have access to the training data and has no knowledge of the backdoor trigger.

5.3.3 Detection with Neuron Noise

Based on previous findings that adversarial noise can activate backdoors [264, 130, 265], we design a two-step method for backdoor detection. First, we search for noise on neurons that can activate the potential backdoor. Then, we decouple the benign and backdoored features using a learnable mask of the latent feature (the output before the final classification layer).

Neuron Noise. In backdoor models, there are two types of neurons: benign and backdoor [127]. The backdoor neurons build a shortcut from $G_X(\mathbf{x})$ to $G_Y(y)$. *Neuron noise* is generated noise added to neurons to maximize model loss on clean samples in an adversarial manner [12]. The noise on benign neurons evenly misleads prediction to all classes, while the noise on backdoor neurons tends to mislead prediction to the target label due to the backdoor shortcut, as shown in Figure 5.1. Therefore, backdoor models with noise behave differently from benign models with noise, as there are no backdoor neurons in the benign models.

Given the j_{th} neuron connecting the i_{th} layer and $(i-1)_{\text{th}}$ layer, we denote its weight and bias with $\mathbf{w}_{ij}$ and b_{ij}, respectively. Neuron noise can be added to the neuron by multiplying its weight and bias with a small number: $(1+\delta_{ij})\mathbf{w}_{ij}$ and $(1+\xi_{ij})b_{ij}$. Then, the output of the neuron is:

$$h_{ij} = \sigma\Big((1+\delta_{ij})\mathbf{w}_{ij}^{\top}\mathbf{h}_{(i-1)} + (1+\xi_{ij})b_{ij}\Big), \tag{5.3}$$

where $\mathbf{h}_{(i-1)}$ is the output of the previous layer and $\sigma(\cdot)$ is the nonlinear function (activation). The noise on all neurons are represented by $\boldsymbol{\delta} = [\delta_{1,1}, \cdots, \delta_{l_1,1}, \cdots, \delta_{1,L}, \cdots, \delta_{l_L,L}]$ and $\boldsymbol{\xi} = [\xi_{1,1}, \cdots, \xi_{l_1,1}, \cdots, \xi_{1,L}, \cdots, \xi_{l_L,L}]$. The $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$ are optimized via a maximization to increase the cross-entropy loss on the clean data:

$$\begin{gathered}\max_{\boldsymbol{\delta},\boldsymbol{\xi}\in S} \mathcal{L}_{\mathcal{D}_c}((1+\boldsymbol{\delta})\odot\mathbf{w}, (1+\boldsymbol{\xi})\odot\mathbf{b}), \\ S = B(\mathbf{w};\epsilon) = \{\delta \in \mathbb{R}^l | \delta \leq \epsilon\},\end{gathered} \tag{5.4}$$

where S is the ball function of the radius ϵ in the l dimensional space. $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$ share the ball function S as the maximum noise size. The maximization in Eq. (5.4) can be solved by PGD [12] algorithm with a random start, as PGD can better explore the entire searching space to mitigate local minima [12].

Feature Decoupling with Mask. The neuron noise activates the backdoor (see Figure 5.1) and misleads the predictions to the target label. However, the performance has high variance when searching for noise multiple times, which we conjecture is caused by random initialization. Therefore, inspired by [112], we further introduce a feature decoupling process to enhance the effect of noise on backdoored features but maintain a decreased effect on benign features. Specifically, the network f is decomposed into $g = f_1 \circ \cdots \circ f_{L-1}$ and f_L, where f_{L-1} extracts the latent features from $\mathbf{x}$, while f_L is the classification layer. Then, we use a mask $\mathbf{m}$ on top of the latent feature $g(\mathbf{x})$ to decouple benign and backdoored features. The optimization objective can be written as:

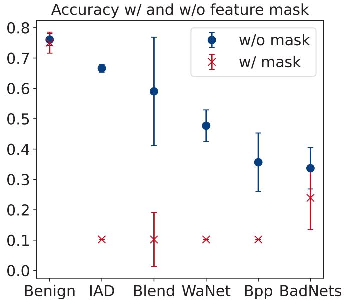


Figure 5.2: Model's clean accuracy with (red dots) and without (blue dots) the mask defined in Eq. (5.6). Only the backdoored models are affected by the noise.

$$\min_{\mathbf{m}} \mathcal{L}(f_L \circ (g(\mathbf{x})\odot\mathbf{m}), y) - \mathcal{L}(f_L \circ (g(\mathbf{x})\odot(1-\mathbf{m})), y) + \lambda_1|\mathbf{m}|, \tag{5.5}$$

where $\odot$ is the element-wise product. This optimization divides the latent features into two parts through the mask while maintaining a relatively small size of $\mathbf{m}$. Note that the regularizer for $\mathbf{m}$ in Eq. (5.5) is necessary. Otherwise, $\mathbf{m}$ will become a dense matrix full of ones to focus on the positive part of Eq. (5.5) because maintaining only the positive part (without penalty of $|\mathbf{m}|$) already satisfies the optimization objective. Finally, we apply the negative mask $(1 - \mathbf{m})$ on top of latent feature of f to enhance the backdoor effect. The final output is:

$$f_L \circ \Big(g\big(\mathbf{x}; (1 + \boldsymbol{\delta}) \odot \mathbf{w}, (1 + \boldsymbol{\xi}) \odot \mathbf{b}\big) \odot (1 - \mathbf{m})\Big). \tag{5.6}$$

In Figure 5.2, the blue dots show the accuracy after adding noise to the model. The red dots show the accuracy (with noise) while applying the feature mask to the model using Eq. (5.6). Our feature masks do not affect the performance of benign models. However, we see that the performance is significantly decreased for backdoored models. This decrease is caused by the model only using backdoor features (through the negative mask), which means the backdoor is activated more frequently. Finally, a suspect model is determined backdoored if the prediction using Eq. (5.6) has a high attack success rate. For all-to-one attacks, the misclassification will be concentrated into one label, which is the target label. For all-to-all attacks, we can do the same as for all-to-one attacks but evaluate the prediction for each class independently.

5.3.4 Improving BTI-DBF

This section shows that the most recent work, BTI-DBF [112], may fail to capture the backdoor features by its decoupling method in Table 5.2. We show how to patch BTI-DBF using a simple solution. The BTI-DBF is the original version, and BTI-DBF* is our improved version. The main pipeline of BTI-DBF consists of two steps: (1) decoupling benign and backdoor features and (2) trigger inversion by minimizing the distance between benign features and maximizing the distance between backdoor features. We found that the defense's first step may introduce errors in the decoupled features. Similar to our Eq. (5.5), the decoupling of BTI-DBF can be written as:

$$\min_{\mathbf{m}} \mathcal{L}(f_L \circ (g(\mathbf{x}) \odot \mathbf{m}), y) - \mathcal{L}(f_L \circ (g(\mathbf{x}) \odot (1 - \mathbf{m})), y). \tag{5.7}$$

However, this equation has no constraints on the feature mask $\mathbf{m}$. Overall, the optimization objective is to decrease the loss. Obviously, BTI-DBF's decoupling encourages the norm of the mask to increase so that the loss will focus on the positive part and ignore the negative part because the negative part goes against the overall objective. Finally, the mask will be a dense matrix that is full of ones. The $f_L \circ (g(\mathbf{x}) \odot (1 - \mathbf{m}))$ is ignored due to multiplying by zero. We propose a simple solution to fix the problem by adding a regularizer of the size of the mask to the loss, i.e., we use our Eq. (5.5) as the first step of BTI-DBF*. According to Table 5.2, BTI-DBF* successfully overcomes BTI-DBF's shortcomings.

5.3.5 Backdoor Defense

After determining whether a suspect model is backdoored, we can fine-tune the backdoored model to remove the backdoor. However, standard fine-tuning using clean data does not effectively remove the backdoor because it does not activate it. Therefore, we propose using optimized neuron noise to fine-tune the model. In the optimization of the neuron noise, the objective is to increase the loss of $\mathcal{L}(f(\mathbf{x}), y)$. We consider both the benign and backdoor neurons to contribute to the increase in loss when optimizing the noise. Indeed, on benign neurons, the neuron noise misleads f to any result other than the true label y, while the noise misleads f to the target label $G_Y(y)$ on backdoor neurons. Therefore, a straightforward method is to decrease the loss between the noise output and the true label. The loss for our noise fine-tuning can be written as:

$$\min_{\mathbf{w},\mathbf{b}} \mathcal{L}(f(\mathbf{x};\mathbf{w},\mathbf{b}), y) + \lambda_2 \mathcal{L}(f(\mathbf{x};(1+\boldsymbol{\delta})\odot\mathbf{w},(1+\boldsymbol{\xi})\odot\mathbf{b}), y). \tag{5.8}$$

5.4 Experimental Results

Datasets and Architectures. The datasets for our experiments include CIFAR-10 [250], GTSRB [272], Tiny-ImageNet [273], and a subset of ImageNet-1K [187]. The ImageNet subset contains 200 classes, which is referred to as ImageNet200. BAN is evaluated using three architectures: ResNet18 [27], VGG16 [26], and DenseNet121 [274]. Please refer to Appendix 5.7.1 for more details.

Attack Baselines. Our experiments are conducted using seven attacks: BadNets [8], Blend [9], WaNet [10], IAD [11], BppAttack [105], Adap-Blend [118], and SSDT [117], which are commonly used in other works [112, 111, 128, 275, 276]. The BadNets [8] and Blend [9] are designed for the input space. WaNet [10], IAD [11], and BppAttack [105] are designed for feature space. Adap-Blend [118] and SSDT [117] (for adaptive evaluation) are state-of-the-art attacks that have been recently introduced to bypass backdoor defenses. The main idea of Adap-Blend [118] and SSDT [117] is to obscure the latent separation in features of benign and backdoor samples. More details can be found in Appendix 5.7.2.

Defense Baselines. BAN is compared to five representative methods: Neural Cleanse (NC) [110], Tabor [144], FeatureRE [111], Unicorn [128], and BTI-DBF [112]. NC and Tabor are designed for input space attacks, while the other three are designed for feature space and dealing with the latest advanced attacks. BAN uses only 1% and 5% of training data for detection and defense, respectively. Note that the data used for our defense is not used for training models, i.e., the defender has no knowledge of the model to be defended. More details can be found in Appendices 5.7.3 and 5.7.4. We use bold font to denote the best results.

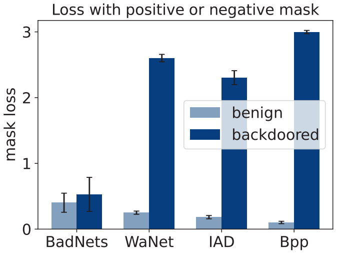


Figure 5.3: BadNets features are weaker when using the mask to disentangle the benign and backdoor features. Defenses that are biased towards large differences may not work in cases like BadNets.

Table 5.1: The detection results under different model architectures on CIFAR-10. The "Bd." refers to the number of models the defense identifies as backdoored. The "Acc." refers to detection success accuracy. The best results are marked in bold. BTI-DBF* refers to an improved version (details in Section 5.3.4).

Model	Attack	NC		Tabor		FeatureRE		Unicorn		BTI-DBF*		Ours	
		Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
ResNet18	No Attack	0	**100%**	0	**100%**	2	90%	6	70%	0	**100%**	0	**100%**
	BadNets	20	**100%**	20	**100%**	14	70%	18	90%	18	90%	20	**100%**
	Blend	20	**100%**	20	**100%**	20	**100%**	19	95%	20	**100%**	18	90%
	WaNet	11	55%	8	40%	15	75%	20	**100%**	18	90%	20	**100%**
	IAD	0	0%	0	0%	15	75%	11	55%	20	**100%**	20	**100%**
	Bpp	0	0%	1	5%	12	60%	17	85 %	20	**100%**	20	**100%**
VGG16	No Attack	0	**100%**	0	**100%**	3	85%	6	70%	6	70%	0	**100%**
	BadNets	18	90%	16	80%	13	65%	16	80%	18	90%	19	**95%**
	Blend	19	**95%**	19	**95%**	16	80%	18	90%	16	80%	17	85%
	WaNet	10	50%	9	45%	12	60%	18	90%	16	80%	20	**100%**
	IAD	0	0%	0	0%	8	40%	17	85%	20	**100%**	20	**100%**
	Bpp	9	45%	10	50%	5	25%	15	75%	14	70%	18	**90%**
DenseNet121	No Attack	0	**100%**	0	**100%**	5	75%	8	60%	3	85%	0	**100%**
	BadNets	18	90%	20	**100%**	19	95%	15	75%	17	85%	20	**100%**
	Blend	20	**100%**	20	**100%**	12	60%	18	90%	19	95%	20	**100%**
	WaNet	13	65%	10	50%	20	**100%**	17	85%	14	70%	19	95%
	IAD	0	0%	0	0%	14	70%	16	80%	14	70%	19	**95%**
	Bpp	0	0%	0	0%	16	80%	8	40%	16	80%	20	**100%**
Average		60.56%		59.17%		72.5%		78.61%		86.39%		**97.22%**	

5.4.1 The Performance of Backdoor Detection

Main Results. In Table 5.1, BAN shows better results on CIFAR-10 than all baselines, especially on advanced attacks. Results on other datasets are presented in Appendix 5.7.5. We note that the advanced detection methods (FeatureRE, Unicorn, and BTI-DBF) perform worse than NC on simple attacks (BadNets and Blend). We hypothesize this is because the backdoor features generated by BadNets are less obvious on feature channels than advanced attacks in the feature space, such as WaNet and IAD. Figure 5.3 shows that BadNets features are weaker than features from advanced attacks using the feature mask in Eq. (5.5). Specifically, we optimize the feature mask to disentangle the benign and backdoor features for four attacks. Then, we compute two cross-entropy loss values using the positive mask ($\mathbf{m}$) and the negative mask ($1 - \mathbf{m}$) for benign and backdoor features, respectively. The average loss values and standard deviation for four models for each attack are plotted in Figure 5.3. The negative loss value of BadNets is much smaller than others, which means BadNets features are weaker than others with regard to misleading the model to the backdoor target. Recent defenses usually add more regularizers to their losses and optimization objectives to counteract powerful backdoor attacks. These regularizers encourage the capturing of strong features but omit weak ones. Thus, recent advanced detections can perform worse on BadNets than NC.

Time consumption. BAN is efficient and scalable as we do not iterate over all target classes. Figure 5.4 demonstrates that BAN uses substantially less time than all baselines.

BAN is also more scalable for larger architectures or datasets. BIT-DBF (76.90s) is 1.37× slower than BAN (55.95s) on CIFAR-10 with ResNet18, and BIT-DBF (5,792.51s) without pre-training is 5.11× slower than ours (1,132.85s) on ImageNet200 with ResNet18. FeatureRE (297.74s) is 5.32× slower than ours on CIFAR-10 with ResNet18, and it (6,053.62s) is 45.14× slower on CIFAR-10 with DenseNet121 than ours (134.10s).

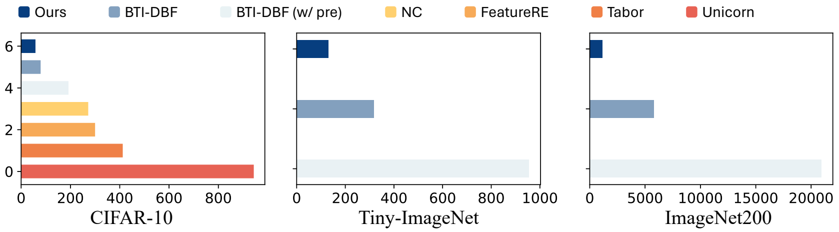


Figure 5.4: Time consumption of detection baselines on ResNet18 (in seconds) for all three datasets. BAN uses significantly less time than the baselines.

Table 5.2: The detection results of BTI-DBF and BTI-DBF* using ResNet18 and CIFAR-10.

Attack	BTI-DBF		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
No Attack	0	**100%**	0	**100%**	0	**100%**
BadNets	5	25%	18	90%	20	**100%**
Blend	0	0%	20	**100%**	18	90%
WaNet	7	35%	18	90%	20	**100%**
IAD	19	95%	20	**100%**	20	**100%**
Bpp	20	**100%**	20	**100%**	20	**100%**

5.4.2 The Performance of Backdoor Defense

A complete backdoor defense framework should include both detection and defense. The goal of defense is to decrease the attack success rate (ASR) of backdoor triggers. Detection before the defense is also necessary because the defense usually decreases the performance on benign inputs [277, 112, 111, 275, 278]. In Section 5.3.5, we propose a simple and effective fine-tuning method using the noise that activates the backdoor. Table 5.3 compares BAN with three baselines: plain fine-tuning, FeatureRE [111], and BTI-DBF(U) [112]. Plain fine-tuning refers to training the backdoor model using the same hyperparameters as BAN but without the noise loss in Eq. (5.8). The FeatureRE [111] and BTI-DBF(U) [112] refer to the defense methods from the respective paper. We use default hyperparameters for FeatureRE [111] and BTI-DBF(U) [112]. For plain fine-tuning and BAN, we use a small learning rate (0.005) to avoid jumping out of the current optimal parameters of the well-trained model. Then, we use λ_2 (0.5) for Eq. (5.8) for the trade-off between robustness against backdoor and clean accuracy. We fine-tune for a short schedule of 25 epochs, as the model is well-trained. Table 5.8 in Appendix 5.7.4 shows defense performance with standard deviation on different hyperparameters, which supports our choice. Tables 5.3

Table 5.3: Defense against 5 attacks using ResNet18. BA refers to benign accuracy on clean data.

Attack	No defense		Fine-tuning		FeatureRE		BTI-DBF(U)		Ours	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets	93.37	99.41	92.93	87.81	**93.15**	99.79	91.26	13.12	92.06	**1.97**
Blend	94.60	100.00	93.07	99.99	**93.20**	39.28	91.86	100.00	92.72	**4.10**
WaNet	93.57	99.37	93.05	1.10	**93.67**	**0.03**	90.30	4.89	92.05	0.91
IAD	93.17	97.88	**94.11**	0.46	92.73	**0.0**	89.54	1.59	92.78	1.48
Bpp	94.29	99.93	93.85	4.46	**94.21**	98.13	90.61	2.73	92.54	**2.58**
Average	93.80	99.32	**93.40**	38.76	93.39	47.45	90.71	24.47	92.43	**2.21**

Table 5.4: Defense of backdoor attacks on Tiny-ImageNet and ImageNet200 using ResNet18.

Dataset	Attack	No defense		Fine-tuning		BTI-DBF(U)		Ours	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR
Tiny-ImageNet	WaNet	58.32	99.85	**51.53**	1.3	39.49	0.96	50.69	**0.86**
	IAD	58.54	99.32	**51.86**	1.72	38.79	**0.60**	50.04	0.76
	Bpp	60.63	99.89	**57.72**	0.15	46.84	0.40	57.66	**0.10**
ImageNet200	WaNet	77.01	99.74	66.71	0.78	63.47	1.0	**69.95**	**0.58**
	IAD	76.72	99.75	69.91	**0.42**	64.33	1.24	**72.18**	1.30
	Bpp	78.56	99.88	70.89	**0.82**	67.02	3.10	**72.59**	2.68

and 5.4 demonstrate that our fine-tuning method effectively removes the backdoor while preserving high accuracy in benign inputs. We provide a comparison between our find-tuning with ANP [130] in Table 5.12, Appendix 5.7.5, as ANP uses the neuron noise for pruning backdoor neurons.

5.4.3 Defense against All-To-All Attacks

Previous works [111, 128] usually only consider all-to-one attacks, which limits the application in practical situations. In this section, we evaluate our fine-tuning method under three all-to-all attacks: WaNet-All [10], IAD-All [11], and Bpp-All [105]. FeatureRE is designed for all-to-one attacks, so we use target label 0 here for FeatureRE. FeatureRE is included to show that the all-to-one defense does not work in the all-to-all setting. BAN is capable of handling all-to-all attacks because we directly explore the neurons themselves instead of optimizing for the potential targeted label. Table 5.5 demonstrates the effectiveness of our method.

Table 5.5: Defense against all-to-all backdoor attacks. BA refers to benign accuracy on clean data.

Attack	No defense		Fine-tuning		FeatureRE		BTI-DBF(U)		Ours	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
WaNet-All	93.60	91.86	92.65	18.03	**93.33**	91.96	91.30	1.72	92.29	**1.11**
IAD-All	92.96	90.62	**93.19**	3.72	93.06	91.20	91.36	3.72	92.31	**1.13**
Bpp-All	94.45	84.68	**93.90**	1.58	94.32	83.87	90.16	2.05	93.23	**1.38**
Average	93.67	89.05	93.25	7.78	**93.57**	89.01	90.94	2.49	92.61	**1.21**

5.4.4 Evaluation under Adaptive Attack

Attackers may design a specific attack for defense when they know its details [239]. In this section, we evaluate BAN against two attacks that attempt to bypass the difference between backdoor and benign features: Adap-Blend [118] and SSDT attack [117]. Both Adap-Blend [118] and SSDT attack [117] try to obscure the difference between benign and backdoor latent features. Adap-Blend [118] achieves it by randomly applying 50% of the trigger to poison the training data, while SSDT attack [117] utilizes source-specific and dynamic-triggers to reduce the impact of triggers on samples from non-victim classes. The source-specific attack refers to backdoor triggers that mislead the model to the target class only when applied to victim class samples. Table 5.6 demonstrates our approach is resistant to these two attacks, while other methods fail. The reason is that the backdoor features of SSDT are close to benign features in the feature space. It is difficult for other methods to distinguish between backdoor and benign features created by SSDT. Our detection method directly analyzes the model itself using neuron noise, which captures the difference between backdoor and benign models concerning parameters. See Appendix 5.7.5 for the mitigation results of our method against these adaptive attacks.

Table 5.6: The detection results under adaptive attacks on using CIFAR-10 and ResNet18.

Attack	NC		Tabor		FeatureRE		Unicorn		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
Adap-Patch	18	90%	15	75%	17	85%	20	**100%**	20	**100%**	20	**100%**
SSDT	0	0%	0	0%	0	0%	0	0%	20	**100%**	20	**100%**

5.4.5 Analysis on Prominent Features

We provide additional analysis of the phenomenon that backdoor features are more prominent for advanced attacks (WaNet, IAD, and Bpp) than weaker attacks (BadNets, Blend). Table 5.7 demonstrates that previous decoupling methods cannot easily pick up backdoor features from weak attacks, such as BadNets. In particular, when detecting without the L_1 regularizer (i.e., w/o norm), the negative feature loss of BadNets is high with a very large L_1 mask norm, while the Bpp has an even higher negative loss with a much smaller mask norm. The high negative loss of BadNets is actually from the sparse feature mask rather than backdoor features, i.e., there are too many zeros in (1 - $\mathbf{m}$). This indicates that BadNet backdoor features are less prominent than Bpp features, making it more challenging to decouple BadNets features.

5.5 Limitations

Similar to existing works [110, 127, 144, 111, 128, 112], BAN assumes a local small and benign dataset. This scenario is common, considering that benign samples are available online, and the model provider may provide some samples to verify the model's performance. In addition, our fine-tuning with neuron noise may slightly decrease the performance in be-

Table 5.7: Loss values when feeding the benign or backdoor features into the final classification layer. The mask is optimized using Eq. (5.5) (w/ norm) or Eq. (5.7) (w/o norm). The shape of the feature mask is 512 × 4 × 4, so the maximal L_1 norm is 8192.

Attack	BA	ASR	L_1 mask norm	Positive loss	Negative loss
BadNets (w/ norm)	93.47	99.71	2258.90	0.21	0.26
BadNets (w/o norm)	93.47	99.71	8054.45	0.14	2.17
Blend (w/ norm)	94.60	100.00	2084.62	0.15	0.20
Blend (w/o norm)	94.60	100.00	8117.90	0.04	2.22
WaNet (w/ norm)	93.88	99.63	7400.97	0.06	2.34
WaNet (w/o norm)	93.88	99.63	7702.56	0.05	2.39
IAD (w/ norm)	93.82	99.64	7898.91	0.03	2.25
IAD (w/o norm)	93.82	99.64	7895.16	0.03	2.25
Bpp (w/ norm)	94.56	99.97	7147.68	0.09	2.80
Bpp (w/o norm)	94.56	99.97	7260.31	0.09	2.78

nign inputs, similar to other defense methods [111, 112, 277, 130, 136, 110, 278]. However, BAN provides better performance and requires less time consumption.

5.6 Conclusions and Future Work

This chapter proposes an effective yet efficient backdoor defense, BAN, that utilizes the adversarial neuron noise and the mask in the feature space. BAN is motivated by the observation that traditional defenses outperformed the latest feature space defenses on input space backdoor attacks. To this end, we provide an in-depth analysis showing that feature space defenses are overly dependent on prominent backdoor features. Experimental studies demonstrate BAN's effectiveness and efficiency against various types of backdoor attacks. We also show BAN's resistance to potential adaptive attacks. Future studies could explore a more practical detection method without assuming access to local benign samples and better strategies for decoupling features because a fixed mask in the feature is not always aligned with the benign and backdoor features.

5.7 Appendix

Broader Impacts. This chapter proposes a complete defense that includes detecting and removing DNN backdoors. We believe our approach has a positive impact on the security of DNNs. A possible negative impact is overconfidence in robustness against backdoor attacks, as there is no theoretical guarantee that it will always remove the backdoor.

5.7.1 Datasets

CIFAR-10. The CIFAR-10 [250] contains 50,000 training images and 10,000 testing images with the size of 3 × 32 × 32 in 10 classes.

GTSRB. The GTSRB [272] contains 39,209 training images and 12,630 testing images in

43 classes. In our experiments, the images are resized to 3 × 32 × 32.

Tiny-ImageNet. Tiny-ImageNet [273] contains 100,000 training images and 10,000 testing images with the size 3 × 64 × 64 in 200 classes.

ImageNet. ImageNet [187] contains over 1.2 million high-resolution images in 1,000 classes. Our experiments use a subset of the full ImageNet dataset, i.e., 200 randomly selected classes. Each class has 1300 training images and 50 testing images. The ImageNet images are scaled to 3 × 224 × 224.

5.7.2 Backdoor Models

BAN and defense baselines are evaluated using seven well-known attacks: BadNets [8], Blend [9], WaNet [10], IAD [11], BppAttack [105], Adap-Blend [118], and SSDT [117].

For all backdoored models, we use SGD with a momentum of 0.9, weight decay of 5×10^{-4}, and a learning rate of 0.01. We train for 200 epochs, and the learning rate is divided by 10 at the 100th and 150th epochs. Same to [130], we use 90% of the training data for training backdoor models and 10% for the validation set.

BadNets. We use a 3 × 3 pattern to build the backdoor trigger. The poisoning rate is 5%.

Blend. We use the random Gaussian noise and the blend ratio of 0.2 for backdoor training. The poisoning rate is 5%.

Adap-Blend. We use the "hellokitty_32.png" and blend ratio of 0.2 to build triggers. The poisoning rate is 5%.

SSDT. SSDT is a Source-Specific attack, which only misleads the victim classes to the targeted class. We use class 1 as the victim and class 0 as the target class.

For other attacks and hyperparameters not mentioned, we use the default settings from the papers or their official open-source implementations.

5.7.3 Defense Baselines

BAN is compared with five representative methods, including Neural Cleanse (NC) [110], Tabor [144], FeatureRE [111], Unicorn [128], and BTI-DBF [112]. In this section, we describe the hyperparameters of these defenses.

NC and Tabor. We use the implementation and default hyperparameters from Trojan-

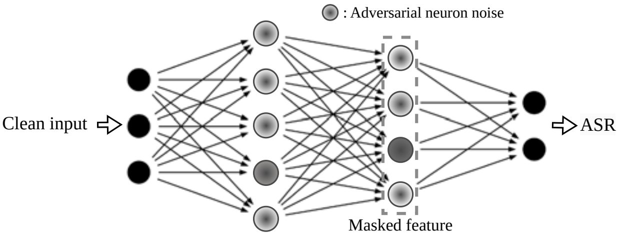


Figure 5.5: Illustrative diagram of BAN

Zoo [257] for NC and Tabor. 1% of the training set is used to conduct 100 epochs of the trigger inversion. The learning rate is 0.01.

FeatureRE. We first tried the default hyperparameters from the FeatureRE paper, but they could not work even for BadNets. We assume this is because the constraints, including feature mask size and similarity between original images and trigger images, are limited to a very small size. Therefore, we relaxed the constraints on masks and norms to find stronger triggers, including loss std bound=1.2, p loss bound=0.2, loss std bound=1.2, mask size=0.06, and learning rate=0.001. We use 1% of the training set and run FeatureRE for 400 epochs for each class.

Unicorn. The default hyperparameters for Unicorn are too powerful in our case, and the recovered triggers can mislead any model to the target label when applied to inputs. Every model, including benign models, is thought of as backdoored. Therefore, we slightly increase the thread values of the constraints on Unicorn optimization to find proper triggers, including loss std bound=0.5, SSIM loss bound=0.1, and mask size=0.01. We use 1% of the training set and run Unicorn 40 epochs for each class.

BTI-DBF. We follow the default settings in the BTI-DBF [112] paper. Note that BTI-DBF uses 5% of training samples for defenses.

5.7.4 BAN Settings

Detection. We use epsilon (ϵ) to limit the noise added to neurons. Otherwise, the model weights are destroyed by the noise. The ϵ values are 0.3, 0.3, 0.2, and 0.1 for CIFAR-10, GTSRB, Tiny-ImageNet, and ImageNet200, respectively. ϵ values are decided according to the size of the images. We use a smaller ϵ for larger datasets. We use the 30-step PGD algorithm to solve the optimization in Eq. (5.4) to find the noise, i.e., $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$. We use SGD and the learning rate of $\epsilon/30$ for the PGD optimization. Then, we use Adam and the learning rate of 0.01 to search for 20 epochs for the feature mask using Eq. (5.5).

Table 5.8: Ablation of hyperparameters for our defense. Each entry is the average of 5 runs.

Attack	LR	Epoch	λ_2	Ours	
				BA	ASR
Bpp	0.005	25	0.2	93.26 ± 0.14	2.53 ± 0.93
	0.005	25	0.5	92.64 ± 0.19	2.28 ± 0.72
	0.005	25	0.8	92.19 ± 0.13	2.89 ± 0.94
	0.005	25	1.0	92.04 ± 0.29	2.74 ± 0.49
	0.005	10	0.5	92.97 ± 0.29	2.62 ± 0.44
	0.005	50	0.5	92.57 ± 0.09	2.80 ± 0.73
	0.005	75	0.5	92.51 ± 0.16	2.34 ± 0.17
	0.001	25	0.5	93.57 ± 0.17	2.89 ± 1.01
	0.010	25	0.5	91.99 ± 0.22	2.52 ± 0.64
	0.020	25	0.5	90.05 ± 0.37	1.91 ± 0.55

The λ_1 for Eq. (5.5) equals 0.75. The two optimizations above use 1% of the training set. Note that this 1% of data is not used to train the backdoor models. The elements in the mask are clamped into continuous values between 0 and 1. In Figure 5.5, we present an illustrative diagram of BAN.

Defense. Our defense uses Eq. (5.8) to remove the backdoor. Due to limited access to benign samples and time consumption, we only use 5% of the training data and 25 epochs for our fine-tuning. The trade-off hyperparameter (λ_2) is 0.5. Then, we use the most commonly used hyperparameters for the optimizer. We use SGD with momentum=0.9, weight decay=5e-4, and learning rate=0.005 as the optimizer. The plain fine-tuning uses the same hyperparameters as BAN.

Table 5.8 shows the ablation results on fine-tuning epochs, λ_2, and learning rate. Considering both performances on benign accuracy and removing the backdoor, our hyperparameters (LR=0.005, λ_2=0.5 and epoch=25) show the best results.

5.7.5 Additional Experimental Results

The Performance under Different Datasets. Table 5.9 shows the detection results using two larger datasets, Tiny-ImageNet and Imagenet200. The ImageNet200 is a subset of ImageNet-1K. We train 10 models for each case, 60 models in total. BAN still performs well under the two larger datasets. In addition, BAN is also scalable in terms of time consumption. To conduct backdoor detection on ResNet18 models trained using CIFAR-10, Tiny-ImageNet, and ImageNet200, BAN takes around 55, 129, and 1,120 seconds, respectively. We do not apply other baselines (except for BTI-DBF) to larger datasets due to high computational requirements. For example, FeatureRE costs more than 2 hours for detection in one class of ImageNet200 (default settings), meaning FeatureRE needs more than 400 hours for a full detection on ImageNet200.

Table 5.9: The detection results using ResNet18 under different datasets.

Dataset	Attack	BTI-DBF		BTI-DBF*		Ours	
		Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
Tiny-ImageNet	No Attack	0	100%	0	100%	0	100%
	WaNet	4	40%	6	60%	10	100%
	IAD	9	90%	8	80%	10	100%
ImageNet200	No Attack	3	70%	0	100%	0	100%
	WaNet	6	60%	9	90%	10	100%
	IAD	10	100%	10	100%	10	100%

Dection on GTSRB. Table 5.10 shows the detection success rate for BAN on GTSRB. Notice that NC and BTI-DBF* perform worse against advanced attacks, while BAN precisely detects all backdoor models.

Defense of Adaptive Attacks. Table 5.11 shows the fine-tuning results using our defense method against the two adap-blend and SSDT. The results show that fine-tuning with neuron noise can remove the backdoor with a slight decrease in clean accuracy.

Table 5.10: The detection results on GTSRB with ResNet18.

Attack	NC		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
BadNets	20	100%	20	100%	20	100%
WaNet	8	40%	18	90%	20	100%
IAD	0	0%	20	100%	20	100%
Bpp	13	65%	14	70%	20	100%

Table 5.11: Fine-tuning for adaptive attacks.

Attack	No defense		Ours	
	BA	ASR	BA	ASR
Adap-Patch	94.20	99.75	90.45	10.24
SSDT	93.86	90.30	93.29	0.90

Comparsion with ANP. ANP [130] uses a similar neuron noise for pruning backdoor neurons. Their optimization objective includes neuron noise and neuron mask. The neuron noise is optimized to fool the network, while the mask controls a trade-off for clean cross-entropy loss. We compare our fine-tuning with ANP in Table 5.12. The hyperparameters of ANP are the default from the original paper.

Table 5.12: Comparison with ANP [130] on CIFAR-10 using ResNet18.

Defense	BadNets		Blend		WaNet		IAD		Bpp	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
No defense	93.37	99.41	94.60	100	93.57	99.37	92.83	97.10	94.29	99.93
ANP	93.16	2.13	94.17	97.24	93.06	13.84	92.50	0.44	93.80	4.77
Ours	92.26	2.04	92.61	1.04	92.18	0.87	92.36	1.47	92.82	2.16

Different Poisoning Rates. BAN is effective against BadNets with different poisoning rates, as shown in Table 5.13. We use BadNets because it is relatively weak at a low poisoning rate (i.e., hard to be mitigated), while more advanced attacks may still be strong at a low poisoning rate.

Table 5.13: The performance of BAN fine-tuning under different poisoning rates of BadNets using CIFAR-10.

Poisoning Rate	BA	ASR	Positive loss	Negative loss	Mitigated BA	Mitigated ASR
0.01	93.48	98.69	0.38	0.17	92.07	2.73
0.05	93.37	99.41	0.37	0.35	92.06	1.97
0.10	90.98	100.00	0.35	2.06	90.29	2.17
0.15	90.32	100.00	0.39	2.23	90.16	1.71
0.20	89.34	100.00	0.44	2.43	90.39	1.01
0.25	88.09	100.00	0.56	2.81	89.55	1.54
0.30	86.09	100.00	0.62	3.13	88.83	1.08
0.40	82.39	100.00	0.67	3.51	88.75	1.67
0.50	77.83	99.97	0.84	4.27	86.87	3.56

Against Backdoors on the MLP. We evaluate the defense performance on a simple model architecture. In particular, a 4-layer multilayer perception (MLP) is trained with benign samples and with BadNets. Table 5.14 demonstrates that BAN is effective on MLP, where the "num. to target" refers to the number of samples (in 5000 validation samples) that are classified as the backdoor target after our detection. We also find that the positive feature loss (i.e., benign feature loss) is very close to the negative loss (i.e., potential backdoor feature loss), which indicates that the backdoor features are more challenging to decouple from benign ones.

Table 5.14: The performance of BAN on the 4-layer MLP.

MLP	BA	ASR	Positve loss	Negative loss	Num. to target	Mitigated BA	Mtigated ASR
Benign	53.24	-	1.95	2.29	419	-	-
BadNets	47.04	100.00	2.03	2.34	3607	45.13	7.11

Discussion about Relationship between the Neuron Noise and Lipschitz Continuity. A small trigger that changes the output of a benign input into a malicious target label can be related to the high Lipschitz constant [131] and a neural network with high robustness tends to have a lower local Lipschitz constant. Moreover, a larger local Lipschitz constant implies steeper output around trigger-inserted points, leading to a smaller trigger effective radius making trigger inversion less effective [279]. Thus, the concepts of adversarial activation noise and Lipschitz continuity are related, and the local Lipschitz constant can serve as an upper bound for the trigger's effective influence.

In addition, introducing theoretical tools like the Lipschitz constant for a backdoor defense may also be very tricky in practice because it needs approximation for implementation. For example, [131] evaluates the channel-wise Lipschitz constant by its upper bound but does not thoroughly discuss the relationship between the channel-wise Lipschitz constant and the network-wise Lipschitz constant, where the theorem of Lipschitz continuity really relies on. Recent work also mentions that empirically estimating the Lipschitz constant is hard from observed data, which usually leads to overestimation [280]. Methods relying on Lipschitz continuity may not require heavy computational load and are also related to our approach. Our method emphasizes more on fine-tuning with the guidance of neuron noise, rather than tuning the trained model.

Choosing the λ_1. As discussed in the method section, the mask norm (L_1 norm) with lambda in Eq. (5.5) is to ensure that the optimization objective is decoupling between benign and backdoor features. Without the constraint of the mask norm in Eq. (5.5), the optimization objective will be simply increasing the mask norm unless there are extremely strong backdoor features. The λ_1 value selection is implemented by checking the value of the mask norm. Table 5.15 provides an example of λ_1 selection on CIFAR-10, where the maximal mask norm is 8192. It can be observed that the mask is almost full of ones when λ_1 is smaller than 0.7, and the negative feature loss (backdoor feature) is high, based on which we can pick a value of λ_1 greater than 0.7. Note that the selection is unaware of potential backdoors.

Table 5.15: Feature loss values with different λ_1

λ_1	Mask norm	Positive loss	Negative loss
0.0	8188.62	0.27	2.30
0.1	8188.75	0.28	2.30
0.2	8184.30	0.27	2.30
0.3	8175.50	0.29	2.30
0.4	8152.40	0.25	2.30
0.5	8131.26	0.27	2.29
0.6	8055.07	0.21	2.27
0.7	7898.25	0.26	2.24
0.8	596.85	0.99	0.23
0.9	22.08	2.33	0.28

Swin Transformer. Table 5.16 provides experimental results of a 12-layer Swin transformer [33] on CIFAR-10. For BadNets and Blend, we train the backdoored network using Adam as the optimizer. For WaNet, IAD, and Bpp, we use the default setting. All backdoored models can be detected by BAN.

Table 5.16: Results on Swin Transformer.

Attack	No defense		FT		BTI-DBF(U)		BAN Fine-tune	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets	86.7	97.43	82.23	44.07	85.21	99.99	83.92	2.91
Blend	85.46	100.00	80.13	100.00	83.9	100.00	79.18	26.20
WaNet	78.25	31.03	75.63	2.3	73.83	2.61	75.28	3.1
IAD	76.35	88.71	74.58	54.63	74.29	61.16	76.26	9.8
Bpp	79.19	63.74	75.35	12.99	74.66	11.04	74.25	6.91

Clean Label Attack. We provide a clean-label backdoor experiment following Label-Consistent Backdoor (LC) [266], where we take default hyperparameters. Table 5.17 demonstrates the detection performance of 20 models, where we randomly select one model from the 20 for the mitigation experiment. Experimental results demonstrate that BAN is also effective against the clean-label attack.

Table 5.17: Clean label attack on CIFAR-10.

Attack	BA	ASR	Detection Success Accuracy	Mitigated BA	Mitigated ASR
LC [266]	92.72	100.00	90.00	89.27	6.16

Table 5.18: Results on different target classes on CIFAR-10.

Target	No defense		BAN Fine-tune	
	BA	ASR	BA	ASR
0	93.41	100.00	92.57	1.56
1	93.51	100.00	92.53	0.84
2	93.54	99.99	91.49	1.08
3	93.59	99.99	92.14	1.93
4	93.84	99.98	92.13	1.49
5	93.52	100.00	91.84	3.29
6	93.46	100.00	92.48	0.93
7	93.56	100.00	92.36	0.73
8	93.57	99.89	92.31	0.58
9	93.36	100.00	91.65	2.40

Semantically Similar Classes. We provide an additional analysis on image form semantically similar classes. From ImageNet200, we select class n02096294 (Australian terrier) as the target class, since there are 19 kinds of terriers in our ImageNet200 datasets, such as n02094433 (Yorkshire terrier) and n02095889 (Sealyham terrier). We train 10 backdoor networks using BadNets with different target classes. Table 5.18 demonstrates that BAN is effective against the backdoor regardless of semantically similar classes.

Chapter 6

Backdoor Stealthiness in Parameter Space

Backdoor attacks maliciously inject covert functionality into machine learning models, which has been considered a security threat. The stealthiness of backdoor attacks is a critical research direction, focusing on adversaries' efforts to enhance the resistance of backdoor attacks against defense mechanisms. Recent research on backdoor stealthiness focuses mainly on indistinguishable triggers in *input space* and inseparable backdoor representations in *feature space*, aiming to circumvent backdoor defenses that examine these respective spaces. However, existing backdoor attacks are typically designed to resist a specific type of backdoor defense without considering the diverse range of defense mechanisms. Based on this observation, this chapter pose a natural question: *Are current backdoor attacks truly a real-world threat when facing diverse practical defenses?* To answer this question, we examine 12 common backdoor attacks that focus on input-space or feature-space stealthiness and 17 diverse representative defenses. Surprisingly, we reveal a critical blind spot that backdoor attacks designed to be stealthy in input and feature spaces can be mitigated by examining backdoored models in *parameter space*. To investigate the underlying causes behind this common vulnerability, we study the characteristics of backdoor attacks in the parameter space. Notably, we find that input- and feature-space attacks introduce prominent backdoor-related neurons in parameter space, which are not thoroughly considered by current backdoor attacks. Taking comprehensive stealthiness into account, we propose a novel supply-chain attack called Grond. Grond limits the parameter changes by a simple yet effective module, Adversarial Backdoor Injection (ABI), which adaptively increases the parameter-space stealthiness during the backdoor injection. Extensive experiments demonstrate that Grond outperforms all 12 backdoor attacks against state-of-the-art (including adaptive) defenses on CIFAR-10, GTSRB, and a subset of ImageNet. In addition, we show that ABI consistently improves the effectiveness of common backdoor attacks. Our code is publicly available: `https://github.com/xiaoyunxxy/parameter_backdoor`.

6.1 Introduction

While deep neural networks (DNNs) have achieved excellent performance on various tasks, they are vulnerable to backdoor attacks. Backdoor attacks insert a secret functionality into a model, which is activated by malicious inputs during inference. Such inputs contain an attacker-chosen property that is called the trigger. Backdoored DNNs can be created by training with poisoned data [8, 9, 10, 266]. More powerful and stealthy backdoors can also be injected through the control of a training process [103, 104, 121, 114, 107, 117, 113], or by direct weights modification of the victim model [51, 108].

In early backdoor attacks [8, 9, 259], triggers could induce noticeable changes that human inspectors or anomaly detectors [109, 110, 127] could easily spot. To enhance the ability to remain undetected against such defenses (i.e., achieve *input-space stealthiness*), smaller or more semantic-aware triggers are designed [10, 281, 105]. Input-space stealthy backdoor attacks usually need to change labels of poisoned samples to the target class (i.e., dirty-label), which makes detection easier [109]. To this end, another line of backdoor attacks poisons the training data without changing the labels [266, 57] (i.e., clean-label), improving backdoor stealthiness.

Despite the stealthiness concerning input images and labels, it has been widely observed that existing backdoor attacks introduce separable representations in the feature space, which can be exploited to develop backdoor defenses [111, 118, 135, 133, 112]. For example, featureRE [111] utilizes feature separability and designs a feature space constraint to reverse engineer the backdoor trigger. In response to feature-space defenses, state-of-the-art (SOTA) backdoor attacks focus on eliminating the separability in the feature space [117, 118, 279, 103] to increase the *feature-space stealthiness*, i.e., the undetectability against feature-space defenses. Considering a different threat model, supply-chain backdoor attacks assume control over the training or directly modify the model's weights [121, 51, 108], and the backdoored model is provided as a service or as the final product. For example, supply-chain attacks could introduce a penalty to the training loss that decreases the distance between the backdoor and benign features to increase feature-space stealthiness [103, 113, 115, 114].

An important observation is that most backdoor attacks are designed to be stealthy to resist a specific type of defense. For example, WaNet [10] and Bpp [105] design imperceptible triggers to bypass input-space defenses (such as NC [110]), but introduce significant separability in the feature space [111]. Adap-Patch [118] avoids feature separability but uses patch-based triggers, which a human inspector can detect. More critically, current backdoor attacks are barely evaluated against *parameter-space defenses* [133, 282, 139, 130, 131, 132]. This oversight is significant because backdoor behaviors are ultimately embedded in and reflected by the parameters of the backdoored model, which is the fi-

Table 6.1: A summary of the existing defenses evaluated in this chapter. *"Proactively training"* refers to the strategy that the defender could proactively control the training on poisoned training data to produce a clean model without a backdoor in it. Additionally, all the defenses have been tested against the all-to-one attack, so we omitted it from the attack assumptions. A summary of backdoor attacks is provided in Table 6.12.

Defense		Defense Task			Threat Model			Attack	
		Input	Model	Mitigation	Black-box	Clean data	Proactively	A2A	Dynamic
Model Inspection	NC [110]	○	●	○	○	●	○	○	○
	Tabor [144]	○	●	○	○	●	○	○	○
	FeatureRE [111]	○	●	○	○	●	○	○	●
	Unicorn [128]	○	●	○	○	●	○	○	●
	BTI-DBF [112]	○	●	○	○	●	○	●	●
Input Inspection	Scale-up [263]	●	○	○	●	○	○	●	●
	IBD-PSC [283]	●	○	○	○	●	○	●	●
	CT [137]	●	○	○	○	●	●	●	●
Pruning	FP [129]	○	○	●	○	●	○	●	●
	ANP [130]	○	○	●	○	●	○	●	●
	CLP [131]	○	○	●	○	○	○	●	●
	RNP [132]	○	○	●	○	●	○	●	●
Fine-tune	vanilla FT	○	○	●	○	●	○	●	●
	FT-SAM [133]	○	○	●	○	●	○	●	●
	I-BAU [134]	○	○	●	○	●	○	●	●
	FST [135]	○	○	●	○	●	○	●	●
	BTI-DBF(U) [112]	○	○	●	○	●	○	●	●

○ the item is not supported by the defense; ● the item is supported by the defense.

nal product of any backdoor attack. As such, there is a lack of systematic evaluation of backdoor attacks against the latest *parameter-space defenses*.

To this end, in this chapter, we first systematically analyze 12 attacks against 17 backdoor defenses. All evaluated defenses and their characteristics, including detection and mitigation, are summarized in Table 6.1. Surprisingly, our experiments demonstrate that parameter-space defenses can easily mitigate SOTA stealthy backdoor attacks (including supply-chain attacks), indicating that existing stealthy backdoor attacks fail to provide *parameter-space stealthiness* and, as a result, still need substantial improvement to be stealthy in the model's parameter space. More importantly, our analysis reveals that even though some backdoor attacks can resist several defenses, bypassing all defense types is far from trivial.

To explore whether it is possible to make backdoor attacks stealthy simultaneously against diverse defenses, we propose a novel attack called Grond that considers *comprehensive stealthiness*, meaning that a backdoor attack is stealthy in the input, the feature, and the parameter space of the model. Grond achieves the input space stealthiness by using adversarial perturbations as the trigger. To achieve parameter-space stealthiness, we propose a novel *Adversarial Backdoor Injection* module that adaptively injects the backdoor during the backdoor training to achieve parameter space stealthiness. We also show that the feature-space stealthiness is a by-product of input- and parameter-space stealthiness with empirical results in Figures 6.3 and 6.6. Specifically, guided by our Trigger-Activated Change (TAC) analysis, we leverage the Lipschitz continuity of neuron activations to find backdoor-related suspicious and sensitive neurons. Then, we conduct pruning on these neurons to eliminate the backdoor effect. As a result, the backdoor is associated with neurons throughout the DNN rather than just focusing on a few prominent neurons after Adversarial Backdoor Injection, as illustrated in Figure 6.1.

We make the following contributions:

- We revisit SOTA backdoor attacks regarding their stealthiness, showing that most attacks are designed to increase input-space indistinguishability or/and feature-space inseparability without considering parameter-space stealthiness. Based on this finding, we examine common backdoor attacks and reveal a critical *blind spot* regarding real-world scenarios: SOTA stealthy backdoor attacks are highly vulnerable to parameter-space defenses.
- To investigate the underlying reasons behind this common vulnerability of backdoor attacks, we take a closer look at the backdoor characteristics in the parameter space, showing that input- and feature-space attacks introduce prominent backdoor-related neurons, which cannot be avoided by current backdoor attacks.
- To accomplish comprehensive stealthiness, we propose a novel backdoor attack, Grond, that considers input, feature, and parameter-space defenses. Extensive experiments demonstrate that Grond outperforms SOTA attacks against four pruning- and five fine-tuning-based defenses on CIFAR-10, GTSRB, and ImageNet200. Moreover, we demonstrate that Grond is resistant against five model detection defenses, two input detection defenses, and a proactive defense.
- We verify the effectiveness of the Adversarial Backdoor Injection module by binding it with other attacks. Experimental results demonstrate that Adversarial Backdoor Injection could substantially improve the parameter-space robustness of most common backdoor attacks.

6.2 Related Work

In this section, we introduce the background and provide relevant literature on backdoors.

6.2.1 Preliminaries on Backdoor Training

This chapter considers a C-class classification problem with an L-layer CNN network $f = f_L \circ \cdots f_1$. Suppose that $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ is the original training data, containing N samples of $\boldsymbol{x}_i \in \mathbb{R}^{d_c \times d_h \times d_w}$ and its label $y \in \{1, 2, \ldots, C\}$. d_c, d_h, and d_w are the number of input channels, the height, and the width of the image, respectively. The attacker chooses a target class t and creates a partially poisoned dataset $\mathcal{D}_p$ by poisoning generators G_x and G_y, i.e., $\mathcal{D}_p = \mathcal{D}_c \cup \mathcal{D}_b$. $\mathcal{D}_c$ is the benign data from original dataset,

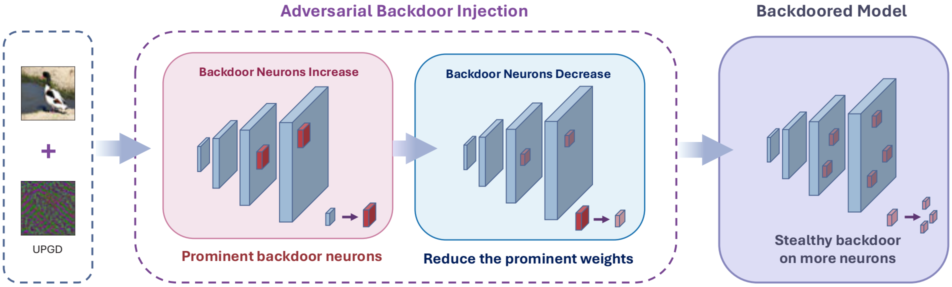

Adversarial Backdoor Injection
Backdoored Model
+
UPGD
Backdoor Neurons Increase
Prominent backdoor neurons
Backdoor Neurons Decrease
Reduce the prominent weights
Stealthy backdoor
on more neurons

Figure 6.1: Diagram illustrating the working mechanism of Grond. On the left, universal PGD (UPGD) perturbation is generated as backdoor patterns to be injected. In the middle, ABI is applied where perturbed samples are iteratively used to train the model, and the model parameters are pruned to limit the magnitude of prominent backdoored weights. On the right, the output backdoored model that considers comprehensive stealthiness is deployed, where 1) the triggers are invisible, 2) the features of trigger samples are inseparable, and 3) the backdoored model weights are hardly distinguishable from benign model weights. Perturbations generated by UPGD are scaled up 10× for visualization.

$\mathcal{D}_b = \{(\boldsymbol{x}', y')|\boldsymbol{x}' = G_x(\boldsymbol{x}), y' = G_y(y), (\boldsymbol{x}, y) \in \mathcal{D} - \mathcal{D}_c\}$. In the clean-label setting, $G_y(y) = y$. For the dirty-label attacks, $G_y(y) = t$. In the training stage, the backdoor is inserted into f by minimizing the loss on $\mathcal{D}_p$:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}_p}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\in\mathcal{D}_p} \ell(f(\boldsymbol{x};\boldsymbol{\theta}), y). \tag{6.1}$$

In the inference stage, the trained f performs well on benign data $\hat{\boldsymbol{x}}$, but predicts $G_x(\hat{\boldsymbol{x}})$ as $G_y(\hat{y})$.

6.2.2 Backdoor Attacks

Backdoor attacks compromise the integrity of the victim model so that the model performs naturally on benign inputs but is misled to the target class by inputs containing the backdoor trigger. The trigger can be a visible pattern inserted into the model's input in the **input space** or a property that affects the feature representation of the model's input in the **feature space**. Eventually, however, the backdoored model's parameters in the **parameter space** will be altered regardless of the exact backdoor attack (see Figure 6.2). To insert a backdoor, the attacker is assumed to only control a small portion of the training data under the *poison training* scenario [8, 9, 102]. In the *supply-chain* setting (backdoor models provided to users), the attacker also control the training process [103, 11, 104, 10, 105]. Moreover, the backdoor can also be created by directly modifying the model's weights [106, 51, 107, 108].

Input-space attacks. Traditional attacks typically use simple patterns as their triggers. For example, BadNets [8] uses a fixed patch, and Blend [9] mixes a Hello Kitty pattern into the images as the trigger. These non-stealthy triggers introduce abnormal data into training data and can be easily detected by human inspectors or defenses [109, 110]. To improve the stealthiness, various triggers are proposed to achieve *invisibility* in the input space. IAD [11] designed a dynamic solution in which the triggers vary among different inputs. WaNet [10] proposed the warping-based trigger, which is invisible to human inspection. Bpp [105] used image quantization and dithering as the trigger, which makes imperceptible changes to images. Although these methods successfully build invisible triggers and bypass traditional defenses [110], they still introduce separable features and can be detected by feature-space defenses [111, 112]. These input-invisible attacks can be even more noticeable than input-visible attacks (BadNet, Blend) in the feature space [139]. We conjecture this is because they have less modification on input pixels than input-visible attacks. Therefore, input-invisible attacks require more influential features to achieve a successful attack.

Feature-space attacks. Knowing the vulnerability of input-space attacks against feature-space defenses, backdoor attacks are improved for feature-space stealthiness. A common threat model of this attack type is to assume additional control over the training process. For example, [103, 113, 114] directly designed loss functions to minimize the difference between the backdoor and benign features. [115] formulated the difference between backdoor and benign features by Wasserstein-2 distance and used the difference as a regularization constraint in backdoor training. Aside from design loss penalties, TACT [116] and SSDT [117] point out that source-specific (poison only the specified source classes) attack helps to obscure the difference in features between benign and backdoor samples. In addition, [118] proposed Adap-Blend and Adap-Patch, which obscures benign and backdoor features by 1) including poisoned samples with the correct label, 2) asymmetric triggers (using a stronger trigger at inference time), and 3) trigger diversification (using diverse variants of the trigger during training). Unfortunately, existing attacks lack systematic evaluation against the latest defenses. For example, Adap-Blend can be thoroughly mitigated by recent works [133, 112, 139]. SSDT can be mitigated by traditional defenses, such as fine-pruning [129] and vanilla fine-tuning according to Tables 6.2 and 6.3. In summary, feature-space attacks usually introduce visible triggers and cannot defeat the latest defenses.

Supply-chain attacks. Supply-chain attacks are getting more attention due to their potential in real-world applications where backdoored models are provided as the final product to users. In supply-chain attacks, adversaries could control both training data and the training process. Note that feature-space attacks [103, 119, 115, 113, 120, 121, 114, 122, 117] with the assumption of control over the training process are a subset of supply-chain attacks, as their output is the backdoor model. In addition to training control, another kind of supply-chain attack directly adjusts the model's weights in parameter space to introduce a backdoor, i.e., *parameter-space attack*. T-BFA [123], TBT [124], and ProFlip [125] explore modifying a sequence of susceptible bits of DNN parameters stored in the main memory (e.g., DRAM) to inject backdoor. SRA [107] and handcrafted Backdoor [51] directly modify a subset of models' parameters to increase the logits of the target class. However, these attacks require a local benign dataset to guide the search of the subset of parameters to be modified. Data-free backdoor [126] releases the requirement of benign data by collecting substitute data irrelevant to the main task and fine-tuning using the substitute data. DFBA [108] further proposes a retraining-free and data-free backdoor attack by injecting a backdoor path (a single neuron from each layer except the output layer) into the victim model. Supply-chain attacks focus on increasing the backdoor effectiveness without comprehensively considering parameter-space defenses. We argue that supply-chain backdoor attacks should also take parameter-space stealthiness into account.

6.2.3 Backdoor Defenses

Backdoor defenses can be classified into detection and mitigation. Detection refers to determining whether a model is backdoored (*model detection*) [110, 127, 113, 128, 112] or a given input is applied with a trigger (*input detection*) [284, 263, 117]. Mitigation refers to erasing the backdoor effect from the victim model by pruning the backdoor-related neurons (*pruning-based* defenses) [129, 130, 131, 132] or unlearning the backdoor trigger (*fine-tuning-based* defenses) [133, 134, 135, 112]. In addition, recent works [136, 137, 138] also consider the home-field advantage* to design more powerful *proactive defenses.*

Backdoor detection. Backdoor trigger reverse engineering (also known as trigger inversion) is considered one of the most practical defenses for backdoor detection as it can be applied to both poisoning training and supply-chain scenarios [111, 128, 112, 139], i.e., it is a post-training method. Specifically, trigger inversion works by searching for a potential backdoor trigger for a specific model. The model is determined as backdoored if a trigger is found, and the trigger can be used to unlearn the backdoor. The searching is implemented as an optimization process corresponding to the model and a local benign dataset. For example, NC [110] firstly proposes trigger inversion for detection by optimizing the mask and pattern in the input space that can mislead the victim to the target class. This optimization is repeated for all classes. The model is considered backdoored if an outlier significantly smaller than triggers for all other classes exists. Tabor [144] designs better optimization objects due to overly large (large size triggers but no overlaps with the true trigger) and overlaying (overlap the true trigger but with redundant noise) problems of NC. Although methods similar to NC perform well against fixed patch trigger attacks, such as BadNets [8] and Blend [9], they may not be effective against input-stealthy attacks like WaNet [10]. To address this problem, FeatureRE [111] moves trigger inversion from input space to feature space. Unicorn [128] further proposes a transformation function

*The defender has full control of the system and could access the training process.

for attacks in other spaces, such as numerical space [105]. Recent works [112, 139] focus on exploring new optimization objectives addressing the inefficiency problem of previous trigger inversion methods due to optimization over all classes. BTI-DBF [112] trains a trigger generator by maximizing the backdoor feature difference between benign samples and their generated version (by the trigger generator) and minimizing the benign feature difference. BAN [139] optimizes the noise on neuron weights rather than input pixels to activate the potential backdoor, which further improves both effectiveness and efficiency.

Backdoor mitigation. Backdoor mitigation consists of fine-tuning and pruning, which are effective and do not assume knowledge of backdoor triggers. Pruning methods aim to find and remove backdoor-related neurons. FP [129] eliminates dormant neurons on benign inputs and then fine-tunes the pruned network. ANP [130] searches for backdoor-related neurons by adding adversarial noise to neuron weights to activate the backdoor. RNP [132] uses an unlearning and recovering process on benign data to expose backdoor neurons, as the recovering will force the backdoor neurons to be silent for the main benign task. Unlike these pruning methods guided by benign data, CLP [131] directly analyses the Channel Lipschitzness Constant of the network and prunes the high Lipschitz constant channels in a data-free manner.

Traditional fine-tuning as defense usually needs trigger inversion methods to recover the trigger and then unlearn the trigger. For example, BTI-DBF(U) [112] fine-tunes backdoor models using triggers recovered by their inversion algorithm. However, there is no guarantee that the recovered trigger is the true trigger for the backdoor. Recent works also consider fine-tuning without the trigger information but with prior human knowledge. For example, FT-SAM [133] observes a positive correlation between the weight norm of neurons and backdoor-related neurons. Then, they propose a fine-tuning method to revise the large outliers of weight norms using Sharpness-Aware Minimization (SAM). I-BAU [134] forms a min-max fine-tuning similar to adversarial training, where the inner maximizing searches for perturbations that mislead the model and the outer minimizing is to keep the model's capability on benign data. FST [135] assumes the backdoor and benign features should be disentangled and actively shifting features while fine-tuning by encouraging the discrepancy between the original backdoor model and the fine-tuned model.

Proactive defense. Several methods have been proposed to exploit the home-field advantage. ABL [136] proposes two techniques to avoid learning the backdoor task while training on the poisoned data: 1) trapping the loss value of each example around a certain threshold because backdoor tasks are learned much faster than the main task, and its loss decreases much faster. The samples with lower loss are recorded as poisoned samples; 2) unlearning the backdoor with the recorded poisoned samples. CT [137] detects poisoned samples in the training set by introducing confusing batches of benign data with randomly modified labels. The confusing batches with random labeling corrupt the benign correlations between normal semantic features and semantic labels, so the inference model trained with confusing batches and the poisoned dataset will find it hard to distinguish benign samples. However, the correlation between the backdoor trigger and the target label remains intact, as the confusing batches contain no trigger. Therefore, samples with correctly predicted labels by the inference model are considered poisoned. PDB [138] proactively injects a defensive backdoor into the model during training, which overrides the potential backdoor to be injected by the poisoned training data. In summary, proactive defenses assume a stronger defender for better defensive performance.

6.3 Comprehensive Backdoor Stealthiness

In this section, we first introduce the threat model (Section 6.3.1). Then, we analyze the behavior of neurons of backdoored models, suggesting that stealthy input- and feature-space backdoor attacks can be identified in parameter space (Section 6.3.2). Then, to achieve comprehensive stealthiness, we propose Grond that includes backdoor generation and Adversarial Backdoor Injection (Section 6.3.3).

6.3.1 Threat Model

Attacker's goal. The attacker provides pre-trained models to users. The aim is to inject backdoors into the pre-trained model so that the model performs well on clean inputs but predicts the attacker-chosen target label when receiving inputs with a backdoor trigger, i.e., an all-to-one attack.

Attacker's knowledge. The attacker has white-box access to the training processes, the training data, and the model weights, i.e., the supply-chain threat model. During inference, the backdoor trigger is imperceptible to human inspectors.

Attacker's capabilities. The attacker can train a well-performed surrogate model to generate UPGD, which is used to perturb the victim model's input. Additionally, the attacker can alter the model's weights during training. Table 6.12 in Appendix 6.7.4 shows that the threat model of Grond is aligned with baseline attacks.

6.3.2 Lack of Parameter-Space Stealthiness

As introduced in the related work, early backdoor attacks that introduce noticeable changes in either input [8, 9] or feature space [10, 11] have been empirically shown powerful, even with very low poisoning rates [8, 57]. Focusing on the backdoor-introduced noticeable changes, backdoor defenses are improved to distinguish backdoor patterns in either input or feature space [111, 282]. Meanwhile, backdoor attacks are optimized to increase stealthiness in input [10] or feature space [118]. However, regardless of the implementation of input- or feature-space attack logic, backdoor behaviors are eventually embedded in the backdoored model's parameters. For this reason, it is important to investigate whether backdoor attacks introduce visible changes in the parameter space of the attacked models that can be used by the parameter-space defenses.

Taking this observation into consideration, we ran an initial experiment to understand the behavior of neurons of backdoored models. We use the Trigger Activated Change (TAC) values [131] to quantify the relevance of a neuron to the backdoor behavior according to the difference when the network accepts benign and backdoor inputs. A higher TAC value indicates that the neuron is strongly relevant for backdoor behaviors. Specifically, TAC is defined as:

$$\mathrm{TAC}_l^{(k)}(\mathcal{D}_c) = \frac{1}{|\mathcal{D}_c|} \sum_{\boldsymbol{x} \in \mathcal{D}_c} ||f_l^{(k)}(\boldsymbol{x}) - f_l^{(k)}(G_x(\boldsymbol{x}))||_2, \tag{6.2}$$

where $f_l^{(k)}$ is the k_{th} channel of the l_{th} layer. $G_x(\boldsymbol{x})$ is the poisoned sample. $\mathcal{D}_c$ consists of a few benign samples. Note that TAC can only be used to analyze backdoor behaviors and cannot be deployed as a practical defense, as it requires access to backdoor triggers, which is unrealistic in practice.

TAC analysis of different backdoor attacks is shown in Figure 6.2, where each dot represents the TAC value for one of the 512 individual neurons. We can observe that the TAC values of neurons of backdoored models are substantially higher than those of benign models. In particular, neurons with higher TAC values contribute more to the backdoor behavior. The working mechanism of pruning- and fine-tuning-based backdoor defenses can be understood as targeting and eliminating neurons with TAC values that are substantially higher than those of others. Our observations from the TAC analysis suggest that backdoor attacks are designed to be stealthy in input space, and feature space can, in fact, be identified in parameter space, making them susceptible to parameter-space defenses. Our experimental analysis further substantiates this assumption (See Section 6.4). Thus, we conclude that current backdoor attacks may not be robust against parameter-space defenses.

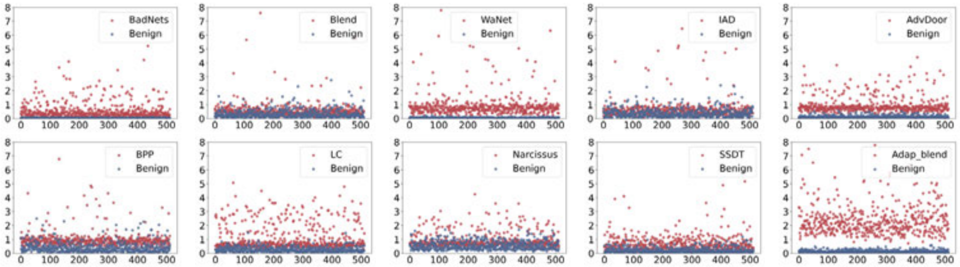


Figure 6.2: TAC [131] analysis of different backdoor attacks on 512 neurons. The y axis contains the TAC values, and the x axis depicts the index of neurons. Higher TAC values suggest a stronger relation between corresponding neurons and the backdoor effect.

6.3.3 Grond for Comprehensive Stealthiness

To address the vulnerabilities identified in the parameter space, we propose a stealthy backdoor attack, Grond, that considers comprehensive stealthiness, i.e., stealthiness in input, feature, and parameter space. Grond includes two key parts: UPGD trigger generation and Adversarial Backdoor Injection (ABI).

Backdoor trigger generation for input-space stealthiness. We use imperceptible adversarial perturbations to generate *imperceptible* backdoor triggers inspired by adversarial example studies [285, 102]. We modify the original PGD algorithm to generate a universal PGD (UPGD) perturbation as the backdoor trigger. UPGD contains non-robust but generalizable semantic information [286], which correlates with the benign functions of the victim model and shortens the distance between poisoned data and the target classification region [102]. Consequently, backdoor patterns tend to make fewer prominent changes to the victim network.

Similar to [57, 102], UPGD is generated on a well-trained surrogate model trained on the clean training set. The architecture and parameters of the surrogate model do not necessarily need to be the same as the victim model (see Table 6.15 in Appendix 6.7.9). UPGD is optimized following the PGD [12] algorithm to decrease the surrogate model's cross-entropy loss that takes as inputs the adversarial examples (the poisoned samples in our case) and the target class label. This procedure is described formally in Algorithm 6.1. The $\boldsymbol{\delta}$ is the generated UPGD that will be used as a backdoor trigger; thus, $G_x(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\delta}$. S is the ball function with the radius ϵ, and the small ϵ guarantees the imperceptibility of the backdoor trigger as it controls the perturbation's magnitude.

The backdoor is injected during training by poisoning some training data from the target class, i.e., applying the UPGD trigger to the training data. In the inference stage, our backdoor is activated by the same trigger. The motivation for our small-size trigger ($\epsilon = 8$) is imperceptibility.

Adversarial Backdoor Injection for parameter-space stealthiness. Backdoor neurons (i.e., trigger-related neurons) regularly show higher activation values for inputs that contain the trigger, which results in powerful performance [127, 111, 282]. To this end, backdoor training needs to substantially increase the magnitude of parameters of backdoor neurons [130, 132, 131], which harms the parameter-space stealthiness of backdoor attacks.

One way to find the sensitive neurons with higher activation values is to analyze the Lipschitz continuity of the network. Leveraging this fact, we introduce a novel backdoor training mechanism, *Adversarial Backdoor Injection*, to increase the parameter-space backdoor stealthiness. Specifically, each neuron's Upper bound of Channel Lipschitz Condition (UCLC [131]) is calculated, based on which the weights of these suspicious neurons are set

Algorithm 6.1 UPGD Generation Algorithm

```
Input: Surrogate model $f_{\theta_{sur}}$, training data $\mathcal{D}$, perturbation budget $\epsilon$, the number
    of iteration $\mathcal{I}$, the target class $t$.
Output: UPGD $\boldsymbol{\delta}$
$S = B(\boldsymbol{\delta}; \epsilon) = \{\boldsymbol{\delta} \in \mathbb{R}^{d_c \times d_h \times d_w} : ||\boldsymbol{\delta}||_\infty \leq \epsilon\}$
$\boldsymbol{\delta} \leftarrow$ random_initialization $\wedge \boldsymbol{\delta} \in S$
for $i \in (0, \mathcal{I} - 1)$ do
    $\boldsymbol{x} \leftarrow$ sample_batch($\mathcal{D}$)
    $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\in\mathcal{D}} \ell(f_{\theta_{sur}}(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}), t),$
    $\boldsymbol{\delta} \leftarrow \min_{\boldsymbol{\delta}\in S} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$
end for
```

to the mean of all neurons' weights in the corresponding layer after every training epoch. In our implementation, we use the weights before every batch normalization as the neuron weights corresponding to the channel setting in UCLC. We prune neurons by substituting their weights with the mean ones because pruning to zeros makes the training unable to converge in our experiments. Formally, the k_{th} parameter of the l_{th} layer, $\boldsymbol{\theta}_l^{(k)}$, is updated as follows:

$$\boldsymbol{\theta}_l^{(k)} := \begin{cases} \text{mean}(\boldsymbol{\theta}_l), & \sigma(\boldsymbol{\theta}_l^{(k)}) > \text{mean}(\sigma(\boldsymbol{\theta}_l)) + u \times \text{std}(\sigma(\boldsymbol{\theta}_l)) \\ \boldsymbol{\theta}_l^{(k)}, & \text{otherwise,} \end{cases} \tag{6.3}$$

where u is a fixed threshold and σ is the UCLC value of the given weights. The measure for quantifying backdoor relevance can be changed from UCLC to others, such as the distance of neuron outputs when receiving benign and backdoor inputs, where a larger distance means the neuron is more relevant to backdoor behaviors and can be pruned. We use the modified UCLC for training efficiency as UCLC is data-free, which does not require calculation based on the outputs of neurons.

In adversarial training [12], adversarial examples are introduced during training to increase the model's robustness during inference. Similarly, during the Adversarial Backdoor Injection, we use backdoor defenses to increase the resistance of backdoor attacks to parameter-space defenses. At the end of each training epoch, Adversarial Backdoor Injection prunes the trained model to decrease the weights of backdoor neurons. Iteratively, backdoored neurons spread across the whole model instead of forming a few prominent backdoor neurons, as illustrated in Figure 6.1.

Feature-space stealthiness. We hypothesize that feature-space stealthiness is a by-product of parameter-space and input-space stealthiness since the variation of feature maps is strongly correlated with model parameters and inputs. Figures 6.3 and 6.6 show that Grond can substantially increase the feature-space stealthiness. Detailed experimental

Table 6.2: Pruning-based mitigations against backdoored ResNet18 on CIFAR-10. BA refers to benign accuracy on clean data, ASR to attack success rate, and PR to the poisoning rate of the training set. The average drop of BA and ASR is also shown with downward arrows compared to the performance without any defense. Red marks indicate the attack failed to resist the defense with an ASR lower than 60%, and green means that the ASR is higher than 60%.

Attack	No Defense		FP [129]		ANP [130]		CLP [131]		RNP [132]		Average	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets [8]	93.13	100	92.42	71.71	91.6	1.06	88.99	49.02	84.04	13.82	89.26 ↓3.87	33.90 ↓66.1
Blend [9]	94.42	100	93.08	99.99	93.57	0.33	90.3	0.54	94.63	57.98	92.89 ↓1.53	39.71 ↓60.29
WaNet [10]	93.60	99.37	92.96	4.60	91.08	0.49	91.53	2.12	92.86	3.17	92.11 ↓1.49	2.59 ↓96.78
IAD [11]	92.88	97.10	91.96	1.22	92.84	0.71	92.24	0.74	92.72	0.42	92.44 ↓0.44	0.77 ↓96.33
AdvDoor [102]	93.97	100	93.37	98.69	91.46	28.83	89.22	6.13	90.17	44.60	91.05 ↓2.92	44.56 ↓55.44
Bpp [105]	94.19	99.93	93.38	18.89	92.96	2.97	93.37	1.89	92.2	5.79	92.98 ↓1.21	7.39 ↓92.54
LC [266]	94.31	100	92.22	93.57	91.02	24.43	90.96	0.38	82.70	33.60	89.23 ↓5.08	37.99 ↓62.01
Narcissus [57]	93.58	99.64	93.49	96.54	89.76	49.18	93.19	97.82	91.10	94.59	91.88 ↓1.7	84.53 ↓15.11
SSDT [117]	93.70	90.30	93.41	0.80	93.88	0.60	93.66	1.20	93.99	3.30	93.74 ↑0.04	1.47 ↓88.83
Adap-Blend [118]	92.74	99.67	92.06	95.50	86.48	67.73	92.49	99.62	78.63	1.56	87.42 ↓5.32	66.10 ↓33.57
Grond (PR=5%)	93.43	98.04	93.09	99.73	91.43	94.01	93.29	87.89	91.83	85.22	92.41 ↓1.02	91.71 ↓6.33
Grond (PR=1%)	94.26	93.51	93.31	96.32	92.94	91.48	94.33	87.56	92.13	94.87	93.18 ↓1.08	92.56 ↓0.95
Grond (PR=0.5%)	94.36	92.91	93.32	90.96	93.87	84.04	94.52	86.82	91.99	84.63	93.43 ↓0.93	86.61 ↓6.3

analysis can be found in Section 6.4.2.

Table 6.3: Fine-tuning-based mitigations against backdoored ResNet18 on CIFAR-10.

Attack	vanilla FT		FT-SAM[133]		I-BAU[134]		FST[135]		BTI-DBF[112]		Average	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets [8]	91.07	43.96	92.01	2.84	90.87	97.48	92.4	13.10	91.26	13.12	91.52 ↓2.06	34.10 ↓65.9
Blend [9]	91.64	99.61	92.52	1.73	91.84	8.84	93.4	100	91.86	100	92.25 ↓2.17	62.04 ↓37.96
WaNet [10]	91.11	0.99	90.89	1.03	87.98	0.81	92.17	0.04	90.3	4.89	90.49 ↓3.11	1.55 ↓97.82
IAD [11]	90.83	2.16	92.18	2.87	88.4	15.68	91.29	0.0	89.54	1.59	90.45 ↓2.43	4.46 ↓92.64
AdvDoor [102]	91.25	68.68	92.18	1.23	89.29	16.99	91.06	99.99	90.25	100	90.81 ↓3.16	57.38 ↓42.62
Bpp [105]	91.36	3.4	91.38	1.00	92.06	6.46	93.23	26.83	90.61	2.73	91.73 ↓2.46	8.08 ↓91.85
LC [266]	90.26	88.52	91.46	1.91	85.87	5.11	91.8	13.11	90.71	4.37	90.02 ↓4.29	22.60 ↓77.4
Narcissus [57]	91.70	92.91	91.76	23.98	91.48	51.74	90.06	54.22	90.94	98.11	91.19 ↓2.39	64.19 ↓35.45
SSDT [117]	93.74	0.70	93.15	0.60	90.27	3.10	92.85	0.2	90.79	1.40	92.16 ↓1.54	1.20 ↓89.10
Adap-Blend [118]	92.42	98.73	91.23	22.4	85.38	37.31	90.91	1.19	89.17	7.09	89.82 ↓2.92	33.34 ↓66.33
Grond (PR=5%)	91.75	94.28	92.02	80.07	90.39	93.92	93.27	99.92	91.88	99.00	91.86 ↓1.57	93.44 ↓4.6
Grond (PR=1%)	91.41	85.52	92.83	79.17	87.89	91.34	93.21	96.59	90.66	88.69	91.20 ↓3.06	88.26 ↓5.25
Grond (PR=0.5%)	91.42	82.96	92.34	76.92	89.83	79.68	93.44	92.71	90.39	91.83	91.48 ↓2.88	84.82 ↓8.09

6.4 Experimental Evaluation

This section contains the main experimental results and analysis. Section 6.4.1 covers the datasets, baseline attacks, and defenses used in our experiments. Section 6.4.2 evaluates common backdoor attacks and Grond against pruning- and fine-tuning-based defenses. Section 6.4.3 provides an in-depth backdoor analysis, followed by Section 6.4.4, which explores how ABI can enhance common attacks. Section 6.4.5 covers backdoor model and input detection results, and Section 6.4.6 provides a comparison to supply-chain attacks. Finally, Section 6.4.7 covers the ablation study.

Table 6.4: Backdoor performance of Grond and baseline attacks on ImageNet200 and GTSRB.

Datasets	Attack	No Defense		FT-SAM [133]		I-BAU [134]		CLP [131]		Average	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
IN200	BadNets [8]	80.65	91.03	79.89	2.21	70.28	26.06	70.74	64.86	73.64 ↓7.01	31.04 ↓59.99
	Blend [9]	80.70	95.63	80.19	0.39	76.13	30.81	80.02	23.38	78.78 ↓1.92	18.19 ↓77.44
	WaNet [10]	81.24	99.97	80.41	0.66	75.67	47.27	77.18	99.78	77.75 ↓3.49	49.24 ↓50.73
	IAD [11]	79.74	99.98	75.49	0.68	77.44	15.18	76.97	84.49	76.63 ↓3.11	33.45 ↓66.53
	AdvDoor [102]	80.72	100	79.52	98.90	74.03	61.31	77.90	100	77.15 ↓3.57	86.74 ↓13.26
	Bpp [105]	81.36	92.74	79.37	1.05	76.53	3.21	80.10	2.34	78.67 ↓2.69	2.19 ↓90.55
	Narcissus [57]	81.73	81.28	80.00	83.37	77.03	56.19	80.99	86.37	79.34 ↓2.39	75.31 ↓5.97
	SSDT [117]	75.45	100	78.19	76.00	76.26	22.00	76.02	94.00	76.82 ↑1.37	64.00 ↓36.00
	Grond	80.92	94.11	79.05	95.05	76.89	87.75	80.29	93.83	78.74 ↓2.18	92.21 ↓1.9
GTSRB	BadNets [8]	97.19	100	95.57	0.48	92.02	29.22	96.38	0.47	94.66 ↓2.53	10.06 ↓89.94
	Blend [9]	95.92	100	93.36	0.21	92.64	38.27	93.21	0.00	93.07 ↓2.85	12.83 ↓87.17
	WaNet [10]	98.69	99.77	92.18	0.45	91.25	0.00	90.14	18.14	91.19 ↓7.50	6.19 ↓93.58
	IAD [11]	99.08	99.65	92.72	0.10	90.11	0.35	98.08	14.63	93.64 ↓5.44	5.03 ↓94.62
	AdvDoor [102]	95.80	99.99	93.94	32.26	92.67	38.20	90.09	66.39	92.23 ↓3.57	45.62 ↓54.37
	Bpp [105]	98.69	99.93	91.27	0.00	92.61	0.23	97.16	2.29	93.68 ↓5.01	0.84 ↓99.09
	Narcissus [57]	95.60	97.18	93.61	54.55	92.87	80.74	93.99	97.60	93.49 ↓2.11	77.63 ↓19.55
	SSDT [117]	96.02	77.78	93.11	0.00	90.82	0.00	94.65	19.31	92.86 ↓3.16	6.44 ↓71.34
	Grond	95.83	95.36	93.80	71.84	93.13	94.30	91.28	93.19	92.74 ↓3.09	86.44 ↓8.92

6.4.1 Experimental Setup

Datasets and Architectures. We follow the common settings in existing backdoor attacks and defenses and conduct experiments on CIFAR-10 [250], GTSRB [272], and a subset of ImageNet [187] with 200 classes and 1,300 images per class (ImageNet200). More details about the datasets can be found in Appendix 6.7.2. The primary evaluation is performed using ResNet18 [27]. Moreover, we evaluate Grond using four additional architectures, VGG16 [26], DenseNet121 [274], EfficientNet-B0 [249], and one recent architecture InceptionNeXt [287] (see Table 6.15 in Appendix 6.7.9).

Attack Baselines. Grond is compared with 12 representative attacks: BadNets [8], Blend [9], WaNet [10], IAD [11], AdvDoor [102], BppAttack [105], LC [266], Narcissus [57], Adap-Blend [118], SSDT [117], DFST [121], and DFBA [108]. The default poisoning rate is set at 5% (of the training set) for all attacks following previous work [112, 139]. Additionally, Grond is evaluated under various poisoning rates to provide a thorough analysis of its effectiveness. Following related works, the training schedule for attacks is 200 epochs when using CIFAR10 and GTSRB, and 100 epochs for ImageNet200. We use 1,000 images as the validation set to select the best-performing checkpoint. More implementation details and reasoning are provided in Appendix 6.7.3.

Defense Baselines. We evaluate Grond and baseline attacks with 17 defenses, including **four pruning-based** methods (FP [129], ANP [130], CLP [131], and RNP [132]), **five fine-tuning-based** methods (vanilla FT, FT-SAM [133], I-BAU [134], FST [135], and BTI-DBF(U) [112]), **five backdoor model detections** (NC [110], Tabor [144], FeatureRE [111], Unicorn [128], and BTI-DBF [112]), **two backdoor input detections** (Scale-up [263] and IBD-PSC [283]), and a **proactive defense** CT [137]. Following their default settings, BTI-DBF [112] and FP [129] use 5% of training data, and other defenses use 1% of training data for detection or mitigation. CLP is a data-free backdoor pruning tool that uses no clean data. CT has access to the complete training set without knowing which samples are poisoned and is also able to interact with the model during training. Backdoor defense details and hyperparameters can be found in Appendix 6.7.5.

6.4.2 Main Results on Backdoor Mitigation

All evaluated backdoor attacks are ineffective to at least one parameter-space backdoor defense on the CIFAR-10, as demonstrated in Tables 6.2 and 6.3. It suggests that common backdoor attacks designed to be stealthy in input and feature spaces are vulnerable to parameter-space defenses. Given that all backdoor behaviors are embedded in parameters of backdoored models, this finding suggests that future backdoor attacks should take parameter-space defenses into account as a standard step to evaluate comprehensive stealthiness.

Not surprisingly, Grond's attack performance is better than all baseline attacks when considering evaluated backdoor defenses since Grond is designed to consider comprehensive stealthiness. On four pruning-based mitigations, Grond achieves 7.18% absolute higher ASR on average than the best backdoor attack, Narcissus. On five fine-tuning mitigations that show more powerful defense capability than pruning-based mitigations, Grond achieves 29.25% absolute higher ASR on average than Narcissus. In addition, Grond bypasses the five model detection and two input-space detections (see Section 6.4.5).

Pruning-based mitigation. We take a closer look at the details of pruning-based backdoor mitigation experiments in Table 6.2, presenting the results of all attacks against four pruning-based defenses. BadNets and Blend perform better on average than input-space stealthy attacks, e.g., WaNet and Bpp, because input-space stealthy attacks introduce significant separability in the feature space (see Figure 6.3). Across all pruning-based defenses, FP performs the worst, as expected, since it follows regular model pruning practice and is not a tailored backdoor pruning method.

Fine-tuning-based mitigations. Table 6.3 presents the backdoor performance against five fine-tuning-based defenses. In general, fine-tuning-based defenses are more effective than pruning-based defenses. For example, Narcissus and Adap-Blend can achieve ASRs higher than 60% against three out of four pruning-based defenses but are much less effective against most fine-tuning-based methods. FT-SAM is the most effective across all defenses, as shown in Tables 6.2 and 6.3, being able to compromise the effectiveness of all attack baselines. One important reason is that FT-SAM adopts Sharpness-Aware Minimization [288] to adjust the outlier of weight norm (large norms) to remove the potential backdoor. Larger weights of neurons are introduced by existing attacks to guarantee a high ASR [127], which also causes large differences when receiving benign and backdoor inputs (see Figure 6.4). Grond can bypass FT-SAM, as expected, since it deliberately decreases the weights of backdoor neurons, compromising the core working mechanism of FT-SAM.

On ImageNet200 and GTSRB. Real-world classification tasks may involve more categories, such as GTSRB (43 classes) and ImageNet200 (200 classes), and the percentage of each class in the dataset will commonly be much less than 10%. We target InceptionNext-Small on Imagenet200 and ResNet18 on GTSRB. The l_∞ norm perturbation budget of UPGD is $\epsilon = 16$ for GTSRB and $\epsilon = 8$ for ImageNet200 to achieve imperceptible perturbations. Table 6.4 demonstrates that Grond is still effective on datasets with more classes and higher resolutions, especially against the most powerful parameter-space defense, FT-SAM.

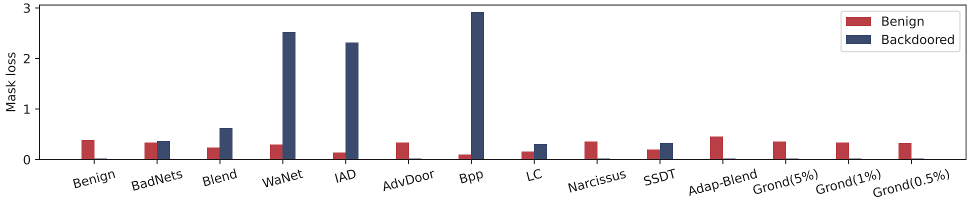


Figure 6.3: Benign and inversed backdoor feature loss (Eq. (6.4)) for all baseline attacks. Large backdoored loss indicates that the backdoor is prominent in the feature space.

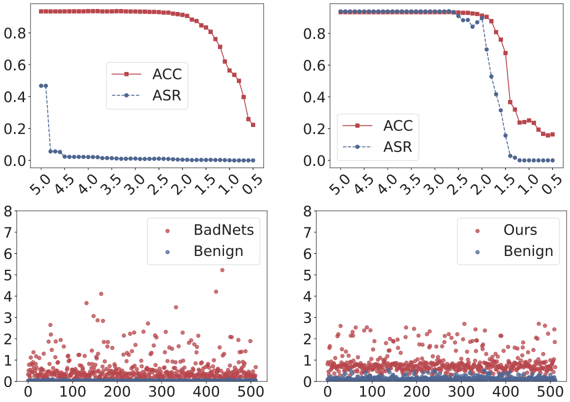


Figure 6.4: We show the performance of pruning neurons with high TAC values using different thresholds. The left column is BadNets, and the right is Grond.

6.4.3 Backdoor Analysis

Adaptive backdoor analysis by TAC pruning. Backdoor triggers are essential for calculating TAC values, making our TAC-based analysis highly adaptive to evaluating backdoor attacks. We directly utilize the trigger information to build a new pruning method based on the TAC analysis. In particular, we prune neurons with high TAC values in the backdoored model. Figure 6.4 shows the pruning results of Grond and BadNets, and TAC analysis for other attacks are in Appendix 6.7.11. The first row of Figure 6.4 provides the pruning results. The second row contains the TAC values plots of neurons in the 4_{th} layer (the layer before the classification head) of ResNet18. For BadNets, the backdoor and benign behaviors of baseline attacks can be disentangled by pruning neurons with high TAC values. However, for Grond, pruning neurons with high TAC values decreases benign accuracy, which means the backdoor neurons are not easily distinguishable from benign neurons without harming benign performance. The analysis supports our statement that Grond spreads the backdoor to more neurons instead of a few prominent ones. We provide a sorted TAC value plot in Figure 6.9 in the appendix, showing the prominent neurons with high TAC values are quite limited in Grond.

Backdoor analysis by feature space inversion. We also provide a feature space analysis for different attacks by using a feature mask to decouple the benign and backdoor features following BTI-DBF [112] and BAN [139]. The decoupling assumes that benign features related to the correct prediction introduce a lower loss, and the backdoor features related to backdoor prediction introduce a higher loss. In particular, the benign and inversed backdoor features are decoupled as follows:

$$\min_{\boldsymbol{m}} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_l} \Big[\mathcal{L}\big(f_L \circ (g(\boldsymbol{x}) \odot \boldsymbol{m}), y\big) - \mathcal{L}\big(f_L \circ (g(\boldsymbol{x}) \odot (1-\boldsymbol{m}), y)\big) + \lambda|\boldsymbol{m}|\Big], \tag{6.4}$$

where $g = f_{L-1} \circ \cdots f_1$, without the classification head. $\boldsymbol{m}$ is the learned feature mask, and $\mathcal{D}_l$ is a small set of benign samples with correct labels. As validated with BTI-DBF [112] and BAN [139], benign and backdoor features can be decoupled by the mask $\boldsymbol{m}$, after which backdoor features will introduce a substantially higher loss with respect to the ground truth label. More details can be found in Appendix 6.7.7.

Decoupled benign feature loss and backdoor feature loss of all evaluated attacks are demonstrated in Figure 6.3, where benign feature loss is represented by the first term in Eq. (6.4) and backdoor feature loss by the second term. It can be observed that several backdoor attacks introduce prominent backdoor features, such as WaNet, IAD, and Bpp, which result in substantially higher backdoor feature loss than benign feature loss. In contrast, several backdoor attacks introduce less prominent backdoor features, including AdvDoor, Narcissus, Adap-Blend, and Grond. Revisiting Tables 6.2, 6.3, and 6.4, attacks that introduce prominent backdoor features are more susceptible to backdoor defenses than attacks with less prominent backdoor features. Both TAC and feature space inversion analyses further confirm that Grond provides comprehensive stealthiness.

6.4.4 ABI Improves Common Backdoor Attacks

In this section, we show that our Adversarial Backdoor Injection (ABI) strategy generalizes to all evaluated common backdoor attacks. We combine the ABI module with baseline attacks to improve their resistance against parameter-space defenses. Figure 6.5 demonstrates that ABI is effective for all attacks when evaluating against the parameter-space defense ANP, where ASRs increase after adversarial injection, especially for BadNets, Blend, AdvDoor, Narcissus, and Adap-Blend. The improvement for feature space attacks (WaNet, IAD, and Bpp) is incremental. We speculate that feature space attacks rely too much on prominent features as their modification in the input space is minor. To activate the backdoor with such minor input modifications, the prominent features are required in the feature space. In addition, Figure 6.8 in Appendix 6.7.10 shows the results of ABI without defense, demonstrating that it does not harm in general the BA and ASR when no defense is applied. Following our finding, we suggest that future backdoor attacks can use ABI to increase parameter-space stealthiness.

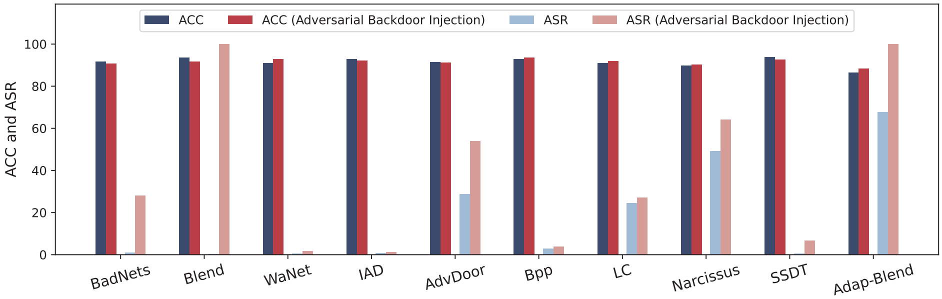


Figure 6.5: BA and ASR of backdoor attacks before and after ABI against parameter-space defense ANP.

Table 6.5: Backdoor detection performance on CIFAR-10. 20 ResNet18 models are trained at each poisoning rate. Bd. refers to the number of models determined as backdoor models. Acc. refers to the detection accuracy.

Defense	PR=5%		PR=1%		PR=0.5%	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
NC [110]	5	25%	2	10%	1	5%
Tabor [144]	5	25%	2	10%	0	0%
FeatureRE [111]	0	0%	0	0%	0	0%
Unicorn [128]	0	0%	0	0%	0	0%
BTI-DBF [112]	3	15%	5	25%	3	15%

Table 6.6: Comparsion with supply-chain attacks.

Attack	No Defense		CLP [131]		FT-SAM [133]	
	BA	ASR	BA	ASR	BA	ASR
DFST [121]	95.23	100	92.43	3.53	94.70	0.00
DFBA [108]	88.99	100	88.96	9.57	86.03	5.24
SSDT [117]	93.70	90.30	93.66	1.20	93.15	0.60
Grond	93.43	98.04	93.29	87.89	92.02	80.07

6.4.5 Backdoor Detection

Following previous works [112, 139], we choose five representative backdoor model detections for evaluation. The model detection refers to determining whether a given model is backdoored. We use 20 models for each poisoning rate with different random seeds. Then, we report the number of models detected as backdoor models out of the 20. Table 6.5 shows that all detections fall short when detecting Grond. In particular, NC [110], Tabor [110], and BTI-DBF [112] can detect a small part of backdoored models, while FeatureRE [111] and Unicorn cannot detect any of them. For featureRE [111], we conjecture it is over-dependent on the separability in the feature space, but Grond does not rely on prominent backdoor features according to Figure 6.3. For Unicorn [128], the false positive rate is high, and it tends to report every class as the backdoor target, even on models trained with benign data only. Except for model detection, Grond can also bypass input-space detections as demonstrated in Appendix 6.7.8.

6.4.6 Comparison with Supply-Chain Attacks

Supply-chain backdoor attacks assume the adversary could directly manipulate models' parameters or control the backdoor training process for more powerful and stealthy backdoors. The backdoored model is provided as a service or final product to end users. Supply-chain backdoor attacks are attracting increasing industry and research community attention because of their stealthiness and significant real-world impact [51].

Sharing a similar threat model to supply-chain attacks, we compare Grond and three state-of-the-art supply-chain attacks, where these attacks are also designed to be robust against backdoor defenses. In particular, DFST [121] proposes to include a controlled detoxification technique in the training process, which restrains the model from picking up simple features. DFBA [108] directly modifies a few parameters of a classifier to inject a backdoor. SSDT [117] introduces additional terms in the loss for the Source-Specific and Dynamic-Triggers (i.e., SSDT) attack, which obscures the difference between normal samples and malicious samples. Table 6.6 shows that supply-chain attacks can also be defeated by existing backdoor defenses. The ASR of DFST [121], DFBA [108], and SSDT [117] are decreased to less than 10% while the BA drop is less than 3%.

Table 6.7: Comparsion with different strategies for the generation of backdoor triggers.

Strategy	No Defense		CLP [131]		FT-SAM [133]	
	BA	ASR	BA	ASR	BA	ASR
Random noise	94.24	1.28	94.13	0.97	93.90	1.84
PGD	94.77	69.33	92.57	46.63	92.40	24.56
UPGD	93.43	98.04	93.29	87.89	92.02	80.07

Table 6.8: Ablation study for Grond.

Arch	Method	No Defense		CLP [131]		FT-SAM [133]	
		BA	ASR	BA	ASR	BA	ASR
ResNet18	UPGD	93.86	98.61	91.15	3.97	91.80	51.77
	+ABI	93.43	98.04	93.29	87.89	92.02	80.07
InceptionNeXt	UPGD	87.81	96.81	87.72	96.57	87.06	2.37
	+ABI	87.06	96.86	86.93	96.87	86.50	92.02

6.4.7 Ablation Study

Trigger generation. To explore the influence of trigger patterns, we employ and evaluate three types of triggers: random noise, PGD perturbation, and UPGD perturbation, using ResNet18 on CIFAR-10. The random noise is sampled from a uniform distribution, and the PGD employs a projected gradient descent to generate sample-wise perturbations [12]. The generation of UPGD is described in Algorithm 6.1. All three triggers are limited to 8/255 (l_∞ norm) for imperceptibility and use the same training settings described in Table 6.11 in Appendix 6.7.3.

We show in Table 6.7 that random noise is ineffective as a backdoor trigger, with an ASR around 1%, even if no defense is applied. The sample-wise PGD perturbation is more effective than random noise and shows (limited) robustness against CLP and FT-SAM. UPGD generates the most effective backdoor trigger with an ASR higher than 80% after CLP and FT-SAM, and we speculate that the reason is that UPGD exploits features from the target class, similar to Narcissus [57].

Adversarial backdoor injection is critical. There are two components in Grond: the UPGD trigger generation and Adversarial Backdoor Injection. We conduct an ablation study with two architectures on CIFAR-10 to analyze the impact of the ABI component. As shown in Table 6.8, after removing the ABI component, CLP or FT-SAM can defend against the clean-label attack with the UPGD trigger. Thus, the Adversarial Backdoor Injection is the key component in maintaining the effectiveness of backdoor attacks against parameter-space defenses.

Dirty-label Grond. Grond works well in a clean-label setting that uses an invisible

Table 6.9: Evaluation of dirty-label Grond using ResNet18 on CIFAR-10.

Dirty-Label Grond	No Defense		CLP [131]		FT-SAM [133]	
	BA	ASR	BA	ASR	BA	ASR
PR=5%	91.60	100	91.41	100	90.24	0.00
PR=1%	93.64	100	91.13	40.83	91.20	46.97
PR=0.5%	94.35	100	91.84	97.03	91.77	99.04

Table 6.10: Evaluation with the proactive defense, CT [137], under different poisoning rates (PR).

Attack	PR	ACC	ASR	Recall	FPR
BadNets [8]	5%	93.18	99.96	2500/2500	1568/47500
	2.5%	93.35	99.83	1250/1250	518/48750
	1%	93.30	100	500/500	73/49500
	0.5%	93.43	100	250/250	5/49750
	0.3%	93.63	99.94	150/150	222/49850
Adap-Patch [118]	5%	93.28	100	1808/2500	116/47500
	2.5%	93.68	100	1088/1250	20/48750
	1%	93.73	100	494/500	570/49500
	0.5%	93.31	100	160/250	154/49750
	0.3%	93.26	100	86/150	3825/49850
Grond	5%	93.84	99.41	2499/2500	671/47500
	2.5%	93.81	95.83	115/1250	7220/48750
	1%	94.09	92.48	208/500	6690/49500
	0.5%	94.36	92.91	90/250	6738/49750
	0.3%	94.22	90.10	29/150	6349/49850

trigger and does not change the original labels of the poisoned samples. However, as we use the supply-chain threat model (the attacker has access to the training process), we could also explore the effect of a dirty-label backdoor attack. A dirty-label threat model could simplify the backdoor by poisoning samples from any class while the clean-label is limited to a single class.

As shown in Table 6.9, Grond with a dirty-label threat model can still perform well and even with a higher ASR than the clean-label setting. However, dirty-label Grond is less robust than clean-label Grond against backdoor defenses because it is less stealthy. For example, FT-SAM [133] can decrease the ASR of dirty-label Grond (PR=5%) from 100% to 0%. When PR=1%, both CLP [131] and FT-SAM [133] can decrease the ASR from 100% to below 50%. We conjecture this is because the dirty-label setting obfuscates the benign semantics of images to their true labels, as the poisoned samples in each class have been assigned the backdoor target label. In contrast, in a clean-label setting, only the target class is poisoned.

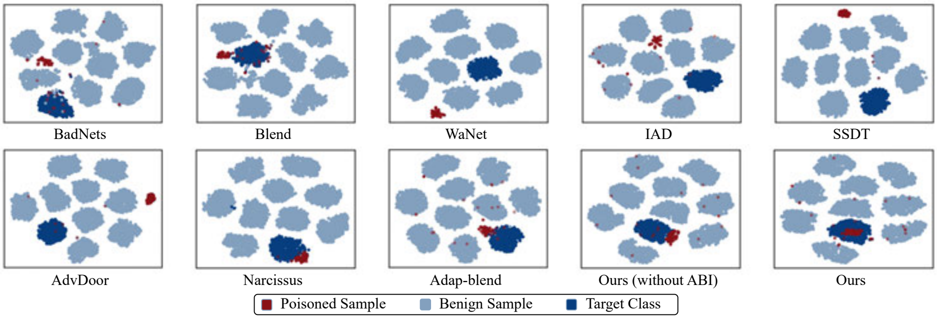


Figure 6.6: Examples of feature visualization of Grond and baseline attacks.

6.5 Stronger Defenders and Additional Analysis

This section dives deeper into stronger defenses and analysis of Grond. We evaluate Grond against a proactive defense in Section 6.5.1. Then, we provide analyses of Grond by Grad-CAM (defense) and feature spaces of different backdoor attacks in Section 6.5.2.

6.5.1 Proactivate Defense

Real-world powerful defenders could take more initiative by intervening proactively in the attack process and exploiting poisoned data. We evaluate Grond against the state-of-the-art proactive defense, CT [137], that detects poisoned samples in the training data. Specifically, CT considers data from the original poisoned training data as regular batches and introduces randomly labeled benign data as confusing batches. Then, CT performs normal supervised training on both regular and confusing batches to produce an inference model, aiming to corrupt benign semantic features and correlations with correct labels in the inference model by confusing batches. The backdoor effect remains in the inference model because there is no trigger information in the confusing batches, and correctly predicted samples by the inference model are recorded as poisoned.

We apply CT to our adversarial backdoor injection process. Table 6.10 presents the detection results on two baseline attacks (BadNets [8] and Adap-Patch [118]) and Grond. CT is effective against the two baseline attacks, where most poisoned samples in the training set are detected with a relatively low false positive rate. However, CT is not capable of detecting Grond when the poisoning rate is lower than 5% due to a high false positive rate and low recall. To understand why CT is not effective against Grond, we recall the main idea of CT is to corrupt benign semantic features and their correct label but not corrupt backdoor semantic features. However, Grond utilizes the benign semantic features of the target class to generate UPGD perturbation as the trigger. CT's mechanism also corrupts the backdoor features of Grond. Therefore, CT cannot effectively detects Grond poisoned samples.

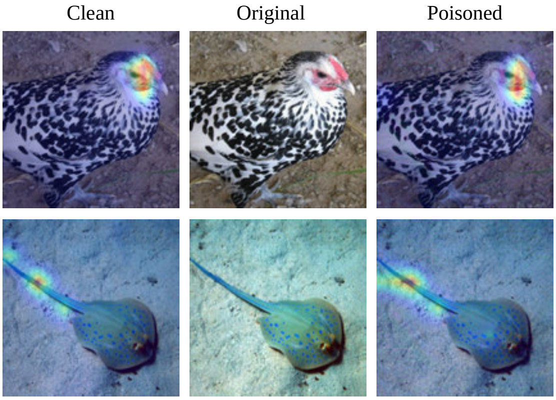


Figure 6.7: Examples of Grad-CAM activation map with ImageNet200 images by clean and Grond models. The first column is Grad-CAM maps with clean images, and the third column is Grad-CAM maps with Grond-poisoned images.

6.5.2 Visualization

Grad-CAM cannot spot the trigger area of Grond. Grad-CAM [289] was originally designed to visualize the network's preference when taking an input image. In backdoor defense research, Grad-CAM is leveraged to highlight the important areas in order to detect the potential backdoor trigger area [290]. Figure 6.7 shows the activated area of a clean model and Grond backdoored model using Grad-CAM. The activated area of Grond backdoored model is indistinguishable from the clean model, so the Grad-CAM-based defense [290] is also ineffective against Grond.

t-SNE visualization of feature space. Figure 6.6 shows the latent feature (feature space of the last convolutional layers) from Grond backdoor models with and without adversarial backdoor injection in 2-D space and other baselines attacks by t-SNE [291]. The poisoning rate for all is 0.5%. WaNet cannot achieve satisfactory ASR at this very low poisoning rate, so we use the default setting according to their open-source implementation. Specifically, we perform dimensionality reduction for the activation of the last convolutional layers by t-SNE. The model architecture is ResNet18 and trained on CIFAR-10. Each class of samples forms a tight cluster, and Grond poisoned samples are better mixed with the target class samples when the model is trained with adversarial backdoor injection.

6.6 Conclusions & Future Work

This chapter studies whether backdoor attacks can resist diverse practical defenses and provides an affirmative answer: current common stealthy backdoor attacks are vulnerable to parameter-space defenses. We further explore how to increase the stealthiness of backdoor attacks against parameter-space defenses. We propose a novel supply-chain backdoor attack, Grond, that considers comprehensive stealthiness, including input, feature, and parameter-space stealthiness. Grond achieves state-of-the-art performance by leveraging adversarial examples and adaptively limiting the backdoored model's parameter changes during the backdoor injection to improve the stealthiness. We also show that Grond's Adversarial Backdoor Injection can consistently improve other backdoor attacks against parameter space defenses. We suggest that future backdoor attacks should be evaluated against parameter-space defense. We also recommend that backdoor research explore Adversarial Backdoor Injection to enhance parameter-space stealthiness.

6.7 Appendix

6.7.1 Additional Details about Experimental Settings

6.7.2 Datasets

CIFAR-10. The CIFAR-10 [250] contains 50,000 training images and 10,000 testing images with the size of 3 × 32 × 32 in 10 classes.

GTSRB. The GTSRB [272] contains 39,209 training images and 12,630 testing images in 43 classes. In our experiments, the images are resized to 3 × 32 × 32.

ImageNet200. ImageNet [187] contains over 1.2 million high-resolution images in 1,000 classes. In our experiments, we randomly select 200 classes from the ImageNet dataset as our ImageNet200 dataset. Each class has 1,300 training images and 50 testing images. The ImageNet images are resized to 3 × 224 × 224.

6.7.3 Backdoor Attacks

Our attack is compared with 12 well-known and representative attacks: BadNets [8], Blend [9], WaNet [10], IAD [11], AdvDoor [102], BppAttack [105], LC [266], Narcissus [57], Adap-Blend [118], SSDT [117], DFST [121], and DFBA [108].

Like Narcissus, our attack uses the class bird (CIFAR10) as the target class. For ImageNet200, we use the stingray as the target class. The Grond poisoning rate (ImageNet200) used for results in Table 6.4 is 0.5%. For GTSRB, we use the speed limit (50) as the target class. The Grond poisoning rate (GTSRB) used for results in Table 6.4 is 1.74%. AdvDoor uses the same trigger and target class as ours. More details are provided in Table 6.11. For other attacks and hyperparameters not mentioned, we use the default setting from the original papers or open-source implementations.

Table 6.11: The backdoor training settings.

Config	Value
Optimizer	SGD, AdamW (InceptionNeXt)
Weight decay	5×10^{-4}
learning rate	0.01
epoch	200 (GTSRB, CIFAR10), 100 (ImageNet200)
learning rate schedule	MultiStepLR (100, 150) for CIFAR10 and GTSRB, CosineAnnealingLR for ImageNet200
poison rate	0.05
u in Eq. (6.3)	3.0
BadNets trigger	3×3
Blend trigger	random Gaussian noise and blend ratio 0.2
Adap-Blend trigger	“hellokitty_32.png” and blend ratio of 0.2
Narcissus trigger size	$\epsilon = 16$ for both inference and training

6.7.4 Attack Summary

In Table 6.12, we summarize the attacks evaluated in this chapter and compare them with Grond. Grond is the only one that achieves stealthiness in input, feature, and parameter spaces.

Table 6.12: A summary of attacks evaluated in this chapter. SS refers to Source Specific.

Attack	Threat Model			Trigger Type			Trigger Strategy		Stealthy Level		
	Data	Label	Training	Patch	Blend	Dynamic	A2A	SS	Input	Feature	Parameter
BadNets	●	○	○	●	○	○	●	○	○	○	○
Blend	●	○	○	○	●	○	●	○	○	○	○
WaNet	●	○	○	○	○	●	●	○	●	○	○
IAD	●	○	○	○	○	●	●	○	○	○	○
AdvDoor	●	○	○	○	●	○	●	○	●	●	○
Bpp	●	○	○	○	○	●	●	○	●	○	○
LC	●	●	○	●	○	○	○	○	○	○	○
Narcissus	●	●	○	○	●	○	○	○	○	●	○
SSDT	●	○	●	○	○	●	○	●	○	●	○
Adap-Blend	●	○	○	○	●	○	○	○	○	●	○
DFST	●	○	●	○	●	○	○	○	○	●	○
DFBA	○	○	●	●	○	○	○	○	○	●	○
Grond	●	●	●	○	●	○	○	○	●	●	●

○ the item is not supported by the defense; ● the item is supported by the defense.

6.7.5 Backdoor Defenses

We evaluate our attack and baseline attacks against 17 defenses, including **4 pruning-based methods** (FP [129], ANP [130], CLP [131], and RNP [132]), **5 fine-tuning-based methods** (vanilla FT, FT-SAM [133], I-BAU [134], FST [135], and BTI-DBF(U) [112]), **5 backdoor model detections** (NC [110], Tabor [144], FeatureRE [111], Unicorn [128], and BTI-DBF [112]), **2 backdoor input detections** (Scale-up [263] and IBD-PSC [283]), and **a proactive detection** CT [137].

ANP[†], **CLP**[‡], **RNP**[§], **FST**[¶], **BTI-DBF**[‖], **BTI-DBF(U)**[‖], **FeatureRE**[**], **Unicorn**[††]. We use the implementation and default hyperparameters from their open-source code.

FP[‡‡], **vanilla FT**[§§], **FT-SAM**[¶¶], **I-BAU**[***]. We use the implementation and default hyperparameters from BackdoorBench [292]. For FT-SAM on ImageNet200, the default setting will decrease benign accuracy to 0.465, so we reduce its training schedule to 25 epochs. Please note that the experiments on CIFAR10 with FT-SAM usually converge within 20 epochs in our experiments. Thus, decreasing the training schedule is not harmful to the defense performance.

NC[†††] **and Tabor**[‡‡‡]**.** We use the implementation from TrojanZoo [257]. 1% training set and 100 epoch are used for trigger inversion.

Scale-up[§§§], **IBD-PSC**[¶¶¶]. We use the implementation and default hyperparameters from BackdoorBox [293].

CT. We use the open-source code implementation. We reduced the number of distillation iterations to 200 for efficiency reasons.

6.7.6 Hyperparameters for Training Surrogate Models

Table 6.13 provides the hyperparameters for training surrogate models to generate UPGD.

6.7.7 Hyperparameters for the Inversed Backdoor Feature Loss

Following the settings in BTI-DBF [112] and BAN [139], we use Adam and the learning rate of 0.01 to search for 20 epochs for the feature mask in Eq. (6.4). The optimization of

[†] https://github.com/csdongxian/ANP_backdoor/tree/main

[‡] https://github.com/rkteddy/channel-Lipschitzness-based-pruning

[§] https://github.com/bboylyg/RNP

[¶] https://github.com/AISafety-HKUST/Backdoor_Safety_Tuning

[‖] https://github.com/xuxiong0214/BTIDBF/tree/master

[**] https://github.com/RU-System-Software-and-Security/FeatureRE/tree/main

[††] https://github.com/RU-System-Software-and-Security/UNICORN

[‡‡] https://github.com/SCLBD/BackdoorBench/blob/main/defense/fp.py

[§§] https://github.com/SCLBD/BackdoorBench/blob/main/defense/ft.py

[¶¶] https://github.com/SCLBD/BackdoorBench/blob/main/defense/ft-sam.py

[***] https://github.com/SCLBD/BackdoorBench/blob/main/defense/i-bau.py

[†††] https://github.com/ain-soph/trojanzoo/blob/main/trojanvision/defenses/backdoor/model_inspection/neural_cleanse.py

[‡‡‡] https://github.com/ain-soph/trojanzoo/blob/main/trojanvision/defenses/backdoor/model_inspection/tabor.py

[§§§] https://github.com/THUYimingLi/BackdoorBox/blob/main/core/defenses/SCALE_UP.py

[¶¶¶] https://github.com/THUYimingLi/BackdoorBox/blob/main/core/defenses/IBD_PSC.py

Table 6.13: The settings for training surrogate models.

Config	Value
Optimizer	SGD, AdamW (InceptionNeXt)
Weight decay	5×10^{-4}
learning rate	0.01 (CIFAR10, GTSRB), 0.001 (ImageNet200)
epoch	200 (GTSRB, CIFAR10), 100 (ImageNet200)
learning rate schedule	MultiStepLR (100, 150) for CIFAR10 and GTSRB, CosineAnnealingLR for ImageNet200

the mask uses 1% of training data. The λ is 0.72. The elements in the mask are limited to continuous values between 0 and 1.

Table 6.14: Input-space detection results.

Attack	Scale-up [263]				IBD-PSC [283]			
	TPR	FPR	AUC	F1	TPR	FPR	AUC	F1
BadNets [8]	81.93	32.90	0.7627	0.7524	100	7.90	0.9996	0.9606
Blend [9]	99.32	38.74	0.8681	0.8275	100	0.90	1.00	0.9953
Adap-Blend [118]	68.72	18.99	0.7621	0.7297	53.95	11.77	0.8731	0.6495
Grond (PR=5%)	24.40	17.69	0.5463	0.3409	0.00	10.33	0.5698	0.0
Grond (PR=1%)	18.39	17.96	0.4879	0.2656	0.00	5.82	0.0626	0.0
Grond (PR=0.5%)	7.05	16.19	0.4034	0.1113	0.00	4.82	0.1087	0.0

Table 6.15: Grond against defenses using different architectures on CIFAR-10 with a poisoning rate of 5%. The surrogate indicates the architecture used to generate UPGD as the trigger.

Victim	Surrogate	No Defense		FT-SAM [133]		I-BAU [134]		FST [135]	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR
VGG16	ResNet18	92.69	95.31	92.72	78.42	90.10	14.53	89.12	92.68
	VGG16	92.57	90.10	92.22	95.14	90.20	76.51	91.72	90.58
DenseNet121	ResNet18	92.39	95.62	90.98	23.88	86.73	48.14	90.77	88.94
	DenseNet121	92.38	81.07	91.10	16.91	90.90	54.76	91.13	71.29
EfficienNet-B0	ResNet18	87.7	96.23	84.05	71.07	87.64	95.41	82.07	97.67
	EfficienNet-B0	86.92	92.61	83.77	71.17	86.93	92.13	82.45	68.72
InceptionNeXt	ResNet18	85.07	91.83	85.07	2.17	85.25	91.67	82.78	3.82
	InceptionNeXt	85.54	96.24	85.64	90.14	85.49	97.21	83.92	97.29

6.7.8 Detection of backdoor input

Backdoor input detection is a defense technique that determines whether or not a given input includes a backdoor trigger. We show that Grond-generated backdoor samples can resist established backdoor detection methods. Table 6.14 shows the input-space detection results using Scale-up [263] and IBD-PSC [283]. We report the True Positive Rate (TPR), False Positive Rate (FPR), AUC, and F1 score in Table 6.14 for baseline attacks and Grond, where Scale-up and IBD-PSC are effective against three baseline attacks but cannot detect Grond-generated backdoor samples.

6.7.9 Different Architectures with Different Surrogate Models

We evaluate Grond with four additional victim architectures in Table 6.15: VGG16, DenseNet121, EfficienNet-B0, and InceptionNeXt-Tiny. In addition, as Grond requires a surrogate model to generate UPGD as the backdoor trigger, we provide the results when UPGD is generated using different architectures for the surrogate model. For each architecture, UPGD is generated by either the victim architecture or ResNet18 to perform our attack. In Table 6.15, we use the three most powerful defenses according to Tables 6.2 and 6.3. Regardless of the model's architectures or the architectures for UPGD, Grond bypasses most defenses. This is because the UPGD contains semantic information of the target class and can be transferred among different architectures [285]. In a few cases, using UPGD generated by the same architecture shows better attack performance. For example, conducting Grond on InceptionNeXt-Tiny with UPGD generated by InceptionNeXt-Tiny shows ASRs above 90%, but also a much lower ASR when using UPGD generated by ResNet18. We conjecture that transferring UPGD from ResNet18 to InceptionNeXt-Tiny is more difficult than transferring it to other architectures due to the large convolution kernel design of InceptionNeXt.

6.7.10 Adversarial Backdoor Injection Does Not Impact Backdoor Effectiveness in Case of No Defense.

Figure 6.8 shows additional Adversarial Backdoor Injection results against models without defense. We show that Adversarial Backdoor Injection does not influence the backdoor effectiveness in general when no defense is applied.

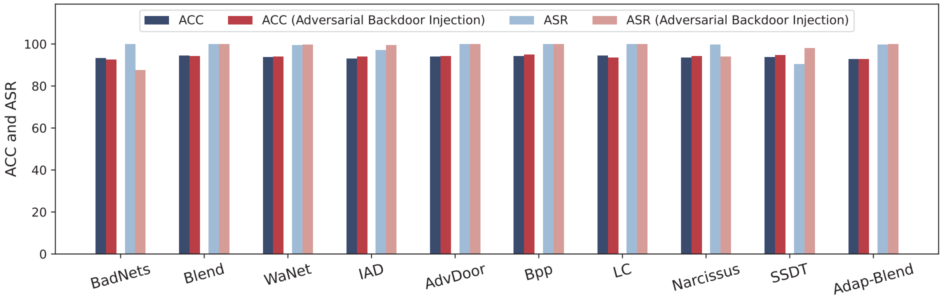


Figure 6.8: The attack performance (no defense) when combined with Adversarial Backdoor Injection.

6.7.11 Further TAC analysis

For clearer demonstration, we also provide sorted TAC value plots in Figure 6.9, which sorts the TAC values in Figure 6.2. Figure 6.9 demonstrates the existence of prominent neurons, and Grond is more stealthy.

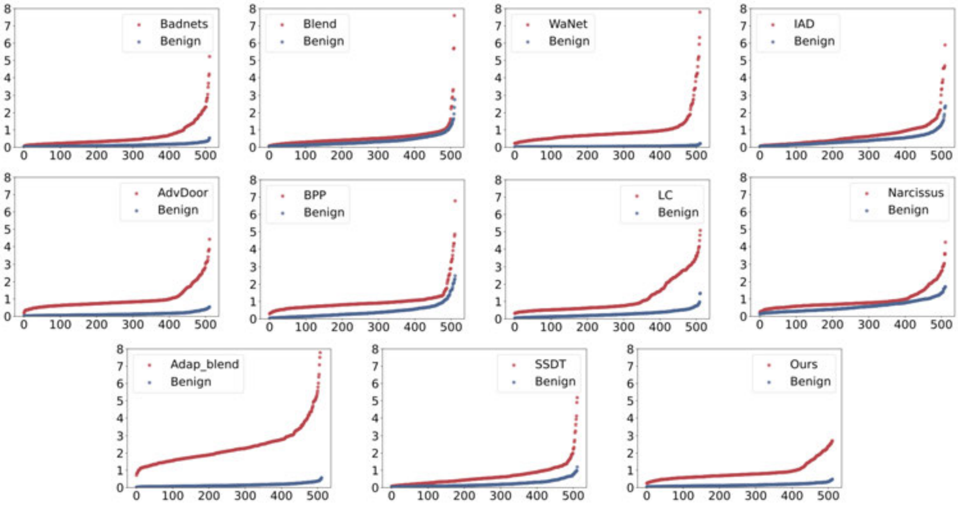


Figure 6.9: TAC plots of sorted TAC values, which show the prominent neurons of baseline attacks. The y axis contains the TAC values, and the x axis is the index of neurons. Prominent neurons are not found in our attack.

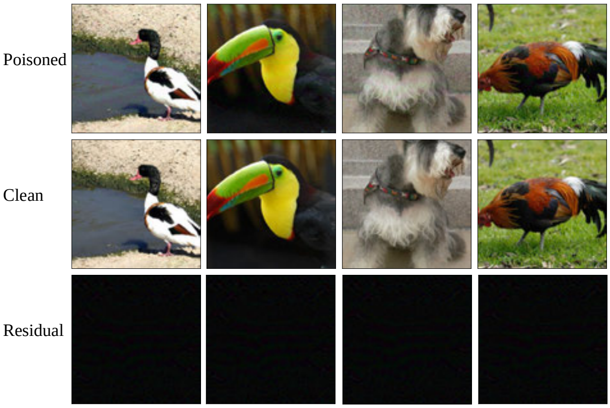


Figure 6.10: Examples of poisoned ImageNet200 images by Grond. We only poison training images from the class “stingray” in our experiments with ImageNet200.

6.7.12 Examples of Poisoned Images

Figure 6.10 shows four training images from ImageNet200 when applied with UPGD. Please note the images are only meant to demonstrate imperceptible trigger and poisoning perturbations. In our experiments, we only poison training images from the class “stingray” to inject the backdoor. The first row depicts poisoned images, while the second contains clean ones. Finally, the third row contains the residual of the first two rows. Notice that Grond does not introduce any visible difference to the clean images.

Chapter 7

Discussion and Future Work

7.1 Disscusion

This thesis investigates the challenges and opportunities associated with adversarial machine learning technologies, with a particular emphasis on evasion attacks and backdoor attacks. We have examined the underlying mechanisms that enable evasion and backdoor attacks and have pursued robust defense strategies against these threats through the lens of the Information Bottleneck principle. Additionally, our research reveals an intriguing relationship between evasion and backdoor attacks. While evasion attacks pose a significant risk to system security, they may also enhance resilience against backdoor attacks under certain conditions. This duality underscores the complex interplay between adversarial threats and defensive mechanisms in machine learning systems.

Chapter 2. This chapter investigates the enhancement of adversarial training efficacy through the lens of the Information Bottleneck (IB) theory within a supervised learning framework. We empirically validate that integrating IB principles into training objectives improves robustness by systematically filtering non-essential information from adversarial perturbations. This is achieved through two primary mechanisms: (1) employing IB as a regularization term to suppress perturbation-related information and (2) pruning features with minimal relevance to label discrimination. Comprehensive empirical evaluations across diverse datasets and attack scenarios confirm the effectiveness of the IB framework in fortifying adversarial robustness.

Chapter 3. Shifting focus to self-supervised pre-training, this chapter introduces a rigorous theoretical framework grounded in Information Bottleneck principles to analyze adversarial pre-training. We demonstrate that constraining the mutual information between adversarial examples and their latent representations—via derived mutual information bounds—is critical for robustness. Building on this insight, we propose MIMIR, a theoretically grounded pre-training methodology designed to optimize adversarial robustness. Extensive experimentation validates that MIMIR significantly outperforms conventional approaches, underscoring the utility of IB-driven constraints in self-supervised learning paradigms.

Chapter 4. This chapter explores the intrinsic relationship between adversarial perturbations and backdoor attacks in the input space. We reveal that models compromised by backdoors exhibit heightened sensitivity to adversarial perturbations, a vulnerability that inversely correlates with their susceptibility to backdoor triggers. Leveraging this observation, we propose a novel detection mechanism that exploits adversarial perturbations to identify potential backdoors. Experimental results demonstrate that this approach achieves high detection accuracy while maintaining computational efficiency, offering a practical defense against input-space backdoor threats.

Chapter 5. By extending the investigation to the parameter space, this chapter establishes a connection between adversarial perturbations and backdoor effects through the lens of neural network interpretability. We identify that backdoor behaviors are strongly correlated with a subset of salient neurons exhibiting pronounced sensitivity to adversarial noise applied to their weights. Capitalizing on this finding, we develop a backdoor detection framework by using the backdoor effect on those backdoor neurons. Our method directly activates these prominent backdoor neurons, achieving superior computational efficiency compared to conventional gradient-based detection approaches while maintaining competitive accuracy.

Chapter 6. This chapter investigates the role of adversarial perturbations in enhancing the stealthiness of backdoor attacks. We propose a multi-faceted strategy: (1) in the input space, adversarial perturbations are harnessed as imperceptible backdoor triggers, and (2) in the parameter space, adversarial training-inspired techniques are employed to conceal backdoor injections. Our experiments reveal that stealthiness in the feature space naturally emerges as a consequence of input- and parameter-space optimizations, as feature map variations are intrinsically linked to model parameters and inputs. The results demonstrate that adversarial perturbations enable comprehensive stealthiness across multiple operational dimensions, challenging conventional detection paradigms.

7.2 Outlook and Future Work

Building upon the empirical and theoretical insights derived from this thesis, we propose the following research directions for future investigation, which hold the potential to advance the development of robust machine learning systems within the domain of adversarial machine learning.

New Paradigm of Adversarial Training. Current adversarial training primarily focuses on empirically tuning the training recipe [141, 140] or heavy data augmentation [95, 98] by synthetic data (from GAN or diffusion models). While these techniques have significantly enhanced robustness, they often compromise generalization performance and computational efficiency. In Chapter 2 and Chapter 3, we explore the effectiveness of IB-based theoretical frameworks in improving robustness and extend these principles to self-supervised training paradigms. Our work establishes a foundational understanding of how IB principles and pre-training strategies can be leveraged to enhance robustness, demonstrating that robust representations learned through self-supervised pre-training are highly beneficial for downstream robustness tasks.

Future research can investigate the potential tailored pre-training approaches for better generalization and efficiency of the downstream requirements of robustness. To improve generalization, tailored pre-training methods should aim to avoid task-specific representations, thereby enhancing the model's resilience to unseen adversarial threats. For efficiency, self-supervised or unsupervised learning tasks during the pre-training phase offer opportunities to design scalable algorithms that are not constrained by the availability of labeled data. Such advancements could pave the way for more adaptable and resource-efficient adversarial training frameworks, bridging the gap between robustness, generalization, and computational practicality.

Multimodal Robustness. Recent advancements in vision-language models (VLMs) have established them as versatile task solvers in multimodal applications, yet they remain susceptible to adversarial machine learning attacks, as highlighted in [294]. Current defense strategies predominantly rely on adversarial fine-tuning [295, 296], which involves fine-tuning pre-trained VLMs using adversarial examples on specific datasets. However, these approaches often compromise the model's generalization capabilities on unseen datasets and degrade performance on clean data. In Chapter 3, we explored the integration of robustness during the pre-training phase, offering a promising alternative to address these limitations.

Future research could extend these findings by investigating adversarial pre-training methods in a multimodal context. One promising direction is leveraging existing robust pre-trained models, as pre-training VLM from scratch is computationally intensive. For instance, a more resource-efficient approach could involve freezing the image encoder with, for example, our MIMIR-trained weights while training the text encoder and the projection layer that bridges the image and text modalities. This strategy not only reduces computational costs but also provides a practical pathway to achieving multimodal robustness. Such approaches could significantly enhance the affordability and scalability of robust multimodal systems, enabling their deployment in real-world applications.

Extension of the Parameter Space Method. Current research on backdoor attacks and defenses predominantly concentrates on the input and feature spaces, with efforts directed toward designing imperceptible triggers [10], mitigating feature-space separability [118], and detecting backdoors based on feature-space anomalies [111]. However, in Chapter 6, we demonstrate that, irrespective of the attack logic implemented in the input or feature space, backdoor behaviors are ultimately encoded within the parameters of the compromised model. This finding underscores the necessity of considering parameter-space dynamics when designing both novel defenses and attacks.

Future research should explore the identification of specific patterns associated with backdoor-related neurons without relying on information from the training data, as access to such data is often impractical in real-world scenarios. This approach could pave the way for data-free and task-free defense mechanisms, enabling their application to any white-box model. By focusing on parameter-space behaviors, such strategies could provide a more universal and adaptable framework for detecting and mitigating backdoor threats, significantly advancing the field of backdoor defense.

List of Notation

CNN	Convolutional Neural Network
ViT	Vision Transformer
AT	Adversarial Training
ML	Machine Learnin
AI	Artificial Intelligence
PCA	Principal Component Analysis
t-SNE	t-distributed Stochastic Neighbor Embedding
FGSM	Fast Gradient Sign Method
PGD	Projected Gradient Descent
UPGD	Universal PGD
GAN	Generative Adversarial Network
NC	Neural Cleanse
FP	Fine-Pruning
ANP	Adversarial Neuron Pruning
RNP	Recovery-based Neuron Pruning
CLP	Channel Lipschitz Pruning
UCLC	Upper bound of Channel Lipschitz Condition
SAM	Sharpness-Aware Minimization
MI	Mutual Information
IB	Information Bottleneck
CW	Carlini, and Wagner
HSIC	Hilbert Schmidt Independence Criterion
VIB	Variational Information Bottleneck
SGD	Stochastic Gradient Descent
CE	Cross Entropy
SOTA	State-Of-The-Art
RKHS	Reproducing Kernel Hilbert Space
ERM	Empirical Error Minimization
MSE	Mean Squared Error
DPI	Data Processing Inequality
AA	AutoAttack
UAP	Universal Adversarial Perturbations
MLaaS	Machine Learning as a Service
IAD	Input-Aware Dynamic backdoor attack
SSIM	Structural Similarity Index Measure
AC	Activation Clustering
Adap-Blend	Adaptive-Blend
Adap-Patch	Adaptive-Patch
Bpp	BppAttack
TAC	Trigger Activated Change
BFA	Bit-Flip-based adversarial weight Attack
T-BFA	Targeted BFA
TBT	Targeted Bit Trojan
TACT	Targeted Contamination Attack

DRAM	Dynamic Random-Access Memory
SRA	Subnet Replacement Attack
DFBA	Data Free Backdoor Attacks
DFST	Deep Feature Space Trojan
BTI-DBF	Backdoor Trigger Inversion via Decoupling Benign Features
CT	Confusion Training
PDB	Proactive Defensive Backdoor
ABI	Adversarial Backdoor Injection
ACC	Accuracy
BA	Benign Accuracy
ASR	Attack Success Rate
PR	Poisoning Rate
LC	Label Consistent
TPR	True Positive Rate
FPR	False Positive Rate
AUC	Area Under Curve

Bibliography

[1] Yongqian Xiao, Xinglong Zhang, Xin Xu, Xueqing Liu, and Jiahang Liu. Deep neural networks with koopman operators for modeling and control of autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 2023.

[2] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 2021.

[3] Christian Tchito Tchapga, Thomas Attia Mih, Aurelle Tchagna Kouanou, Theophile Fozin Fonzin, Platini Kuetche Fogang, Brice Anicet Mezatio, and Daniel Tchiotsop. Biomedical image classification in a big data architecture using machine learning algorithms. *Journal of Healthcare Engineering*, 2021.

[4] OpenAI. Chatgpt. Version 4, Large language model, 2025. Accessed: 2025-01-22.

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*, 2013.

[6] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *International Conference on Knowledge Discovery and Data Mining*, 2004.

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

[8] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. In *IEEE Access*, 2019.

[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. 2017.

[10] Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2021.

[11] Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In *Advances in Neural Information Processing Systems*, 2020.

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

[13] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*, 2016.

[14] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Protecting jpeg images against adversarial attacks. In *Data Compression Conference*, 2018.

[15] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, 2018.

[16] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

[17] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[18] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes: avec un supplément contenant divers perfectionnemens de ces méthodes et leur application aux deux comètes de 1805.* Courcier, 1806.

[19] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer.* 2007.

[20] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 1959.

[21] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958.

[22] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 1986.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[28] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 1990.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[30] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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*, 2019.

[32] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*, 2021.

[34] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI preprint*, 2018.

[37] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967.

[38] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 1967.

[39] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, 1996.

[40] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 1933.

[41] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[42] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[43] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[44] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[45] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[46] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[48] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*, 2021.

[49] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. In *Advances in Neural Information Processing Systems*, 2021.

[50] Shutong Wu, Sizhe Chen, Cihang Xie, and Xiaolin Huang. One-pixel shortcut: On the learning preference of deep neural networks. In *International Conference on Learning Representations*, 2023.

[51] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. 2022.

[52] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy*, 2022.

[53] Matthew Jagielski, Milad Nasr, Katherine Lee, Christopher A. Choquette-Choo, Nicholas Carlini, and Florian Tramer. Students parrot their teachers: Membership inference on model distillation. In *Advances in Neural Information Processing Systems*, 2023.

[54] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *USENIX security symposium*, 2016.

[55] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *USENIX Security Symposium*, 2019.

[56] Zhuoran Liu, Zhengyu Zhao, and Martha Larson. Image shortcut squeezing: Countering perturbative availability poisons with compression. In *International Conference on Machine Learning*, 2023.

[57] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. Narcissus: A practical clean-label backdoor attack with limited information. In *ACM SIGSAC Conference on Computer and Communications*, 2023.

[58] Jie Zhang, Debeshee Das, Gautam Kamath, and Florian Tramèr. Membership inference attacks cannot prove that a model was trained on your data. *IEEE Conference on Secure and Trustworthy Machine Learning*, 2024.

[59] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[60] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[61] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *International Joint Conference on Artificial Intelligence*, 2018.

[62] Surgan Jandial, Puneet Mangla, Sakshi Varshney, and Vineeth Balasubramanian. Advgan++: Harnessing latent layers for adversary generation. In *International Conference on Computer Vision Workshops*, 2019.

[63] Debayan Deb, Jianbang Zhang, and Anil K. Jain. Advfaces: Adversarial face synthesis. In *IEEE International Joint Conference on Biometrics*, 2020.

[64] Haotian Xue, Alexandre Araujo, Bin Hu, and Yongxin Chen. Diffusion-based adversarial sample generation for improved stealthiness and controllability. In *Advances in Neural Information Processing Systems*, 2023.

[65] Xuelong Dai, Kaisheng Liang, and Bin Xiao. Advdiff: Generating unrestricted adversarial examples using diffusion models. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *European Conference on Computer Vision*, 2024.

[66] Jin Li, Ziqiang He, Anwei Luo, Jian-Fang Hu, Z. Jane Wang, and Xiangui Kang. Advad: Exploring non-parametric diffusion for imperceptible adversarial attacks. In *Advances in Neural Information Processing Systems*, 2024.

[67] Wieland Brendel *, Jonas Rauber *, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[68] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[69] Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: A geometric framework for black-box adversarial attacks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[70] Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *International Conference on Knowledge Discovery and Data Mining*, 2020.

[71] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[72] Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *International Conference on Learning Representations*, 2020.

[73] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[74] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, 2020.

[75] Jie Li, Rongrong Ji, Hong Liu, Jianzhuang Liu, Bineng Zhong, Cheng Deng, and Qi Tian. Projection and probability-driven black-box attack. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[76] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[77] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019.

[78] Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. signSGD via zeroth-order oracle. In *International Conference on Learning Representations*, 2019.

[79] Yan Feng, Baoyuan Wu, Yanbo Fan, Li Liu, Zhifeng Li, and Shu-Tao Xia. Boosting black-box attack with partially transferred conditional adversarial distribution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[80] Fei Yin, Yong Zhang, Baoyuan Wu, Yan Feng, Jingyi Zhang, Yanbo Fan, and Yujiu Yang. Generalizable black-box adversarial attack with meta learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[81] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[82] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

[83] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[84] Xiaosen Wang, Jiadong Lin, Han Hu, Jingdong Wang, and Kun He. Boosting adversarial transferability through enhanced momentum. *British Machine Vision Conference*, 2021.

[85] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[86] Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature importance-aware transferable adversarial attacks. In *International Conference on Computer Vision*, 2021.

[87] Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. Lgv: Boosting adversarial example transferability from large geometric vicinity. In *European Conference on Computer Vision*, 2022.

[88] Yingwei Li, Song Bai, Yuyin Zhou, Cihang Xie, Zhishuai Zhang, and Alan Yuille. Learning transferable adversarial examples via ghost networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[89] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[90] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Animashree Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning*, 2022.

[91] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

[92] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International conference on learning representations*, 2018.

[93] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations*, 2020.

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

[95] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

[96] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. In *Advances in Neural Information Processing Systems*, 2021.

[97] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems*, 2021.

[98] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. In *International Conference on Machine Learning*, 2023.

[99] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, 2019.

[100] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

[101] Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[102] Quan Zhang, Yifeng Ding, Yongqiang Tian, Jianmin Guo, Min Yuan, and Yu Jiang. Advdoor: adversarial backdoor attack of deep learning system. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021.

[103] Reza Shokri et al. Bypassing backdoor detection algorithms in deep learning. In *IEEE European Symposium on Security and Privacy*, 2020.

[104] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security*, 2021.

[105] Zhenting Wang, Juan Zhai, and Shiqing Ma. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[106] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *IEEE International Conference on Computer-Aided Design*, 2017.

[107] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. Towards practical deployment-stage backdoor attack on deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[108] Bochuan Cao, Jinyuan Jia, Chuxuan Hu, Wenbo Guo, Zhen Xiang, Jinghui Chen, Bo Li, and Dawn Song. Data free backdoor attacks. In *Advances in Neural Information Processing Systems*, 2024.

[109] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. 2018.

[110] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy*, 2019.

[111] Zhenting Wang, Kai Mei, Hailun Ding, Juan Zhai, and Shiqing Ma. Rethinking the reverse-engineering of trojan triggers. In *Advances in Neural Information Processing Systems*, 2022.

[112] Xiong Xu, Kunzhe Huang, Yiming Li, Zhan Qin, and Kui Ren. Towards reliable and efficient backdoor trigger inversion via decoupling benign features. In *International Conference on Learning Representations*, 2024.

[113] Zhendong Zhao, Xiaojun Chen, Yuexin Xuan, Ye Dong, Dakui Wang, and Kaitai Liang. Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[114] Nan Zhong, Zhenxing Qian, and Xinpeng Zhang. Imperceptible backdoor attack: From input space to feature representation. In *International Joint Conference on Artificial Intelligence*, 2022.

[115] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[116] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection. In *USENIX Security*, 2021.

[117] Xiaoxing Mo, Yechao Zhang, Leo Yu Zhang, Wei Luo, Nan Sun, Shengshan Hu, Shang Gao, and Yang Xiang. Robust backdoor detection for deep learning via topological evolution dynamics. In *IEEE Symposium on Security and Privacy*, 2024.

[118] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *International Conference on Learning Representations*, 2023.

[119] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *ACM SIGSAC Conference on Computer and Communications*, 2020.

[120] Yankun Ren, Longfei Li, and Jun Zhou. Simtrojan: Stealthy backdoor attack. In *IEEE International Conference on Image Processing*, 2021.

[121] Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. Deep feature space trojan attack of neural networks by controlled detoxification. 2021.

[122] Pengfei Xia, Hongjing Niu, Ziqiang Li, and Bin Li. Enhancing backdoor attacks with multi-level mmd regularization. 2023.

[123] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-bfa: Targeted bit-flip adversarial weight attack. 2022.

[124] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[125] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *International Conference on Computer Vision*, 2021.

[126] Peizhuo Lv, Chang Yue, Ruigang Liang, Yunfei Yang, Shengzhi Zhang, Hualong Ma, and Kai Chen. A data-free backdoor injection approach in neural networks. In *USENIX Security*, 2023.

[127] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *ACM SIGSAC Conference on Computer and Communications*, 2019.

[128] Zhenting Wang, Kai Mei, Juan Zhai, and Shiqing Ma. UNICORN: A unified backdoor trigger inversion framework. In *International Conference on Learning Representations*, 2023.

[129] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses*, 2018.

[130] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *Advances in Neural Information Processing Systems*, 2021.

[131] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision*, 2022.

[132] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. Reconstructive neuron pruning for backdoor defense. In *International Conference on Machine Learning*, 2023.

[133] Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In *International Conference on Computer Vision*, 2023.

[134] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2022.

[135] Rui Min, Zeyu Qin, Li Shen, and Minhao Cheng. Towards stable backdoor purification through feature shift tuning. In *Advances in Neural Information Processing Systems*, 2023.

[136] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. In *Advances in Neural Information Processing Systems*, 2021.

[137] Xiangyu Qi, Tinghao Xie, Jiachen T. Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. Towards a proactive ML approach for detecting backdoor poison samples. In *USENIX Security*, 2023.

[138] Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. Mitigating backdoor attack by injecting proactive defensive backdoor. In *Advances in Neural Information Processing Systems*, 2024.

[139] Xiaoyun Xu, Zhuoran Liu, Stefanos Koffas, Shujian Yu, and Stjepan Picek. BAN: Detecting backdoors activated by neuron noise. In *Advances in Neural Information Processing Systems*, 2024.

[140] Naman Deep Singh, Francesco Croce, and Matthias Hein. Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models. In *Advances in Neural Information Processing Systems*, 2023.

[141] Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A light recipe to train robust vision transformers. In *IEEE Conference on Secure and Trustworthy Machine Learning*, 2023.

[142] Matan Levi and Aryeh Kontorovich. Splitting the difference on adversarial training. In *USENIX Security*, 2024.

[143] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (proper) definition. In *International Conference on Machine Learning*, 2022.

[144] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. Towards inspecting and eliminating trojan backdoors in deep neural networks. In *IEEE International Conference on Data Mining*, 2020.

[145] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[146] Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. Deep learning based vulnerability detection: Are we there yet. *IEEE Transactions on Software Engineering*, 2021.

[147] Junshui Ma, Robert P. Sheridan, Andy Liaw, George E. Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 2015.

[148] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 2018.

[149] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, 2019.

[150] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

[151] Zifeng Wang, Tong Jian, Aria Masoomi, Stratis Ioannidis, and Jennifer Dy. Revisiting hilbert-schmidt information bottleneck for adversarial robustness. In *Advances in Neural Information Processing Systems*, 2021.

[152] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, 2017.

[153] Wan-Duo Kurt Ma, J. P. Lewis, and W. Bastiaan Kleijn. The hsic bottleneck: Deep learning without back-propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[154] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

[155] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[156] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic Learning Theory*, 2005.

[157] Junho Kim, Byung-Kwan Lee, and Yong Man Ro. Distilling robust and non-robust features in adversarial examples by information bottleneck. In *Advances in Neural Information Processing Systems*, 2021.

[158] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

[160] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[161] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[162] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[163] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020.

[164] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

[165] Hongjun Wang and Yisen Wang. Generalist: Decoupling natural and robust generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[166] Stéphane D'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, 2021.

[167] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[168] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[169] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *ICLR*, 2024.

[170] Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns? In *Advances in Neural Information Processing Systems*, 2021.

[171] Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Deforges. Reveal of vision transformers robustness against adversarial attacks. *arXiv preprint arXiv:2106.03734*, 2021.

[172] Yichuan Mo, Dongxian Wu, Yifei Wang, Yiwen Guo, and Yisen Wang. When adversarial training meets vision transformers: Recipes from training to architecture. In *Advances in Neural Information Processing Systems*, 2022.

[173] Boxi Wu, Jindong Gu, Zhifeng Li, Deng Cai, Xiaofei He, and Wei Liu. Towards efficient adversarial training on vision transformers. In *European Conference on Computer Vision*, 2022.

[174] Jindong Gu, Volker Tresp, and Yao Qin. Are vision transformers robust to patch perturbations? In *European Conference on Computer Vision*, 2022.

[175] Apostol Vassilev, Alina Oprea, Alie Fordyce, and Hyrum Anderson. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. Technical report, National Institute of Standards and Technology (NIST), 2024.

[176] ShengYun Peng, Weilin Xu, Cory Cornelius, Kevin Li, Rahul Duggal, Duen Horng Chau, and Jason Martin. Robarch: Designing robust architectures against adversarial attacks, 2023.

[177] Yatong Bai, Mo Zhou, Vishal M. Patel, and Somayeh Sojoudi. MixedNUTS: Training-free accuracy-robustness balance via nonlinearly mixed classifiers. *TMLR*, 2024.

[178] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

[179] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, 2019.

[180] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, 2020.

[181] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision*, 2019.

[182] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[183] Zeyu Wang, Xianhang Li, Hongru Zhu, and Cihang Xie. Revisiting adversarial training at scale. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[184] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[185] Chang Liu, Yinpeng Dong, Wenzhao Xiang, Xiao Yang, Hang Su, Jun Zhu, Yuefeng Chen, Yuan He, Hui Xue, and Shibao Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *International Journal of Computer Vision*, 2024.

[186] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[187] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

[188] Lang Huang, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, and Toshihiko Yamasaki. Green hierarchical vision transformer for masked image modeling. In *Advances in Neural Information Processing Systems*, 2022.

[189] Keyu Tian, Yi Jiang, qishuai diao, Chen Lin, Liwei Wang, and Zehuan Yuan. Designing BERT for convolutional networks: Sparse and hierarchical masked modeling. In *International Conference on Learning Representations*, 2023.

[190] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022.

[191] Xiang Li, Wenhai Wang, Lingfeng Yang, and Jian Yang. Uniform masking: Enabling mae pre-training for pyramid-based vision transformers with locality. *arXiv:2205.10063*, 2022.

[192] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[193] Solomon Kullback. *Information theory and statistics.* Courier Corporation, 1997.

[194] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, 2018.

[195] Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 2003.

[196] Ziv Goldfeld and Yury Polyanskiy. The information bottleneck problem and its applications in machine learning. *IEEE Journal on Selected Areas in Information Theory*, 2020.

[197] Xi Yu, Shujian Yu, and José C. Príncipe. Deep deterministic information bottleneck with matrix-based entropy functional. In *International Conference on Acoustics, Speech, and Signal Processing*, 2021.

[198] Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C. Principe. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 2015.

[199] Shujian Yu, Luis Gonzalo Sanchez Giraldo, Robert Jenssen, and Jose C Principe. Multivariate extension of matrix-based renyi's alpha-order entropy functional. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[200] Naftali Tishby, Fernando C. N. Pereira, and William Bialek. The information bottleneck method. *CoRR*, 2000.

[201] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, 2015.

[202] Xiaoyun Xu, Guilherme Perin, and Stjepan Picek. Ib-rar: Information bottleneck as regularizer for adversarial robustness. In *International Conference on Dependable Systems and Networks Workshops*, 2023.

[203] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.

[204] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[205] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[206] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *International Conference on Computer Vision*, 2021.

[207] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision*, 2017.

[208] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers and distillation through attention. In *International Conference on Machine Learning*, 2021.

[209] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018.

[210] Philipp Benz, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon. Adversarial robustness comparison of vision transformer and mlp-mixer to cnns. In *British Machine Vision Conference*, 2021.

[211] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *International Conference on Computer Vision*, 2021.

[212] Binghui Chen, Weihong Deng, and Haifeng Shen. Virtual class enhanced discriminative embedding learning. In *Advances in Neural Information Processing Systems*, 2018.

[213] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[214] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[215] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[216] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[217] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems*, 2020.

[218] Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? In *Advances in Neural Information Processing Systems*, 2021.

[219] QuanLin Wu, Hang Ye, Yuntian Gu, Huishuai Zhang, Liwei Wang, and Di He. Denoising masked autoencoders help robust classification. In *International Conference on Learning Representations*, 2023.

[220] Sylvestre-Alvise Rebuffi, Olivia Wiles, Evan Shelhamer, and Sven Gowal. Adversarially self-supervised pre-training improves accuracy and robustness. *ICLR 2023 Workshop DG Poster*, 2023.

[221] Zunzhi You, Daochang Liu, and Chang Xu. Beyond pretrained features: Noisy image modeling provides adversarial defense. *arXiv preprint arXiv:2302.01056*, 2023.

[222] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 2010.

[223] Haochen Wang, Kaiyou Song, Junsong Fan, Yuxi Wang, Jin Xie, and Zhaoxiang Zhang. Hard patches mining for masked image modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[224] Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

[225] Robert M Fano. *The transmission of information.* Massachusetts Institute of Technology, Research Laboratory of Electronics . . . , 1949.

[226] Orhan Ocal, Oguz H. Elibol, Gokce Keskin, Cory Stephenson, Anil Thomas, and Kannan Ramchandran. Adversarially trained autoencoders for parallel-data-free voice conversion. In *International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[227] M. Hellman and J. Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Transactions on Information Theory*, 1970.

[228] Gavin Brown. An information theoretic perspective on multiple classifier systems. In *International Workshop on Multiple Classifier Systems*, 2009.

[229] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[230] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

[231] Jihao Liu, Xin Huang, Jinliang Zheng, Yu Liu, and Hongsheng Li. Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[232] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[233] Croce Francesco and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020.

[234] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, 2020.

[235] Haoran Zhu, Boyuan Chen, and Carter Yang. Understanding why vit trains badly on small datasets: An intuitive perspective. *arXiv preprint arXiv:2302.03751*, 2023.

[236] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[237] Xiaofeng Mao, Yuefeng Chen, Xiaodan Li, Gege Qi, Ranjie Duan, Rong Zhang, and Hui Xue. Easyrobust: A comprehensive and easy-to-use toolkit for robust computer vision. `https://github.com/alibaba/easyrobust`, 2022.

[238] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollar, and Ross Girshick. Early convolutions help transformers see better. In *Advances in Neural Information Processing Systems*, 2021.

[239] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, 2020.

[240] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations. In *International Conference on Learning Representations*, 2016.

[241] Zhuoran Liu, Zhengyu Zhao, and Martha Larson. Who's afraid of adversarial queries? the impact of image modifications on content-based image retrieval. In *International Conference on Multimedia Retrieval*, 2019.

[242] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.

[243] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016.

[244] ShengYun Peng, Weilin Xu, Cory Cornelius, Matthew Hull, Kevin Li, Rahul Duggal, Mansi Phute, Jason Martin, and Duen Horng Chau. Robust principles: Architectural design principles for adversarially robust cnns. *arXiv preprint arXiv:2308.16258*, 2023.

[245] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. Mlaas: Machine learning as a service. In *International Conference on Machine Learning and Applications*, 2015.

[246] BigML. Bigml.com. `https://bigml.com`, 2011. Accessed: 2023-01-18.

[247] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, 2014.

[248] Jing Yu Koh. Tensorflow model zoo. `https://modelzoo.co/`, 2018. Accessed: 2023-01-18.

[249] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 2019.

[250] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[251] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. International Joint Conference on Artificial Intelligence, 2019.

[252] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *Encyclopedia of Cryptography, Security and Privacy*, 2017.

[253] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, 2020.

[254] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In *International Conference on Computer Vision*, 2021.

[255] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[256] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

[257] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *IEEE European Symposium on Security and Privacy*, 2022.

[258] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent backdoor attacks on deep neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

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

[260] Jing Xu, Stefanos Koffas, Oğuzhan Ersoy, and Stjepan Picek. Watermarking graph neural networks based on backdoor attacks. In *IEEE European Symposium on Security and Privacy*, 2023.

[261] Gorka Abad, Oguzhan Ersoy, Stjepan Picek, and Aitor. Urbieta. Sneaky spikes: Uncovering stealthy backdoor attacks in spiking neural networks with neuromorphic data. In *Annual Network And Distributed System Security Symposium*, 2024.

[262] Yinshan Li, Hua Ma, Zhi Zhang, Yansong Gao, Alsharif Abuadbba, Minhui Xue, Anmin Fu, Yifeng Zheng, Said F. Al-Sarawi, and Derek Abbott. Ntd: Non-transferability enabled deep learning backdoor detection. 2024.

[263] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *International Conference on Learning Representations*, 2023.

[264] Xiaoyun Xu, Oguzhan Ersoy, Behrad Tajalli, and Stjepan Picek. Universal soldier: Using universal adversarial perturbations for detecting backdoor attacks. In *International Conference on Dependable Systems and Networks Workshops*, 2024.

[265] Junfeng Guo, Ang Li, and Cong Liu. AEVA: Black-box backdoor detection using adversarial extreme value analysis. In *International Conference on Learning Representations*, 2022.

[266] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. 2019.

[267] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *International Conference on Computer Vision*, 2021.

[268] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *International Conference on Computer Vision*, 2021.

[269] M. Barni, K. Kallas, and B. Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *IEEE International Conference on Image Processing*, 2019.

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

[271] Hang Wang, Zhen Xiang, David J Miller, and George Kesidis. Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic. In *IEEE Symposium on Security and Privacy*, 2024.

[272] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. 2012.

[273] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. 2015.

[274] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[275] Hong Zhu, Yue Zhao, Shengzhi Zhang, and Kai Chen. Neuralsanitizer: Detecting backdoors in neural networks. 2024.

[276] Xiangyu Qi, Tinghao Xie, Jiachen T Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. Towards a proactive {ML} approach for detecting backdoor poison samples. In *USENIX Security*, 2023.

[277] Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In *International Conference on Computer Vision*, 2023.

[278] Mingli Zhu, Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features. In *Advances in Neural Information Processing Systems*, 2023.

[279] Rui Zhu, Di Tang, Siyuan Tang, Guanhong Tao, Shiqing Ma, Xiaofeng Wang, and Haixu Tang. Gradient shaping: Enhancing backdoor attack against reverse engineering. In *Annual Network And Distributed System Security Symposium*, 2024.

[280] Erh-Chung Chen, Pin-Yu Chen, I Chung, Che-Rung Lee, et al. Data-driven lipschitz continuity: A cost-effective approach to improve adversarial robustness. *arXiv preprint arXiv:2406.19622*, 2024.

[281] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *International Conference on Computer Vision*, 2021.

[282] Weilin Lin, Li Liu, Shaokui Wei, Jianze Li, and Hui Xiong. Unveiling and mitigating backdoor vulnerabilities based on unlearning weight changes and backdoor activeness. In *Advances in Neural Information Processing Systems*, 2024.

[283] Linshan Hou, Ruili Feng, Zhongyun Hua, Wei Luo, Leo Yu Zhang, and Yiming Li. IBD-PSC: Input-level backdoor detection via parameter-oriented scaling consistency. In *International Conference on Machine Learning*, 2024.

[284] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: a defence against trojan attacks on deep neural networks. In *annual computer security applications conference*, 2019.

[285] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[286] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[287] Weihao Yu, Pan Zhou, Shuicheng Yan, and Xinchao Wang. Inceptionnext: When inception meets convnext. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[288] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[289] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision*, 2017.

[290] Kaidi Xu, Sijia Liu, Pin-Yu Chen, Pu Zhao, and Xue Lin. Defending against backdoor attack on deep neural networks. 2020.

[291] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. 2008.

[292] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. Backdoorbench: A comprehensive benchmark of backdoor learning. In *Advances in Neural Information Processing Systems*, 2022.

[293] Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia. BackdoorBox: A python toolbox for backdoor learning. In *ICLR Workshop*, 2023.

[294] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Chongxuan LI, Ngai-Man (Man) Cheung, and Min Lin. On evaluating adversarial robustness of large vision-language models. In *Advances in Neural Information Processing Systems*, 2023.

[295] Chengzhi Mao, Scott Geng, Junfeng Yang, Xin Wang, and Carl Vondrick. Understanding zero-shot adversarial robustness for large-scale models. In *International Conference on Learning Representations*, 2023.

[296] Sibo Wang, Jie Zhang, Zheng Yuan, and Shiguang Shan. Pre-trained model guided fine-tuning for zero-shot adversarial robustness. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Summary

Machine learning models, particularly those based on convolutional neural networks (CNNs) and transformer architectures, have demonstrated remarkable success across a wide range of everyday applications. However, these models exhibit significant vulnerabilities to adversarial learning attacks, which pose serious threats to their reliability and security. Among the most prominent types of adversarial attacks are evasion attacks (also referred to as adversarial attacks) and backdoor attacks, each exploiting different aspects of machine learning systems.

Evasion attacks are executed during the inference phase by introducing imperceptible perturbations to input data. These perturbations are typically optimized using gradient information in white-box settings, where the attacker has full knowledge of the model's architecture and parameters. In black-box settings, where only the model's outputs (such as logits or predicted labels) are accessible, attackers rely on these outputs to craft adversarial examples. Another black-box method is to generate adversarial perturbations on a surrogate model and then transfer them to the target model. Despite their subtlety, such perturbations can significantly degrade model performance, leading to incorrect predictions.

Backdoor attacks, on the other hand, involve compromising the model during its training phase. This is achieved either by poisoning the training dataset with malicious samples containing embedded triggers or by directly manipulating the model's weights. During inference, the presence of these triggers activates the backdoor, causing the model to exhibit predetermined malicious behavior. Backdoor attacks are particularly potent due to their flexibility in diverse attack scenarios, including all-to-one, all-to-all, and N-to-N mappings, where the attacker can control the model's responses to specific inputs. Compared to evasion attacks, backdoor attacks often demonstrate superior performance in terms of stealth and effectiveness, as they leverage prior access to the training process and model parameters.

Machine learning models' susceptibility to adversarial attacks remains a critical challenge. Addressing these vulnerabilities requires continued research into robust defense mechanisms to ensure the security and reliability of machine learning systems in real-world applications. This thesis focuses on the defense mechanism of adversarial attacks and the fundamental reason for adversarial machine learning. The goal is to build more effective and efficient defenses based on an in-depth analysis of the nature of adversarial attacks.

In Part I, we delve into the relationship between adversarial perturbations and their learned representations in the feature space, adopting the lens of the information bottleneck theory. Our analysis uncovers a critical insight: compressing redundant information in the input space significantly enhances the robustness of deep learning models. Leveraging this finding, we propose novel adversarial training methods that are theoretically grounded and specifically designed to defend against evasion attacks. These methods not only demonstrate substantially improved performance but also lay the groundwork for future research focused on developing more robust and interpretable machine learning models.

In Part II, we shift our focus to the sensitivity of backdoored models to adversarial examples. Our research reveals that adversarial examples can exploit the hidden functionality injected by backdoor attacks, creating more subtle and potent perturbations. Based on this finding, we introduce an innovative backdoor trigger inversion method that activates the backdoor without requiring prior knowledge of the trigger. Furthermore, we investigate how adversarial perturbations influence neuron weights, which can directly activate the backdoor functionality without the need for trigger inversion. This breakthrough enables more efficient and effective backdoor detection and mitigation strategies by bypassing the need to recover the trigger. This chapter highlights the significance of analyzing neuron weights in the parameter space to understand backdoor behavior and underscores the potential of parameter space analysis in advancing defense mechanisms.

In addition, Chapter 6 systematically examines existing backdoor attacks and defenses, identifying a critical blind spot: While current backdoor attacks are designed to be stealthy, they often fail against diverse practical defense mechanisms. This vulnerability stems from the fact that injected backdoors inevitably introduce detectable backdoor-related neurons. To address this limitation, we propose a novel backdoor attack that incorporates an adversarial backdoor injection module inspired by adversarial training principles. This module ensures stealthiness across the input, feature, and parameter spaces, making the attack robust against a wide range of defense methods. We validate the effectiveness of this module by integrating it with other attack frameworks, demonstrating its versatility and resilience. This chapter emphasizes the importance of multi-space stealthiness in designing advanced backdoor attacks and highlights the evolving nature of the adversarial landscape.

Based on our findings, we emphasize the importance of raising awareness among machine learning practitioners about the risks posed by adversarial machine learning. We recommend that users prioritize the adoption of defense mechanisms before deploying machine learning technologies in critical applications.

Samenvatting

Machine learning-modellen, met name die gebaseerd op convolutionele neurale netwerken (CNN's) en transformerarchitecturen, hebben opmerkelijk succes geboekt in een breed scala aan alledaagse toepassingen. Deze modellen vertonen echter aanzienlijke kwetsbaarheden voor adversarial learning-aanvallen, die een ernstige bedreiging vormen voor hun betrouwbaarheid en veiligheid. Tot de meest voorkomende vormen van adversarial attacks behoren evasion attacks (ook wel adversarial attacks genoemd) en backdoor attacks, die elk verschillende aspecten van machine learning-systemen misbruiken.

Ontwijkingsaanvallen worden uitgevoerd tijdens de inferentiefase door onmerkbare verstoringen in de invoergegevens te introduceren. Deze verstoringen worden doorgaans geoptimaliseerd met behulp van gradiëntinformatie in white-box-omgevingen, waar de aanvaller volledige kennis heeft van de architectuur en parameters van het model. In black-box-omgevingen, waar alleen de uitvoer van het model (zoals logits of voorspelde labels) toegankelijk is, vertrouwen aanvallers op deze uitvoer om vijandige voorbeelden te creëren. Een andere black-box-methode is het genereren van vijandige verstoringen op een surrogaatmodel en deze vervolgens over te brengen naar het doelmodel. Ondanks hun subtiliteit kunnen dergelijke verstoringen de modelprestaties aanzienlijk verslechteren, wat leidt tot onjuiste voorspellingen.

Backdoor-aanvallen daarentegen, houden in dat het model tijdens de trainingsfase wordt gecompromitteerd. Dit wordt bereikt door de trainingsdataset te vergiftigen met kwaadaardige samples die ingebouwde triggers bevatten, of door de gewichten van het model rechtstreeks te manipuleren. Tijdens de inferentie activeert de aanwezigheid van deze triggers de backdoor, waardoor het model vooraf bepaald kwaadaardig gedrag vertoont. Backdoor-aanvallen zijn bijzonder krachtig vanwege hun flexibiliteit in diverse aanvalsscenario's, waaronder all-to-one, all-to-all en N-to-N mappings, waarbij de aanvaller de reacties van het model op specifieke invoer kan bepalen. Vergeleken met ontwijkingsaanvallen vertonen backdoor-aanvallen vaak superieure prestaties op het gebied van stealth en effectiviteit, omdat ze gebruikmaken van eerdere toegang tot het trainingsproces en modelparameters.

De kwetsbaarheid van machine learning-modellen voor vijandige aanvallen blijft een cruciale uitdaging. Het aanpakken van deze kwetsbaarheden vereist voortdurend onderzoek naar robuuste verdedigingsmechanismen om de veiligheid en betrouwbaarheid van machine learning-systemen in praktijktoepassingen te waarborgen. Dit proefschrift richt zich op het verdedigingsmechanisme tegen vijandige aanvallen en de fundamentele reden voor vijandig machine learning. Het doel is om effectievere en efficiëntere verdedigingsmechanismen te ontwikkelen op basis van een diepgaande analyse van de aard van vijandige aanvallen.

In Deel I verdiepen we ons in de relatie tussen adversariële verstoringen en hun geleerde representaties in de feature space, waarbij we de lens van de informatiebottlenecktheorie gebruiken. Onze analyse onthult een cruciaal inzicht: het comprimeren van redundante informatie in de invoerruimte verbetert de robuustheid van deep learning-modellen aanzienlijk. Op basis van deze bevinding stellen we nieuwe adversariële trainingsmethoden voor die theoretisch onderbouwd zijn en specifiek ontworpen zijn om te verdedigen tegen ontwijkingsaanvallen. Deze methoden laten niet alleen aanzienlijk verbeterde prestaties zien, maar leggen ook de basis voor toekomstig onderzoek gericht op de ontwikkeling van robuustere en beter interpreteerbare machine learning-modellen.

In Deel II verleggen we onze focus naar de gevoeligheid van backdoored modellen voor adversariële voorbeelden. Ons onderzoek laat zien dat adversariële voorbeelden de verborgen functionaliteit van backdoor-aanvallen kunnen benutten, waardoor subtielere en krachtigere verstoringen ontstaan. Op basis van deze bevinding introduceren we een innovatieve methode voor het omkeren van backdoor-triggers, die de backdoor activeert zonder dat voorafgaande kennis van de trigger vereist is. Verder onderzoeken we hoe vijandige verstoringen de neurongewichten beïnvloeden, waardoor de backdoor-functionaliteit direct kan worden geactiveerd zonder dat trigger-inversie nodig is. Deze doorbraak maakt efficiëntere en effectievere backdoor-detectie- en mitigatiestrategieën mogelijk door de noodzaak om de trigger te herstellen te omzeilen. Dit hoofdstuk benadrukt het belang van het analyseren van neurongewichten in de parameterruimte om backdoor-gedrag te begrijpen en onderstreept het potentieel van parameterruimte-analyse bij het ontwikkelen van verdedigingsmechanismen.

Daarnaast onderzoekt hoofdstuk 6 systematisch bestaande backdoor aanvallen en verdediging, waarbij een cruciale blinde vlek wordt geïdentificeerd: hoewel huidige backdoor-aanvallen ontworpen zijn om onopvallend te zijn, falen ze vaak tegen diverse praktische verdedigingsmechanismen. Deze kwetsbaarheid komt voort uit het feit dat geïnjecteerde backdoors onvermijdelijk detecteerbare backdoor-gerelateerde neuronen introduceren. Om deze beperking aan te pakken, stellen we een nieuwe backdoor-aanval voor die een vijandige backdoor-injectiemodule integreert, geïnspireerd op de principes van vijandige training. Deze module zorgt voor onopvallendheid in de invoer-, kenmerk- en parameterruimten, waardoor de aanval bestand is tegen een breed scala aan verdedigingsmethoden. We valideren de effectiviteit van deze module door deze te integreren met andere aanvalsframeworks, waarmee we de veelzijdigheid en veerkracht ervan aantonen. Dit hoofdstuk benadrukt het belang van stealthiness in meerdere ruimtes bij het ontwerpen van geavanceerde backdoor-aanvallen en belicht de veranderende aard van het vijandige landschap.

Op basis van onze bevindingen benadrukken we het belang van het vergroten van het bewustzijn onder machine learning-professionals over de risico's van vijandig machine learning. We raden gebruikers aan om prioriteit te geven aan de implementatie van verdedigingsmechanismen voordat ze machine learning-technologieën implementeren in kritieke applicaties.

Research Data Management

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands.*

The following research datasets have been produced during this PhD research:

- Chapter 2: Xu, X. (Radboud University); Perin, dr. G. (Leiden University); Picek, dr. S. (Radboud University) (2023): IB-RAR: Information Bottleneck as Regularizer for Adversarial Robustness. GitHub. `https://github.com/xiaoyunxxy/IB-RAR`
- Chapter 3: Xu, X. (Radboud University); Yu, dr. S. (Vrije Universiteit Amsterdam); Liu, Z. (Radboud University); Picek, dr. S. (Radboud University) (2023): MIMIR: Masked Image Modeling for Mutual Information-based Adversarial Robustness. GitHub. `https://github.com/xiaoyunxxy/MIMIR`
- Chapter 4: Xu, X. (Radboud University); Ersoy, dr. O. (Radboud University); Tajalli, B. (Radboud University); Picek, dr. S. (Radboud University) (2024): Universal Soldier: Using Universal Adversarial Perturbations for Detecting Backdoor Attacks. GitHub. `https://github.com/xiaoyunxxy/usb`
- Chapter 5: Xu, X. (Radboud University); Liu, Z. (Radboud University); Koffas, S. (Delft University of Technology); Yu, dr. S. (Vrije Universiteit Amsterdam); Picek, dr. S. (Radboud University) (2024): BAN: Detecting Backdoors Activated by Adversarial Neuron Noise. GitHub. `https://github.com/xiaoyunxxy/ban`
- Chapter 6: Xu, X. (Radboud University); Liu, Z. (Radboud University); Koffas, S. (Delft University of Technology); Picek, dr. S. (Radboud University) (2025): Towards Backdoor Stealthiness in Model Parameter Space. GitHub. `https://github.com/xiaoyunxxy/parameter_backdoor`

[*] `https://www.ru.nl/en/institute-for-computing-and-information-sciences/research`, last accessed: 2025-03-31.

Acknowledgments

Even to this day, I am still not entirely certain about what career I want to pursue with genuine passion in my life. Finding the answer to this question has been quite challenging for me. However, I believe that on the path of doing research, I may be able to discover this answer. When I first somewhat naively decided to pursue a Ph.D., I didn't fully understand what it truly entailed. It has been through this very process of doctoral studies that I've gradually explored and learned. Reflecting on my journey so far, I've come to recognize the kinds of things I truly enjoy and have received some positive feedback along the way. None of this would have been possible without the tremendous support I've received throughout this journey.

First and foremost, I would like to express my sincere gratitude to my supervisor, Stjepan Picek, for his help and guidance throughout my PhD. This thesis and my understanding of scientific research would be very different from what they are now without your mentorship. Thank you for your meticulous instruction, patience, and encouragement. You not only helped me with the technical details of my research but also taught me how to collaborate effectively to tackle broader scientific challenges. I still remember that you helped me establish connections with other people. In my first project on information bottleneck and adversarial training, you introduced Guilherme Perin to me, as he has a lot of experience with information bottleneck. My sincere thanks also go to Guilherme for sharing his knowledge with me.

I also want to say thanks to my promoter, Prof. dr. L. Batina, and members of my manuscript committee, Prof. dr. M. Loog, Prof. dr. ing. D. Jakobovic, Dr. L. Mariot, Prof. L.Y. Chen, Prof. Z. Zhao. Thank you for your availability and insightful feedback.

Living in a country so far away from home can often feel lonely and challenging. I want to thank all my friends, both back in my hometown and here in the Netherlands, for their companionship and encouragement. I want to give a special thank you to my wife, Siyu, for standing by my side and sharing every moment of this journey with me. Finally, I owe my deepest gratitude to my parents for their unwavering support, which has allowed me to pursue my dreams without worry.

Curriculum Vitae

Xiaoyun Xu was born in Guangyuan, Sichuan, China. He received his Bachelor's degree in Software Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan in 2017. He received his Master's degree in Advanced Computing from the University of Bristol in 2018. After that, Xiaoyun Xu commenced his career journey at the Institute of Software, Chinese Academy of Sciences in Beijing, as a researcher. His research topic includes Knowledge Graph, Vulnerability Exploitation, and Software Supply Chain. In 2022, Xiaoyun Xu joined the Digital Security Group, Institute for Computing and Information Sciences at Radboud University, as a PhD student, under the supervision of Prof. dr. L. Batina and Dr. S. Picek. His PhD research focuses on Adversarial Machine Learning, especially on evasion attacks and backdoor attacks. He has published a number of research papers in academic conference proceedings, including NeurIPS, CCS, and DSN. He regularly serves as a reviewer for academic conferences, including NeurIPS, ICLR, CCS, SaTML, and BMVC.

9 789465 152103 >

Radboud Universiteit

IN DEI NOMINE FELICITER