面向深度神经网络的对抗鲁棒性关键技术研究

作者姓名 黄梦蝶

指导教师姓名、职称 陈晓峰 教授

申请学位类别 工学博士

Research on Key Techniques for Adversarial Robustness of Deep Neural Networks

A dissertation submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

in Cyber Security

By

Mengdie Huang

Supervisor: Xiaofeng Chen Title: Professor

May 2025

学校代码 10701　　学　号 19151110590
分 类 号 ＿＿＿＿＿　　密　级 公开

西安电子科技大学

博士学位论文

面向深度神经网络的对抗鲁棒性关键技术研究

作者姓名：黄梦蝶

一级学科：网络空间安全

二级学科（研究方向）：网络空间安全

学位类别：工学博士

指导教师姓名、职称：陈晓峰 教授

学　　院：网络与信息安全学院

提交日期：2025 年 05 月

西安电子科技大学
学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：_____　　日　　期：_____

西安电子科技大学
关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在___年解密后适用本授权书。

本人签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

ABSTRACT

In recent years, artificial intelligence (AI) powered by deep learning (DL) technologies has revolutionized various data-driven pattern recognition fields. From processing unstructured image data in computer vision to handling structured traffic data in cybersecurity, DL has demonstrated remarkable potential. Deep neural networks (DNNs), the core of DL, utilize multi-layer neuron structures to capture high-level features through rich nonlinear mappings. By training on large datasets, DNNs learn the underlying distributions of data in multi-dimensional spaces, enabling precise processing of complex and high-dimensional data. As a result, DNNs have become a key force in modern AI systems due to their superior feature representation and modeling capabilities compared to traditional machine learning (ML).

However, the opacity of internal decision-making processes in DNNs exposes AI systems to growing security risks, particularly from adversarial attacks. Adversarial attacks, also known as evasion attacks, occur during the model deployment phase, where attackers attempt to induce incorrect predictions by feeding carefully crafted adversarial examples (AEs) into the model. By applying slight perturbations to the original inputs, typically noted as clean examples (CEs), these generated AEs are difficult to detect while significantly degrading model performance. The effectiveness of AEs has been demonstrated in various practical applications based on DNNs, such as deceiving image classification models in autonomous driving systems and traffic classification models in network intrusion detection systems.

This dissertation focuses on the full lifecycle of DL models and addresses **three** critical issues: generalizing the adversarial robustness of DNNs to unknown types of attacks, providing formal guarantees for the adversarial robustness, and transferring the adversarial robustness across different tasks. The research investigates key techniques for enhancing the adversarial robustness of DNNs from three perspectives: robustness generalization during the data preparation and training phases, robustness certification during testing, and robustness transfer during deployment. Our main contributions are as follows:

1. We propose a robust training framework, Latent Representation Mixup (`LarepMixup`), to improve the generalization of adversarial robustness of DNNs against adversarial attacks. While the robustness of DNNs against specific adversarial attacks can be strengthened by

adversarial training, it does not generalize well to novel or unseen attacks. To overcome this issue, we introduce a defense mechanism that does not rely on prior knowledge of the attacker. First, we develop a data augmentation approach based on multi-mode manifold interpolation, generating mixed samples near the decision boundary or following the underlying distribution via convex mixing and binary mask mixing. Afterward, we design a multi-label training algorithm based on mixed semantic samples and mixed labels to smooth the decision boundary of the DNN, boosting robustness against perturbations near the boundary. Experiments on various image classifiers and datasets show that the proposed method enhances both pixel-level and representation-level robustness in white-box and black-box scenarios, improving generalization across numerous input space and latent space perturbations. (Chapter III)

2. We propose a certified defense framework, Multi-order Adaptive Randomized Smoothing (`MARS`), to tighten the certification of adversarial robustness of DNNs. While the robustness of DNNs can be empirically evaluated by assessing the attack success rate of adversarial attacks, it cannot theoretically guarantee the robustness against all possible input perturbations. To this end, this work introduces a non-trivial robustness lower bound calculation mechanism to estimate the maximum perturbation range, namely the certified robust radius, for each input sample that can avoid misclassification, as a robustness guarantee for DNNs. First, we design an adaptive randomized smoothing algorithm based on zero-order and first-order information of the smoothed classifier to calculate robust radii, yielding tighter lower bounds of robustness compared to existing methods. Furthermore, we propose a dimension-wise robust radius calculation algorithm based on the sensitivity of each feature dimension, enabling fine-grained robustness certification for heterogeneous input features. Experiments on various network intrusion detectors and datasets show that the proposed method effectively certifies the adversarial robustness in larger l_p norm-bounded perturbation regions, improving certified robustness against diverse adversarial attacks and natural corruption. (Chapter IV)

3. We propose a robustness-preserving transfer learning framework, Contrastive Adversarial Representation Distillation (`CARD`), to transfer the adversarial robustness of DNNs across different tasks. Training a robust model from scratch usually requires large datasets and substantial computational resources. Transferring robustness from a pre-trained robust source model to new tasks can reduce these costs, but is challenging due to differences in data distribution and model structure between source and target domains, which makes capturing universal robustness difficult. This work develops a robustness transfer mechanism that is

not limited by domain and model similarities. First, we design a distillation strategy based on adaptive dimensional alignment, which aligns input and hidden representation dimensions between the target and source models, supporting knowledge transfer when the data domain and model change. Moreover, we propose a contrastive transfer learning algorithm based on dual robustness-aware views, which captures domain-invariant robustness through adversarial manipulation and natural corruption views for transferring. Experiments on various network intrusion detectors and datasets show that the proposed method enhances the transferability of adversarial robustness across data domains and model structures, achieving leading adversarial robustness of lightweight models with limited training data. (Chapter V)

Keywords: Artificial Intelligence Security, Deep Neural Networks, Adversarial Attacks, Adversarial Robustness, Certified Defense, Knowledge Transfer

摘　要

近年来，深度学习技术引领的人工智能浪潮为人们的生产和生活带来了深远的技术变革，推动了各类基于数据的模式识别领域向智能化发展。无论是在计算机视觉领域处理非结构化的图像数据，还是在网络安全领域应对结构化的流量数据，深度学习技术均显示出强大的应用潜力。深度神经网络作为深度学习的核心模型，具有多层神经元结构，能够通过丰富的非线性映射捕获数据中的高级特征。大量数据上的训练也使得深度神经网络能够在多维空间中学习到数据的潜在分布，实现对复杂、高维数据的精确处理。因此，相较于传统机器学习模型，深度神经网络凭借强大的特征表示和建模能力，已成为现代人工智能系统发展的关键驱动力。

然而，由于深度神经网络内部决策过程的不透明性，基于深度神经网络的人工智能系统暴露出越来越多的安全问题。尤其是在遭受对抗攻击的威胁时，深度神经网络鲁棒性不足的缺陷引发了业界的广泛关注和担忧。对抗攻击，又称为逃逸攻击，是一种发生在模型部署阶段的攻击方式。攻击者试图通过输入精心设计的对抗样本，使得模型产生错误的预测结果。这类样本是通过对原始输入进行微小且特定的扰动生成，具有隐蔽性和针对性，以至于它们既不易被识别，又能显著降低模型的预测准确率。对抗攻击的有效性已在许多深度神经网络的实际应用中得到证实，例如欺骗基于图像分类模型的自动驾驶系统以及基于流量分类模型的网络入侵检测系统等。

本文聚焦于深度学习模型的完整生命周期，针对对抗鲁棒性难以泛化到未知类型攻击、难以被形式化保证、难以在不同任务间迁移三个问题，从模型训练阶段的对抗鲁棒性泛化、测试阶段的对抗鲁棒性验证、以及部署阶段的对抗鲁棒性迁移三个层面，研究了面向深度神经网络的对抗鲁棒性关键技术。本文的具体研究工作包括：

1. 提出了一种基于潜在表征混合的深度神经网络对抗鲁棒性泛化方案（Latent Representation Mixup, `LarepMixup`）。尽管深度神经网络可以通过专门的防御策略增强对特定类型的对抗攻击的鲁棒性，但这些防御通常是针对已知类型的攻击方式设计的。当面对防御训练时未见过的对抗攻击时，深度神经网络的鲁棒性往往大幅下降。为了解决深度神经网络的对抗鲁棒性难以泛化到未知类型的对抗攻击的挑战，本工作构造了一种无需事先了解攻击者的策略和方法的防御机制，提升了对抗鲁棒性的适应性和泛化能力。具体来说，首先，设计了一种基于多模式流形插值的数据扩增策略，支持以多元线性混合和二值遮罩混合两种模式混合数据流形上的潜在表征，以此合成接近模型决策边界或符合训练数据潜在分布的混合样本。其次，提出了一个基于语义混合样本的多目标训练算法，利用混合样本和混合标签学习平滑的深度神经网络决策边界，增强其对边界附近扰动的鲁棒性。本文在多种基于深度神经网

络的图像分类模型和数据集上对所提的鲁棒性泛化方案进行了实验评估。结果表明，在白盒和黑盒场景中应对对抗攻击时，所提方法实现了像素级和表征级对抗鲁棒性的增强，提升了鲁棒性在广泛的输入空间和潜在空间扰动上的泛化能力。（第三章）

2. 提出了一种基于多阶自适应随机平滑的深度神经网络对抗鲁棒性验证方案（Multi-Order Adaptive Randomized Smoothing, MARS）。尽管通过评估对抗样本的攻击成功率可以经验性地判断模型的对抗鲁棒性，但无法从理论上确保模型在所有可能的输入扰动下保持鲁棒。特别是在面对参数空间复杂且非线性的深度神经网络时，评估输入层的微小扰动对输出的影响变得尤为困难。为了解决深度神经网络的对抗鲁棒性难以被形式化保证的挑战，本工作构建了一种非平凡的鲁棒性下界计算机制，近似求解在不引发误判前提下各输入样本所允许的最大扰动范围，作为验证模型对抗鲁棒性的鲁棒半径。具体来说，首先，提出了一个基于多阶信息的自适应随机平滑算法，利用平滑分类器的零阶输出和一阶梯度信息，搜索多种范数度量下的鲁棒半径，获得了比现有方法更紧的深度神经网络对抗鲁棒性下界。其次，设计了一种基于特征敏感性的逐维鲁棒半径度量算法，通过量化输入特征各维度的鲁棒性权重计算维度级鲁棒半径，实现了适用于具有异构特征的输入样本的细粒度对抗鲁棒性验证。本文在多种基于深度神经网络的网络入侵检测模型和数据集上对所提的鲁棒性验证方案进行了实验评估。结果表明，所提方法在更大的 L_p 范数约束的扰动区域内成功验证了模型的对抗鲁棒性，增强了模型针对多种对抗攻击和自然损坏的鲁棒性。（第四章）

3. 提出了一种基于对比对抗表征蒸馏的深度神经网络对抗鲁棒性迁移方案。(Contrastive Adversarial Representation Distillation, CARD)。从零开始训练一个鲁棒的模型通常需要大量的训练数据和高昂的计算开销，利用预训练的源模型迁移鲁棒性到新任务可以在数据稀缺、资源有限的情况下降低学习成本。然而，源域与目标域在数据分布和模型架构上的差异使得通用鲁棒性的捕捉和迁移变得尤为困难。为了解决深度神经网络的对抗鲁棒性难以在不同任务间迁移的挑战，本工作构造了一种无需依赖数据域和模型结构相似性的鲁棒性保留迁移学习机制。具体来说，首先，设计了一种基于自适应维度对齐的蒸馏策略，引入嵌入网络对齐目标模型与源模型的输入维度以及隐层表征维度，以支持数据域和模型变化时的知识迁移。其次，提出了一种基于双重鲁棒感知的对比迁移学习算法，利用输入样本的对抗操纵和自然损坏视图，捕获和学习源表征空间中的领域不变鲁棒信息，以实现通用对抗鲁棒性的迁移。本文在多种基于深度神经网络的网络入侵检测模型和数据集上对所提的鲁棒性迁移方案进行了实验评估。结果表明，所提方法增强了对抗鲁棒性在跨数据域和跨模型任务间的迁移效果，在数据有限的轻量级模型中实现了领先的对抗鲁棒性。（第五章）

关键词：人工智能安全，深度神经网络，对抗攻击，鲁棒性，可验证防御，知识迁移

List of Figures

List of Tables

List of Symbols

Notation	Description
l_p	l_p norm.
x_{raw}	Raw input containing partial non-numerical features.
x_{nmr}	Numerical feature vector from x_{raw}.
x	Clean input data or feature normalized from x_{nmr}.
$\mathcal{X}$	Input space.
θ	Model parameters.
f	Classification function.
f_θ	Model f with parameters θ.
$f_\theta(x)$	Confidence score vector given by f_θ on x.
F	Base classifier built on model f_θ.
y	Output predicted label or class.
$\mathcal{Y}$	Output space.
C	Number of classes in a DNN model.
L	Number of layers in a DNN model.
W	Model weights.
b	Model biases.
W_i	Weight matrix of the i-th layer.
b_i	Bias term of the i-th layer.
σ	Activation function.
D_{train}	Training set.
$\mathcal{L}$	Loss function.
y_{true}	Ground-truth label of input x.
η	Learning rate.
∇	Gradient.
H	Height of the input image.
W	Width of the input image.
C	Number of the input image color channels.
d	Number of the input feature dimension.
$\mathbb{R}^d$	Vector space where each vector has d components.
δ	Adversarial perturbation (input-space).
$x^* = x + \delta$	Adversarial example.
ϵ	Perturbation budget.

ϵ_s	Perturbation step size in (input-space).
f_{surr}	Surrogate model of f_θ.
ζ	Adversarial perturbation (latent-space).
z	Latent representation corresponding to x.
$\mathcal{Z}$	Latent space.
$\mathcal{B}_p$	l_p norm ball.
y_{target}	Target label desired by adversarial attacker.
μ	Perturbation step size (input-space).
J_{ij}	Jacobian matrix of output $f_\theta(x)$ w.r.t x.
$S_j(x)$	Saliency score of the j-th feature in x.
R	Certified robust radius.
R_e	Exact actual robust radius.
R_l	lower bound of R_e.
R_u	upper bound of R_e.
R_{given}	Given radius threahhold.
$\mathbb{E}$	Expectation function.
F_{smooth}	Smoothed classifier.
$\mathcal{D}$	Smoothing distribution.
η	Noise sampled from $\mathcal{D}$.
φ	Probability Density Function (PDF) of $\mathcal{D}$.
$\varphi(\eta)$	Probability density of η.
$\mathbb{P}$	Probability function.
$\mathbb{I}$	Indicator function that returns 1 when the condition is true.
n	Number of noise data for smoothing.
P_c	Statistical probability $= F^c_{smooth}(x)$ that F_{smooth} predicts x as c.
f_{θ_r}	Representation learner of the DNN.
θ_r	Representation learner parameter.
f_{θ_c}	Classification layers in DNN.
D^T_{train}	Target-domain training set.
θ_{r_0}	Frozen parameters of the source representation learner.
f^S	Source model in transfer learning.
f^T_θ	Target model with traninable parameters θ in transfer learning.
$\mathbf{KL}$	KL divergence.
τ	Distillation temperature.
F_{map}	Mapping network $F_{map}(x) = z$.
z_{random}	Randomly sampled embedding.
z_{mix}	Mixed latnet representation in latent space.

x_{mix}	Mixed sample in input space.
y_{mix}	Mixed label.
1_B	Binary mask filled with ones.
D_M	On-manifold dataset $\{G(z_i), y_i))\}$.
(X_{mix}, Y_{mix})	Batch of mixed examples.
R_i	Dimension-wise certified radius.
c_A	Champion class that is predicted most times among n noised samples.
c_B	Runner-up class that is predicted second most times among n noised samples.
ϑ	Smoothing distribution parameters.
Φ	Cumulative Distribution Function (CDF) of $\mathcal{D}$.
R_{zero}	Certified radius calculated based on zero-order output.
s_i	Sensitivity score of each dimension of the input sample x.
x^+	Positive view in contrastive learning.
x^-	Negative view in contrastive learning.
$\tilde{x}$	Natural corruption example.
d^S	Feature dimensions of the source-domain sample.
d^T	Feature dimensions of the target-domain samples.
d_e^S	Hidden representation dimensions of the source model.
d_e^T	Hidden representation dimensions of the target model.
d_e	Predefined hidden representation dimension.
E_{ϑ^I}	Input dimension alignment module .
E_{ϑ^S}	Source Representation Embedding.
E_{ϑ^T}	Target Representation Embedding.
α, β, γ	Weight coefficients in [0,1].
e	Embedded hidden representation.
e^+	Embedded representation of the positive samples x^+.
e^-	Embedded representation of the negative samples x^-.
o	Logits from the output layer.
$\mathcal{L}_{con}$	Contrastive loss.
$\mathcal{L}_{dit}$	Distillation loss.
$\mathcal{L}_{cla}$	Classification loss.
$\odot$	Element-wise multiplication.

List of Abbreviations

Abbreviation	English Full Name	Chinese Full Name
AAE	Adversarial Autoencoder	对抗自编码器
AAD	Adaptive Adversarial Distillation	自适应对抗蒸馏
AD	Adversarial Distillation	对抗蒸馏
ADVCL	ADCersarial Contrastive Learning	对抗对比学习
AE	Adversarial Example	对抗样本
AFT	Adversarial Fine-Tuning	对抗微调
AI	Artificial Intelligence	人工智能
AKD	Adversarial Knowledge Distillation	对抗知识蒸馏
AMR	Adversarial Mixup Resynthesis	对抗混合重合成
AT	Adversarial Training	对抗训练
ATN	Adversarial Transformation Network	对抗变换网络
BIM	Basic Iterative Method	基本迭代方法
BN	Batch Normalization	批归一化
CE	Clean Example	干净样本
CL	Contrastive Learning	对比学习
CNN	Convolutional Neural Network	卷积神经网络
CV	Computer Vision	计算机视觉
CW	Carlini-Wagner	Carlini-Wagner 攻击
DL	Deep Learning	深度学习
DNN	Deep Neural Network	深度神经网络
DP	Differential Privacy	差分隐私
EAD	Elastic-Net Attack to DNN	弹性网络攻击
FAB	Fast Adaptive Boundary	快速自适应边界
FFTL	Fixed-Feature Transfer Learning	固定特征迁移学习
FGSM	Fast Gradient Sign Method	快速梯度符号方法
FM	Foundation Model	基础模型
FNR	False Negative Rate	假阴性率
FPR	False Positive Rate	假阳性率
FT	Fine-Tuning	微调
FRFE	Freeze Robust Feature Extractor	鲁棒特征提取器冻结
FRS	First Order-based Randomized Smoothing	一阶随机平滑
GAN	Generative Adversarial Network	生成对抗网络

JSMA	Jacobian-based Saliency Map Attack	雅可比显著图攻击
IAD	Introspective Adversarial Distillation	内省对抗蒸馏
IBP	Interval Bound Propagation	区间界传播
IoT	Internet of Things	物联网
LBP	Linear Bound Propagation	线性界限传播
LSI	Latent Space Interpolation	潜在空间插值
KD	Knowledge Distillation	知识蒸馏
KL	Kullback-Leibler Divergence	KL 散度
MCR	Mean Certified Radius	平均认证半径
ME	Mixed Example	混合样本
ML	Machine Learning	机器学习
MI	Mixup Inference	混合推理
MSE	Mean Squared Error	均方误差
NID	Network Intrusion Detection	网络入侵检测
OM-FGSM	On-Manifold FGSM	流形上 FGSM
OM-PGD	On-Manifold PGD	流形上 PGD
OOD	Out-of-Distribution	分布外
PDF	Probability Density Function	概率密度函数
PGD	Projected Gradient Descent	投影梯度下降法
PGD-DMAT	PGD-Dual Manifold Adversarial Training	PGD 对抗训练
ReLU	REctified Linear Unit	线性整流函数
RS	Randomized Smoothing	随机平滑
SGD	Stochastic Gradient Descent	随机梯度下降法
TL	Transfer Learning	迁移学习
VAD	Vanilla Adversarial Distillation	基础对抗蒸馏
VAE	Variational Auto-encoder	变分自编码器
VRS	Vanilla Randomized Smoothing	基础随机平滑

Contents

目 录

Chapter I Introduction

In this chapter, we first introduce the background and significance of research on the adversarial robustness of deep neural networks. Then, we summarize and analyze the related work. Finally, we present a detailed outline of the main content of this dissertation.

1.1 Background

In recent years, the rapid development of deep learning (DL)-based artificial intelligence (AI) has advanced the smart processes of pattern recognition and classification across various data fields, leading to significant changes in human production and lifestyles. For instance, DL-based image classification techniques improve the safety of autonomous driving systems by accurately recognizing road, traffic signals, vehicles, and pedestrians[1]. DL-based network traffic analysis techniques enhance the adaptability of network intrusion detection systems by effectively identifying abnormal traffic patterns associated with sophisticated cyber-attacks[2-7]. Furthermore, DL is widely used in medical image analysis, intelligent assistants, personalized recommendations, and more, greatly upgrading the level of intelligent services.

Deep neural networks (DNNs), as the backbone of DL, feature a multi-layered neuron structure that allows them to capture high-level features in data through rich nonlinear mappings. From simple features in the initial layers to more abstract features in the deeper layers, the hierarchical representation learning approach helps DNNs understand complex patterns in high-dimensional data that traditional machine learning (ML) models may overlook[8]. Moreover, by training on large datasets, DNNs gather knowledge from experience rather than relying on formal specifications, enabling them to learn underlying distributions in multi-dimensional spaces. This reduces bias from artificially set knowledge and enhances generalization on unseen data. Consequently, with powerful feature representation and modeling capabilities, DNNs have become a key force in advancing modern AI systems.

However, the opacity of the internal decision-making processes of DNNs has exposed AI systems to growing security risks, particularly from adversarial attacks, where the lack of adversarial robustness in DNNs has raised widespread concern. Adversarial attacks, also known as evasion attacks, occur during the deployment phase of DL models, where attackers attempt to induce incorrect predictions by feeding carefully crafted adversarial examples (AEs) into the model[9-10]. These samples are generated by applying subtle perturbations

to the original inputs, commonly known as clean examples (CEs), making AEs difficult to detect while significantly degrading model performance[11-15]. Such attacks have proven effective in various real-world DNN applications. For example, autonomous driving systems based on image classification models can be easily misled by AEs, leading to dangerous misclassification of roads and objects[16-17]. Also, network intrusion detection systems based on traffic classification models can be bypassed by AEs, allowing malicious activities to go undetected[18-22]. The rising threat of adversarial attacks emphasizes the urgent need to strengthen the defense capabilities of DNNs to ensure the security and reliability of AI systems.

Defenses against adversarial attacks are typically categorized into input-based and model-based defense approaches. Input-based defenses focus on filtering and processing data, aiming to prevent AEs from entering the model. This includes adversarial detection aimed at identifying AEs, input reconstruction aimed at denoising AEs[23-25], and more. Model-based defenses focus on enhancing the adversarial robustness of a model against AEs by changing its training or optimization. This includes adversarial training aimed at incorporating AEs for training[9,14,21], robust optimization aimed at adding adversarial regularization terms to limit the model sensitivity to inputs, etc. Although input-based defenses do not modify existing models, they often increase inference latency due to additional input processing. In contrast, model-based defenses add computational overhead during training but enhance internal robustness, reducing their impact on inference performance and offering more sustainable defenses. In this dissertation, we focus on model-based defenses aimed at improving adversarial robustness, fundamentally boosting the defense capabilities of DNNs.

Although significant progress has been made in enhancing the adversarial robustness of DNNs, some issues that seriously affect the reliability of DNNs still need to be solved.

- Firstly, generalizing the adversarial robustness of DNNs to unknown types of adversarial attacks is a struggle. Most existing adversarial training methods rely on predefined attack strategies and lack the adaptability to potential attacks. As a result, models that are robust against known types of adversarial attacks often experience a decline in robustness when faced with novel attack types. Thus, improving the generalization of the adversarial robustness of DNNs across various adversarial attacks is crucial.

- Secondly, formally guaranteeing the adversarial robustness of DNNs is challenging. Many existing defense methods empirically assess adversarial robustness by creating test sets of AEs and calculating attack success rates, which do not theoretically ensure robustness against all possible perturbations. Small input perturbations can lead to significant output changes, especially in complex and highly nonlinear DNNs, making

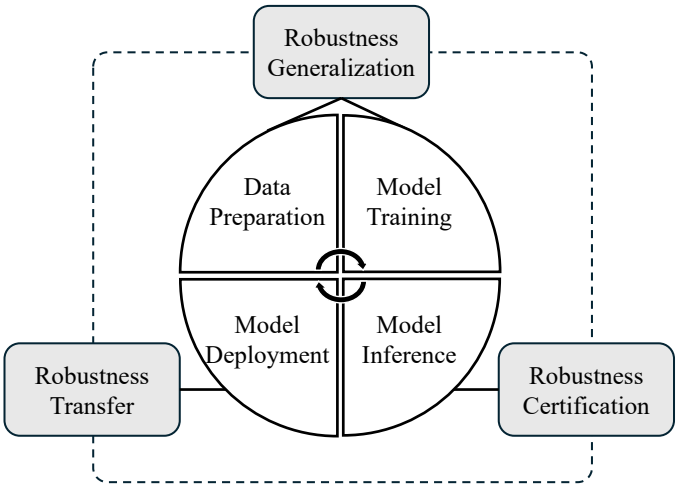


Figure 1.1 Key technologies for adversarial robustness in the lifecycle of DL models.

robustness certification particularly complicated. Therefore, developing certification methods that provide tight guarantees of adversarial robustness is essential.

- Thirdly, transferring the adversarial robustness of DNNs across tasks is difficult. Many existing robustness improvement methods focus on training robust models from scratch, requiring extensive data and computational resources. Transferring the robustness of pre-trained models to new tasks can lower learning costs, especially when data is scarce and the model is lightweight. However, differences in data distributions and model architectures between the source and target tasks complicate capturing and transferring general robustness. Thus, exploring transfer mechanisms for adversarial robustness across tasks is vital for enhancing dynamic defense while saving training costs.

In this dissertation, we focus on the risks posed by adversarial attacks in different DL scenarios and investigate key techniques in the lifecycle of DL models to enhance the adversarial robustness of DNNs. Our research is structured around three perspectives of adversarial robustness (see Figure 1.1): robustness generalization during the data preparation and training phases, robustness certification during testing, and robustness transfer during deployment. Through this study, we aim to tackle critical challenges in defending against adversarial attacks and contribute to the development of trustworthy AI.

1.2 Related Work

Despite the numerous advantages of DL techniques, the vulnerability of DNNs to adversarial attacks seriously threatens the reliability of their predictions, causing widespread concern in safety-critical fields driven by DL, such as autonomous driving and network in-

trusion detection. For this reason, improving the adversarial robustness of DNNs has emerged as an important topic in AI security research. Recently, challenging issues such as generalization, certification, and transfer strategies for adversarial robustness have been explored in academia and industry to safeguard DNNs against a wide range of adversarial attacks.

1.2.1 Adversarial Example Generation

Adversarial examples (AEs), created by adding adversarial perturbations to the original inputs, also known as clean examples (CEs), were first defined by Szegedy et al.[26] in 2014. They observed that carefully crafted small perturbations could cause misclassification in image classification models based on DNNs. These adversarial perturbations are nearly imperceptible to human vision but sufficient to make DNNs output different predicted classes when given seemingly identical CEs and AEs, revealing their adversarial vulnerability in high-dimensional spaces. Specifically, they generated these adversarial perturbations using an optimization algorithm based on box-constrained Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS). This finding exposed the security risks of DNNs against AEs and sparked subsequent research on adversarial attacks and defenses.

Initially, the adversarial vulnerability of DNNs was attributed to deficiencies in modeling the local space around the training data, as DNNs rely on large training data that can't cover all possibilities in the input space, leaving gaps for AEs[26]. Later, Goodfellow et al.[9] further explained this vulnerability through the perspective of local linearity of DNNs. Despite their nonlinear nature, DNNs behave linearly in local regions, like the activation function of the Rectified Linear Unit (ReLU), which is piecewise linear. This local linearity causes small perturbations in the input space to accumulate and amplify through model layers, eventually resulting in significant output changes. In 2015, they introduced the Fast Gradient Sign Method (FGSM), a simple and fast AE generation algorithm based on the gradient of the loss function.

Afterward, Papernot et al.[11] noticed the flaws of DNNs and proposed the Jacobian-based Saliency Map Attack (JSMA) to generate AEs based on the Jacobian matrix of the output function. Moosavi-Dezfooli et al.[12] designed a more precise algorithm, DeepFool, for minimum adversarial perturbation generation using orthogonal projection on the decision boundary. In 2017, Kurakin et al.[27] demonstrated the existence of real-world AEs by designing two iterative versions of FGSM: the Basic Iterative Method (BIM) for generating non-targeted AEs without desired classes and the Iterative Least-Likely Class Method (ILLCM) for generating targeted AEs with desired classes. They validated adversarial attacks

using photos of printed AEs. One year later, Kurakin et al.[14] evolved the simple clipping operation for perturbations exceeding the limit in BIM into a projection operation, proposing an AE generation algorithm, projected gradient descent (PGD), to ensure that the generated perturbations are always within the allowed perturbation range.

Carlini and Wagner et al.[13] introduced three new AE generation methods for the l_1-norm, l_2-norm, and l_∞-norm distance metrics based on the optimization, seeking to minimize the distance between AEs and original inputs while ensuring the model misclassifies the inputs. At the same time, they proposed the concept of high-confidence AEs, which are designed to be strongly misclassified by the original model. Unlike being barely classified as the incorrect label, these AEs are much more likely to be classified as targets than others. CW attacks are known for their precision, often bypassing many defenses that resist gradient-based attacks like PGD. Moosavi-Dezfooli et al.[28] proposed a universal adversarial perturbation algorithm, which generates a global perturbation that misclassifies most input samples by iteratively optimizing across different samples. Baluja et al.[29] proposed a faster AE generation method, the Adversarial Transformation Network (ATN), which can generate perturbations in the Perturbation ATN mode and can also generate the low-dimensional adversarial encodings of AEs in the Autoencoding ATN mode and map them to AEs through the decoder. Sarkar et al.[30] developed two algorithms, UPSET (Universal Perturbations for Steering to Exact Targets) and ANGRI (Antagonistic Network for Generating Rogue Images), for generating targeted AEs with specific desired labels, focusing on the global and sample-specific effectiveness of adversarial perturbations, respectively. In 2019, Su et al.[31] conducted a pioneering study on generating AEs by modifying only a single pixel in extreme cases. They proposed a one-pixel adversarial attack, which searches the position and color of pixels through differential evolution to find a single pixel that maximizes the prediction error. Croce et al.[32] proposed AutoAttack in 2020, an ensemble of diverse parameter-free attacks, including two white-box PGD versions, white-box FAB[33], and black-box[34] attacks.

1.2.2 Adversarial Robustness Generalization

In recent years, robust training methods, such as adversarial training and mix-up training, have been widely studied to enhance the adversarial robustness of DNNs and generalize to unknown types of adversarial attacks.

Adversarial Training. In the first work on AEs, L-BFGS[26], Szegedy et al. found that back-feeding AEs crafted from the clean training set to training can improve the generalization of the resulting models, strengthening the robustness against AEs crafted from the clean

test set. Soon after, Chalupka et al.[35] proposed to enhance adversarial robustness against L-BFGS and generalization on out-of-distribution (OOD) data by learning truly impactful features, also known as causal features, which are less sensitive to noise and perturbations. In 2015, Goodfellow et al.[9] first formally defined the FGSM-based adversarial training (AT), which enhanced the adversarial robustness of the model against FGSM by training on a mixture of FGSM AEs and CEs. Subsequently, Huang et al.[36] formalized the learning process of previous heuristic AT as a min-max problem, making the model inherently robust to AEs. Shaham et al.[37] proposed the concept of robust optimization, achieving simultaneous adversarial perturbation generation and model parameter updates through alternating two goals between the maximum and minimization.

A variety of known-type attacks can be used for AT to improve the generalization of the adversarial robustness. Moosavi-Dezfooli et al.[12] proposed the DeepFool-based AT, using DeepFool AEs to expand the training set, surpassing the adversarial robustness of the model learned by FGSM-AT. Kurakin et al.[38] designed and compared FGSM-AT, BIM-AT, and ILLCM-AT, finding that the transferability of multi-step AEs (BIM and ILLCM) was slightly lower than that of single-step AEs (FGSM), and that the AT defense model performed better against AEs than CEs. In 2018, Madry et al.[14] introduced the PGD into AT. By optimizing the parameters to minimize both the loss on CEs and the loss on PGD AEs, PGD-AT significantly improves the adversarial robustness against FGSM, PGD, and CW. Due to its wide effectiveness, PGD-AT has set a benchmark for evaluating the adversarial robustness of DNNs. In 2019, Zhang et al.[39] proposed Tradeoff-inspired Adversarial Defense via Surrogate-loss minimization (TRADES). This AT approach addresses the tradeoff between adversarial robustness and regular predictive performance by introducing a regularization term that penalizes the difference between outputs on CEs and AEs.

AT defenses are further divided into off-manifold AT and on-manifold AT[40], aiming to involve off-manifold AEs and on-manifold AEs to generalize model robustness. Lin et al.[41] proposed PGD-based Dual Manifold Adversarial Training (PGD-DMAT) in 2020, which boosts the regular predictive performance on CEs and robustness against latent-space adversarial attacks (a.k.a. on-manifold attacks, like OM-FGSM and OM-PGD) by conducting AT in the latent space of inputs (a.k.a. on-manifold AT). Meanwhile, it improves robustness against input-space adversarial attacks (a.k.a. off-manifold attacks, like FGSM and PGD) through AT in the input space (a.k.a. off-manifold AT). However, AT always relies on prior knowledge of the attack algorithm, which limits the generalization of the resulting robustness to new adversarial attacks not seen during training.

Mixup Training. In 2017, Zhang et al.[42] proposed a novel training algorithm, input-space mixup training (InputMixup), from the perspective of data augmentation, which generates mixed examples (MEs) by convexly combining two samples and their labels from the training set and then feeding them into the DNN for training. InputMixup enables the model to learn transition regions between samples in the input space, improving generalization and robustness against unseen adversarial attacks without introducing AEs. Afterward, many works extending from InputMixup were studied, mainly divided into two types: linear combination-based and binary mask combination-based mixup training.

In linear combination-based mixup, training samples are combined by linear interpolation, which can enhance the model performance in smooth transition areas between samples. In 2020, Lee et al.[43] proposed Adversarial Vertex Mixup (AVmixup) to mitigate the overfitting of adversarial features in adversarial training. By convexly combining the clean example and the adversarial vertex, which is located in the same direction as the adversarial example but several times farther away, the AVmixup algorithm generates MEs for training. This improves the generalization of adversarial robustness and eases the trade-off between regular predictive performance and adversarial robustness. Inspired by Mixup, Pang et al.[44] extended mixup training to Mixup Inference (MI) performed during the testing phase. By mixing the input test samples with other clean samples, the MI algorithm reduces the adversarial perturbations that may be contained in the input test samples. Specifically, MI is subdivided into two modes: MI with Predicted Label (MI-PL), performed using the initial predicted label on the input test sample, and MI with Other Labels (MI-OL), operated with labels other than the predicted label. These two strategies focus on the most confident prediction and the generalization to unseen adversarial examples, respectively.

In binary mask combination-based mixup, binary masks composed of 0 and 1 are used to determine the source of each local space, which can generate more complex MEs. In 2019, Yun et al.[45] proposed the first binary mask combination-based mixup algorithm, CutMix, to address issues like violent black pixel coverage and random noise patches in regional dropout by combining training samples with masks. Later, Kim et al.[46] addressed the virtuality issue in synthetic mixed examples by proposing Puzzle Mixup (PuzzleMixup), a mixup training algorithm that uses the salient and statistical information of clean samples to optimize mixed sample space, achieving superior adversarial robustness compared to earlier binary mask combination-based mixup methods. However, mixed examples created in the input space are perceptually unnatural and can't be considered samples drawn from the underlying data distribution. Moreover, it is challenging to effectively use input-space interpolation ratio

information in the feature space to modify the decision boundary.

Guo et al.[47] studied the interpretability of mixup training. From the perspective of off-manifold regularization, mixup training is interpreted as imposing local linear constraints on the input space beyond the data manifold. However, they expressed concerns about manifold intrusion, emphasizing that MEs can invade regions of the input space not represented by the original training data, falling outside the true data manifold. This may lead to less effective generalization as the model may learn to fit mixed data points that do not reflect the underlying data distribution. To this end, they proposed an adaptive version of MixUp (AdaMixUp) to automatically learn the mixup parameters, enhancing the effectiveness of mixup training. To smooth the decision boundary of DNNs, Verma et al. proposed to combine feature maps of different inputs in the random selected hidden layer of a classifier via linear combination-based ManifoldMixup[48] and binary mask combination-based PatchUp[49]. In this way, compared to the input-space interpolation ratio information, hidden representation-space interpolation ratio information provides a more appropriate guiding signal for smoothing the decision boundary. Unfortunately, their work oversimplifies the data manifold as feature maps in the DNN, failing to capture the underlying geometric structure formed by the distribution of data points in a high-dimensional feature space.

Although exact data manifold learning has not been explored in mixup training, work on latent interpolation in unsupervised learning has made essential contributions to this. In 2018, Liu et al.[50] proposed a latent space interpolation (LSI) method for data augmentation. They used an adversarial autoencoder (AAE)[51] to learn feature representations in the latent space and generated new training samples through linear interpolation, which improved the generalization of DNNs. Sainburg et al.[52] introduced Generative Adversarial Interpolative Autoencoding (GAIA), an adversarial training method based on latent space interpolation. GAIA uses an autoencoder (AE)-generative adversarial network (GAN) architecture to capture data representations in the latent space and perform convex combinations. Compared to input-space interpolation, the samples synthesized by latent-space interpolation are closer to the underlying data distribution and, thus, more challenging to distinguish from real data. Berthelot et al.[53] pointed out that synthetic samples decoded by autoencoder from the convex combination of latent representation blend input features and proposed a consistency regularization algorithm, Adversarially Constrained Autoencoder Interpolation (ACAI), to enhance the realism of interpolated outputs. Cemgil et al.[54] proposed a regularization method for variational auto-encoder (VAE)[55] based on a selection mechanism, which can learn representations in the latent space that are insensitive to adversarial pertur-

bations. They demonstrated the positive significance of latent-space data augmentation for robust representation learning. Beckham et al.[56] proposed adversarial mixup resynthesis (AMR) for latent representation mixup, which utilizes an AE-GAN architecture to generate representations and mix them under two modes: linear and binary mask combination.

Mixup training improves the generalization of adversarial robustness against unknown types of adversarial attacks, as it does not rely on specific AE generation algorithms. However, existing mixup training methods primarily address off-manifold (input-space) adversarial attacks, neglecting on-manifold (latent-space) attacks occurring on the data manifold. Additionally, the limited expressivity of classifier hidden layers restricts their ability to capture the complexity of the underlying manifold, while mixing entangled features may fail to yield realistic input samples, potentially disrupting boundary learning. Furthermore, the necessary modifications to hidden layer architectures in existing works reduce the flexibility, complicating the application of these mixup training methods across various DNNs.

1.2.3 Adversarial Robustness Certification

Against adversarial attacks, heuristic strategy-driven empirical defenses, such as adversarial training and mixup training, can usually be adaptively attacked again [57] and cannot verify the resulting robustness. To deal with the endless arms race of adversarial attack and defense, research on the adversarial robustness of DNNs has gradually shifted to certified defense represented by robustness certification[58].

Robustness Certification. Such a defense aims to calculate a certified robust radius (a.k.a. certified radius) for each input as a robustness guarantee for the DNN. The radius is defined as the maximum range of perturbation in the input that does not cause the output label to change. This formal guarantee ensures that, within the specified perturbation range bounded by a certified radius measured by the l_p norm, no AE can cause the model to misclassify the input. By doing so, certified defenses not only provide theoretical robustness guarantees but also serve as a critical metric for evaluating the robustness of DNNs against all possible adversarial manipulations. When using different robustness certification algorithms for the same model and input, a larger certified radius indicates a tighter robustness guarantee provided by the robustness certification method. When different models are certified on the same input with the same robustness certification algorithm, the larger the certified radius obtained, the better the certified robustness of the model.

There are two types of robustness certification approaches: complete certification and incomplete certification. The relationship between the certified radius and the exact robust

radius of the model on a specific input determines whether the robustness certification is complete or incomplete. Complete certification provides a certified radius equal to the exact robust radius, and incomplete certification provides a certified radius as a lower bound of the exact robust radius. Therefore, when complete certification returns failed certification for a sample, it guarantees the existence of an AE nearby. When incomplete certification returns failed certification, it does not guarantee this, as the model could be more robust than the certified radius suggests. From another perspective, robustness certification can also be divided into deterministic and probabilistic approaches. Deterministic certification guarantees to produce not certified when the input is non-robust, whereas probabilistic certification does so with a certain probability. Most complete certification methods are deterministic, while incomplete certification encompasses both deterministic and probabilistic types[58].

Complete Certification. Complete certification provides strong robustness guarantees by ensuring that no AEs are within a certified radius around any input point. In 2017, Cheng et al.[59] utilized mixed integer linear programming (MILP) constraints to encode the non-linear ReLU operations and the whole model, precisely encoding the certification problem as a MILP problem. Tjeng et al.[60] extended the model size supported by MILP-based complete certification to convolutional networks, achieving computational speedups through a presolve algorithm. However, MILP-based methods mainly apply to medium-sized models and are difficult to scale to DNNs.

The boundary propagation-based approach improves the scalability of complete certification on DNNs. In 2018, Zhang et al.[61] proposed CROWN, a general framework to certify the robustness of DNNs through bound propagation. By propagating the input boundary in the network, the upper and lower bounds of each neuron output are calculated layer by layer, thereby obtaining the output bound of the entire network. CROWN bounds activation functions with linear and quadratic functions, allowing it to tackle DNNs with general activation functions. Gowal et al.[62] simplified the bounding technique in CROWN to interval bound propagation (IBP), which applies linear propagation at each layer, fasting the certification but leading to loose bounds. Later, Zhang et al.[63] introduced CROWN-IBP, which combines the fast IBP bounds in a forward bounding pass with a tight CROWN bound in a backward bounding pass, achieving better computational performance than IBP baselines. Xu et al.[64] developed an automatic framework to generalize CROWN to general computational graphs, enabling certification on any DNN structure. Lyu et al.[65] proposed a relaxed version of CROWN, linear bound propagation (LBP), which can certify large DNNs and obtain lower certified errors than IBP. In 2021, Wang et al.[66] introduced a branch-and-bound strategy to

the bound propagation-based CROWN framework, proposing β-CROWN. In 2022, Zhang et al.[67] proposed GCP-CROWN, which enhances the strength and scalability of CROWN by introducing a general cutting plane to the bound propagation. In 2024, Zhang et al.[68] analyzed the theoretical upper bound of certified robust accuracy using Bayes error, proving that robustness inherently reduces accuracy. Complete certification provides strict robustness guarantees but is computationally complex, making it less applicable to high-dimensional inputs and large-scale models due to the time and resources required.

Incomplete Certification. Incomplete certification, which uses an approximation of the exact robust radius of the model as the certified radius, avoids the NP-complete challenge of computing the exact robust radius of a DNN in complete certification[58,69]. However, a notable drawback of incomplete certification is that the robustness guarantee provided is loose; that is, the provided certified radius is far from the exact robust radius. Many approaches have been proposed to upgrade incomplete certification algorithms for image classifiers to compute the non-trivial certified radius for DNNs.

Deterministic incomplete certification mainly involved linear relaxation-based and Lipschitz constant-based certification approaches. Most linear relaxation-based approaches rely on activation polytope[70-74], which use activation values of the hidden layers of the DNN to form the vertices of the polytope, and calculating the volume and surface area of the polytope to evaluate the robustness. Lipschitz constant-based approaches aim at computing a tight Lipschitz bound for general DNNs to quantify the change in the model's output under input perturbations. Since the global Lipschitz constant is usually too loose, Lee et al.[75] improved it by performing fine-grained analysis of the convolutional layers. Fazlyab et al.[76] computed the local Lipschitz bounds by posing the Lipschitz constant estimation problem as a semidefinite program (SDP), increasing the estimation accuracy.

Probabilistic incomplete certification mainly includes differential privacy-based and randomized smoothing-based certification approaches. Most differential privacy (DP)-based approaches usually use the first-order derivative of the output function. In 2019, Lecuyer et al.[77] first introduced a DP mechanism to add noise to hidden layers of the model during training and regularized the output, providing the l_2 bounded robustness guarantee to the model. Later, Phan et al.[78] proposed a scalable certification method that combines DP with adversarial training, where DP is applied to the gradient update process by adding noise during model training, thereby limiting the sensitivity of the model to data points.

An ideal certified defense against adversarial attacks should be model agnostic and applicable to various DL models without relying on their specific structures. Randomized

smoothing (RS)-based approaches are the most competitive for tight and architecturally scalable certification and can be categorized into zero-order and first-order information-based techniques. In 2019, Cohen et al.[79] innovatively proposed vanilla randomized smoothing (VRS), transforming any classifier into a smoothed version. The smoothed classifier makes several predictions on noisy versions of the input with Gaussian noise and outputs the majority vote as its final decision. Later, Lee et al.[69] expand the Gaussian-based RS across broad classes of distributions, offering robustness guarantees against l_0 norm-bounded attackers. In 2022, Hao et al.[80] proposed generalized randomized smoothing (GSmooth), a unified framework for certifying robustness against general semantic transformations. Unlike most RS works that only use the zero-order information, i.e., the output prediction of the smoothed classifier, Mohapatra et al.[81] proposed the First Order-based Randomized Smoothing (FRS), which uses the first-order gradient information of the smoothed classifier together with the zero-order output information to calculate a tighter certified radius bounded by l_2 norm.

Existing randomized smoothing (RS) methods are mainly designed for DNNs with homogeneous input features, like image classifiers. However, they cannot be applied to DNNs with heterogeneous input features, such as network traffic classifiers. In 2023, Wang et al.[82] introduced Boundary-Adaptive Randomized Smoothing (BARS), the first robustness certification method aimed to provide l_2 robustness guarantees for network traffic classifiers. However, since it relies only on zero-order information, its l_2 norm-bounded certified radius may not be tight enough and lacks support for other norm-bounded certificates.

1.2.4 Adversarial Robustness Transfer

Robustness-preserving transfer learning (TL) methods, such as adversarial fine-tuning and adversarial distillation, are the main techniques for transferring adversarial robustness between DNNs. These approaches aim to ensure that the target model performs well on regular samples and remains robust against adversarial attacks in the target domain.

Adversarial Fine-tuning. In 2019, Hendricks et al.[83] proposed adversarial fine-tuning (AFT) to enhance the robustness of a target model by fine-tuning a pre-trained robust source model using target-domain adversarial examples. However, compared with conventional fine-tuning, this mechanism reduces the accuracy on the target domain clean samples[84]. To balance the robustness with regular predictive performance, Hafahi et al.[85] proposed a robustness-preserving TL framework, Freeze Robust Feature Extractor (FRFE), to fine-tune the robust source model using only clean data. In 2021, Fan et al.[86] introduced ADCersarial Contrastive Learning (ADVCL) to explore AFT within self-supervision, avoiding re-

liance on labeled data. Yamada et al.[87] proposed Fixed-Feature Transfer Learning (FFTL) to tackle robustness against common image corruptions like noise and blur. In 2023, Liu et al.[88] developed a finetuning framework, TwoWIng NormliSation (TWINS), which incorporated vanilla AFT with dual batch normalization (BN)[89] for robustness in downstream tasks, leveraging the statistics information from a frozen BN layer. However, existing AFT methods that optimize adversarial and generalization objectives can lead to gradient direction divergence[90]. Moreover, except for FRFE[85], none of these methods consider the data scarcity problem in the target domain. They mainly fine-tune models pre-trained on complex source domains like ImageNet-1000 for simpler target domains like CIFAR-10, neglecting the challenging task of adapting to target domains with more categories, which is crucial for network intrusion detection that often faces emerging threats. Since AFT typically adapts a pre-trained model by adjusting weights and output dimensions, it cannot transfer robustness to a target model with significant structural differences.

Adversarial Distillation. Adversarial distillation (AD)[91], as known as vanilla adversarial distillation (VAD), proposed by Goldblum et al. in 2020, aims to transfer robustness from a source to a smaller target model by minimizing Kullback-Leibler (KL) divergence between teacher and student outputs on adversarial images. Later, Muhammad et al.[92] proposed MixACM, which aims to distill activated channel maps of hidden layers on mixed samples. Considering teachers pre-trained on source-domain adversarial data may become unreliable in predicting target-domain adversarial data queried by students, Zhu et al.[93] designed introspective adversarial distillation (IAD) to encourage student models to trust their teachers for adversarial robustness partially. Zi et al.[94] proposed Robust Soft Label Adversarial Distillation (RSLAD) to train robust small student models, which exploits the robust soft labels produced by an adversarially-trained robust large teacher model to guide the student's learning on both CEs and AEs. In 2022, Maroto et al.[95] developed Adversarial Knowledge Distillation (AKD), a new AD framework to boost the robustness by adversarially training a student on a mixture of ground truth labels and teacher outputs. Bai et al.[96] introduced Guided Adversarial Contrastive Distillation (GACD), which uses contrastive distillation based on AEs to generate robust target models on CIFAR-10. In 2023, huang et al.[97] proposed a new AD framework, Adaptive Adversarial Distillation (AAD), further boosting the robustness of students by dynamically updating AEs in the distillation process. However, these AD methods do not fully interact with the teacher models to minimize prediction discrepancies between source and target models, limiting the robustness and regular predictive performance inherited by the student model. In addition, they focus more on cross-model

distillation in the same learning task, ignoring TL scenarios across data domains and the scarcity of training data in the target domain.

1.3 Our Contributions

In this dissertation, we focus on the threats from adversarial attacks in deep learning (DL) scenarios and explore three key issues: generalizing the adversarial robustness of DNNs to unknown attack types, providing formal guarantees for the adversarial robustness of DNNs, and transferring the adversarial robustness of DNNs across various tasks. We study three key techniques of adversarial robustness in DNNs, including generalization, certification, and transfer. Our primary contributions are as follows:

1. We propose Latent Representation Mixup (`LarepMixup`), a robust training framework to enhance the generalization of adversarial robustness in DNNs. Although adversarial training can strengthen robustness against specific attacks, it struggles to generalize to new or unseen ones. To address this, we present a mixup training-based defense mechanism that does not depend on prior attacker knowledge. Initially, we develop a data augmentation approach based on dual-mode manifold interpolation, creating mixed samples near decision boundaries or following the underlying distribution through convex and binary mask mixing. Then, we design a multi-label training algorithm using mixed semantic samples and mixed labels to smooth the decision boundary of the DNN, enhancing robustness against perturbations near the boundary. Experiments on various image classifiers and datasets demonstrate that our proposed method improves both pixel-level and representation-level robustness in white-box and black-box scenarios, enhancing generalization of robustness across diverse input and latent space perturbations (e.g., FGSM, PGD, AutoAttack, DeepFool, CW, OM-FGSM, and OM-PGD) compared with the state-of-the-art (SOTA) robust training methods (e.g., PGD-AT, PGD-DMAT, InputMixup, CutMix, PuzzleMixup, ManifoldMixup, and PatchUp).

2. We propose Multi-order Adaptive Randomized Smoothing (`MARS`), a certified defense framework to enhance the tightness of adversarial robustness certification in DNNs. While the robustness of DNNs can be empirically evaluated by assessing the attack success rate of adversarial attacks, it doesn't theoretically guarantee robustness against all input perturbations. To address this, we present a probabilistic incomplete certification method to calculate non-trivial robustness lower bound, estimating the certified robust radius for each input sample to avoid misclassification. We first design an adaptive randomized smoothing algorithm that uses zero-order and first-order information of the smoothed classifier to calculate robust radii, offering tighter certified radii than existing methods. Additionally, we

propose a dimension-wise robust radius calculation algorithm based on the sensitivity of each feature dimension, allowing for fine-grained robustness certification of heterogeneous input features. Experiments on various network intrusion detectors (e.g., CADE and ACID) and datasets (e.g., CSE-CIC-IDS-2018-CADE and CSE-CIC-IDS-2018-ACID) demonstrate that our proposed method effectively certifies adversarial robustness in larger l_p norm-bounded perturbation regions compared with the SOTA traffic-specific certification BARS, image-specific certification VRS, and FRS, enhancing robustness against diverse adversarial attacks (e.g., l_∞-PGD, l_2-PGD, l_1-EAD) and natural corruption (e.g., Latency and PacketLoss).

3. We propose Contrastive Adversarial Representation Distillation (`CARD`), a robustness-preserving transfer learning framework to transfer the adversarial robustness of DNNs across different tasks. Training a robust model from scratch typically requires extensive datasets and substantial computational resources. Transferring robustness from a robust pre-trained source model to new tasks can mitigate these demands but is challenging due to differences in data distribution and model structures, making universal robustness hard to capture. This work develops a robustness transfer mechanism for robustness transfer that isn't constrained by domain or model similarities. First, we design a distillation strategy using adaptive dimensional alignment, aligning input and hidden representation dimensions between target and source models to facilitate knowledge transfer despite data domain and model changes. Moreover, we propose a contrastive transfer learning algorithm with dual robustness-aware views, capturing domain-invariant robustness through adversarial manipulation and natural corruption views for effective transferring. Experiments on various network traffic classifiers (e.g., WideResNet-34-10, ResNet-18, and MobileNet) and datasets (e.g., UNSW-NB15 and NSL-KDD) demonstrate that our proposed method enhances the transferability of adversarial robustness across data domains and model structures, achieving superior adversarial robustness against l_∞ PGD attacks in lightweight models with limited training data compared with standard fine-tuning and distillation as well as SOTA adversarial fine-tuning (e.g., FRFE and TWINS) and adversarial distillation (e.g., VAD and AAD) in binary and multi-classification.

1.4 Organization

This dissertation contains six chapters in total, detailing the key techniques for the adversarial robustness of DNNs against adversarial attacks. The overall content structure of this dissertation is shown in Figure 1.2, and the specific content is organized as follows:

In Chapter I, we first introduce the background and significance of studying the adversarial robustness of DNNs. Then, we summarize the related work on adversarial example

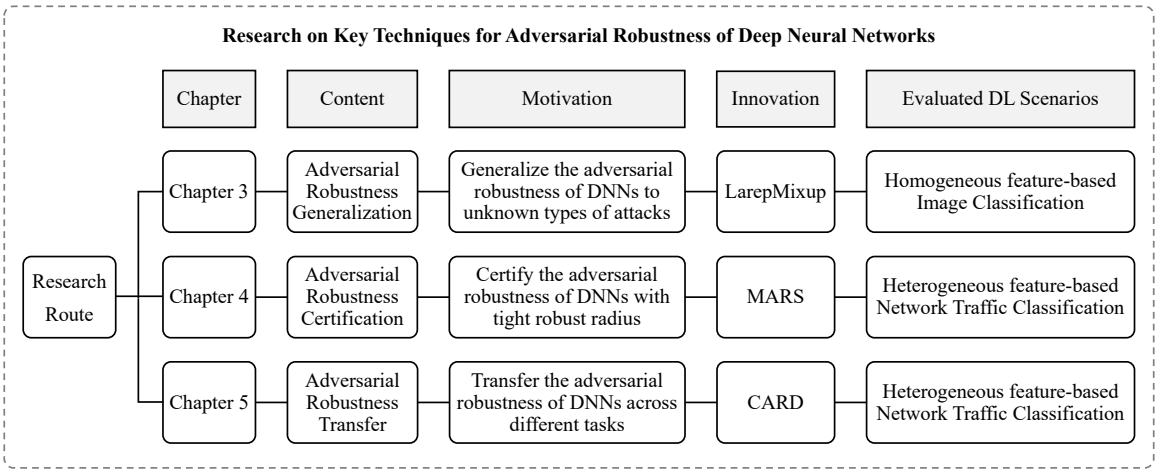


Figure 1.2 Organizational structure of this dissertation.

generation, adversarial robustness generalization, certification, and transfer, highlighting the challenges existing methods face. Next, we briefly outline the main contributions of our work. Finally, we describe the organizational structure of this dissertation.

In Chapter II, we cover the preliminaries of adversarial attacks and defenses for DNNs. First, we introduce the basic structure, working principle, and application scenarios of DNNs. Then, we present the attack types, attack principles, and threat scenarios of adversarial attacks DNNs face. Finally, we introduce the basic concepts, enhancement techniques, and evaluation methods of adversarial robustness.

In Chapter III, we study the generalization technique for adversarial robustness of DNNs and propose a robust training framework, Latent Representation Mixup (`LarepMixup`). First, we introduce a developed data augmentation approach based on dual-mode manifold interpolation, generating mixed samples near the decision boundary or following the underlying distribution. Then, we present a designed multi-label training algorithm using mixed semantic samples and mixed labels to smooth the DNN decision boundary. Finally, we show experimental results on various image classifiers and datasets, demonstrating that the proposed method improves the generalization of adversarial robustness across various input and latent space perturbations in both white-box and black-box scenarios.

In Chapter IV, we study the certification technique for adversarial robustness of DNNs and propose a certified defense framework, Multi-order Adaptive Randomized Smoothing (`MARS`). First, we introduce a designed adaptive randomized smoothing algorithm that leverages zero-order and first-order information to calculate robust radii, providing tighter lower bounds of robustness than existing methods. Then, we present a proposed dimension-wise robust radius calculation algorithm based on the sensitivity of each feature dimension, enabling

fine-grained robustness certification for heterogeneous input features. Finally, we show experimental results on various network intrusion detectors and datasets, demonstrating that the proposed method certifies robustness in larger perturbation regions and improves certified robustness against diverse adversarial attacks.

In Chapter V, we study the transfer technique for adversarial robustness of DNNs and propose a robustness-preserving transfer learning framework, Contrastive Adversarial Representation Distillation (`CARD`). First, we introduce a designed distillation strategy based on adaptive dimensional alignment, aligning input and hidden representations between target and source models. Then, we present a proposed contrastive transfer learning algorithm based on dual robustness-aware views, which captures domain-invariant robustness through adversarial manipulation and natural corruption views. Finally, we show experimental results on various network intrusion detectors and datasets, demonstrating that the proposed method enhances the transferability of adversarial robustness across data domains and model structures, achieving leading adversarial robustness for lightweight models with limited data.

In Chapter VI, we conclude this dissertation and discuss future directions. We review the main contributions, summarize the limitations, and outline possible follow-up work.

Chapter II Preliminaries

In this chapter, we present some preliminaries for this dissertation. We begin by presenting the basic structure, training and inference principles, and application contexts of deep neural networks (DNNs). Following that, we offer an overview of adversarial attacks, covering various attack types, attack principles, and the threat scenarios that DNNs encounter. Finally, we introduce the basic concepts, enhancement techniques, and evaluation methods for adversarial robustness.

2.1 Deep Neural Networks

2.1.1 Basic Structure

A deep neural network (DNN) is a type of deep learning (DL) model inspired by the structure of neurons in the human brain. It typically consists of multiple layers between the input and output, comprising an input layer, multiple hidden layers, and an output layer. The input layer handles the task of receiving data, such as images or text. The hidden layers apply activation functions for nonlinear transformations (e.g., ReLU or sigmoid). The output layer produces the final result, which could be a class label for classification tasks or a prediction for regression tasks. In this dissertation, we focus on DNN-based classification tasks.

DNNs are often used to model complex relationships in high-dimensional data by learning abstract features through multiple layers. For a given input $x \in \mathcal{X}$, the DNN is represented as a classification function f_θ: $\mathcal{X} \to \mathcal{Y}$, where the model's output $f_\theta(x)$ is a C-dimensional probability vector, referred to as the *confidence score vector*, which indicates the confidence levels of x across all C classes. Formally, a DNN with L layers is expressed as Eq. (2-1):

$$f_\theta(x) = f(x;\theta) = f_L(f_{L-1}(...f_1(x))) \tag{2-1}$$

where x represents the input, θ refers to the model's parameters, f is the learned prediction function, and f_i signifies the transformation carried out by the i-th layer.

Each transformation f_i typically involves a linear operation combined with a nonlinear activation function, commonly formulated as Eq. (2-2).

$$f_i(x) = \sigma(W_i \cdot x + b_i) \tag{2-2}$$

where W_i is the weight matrix of the i-th layer, b_i is the bias term of the i-th layer, and σ is a nonlinear activation function, such as ReLU, sigmoid, softmax, etc.

2.1.2 Training and Inference Principles

The working principle of DNN models consists of two main phases: training before deployment and inference after deployment.

Model Training Phase. The training process of the model f_θ typically involves optimizing the model parameters θ, which consist of weights W and biases b, by using a training set D_{train} to minimize a specific loss function $\mathcal{L}$, such as Cross-Entropy or Mean Squared Error (MSE). Specifically, during the training of a classifier with C categories, f_θ is supervised to map each sample x in D_{train} to a C-dimensional confidence score vector $f_\theta(x)$ with minimal loss for the C-dimensional one-hot vector corresponding to its ground-truth label $y_{true} \in [C] \equiv \{1, ..., C\}$. This can be formulated as a minimization problem in Eq. (2-3):

$$\min_{\theta} \mathbb{E}_{(x,y_{true})\sim D_{train}}[\mathcal{L}(f_\theta(x), y_{true})] \tag{2-3}$$

where (x, y_{true}) are the input-output pairs from the training dataset D_{train}, θ represents the trainable parameters of the model, $\mathcal{L}$ indicates the loss function, and $\mathbb{E}$ denotes the expectation function.

To minimize the prediction error assessed by the loss function, gradient-based optimization techniques like Stochastic Gradient Descent (SGD) are commonly employed to iteratively update the model parameters θ using backpropagation. Taking optimization with a single training example as an instance, the updating of θ can be expressed as Eq. (2-4):

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true}) \tag{2-4}$$

where, η signifies the learning rate, $\mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true})$ represents the loss function calculated for an input-output pair $(x^{(i)}, y^{(i)}_{true})$ randomly selected from the training dataset D_{train}, and $\nabla_\theta \mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true})$ indicates the gradient of the loss function $\mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true})$ concerning the parameters θ, pointing in the direction of the steepest increase of the loss function. During training, the goal is to minimize the value of the loss function. By subtracting the gradient $\nabla_\theta \mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true})$, the parameters θ are updated in the direction where the loss function $\mathcal{L}(f_\theta(x^{(i)}), y^{(i)}_{true})$ decreases. Another common form of Eq. (2-4) in related work is shown in Eq. (2-5), which expresses the same meaning. In this dissertation, we uniformly use $\mathcal{L}$ to denote the loss function.

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}, y^{(i)}_{true}) \tag{2-5}$$

where $\nabla_\theta J(\theta; x^{(i)}, y^{(i)}_{true})$ represents the gradient of the loss function $J(\theta; x^{(i)}, y^{(i)}_{true})$ computed based on $(x^{(i)}, y^{(i)}_{true})$) w.r.t the model parameters θ.

Model Inference Phase. The inference process involves applying the trained model to predict the class for new, unseen inputs, keeping the model structure unchanged and utilizing the parameters learned during training. Formally, the base classifier F for inference built on the trained DNN model f_θ is defined in Definition 1.

Definition 1 (Base Classifier)**.** *Given a DNN model f_θ that maps input x to a C-dimensional confidence score vector $f_\theta(x)$, predicted class y output by a base classifier F is defined as:*

$$y = F(x) = \arg\max_{i \in [C]} f_\theta(x)_i \tag{2-6}$$

where $f_\theta(x)_i$ denotes the confidence score output by f_θ on the i-th category for the input x.

In this dissertation, the base classifier is defined as the classifier trained using the standard process without incorporating robust defense training. The output y of the base classifier represents the index of the highest value in the confidence score vector generated by the model f_θ; that is, the model selects the class with the highest confidence score as its predicted label.

2.1.3 Application Scenarios

DNNs have diverse application scenarios across various domains. In computer vision (CV), they excel in image classification, object detection, and segmentation. In natural language processing (NLP), they enable sentiment analysis, machine translation, and text generation. They are also utilized in cybersecurity for intrusion detection, malware analysis, and phishing detection. Their ability to learn complex patterns from large datasets makes DNNs valuable tools for both supervised and unsupervised learning tasks. This dissertation focuses on classification tasks with supervised learning. Two application scenarios are mainly explored: image classification with homogeneous input features and network traffic classification with heterogeneous input features.

Image Classification. DNNs have been extensively researched and utilized for image classification tasks, aiming to accurately assign input images to predefined labels based on their visual features. In image classification, each image sample from the dataset is represented as a multi-dimensional array of pixel values, specifically a $H \times W \times C$ tensor, where, H represents the height, W) signifies the width, and C indicates the color channels. The feature dimensions of the input image vector $x \in \mathcal{X} := \mathbb{R}^{H \times W \times C}$ for the image classifier are homogeneous. Each feature dimension consistently captures a specific attribute of the

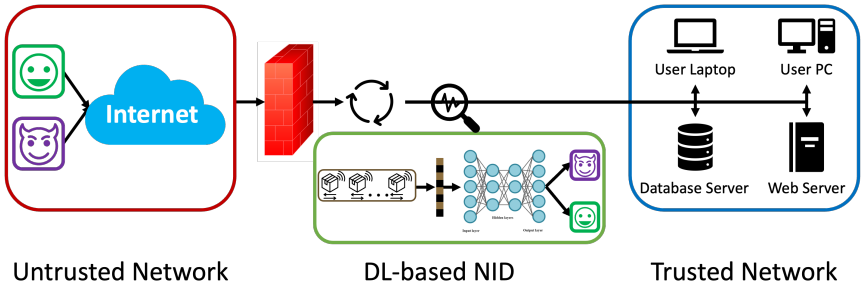


Figure 2.1 Application of DNN-based network traffic classifier in network intrusion detection (NID).

image, such as the pixel values in the RGB channels representing color intensities, which are all measured on a similar scale, ranging from 0 to 255 or normalized between 0 and 1.

The homogeneity of feature dimensions in image samples allows DNNs, particularly convolutional neural networks (CNNs), to learn spatial hierarchies and patterns effectively. the DNN model f_θ: $\mathcal{X} \to \mathcal{Y}$ behind the image classifier F is trained according to Eq. (2-3) and used for inference according to Eq. (2-6). The uniform structure facilitates the application of CNN filters that can capture spatial relationships, such as edges, textures, and shapes, across different parts of the image. As such, DNNs can exploit the consistency in dimensionality to build feature maps that enhance image recognition accuracy.

Generally, DNN-based image classifiers can be applied in various fields, such as detecting anomalies in X-rays or magnetic resonance imaging in medical imaging, identifying objects in autonomous vehicles for safe navigation, recognizing individuals in facial recognition systems for security and access control, and classifying images in multimedia databases for content-based image retrieval.

Network Traffic Classification. DNNs have also been increasingly studied and applied in network traffic classification tasks, where the goal is to identify abnormal behavior by categorizing input network traffic into predefined classes, such as benign or malicious (See Figure 2.1). In addition to the binary classification of benign and malicious, network traffic classification can also be formalized as a multi-class problem to detect different types of network attack behaviors more fine-grained. In network traffic classification, each network traffic sample in the dataset consists of d network traffic features, including source and destination IP addresses, port numbers, protocol types, timestamps, packet sizes, flow durations, and more. Unlike images, the feature dimensions of the input traffic vector $x \in \mathcal{X} \coloneqq \mathbb{R}^d$ for the network traffic classifier are heterogeneous. Each of these features can be of different types and scales. For example, packet sizes are numerical values, representing continuous quantities, while protocol types are categorical, representing distinct classes.

Given the heterogeneity of feature dimensions in network traffic samples, specialized

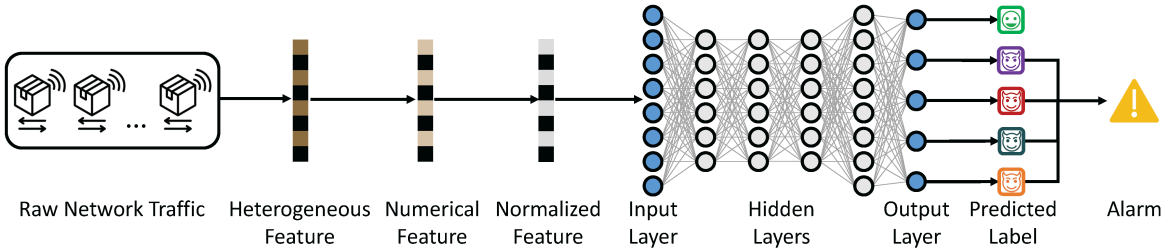


Figure 2.2 Workflow of the DNN-based network traffic classifier.

preprocessing is required to ensure that diverse features are appropriately encoded or normalized for model input. Specifically, for a given d-dimensional raw network traffic feature vector x_{raw} containing some non-numerical feature dimensions (such as protocol, network service, timestamp, etc.), the vector is first transformed into a numerical feature vector x_{nmr} and then normalized into a feature vector $x \in \mathcal{X} \equiv \mathbb{R}^d$ that belongs to a continuous real number range. Then the DNN model f_θ: $\mathcal{X} \to \mathcal{Y}$ behind the network traffic classifier F is trained according to Eq. (2-3) and used for inference according to Eq. (2-6). The DNN-based network traffic classifier workflow is shown in Figure 2.2.

By leveraging the ability of DNNs to learn hierarchical and high-level representations from network traffic data, they can capture nonlinear relationships among heterogeneous features, enabling them to better model the interactions and correlations within network traffic data, thereby being adaptable and effective in identifying complex patterns and anomalies. DNN-based network traffic classifiers are valuable in several critical cybersecurity applications, such as analyzing network traffic to detect unauthorized access or potential threats in network intrusion detection (NID), examining traffic patterns between infected devices and command-and-control servers to detect malware, and identifying web behaviors such as redirect chains, domain characteristics, or URL structures in phishing detection.

2.2 Adversarial Attacks

2.2.1 Basic Concept

Adversarial attacks, commonly referred to as evasion attacks, aim to exploit the robustness vulnerabilities of DNNs with sophisticated techniques, causing the model to produce incorrect predictions. Specifically, adversarial attacks take place during the model's deployment phase, where attackers attempt to induce misclassification by feeding carefully crafted adversarial examples (AEs) into the model. An adversarial example (AE) is a modified version of the original input, a.k.a clean example (CE), created by introducing minor perturba-

tions to the original inputs, expressed as Eq. (2-7):

$$x^* = x + \delta \tag{2-7}$$

where x^* represents the adversarial example, x denotes the clean example, and δ signifies the adversarial perturbation. Despite the adversarial manipulation, the adversarial example (AE) closely resembles the clean example (CE) to human observers and can substantially impair the model's performance, as outlined in Definition 2.

Definition 2 (Adversarial Attack)**.** *Given a classifier F built on trained DNN model f_θ with C classes, a clean input x with the ground-truth label y_{true}, an adversarial attack is characterized as the process of identifying a small adversarial perturbation δ that satisfies:*

$$y = F(x + \delta) = \arg\max_{i \in [C]} f_\theta(x + \delta)_i \neq y_{true} \tag{2-8}$$

where $F(x + \delta)$ denotes the predicted class of the classifier for the perturbated input $x + \delta$.

In this dissertation, we mainly consider l_p adversarial attacks, as defined in Definition 3. The size of the perturbation δ introduced to the input x is limited by the l_p norm (e.g., ${l_0}^{[31]}$, ${l_2}^{[11\text{-}13,26,28\text{-}29]}$, and ${l_\infty}^{[9,11,13,27\text{-}28,38]}$), ensuring that the changes made to the clean input do not exceed a certain *perturbation budget* ϵ so that the perturbation is imperceptible. The adversarial attacks described in this dissertation are all l_p adversarial attacks.

Definition 3 (l_p Adversarial Attack)**.** *Given a classifier F built on trained DNN model f_θ with C classes, a clean input x with the ground-truth label y_{true}, an l_p adversarial attack is described as the process of locating a small adversarial perturbation δ that satisfies:*

$$y = F(x + \delta) = \arg\max_{i \in [C]} f_\theta(x + \delta)_i \neq y_{true}, ||\delta||_p \leq \epsilon \tag{2-9}$$

where $F(x + \delta)$ denotes the predicted class of the classifier for the perturbated input $x + \delta$, $||\delta||_p$ is the l_p norm-measured perturbation magnitude, ϵ is the perturbation budget.

2.2.2 Attack Objectives

Depending on whether the attack target is specified, adversarial attacks can be categorized into two types: untargeted and targeted attacks, corresponding to two different adversarial perturbation optimization objectives when generating AEs.

Untargeted Adversarial Attack. The attacker aims to deceive the model into making a wrong prediction without targeting a specific class. The untargeted adversarial attack with a perturbation budget ϵ constrained by the l_p norm is defined in Definition 4.

Definition 4 (Untargeted l_p Adversarial Attack)**.** *Given a classifier F built on trained DNN model f_θ with C classes, a clean input x with the ground-truth label y_{true}, the attack objective of a l_p adversarial attack is defined as finding a slight adversarial perturbation δ such that:*

$$y = F(x+\delta) = \arg\max_{i \in [C]} f_\theta(x+\delta)_i \neq y_{true} \tag{2-10}$$

where $F(x+\delta)$ indicates the predicted class of the classifier for the perturbed input $x+\delta$. The goal of generating the adversarial perturbation δ is defined as:

$$\max_{||\delta||_p < \epsilon} \mathcal{L}(f_\theta(x+\delta), y_{true}) \tag{2-11}$$

where where $||\delta||_p$ represents the magnitude of the perturbation measured by the l_p norm, ϵ denotes the perturbation budget, and $\mathcal{L}$ signifies the loss function.

Targeted Adversarial Attack. The attacker seeks to cause the model to classify the input as a specific incorrect target class. The targeted adversarial attack with a perturbation budget ϵ constrained by the l_p norm is defined in Definition 5

Definition 5 (Targeted l_p Adversarial Attack)**.** *Given a classifier F built on trained DNN model f_θ with C classes, a clean input x with the ground-truth label y_{true}, a target label y_{target} desired by the attacker, the goal of an l_p adversarial attack is defined as identifying a small perturbation δ that satisfies:*

$$y = F(x+\delta) = \arg\max_{i \in [C]} f_\theta(x+\delta)_i = y_{target} \tag{2-12}$$

where $F(x+\delta)$ denotes the predicted class of the classifier for the perturbated input $x+\delta$. The optimization objective of adversarial perturbation δ generation is defined as:

$$\min_{||\delta||_p < \epsilon} \mathcal{L}(f_\theta(x+\delta), y_{target}) \tag{2-13}$$

where where $||\delta||_p$ represents the perturbation magnitude measured by the l_p norm, ϵ indicates the perturbation budget, and $\mathcal{L}$ denotes the loss function.

2.2.3 Attack Types

According to the knowledge available to the attacker about the victim model f_θ, adversarial attacks can be classified into white-box and black-box attacks.

White-Box Adversarial Attack. The attacker possesses full knowledge of the victim model f_θ, which encompasses its architecture, parameters, and training dataset. This extensive insight enables the attacker to design highly effective AEs by directly altering the input

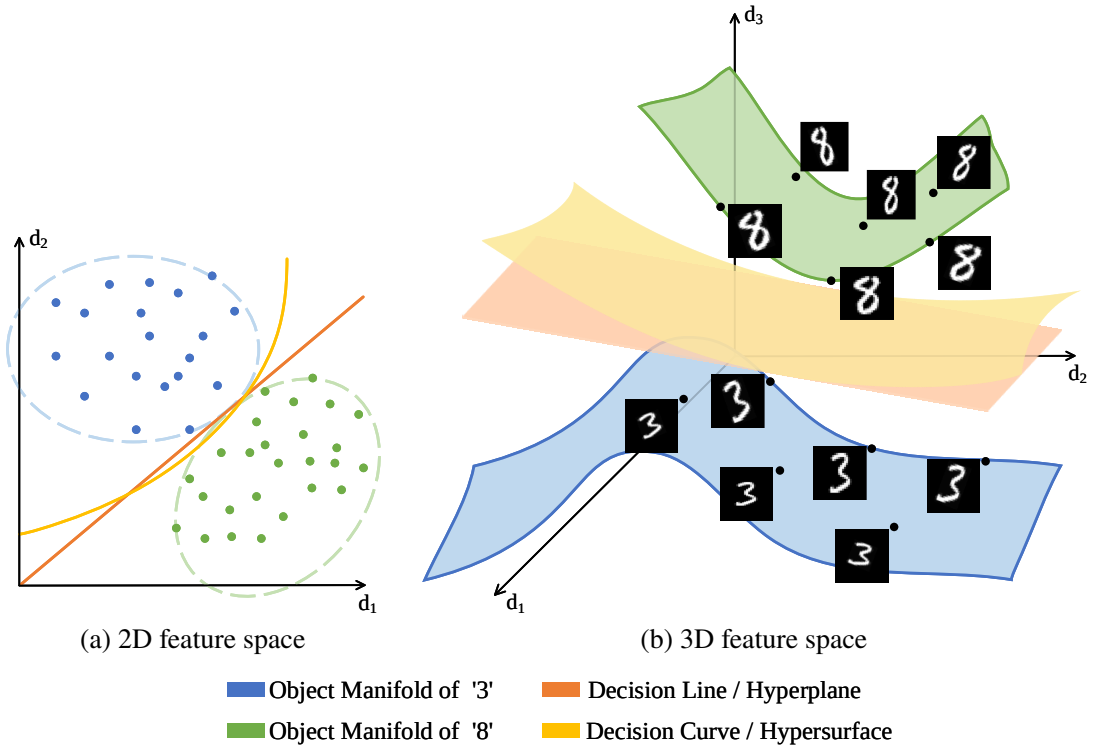
to optimize the victim model's targeted loss.

Black-Box Adversarial Attack. The attacker lacks access to the internal mechanisms of the victim model f_θ. Instead, they can rely on query-based methods by querying the victim model f_θ and using the outputs to generate AEs, or rely on surrogate model-based methods by utilizing the transferability of AEs from surrogate models f_{surr} to f_θ.

It is particularly emphasized that adversarial attacks can be categorized into *off-manifold* attacks and *on-manifold* attacks based on the space where perturbations are generated[40]. A manifold is a geometric object that represents the underlying distribution of a dataset and captures latent factors.

Object Manifold. Based on the manifold hypothesis[98-99], high-dimensional data in practical scenarios is situated on low-dimensional manifolds that are embedded within the higher-dimensional space. For instance, samples of size 28×28 pixels in the MNIST dataset can be viewed as data points residing on a low-dimensional manifold situated within a 784-dimensional feature space, which is supported by the underlying distribution of the dataset. In this dissertation, an object manifold refers to a dataset consisting of samples belonging to the same class, which is consistent with the explanation of object manifolds in the human visual hierarchy[100]. The data points determined by two features (d_1, d_2) and three features (d_1, d_2, d_3) are illustrated in Figure 2.3 (a) and Figure 2.3 (b), respectively. The dimensionality of the data manifold is determined by the number of degrees of freedom that can be adjusted when creating the dataset[101]. When the dataset can be generated by changing the rotation angle, the corresponding object manifold will be a 1-dimentional curve embedded in the feature space. Similarly, when the dataset can be generated by changing the rotation angle and scaling transformation, the corresponding object manifold will be a 2-dimentional hypersurface embedded in the feature space.

Decision Boundary. In the two-class classification task, the feature space learned by the classifier will be partitioned into two subspaces by the decision boundary, one subspace for each class. For the feature space embedded with 1-dimentional object manifolds, the decision boundary of linear classifiers and nonlinear classifiers will be a straight line and a curve, respectively, as shown in Figure 2.3 (a). For the latent space that includes at least one embedded 2-dimensional object manifold, the decision boundary of linear classifiers and nonlinear classifiers will be a hyperplane and a hypersurface, respectively, as shown in Figure 2.3 (b). Similarly, when the problem is extended to a multi-class classification task, assuming $|\mathcal{Y}|$ categories, the feature space will be partitioned into $|\mathcal{Y}|$ subspaces by the decision boundary, one subspace for each class.



(a) 2D feature space (b) 3D feature space

Figure 2.3 Interpreting object manifolds and decision boundaries in the binary classification task.

Off-Manifold Adversarial Attack. An adversarial example x^* in an off-manifold adversarial attack is created by adding imperceptible adversarial perturbation δ to the original input x in the input space $\mathcal{X}$. Formally, the objective of an untargeted off-manifold l_p adversarial attack is defined in Eq. (2-11), and the objective of an targeted off-manifold l_p adversarial attack is specified in Eq. (2-13). Most regular AEs using input-space perturbations leave the manifold orthogonally[40]. Thus, conventional adversarial attacks aiming to manipulate input features are off-manifold attacks, which seek adversarial perturbations in the input space using various methods (e.g., FGSM[9], PGD[14], and AutoAttack[32]), leaving the object manifolds corresponding to the ground-truth classes of clean examples.

On-Manifold Adversarial Attack. Novel on-manifold adversarial attacks aim at adding slight adversarial perturbation ζ to the latent representation z of the original input x in the latent space $\in \mathcal{Z}$. Formally, the goal of an untargeted on-manifold l_p adversarial attack is expressed as Eq. (2-14):

$$\max_{||\zeta||_p<\eta} \mathcal{L}(f_\theta(G_\varphi(z+\zeta)), y_{true}) \tag{2-14}$$

and the objective of an targeted on-manifold l_p adversarial attack is specified as Eq. (2-15):

$$\min_{||\zeta||_p<\eta} \mathcal{L}(f_\theta(G_\varphi(z+\zeta)), y_{target}) \tag{2-15}$$

where G denotes a generative model (e.g., GAN, AutoEncoder, StyleGAN) that can map any latent representation in $\mathcal{Z}$ to its corresponding input-space sample in $\mathcal{X}$, φ denotes the network parameters of the generative model G, η represents the perturbation budget for ζ, ζ is the adversarial perturbation in on-manifold attacks, $||\zeta||_p$ is the l_p norm-measured perturbation magnitude, and $\mathcal{L}$ signifies the loss function. On-manifold AEs are essentially generalization errors and can be generated using an approximation of the data manifold corresponding to the underlying data distribution of the given dataset. Typical attacks in this realm include on-manifold FGSM (OM-FGSM)[40,102] and on-manifold PGD (OM-PGD)[41].

2.2.4 Attack Methods

Popular adversarial attack methods include gradient-based AE generation techniques like FGSM, PGD, OM-FGSM, and OM-PGD; salience map-based techniques like JSMA; optimization-based techniques like CW; and others.

Fast Gradient Sign Method (FGSM). FGSM[9] is a off-manifold adversarial attack technique that generates AEs by leveraging the gradient of the loss function $\mathcal{L}$. The formula of untargeted FGSM in the white-box scenario is given by Eq. (2-16):

$$x^* = x + \epsilon \cdot sign(\nabla_x \mathcal{L}(f_\theta(x), y_{true})) \tag{2-16}$$

where x^* represents the adversarial example, x is the input associated with the true label y_{true}, ϵ signifies the perturbation factor, f_θ denotes the victim model characterized by parameters θ, and $\nabla_x \mathcal{L}(f_\theta(x), y_{true})$ indicates the gradient of $\mathcal{L}(f_\theta(x), y_{true})$ with respect to x. The notation sign refers to the sign function defined as Eq. (2-17):

$$sign(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & if\ x = 0 \\ -1, & if\ x < 0 \end{cases} \tag{2-17}$$

The gradient $\nabla_x \mathcal{L}(f_\theta(x), y_{true})$ points in the direction where the loss function $\mathcal{L}(f_\theta(x), y_{true})$ rises fastest. FGSM creates AEs by updating the input x in the direction where the loss function increases, that is, gradient ascent, thereby achieving the goal of deceiving the model.

Projected Gradient Descent (PGD). PGD[14] is an iterative off-manifold l_p adversarial attack method that generates AEs by applying gradient updates and projecting back into a specified l_p norm ball. The formula of untargeted PGD in the white-box scenario is given by:

$$x^{(k+1)} = Proj_{\mathcal{B}_p(x,\epsilon)} \left(x^{(k)} + \alpha \cdot sign(\nabla_x \mathcal{L}(f_\theta(x^{(k)}), y_{true})) \right) \tag{2-18}$$

where $\mathcal{B}_p$ denotes the l_p norm ball, ϵ is the perturbation budget, and ϵ_s is the step size, that is, the perturbation factor in each step during iteration. PGD can be applied with various l_p norms (e.g., l_2, l_∞, and l_1), allowing for flexible perturbation types.

On-Manifold FGSM (OM-FGSM). OM-FGSM[41] represents a variation of FGSM that ensures adversarial examples remain within the data manifold by adjusting the input in the latent space. The formula of untargeted FGSM in the white-box scenario is given by Eq. (2-19):

$$\begin{aligned} x^* &= G_\varphi(z + \eta \cdot sign(\nabla_z \mathcal{L}(f_\theta(x), y_{true}))) \\ &= G_\varphi(z + \eta \cdot sign(\nabla_z \mathcal{L}(f_\theta(G_\varphi(z)), y_{true}))) \end{aligned} \tag{2-19}$$

where G is a generative model that can map latent-space representation $z \in \mathcal{Z}$ to the input-space sample $x \in \mathcal{X}$, φ denotes the parameters of the generative model G, η is the perturbation factor for latent representation z, and $\mathcal{L}$ is the loss function.

On-Manifold PGD (OM-PGD). OM-PGD[41] is the variant of PGD that manipulates the input by adding perturbation to the corresponding latent representation in the latent space. The formula of untargeted PGD in the white-box scenario is given by Eq. (2-20):

$$\begin{aligned} x^{(k+1)} &= Proj_{\mathcal{B}_p(z,\eta)}\left(x^{(k)} + \mu \cdot sign(\nabla_z \mathcal{L}(f_\theta(x^{(k)})), y_{true}))\right) \\ &= Proj_{\mathcal{B}_p(z,\eta)}\left(G_\varphi(z^{(k)}) + \mu \cdot sign(\nabla_z \mathcal{L}(f_\theta(G_\varphi(z^{(k)})), y_{true}))\right) \end{aligned} \tag{2-20}$$

where $\mathcal{B}_p$ denotes the l_p norm ball, η is the perturbation budget for latent representation z, and μ is the step size in latent space, that is, the perturbation factor in each step during iteration.

Jacobian-based Saliency Map Attack (JSMA). JSMA[11] is a targeted off-manifold adversarial attack technique that generates AEs by focusing on the most influential features of an input x w.r.t the target label y_{target}. It relies on the Jacobian matrix J_{ij} of the model output $f_\theta(x)$ w.r.t the input x, fomulated as Eq. (2-21):

$$J_{ij} = \nabla_x f_\theta(x) = \left[\frac{\partial f_{\theta_i}(x)}{\partial x_j}\right]_{i,j} \tag{2-21}$$

where $f_{\theta_i}(x)$ denotes the model's output corresponding to class i, and x_j represents the j-th feature of the input x. Then, the important features for the target class y_{target} are identified using the saliency map. For a targeted attack towards the t-th class y_{target} of the victim model f_θ, the saliency score $S_j(x)$ of the j-th feature in x is calculated according to Eq. (2-22):

$$S_j(x) = \begin{cases} 0, & if\ \frac{\partial f_{\theta_t}(x)}{\partial x_j} < 0\ or\ \sum_{i \neq t} \frac{\partial f_i(x)}{\partial x_j} > 0 \\ \frac{\partial f_{\theta_t}(x)}{\partial x_j} \cdot \left(\sum_{i \neq t} \left|\frac{\partial f_i(x)}{\partial x_j}\right|\right), & otherwise \end{cases} \tag{2-22}$$

where $\frac{\partial f_{\theta t}(x)}{\partial x_j}$ measures how much changing the feature x_j affects the probability of the model predicting class y_{target}, and $\left(\sum_{i \neq t} \left| \frac{\partial f_i(x)}{\partial x_j} \right|\right)$ indicates its impact on other non-target classes. JSMA aims to modify the most salient features to increase the confidence score of the model on target class y_{target}, formulated as Eq. (2-23):

$$x^* = x + \Delta x \tag{2-23}$$

where Δx is the perturbation, determined using saliency maps $S_(x)$ derived from the Jacobian matrix J_{ij}. JSMA is a feature-wise targeted attack that selectively perturbs a certain number of salient features of an input, generating AEs with minimal modifications.

Carlini & **Wagner Attack (CW)** CW[13] is an optimization-based targeted off-manifold l_p adversarial attack that minimizes a specific objective function to generate AEs while maintaining low perceptibility of the perturbation. The formula of targeted CW in the white-box scenario is given by Eq. (2-24):

$$\begin{aligned} &\min_{\delta} ||\delta||_p \\ &subject\ to\ x + \delta \in [0,1]^n \\ &f_\theta(x+\delta) = y_{target} \end{aligned} \tag{2-24}$$

where $||\delta||_p$ represents the l_p norm of the perturbation, typically using l_0, l_2, or l_∞. To determine δ, an objective function o can be established, where $f_\theta(x+\delta) = y_{target}$ holds true if and only if $o(x+\delta) \leq 0$. Then the optimization problem described in Eq. (2-24) can be expressed as Eq. (2-25):

$$\begin{aligned} &\min_{\delta} ||\delta||_p \\ &subject\ to\ x + \delta \in [0,1]^n \\ &o(x+\delta) \leq 0 \end{aligned} \tag{2-25}$$

A different formulation of Eq. (2-26) is presented as follows:

$$\begin{aligned} &\min_{\delta} ||\delta||_p + c \cdot o(x+\delta) \\ &subject\ to\ x + \delta \in [0,1]^n \end{aligned} \tag{2-26}$$

where $c > 0$ is a suitably chosen constant that balances the perturbation size and the attack success rate. The CW attack frames the adversarial attack as a constrained optimization problem, making it more powerful in generating minimal, harder-to-detect AEs but significantly more computationally expensive and slower.

2.2.5 Attack Scenarios

Adversarial attacks severely threaten real-world application scenarios, including computer vision with image classifiers and cybersecurity relying on network traffic classifiers.

Image Classification. Adversarial attacks can deceive models into misclassifying images, which can have serious consequences in various multimedia applications. For example, in autonomous driving, attackers can craft adversarial images of traffic signs, causing the image recognition system of vehicles to misinterpret a stop sign as a yield sign, leading to dangerous situations on the road. Similarly, in security surveillance, AEs could enable intruders to evade detection by adversarially altering their appearance in ways that exploit vulnerabilities in the image classifiers, potentially compromising access control.

Network Traffic Classification. An attacker can generate AEs that mimic legitimate network traffic, allowing malicious activities to bypass security protections. For example, in network intrusion detection, attackers can evade detection by slightly modifying network traffic characteristics, allowing malicious activities (such as data exfiltration or system intrusion) to be performed without triggering alerts. Also, in anomaly detection, attackers can deceive the system by generating adversarial traffic that resembles normal patterns, such as launching a distributed denial of service (DDoS) attack while disguising the traffic as legitimate user behavior. This not only affects the network's availability but also leads to a failure to recognize the ongoing attack. Thus, adversarial attacks against network traffic classification systems can undermine the effectiveness of multiple detection systems.

2.3 Adversarial Robustness

2.3.1 Basic Concepts

Adversarial robustness denotes the capacity of DL models, particularly DNNs, to maintain performance under adversarial attacks. According to evaluation methods, they can be divided into two categories: empirical and certified adversarial robustness.

Empirical Adversarial Robustness. The empirical adversarial robustness of a DL model is demonstrated by observing its experimental performance against adversarial attacks during testing. It is usually evaluated by generating AEs from a clean test set and evaluating the prediction accuracy (typically recorded as robust accuracy) of the model on an adversarial test set composed of these AEs.

Certified Adversarial Robustness. The certified adversarial robustness of a model is established by offering guarantees about its robustness to adversarial perturbations through

theoretical evaluations. It is usually assessed using statistical tools to determine that perturbations within a certain range (typically recorded as a l_p norm-bounded certified radius) around an input will be ensured not to lead to misclassification.

2.3.2 Evaluation Methods

Evaluating the adversarial robustness of DNNs involves its ability to resist adversarial attacks. The methods for evaluating robustness can be divided into empirical evaluation with robust accuracy and certified evaluation with a robustness guarantee.

Robust Accuracy. The empirical assessment evaluates the adversarial robustness of the model by measuring its robust accuracy in response to various adversarial attacks, such as FGSM, PGD, CW, etc. Specifically, robust accuracy is defined as Eq. (2-27):

$$Robust\ Accuracy\ = \frac{Number\ of\ Correct\ Predictions\ on\ AEs}{Total\ Number\ of\ Tested\ AEs} = \frac{N_{(F(x^*)=y_{true})}}{N_{TotalTest}} \tag{2-27}$$

where $x^* = x+\delta$ denotes the generated adversarial examples, y_{true} is the ground-truth label of the clean example x, F is the classifer built on the trained model f_θ (defined in Definition 1).

Robustness Guarantee. The certified evaluation demonstrates the adversarial robustness of the model by providing a robustness guarantee for the predicted label of the model on each test input. The robustness guarantee provided by a classifier for an input x is formalized as a l_p norm-measured *certified radius* R of the robust region containing x, typically centered at x. The certification of robust regions bounded under different l_p norms is specifically denoted as l_p robustness guarantee, as defined in Definition 6.

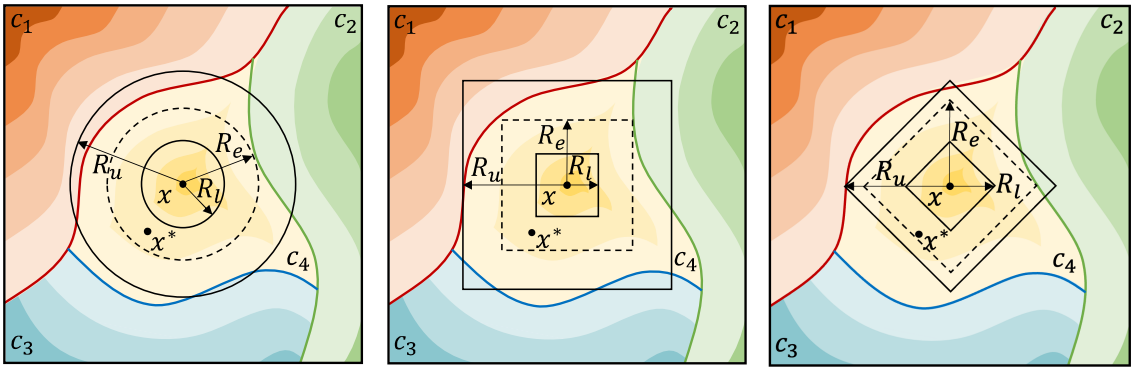
Definition 6 (l_p Robustness Guarantee)**.** *Given a base classifier F, an input x, a perturbation δ, a distance measure l_p, robustness guarantee of F on x against l_p-bounded δ is defined as:*

$$For\ all\ \delta\ such\ that\ ||\delta||_p < R,\ F(x+\delta) = F(x) \tag{2-28}$$

where R is the certified robust radius (a.k.a certified radius).

Common l_p robustness guarantees include the certified radii measured by l_2 norm and l_∞ norm shown in Figure 2.4. In this dissertation, we focus on incomplete certification, which aims to calculate the lower bound R_l of the exact robust radius R_e as tight as possible. Thus, the certified radius R provided in the dissertation refers to the lower bound R_l.

Certified Accuracy. Different from robust accuracy that measures empirical robustness against AEs, certified accuracy is the fraction of test inputs for which the model is certified



(a) l_2 radius: $||\delta||_2 < R$ (b) l_∞ radius: $||\delta||_\infty < R$ (c) l_1 radius: $||\delta||_1 < R$

Figure 2.4 l_p-bounded certified radius of the multi-class classifier on the input x.

Darker colors indicate higher confidence scores in the predicted class. Suppose the input x is predicted as c_4, the distance between the clean example x and the adversarial example x^* is $||\delta||_p$, the actual robust radius of the model on x is R_e, if $||\delta||_p < R_e$, the model is certified to give the same prediction on x and x^*. R_u and R_l are the upper and lower bounds of R_e on x. The certified radius R provided in the dissertation refers to the lower bound R_l, the radius aimed to be tightened.

to be robust within a specified perturbation radius R_{given}, given as Eq. (2-29):

$$\begin{aligned} Certified\ Accuracy(R_{given}) &= \frac{Number\ of\ Successfully\ Certified\ Inputs}{Total\ Number\ of\ Tested\ Inputs} \\ &= \frac{N_{(F(x)=y_{true})\&(R\geq R_{given})}}{N_{TotalTest}} \end{aligned} \tag{2-29}$$

where x is the test input with the ground-truth label y_{true}, R is certified radius figured out for x with robustness certification algorithm. Certified accuracy provides a worst-case guarantee of adversarial robustness against adversarial attacks, ensuring that within the certified radius, no AE can fool the model.

2.3.3 Generalization Techniques

For empirical adversarial robustness, generalization techniques represented by adversarial training and mixup training aim to improve the robust accuracy on various adversarial attacks, that is, its ability to correctly classify diverse AEs.

Adversarial Training (AT). AT serves as an essential defense mechanism against adversarial attacks by enriching the training dataset D_{train} with adversarial inputs to enhance the empirical robustness of the model f_θ. This approach seeks to reduce the worst-case empirical loss of f_θ throughout the training process by introducing adversarial modifications to each input in the clean training set D_{train}. The training objective of AT is defined as Eq. (2-30):

$$\min_\theta \mathbb{E}_{(x,y_{true})\sim D_{train}} \left[\max_{||\delta||_p<\epsilon} \mathcal{L}(f_\theta(x+\delta), y_{true}) \right] \tag{2-30}$$

where $\mathbb{E}$ is the expectation function, f_θ is the model to be defended with parameters θ, D_{train}

is the clean training dataset consisting of clean examples x with ground-truth labels y_{true}, δ is the adversarial perturbation, ϵ is the perturbation budget, $\mathcal{L}$ denotes the loss function. The most common AT relying on PGD AEs[14] generated according to Eq. (2-18), so it is called PGD-based adversarial training (PGD-AT).

Mixup Training (Mixup). Mixup training is a technique for data augmentation that generates new training instances through linear interpolation between pairs of input samples along with their associated labels. This method encourages the model to learn smoother decision boundaries, improving its robustness against adversarial attacks. The objective for mixup training is formulated as Eq. (2-31):

$$\min_{\theta} \mathbb{E}_{(x_i,y_{itrue}),(x_j,y_{jtrue})\sim D_{train}} \left[\mathcal{L}(f_\theta(\lambda x_i + (1-\lambda)x_j), \lambda y_{itrue} + (1-\lambda)y_{jtrue})\right] \quad (2\text{-}31)$$

where x_i and x_j represent input samples drawn from the training dataset D_{train}, each associated with their respective ground-truth labels y_{itrue} and y_{jtrue}. The mixing parameter λ is drawn from a Beta distribution, specifically Beta(α, α), which determines the interpolation ratio, while $\mathcal{L}$ denotes the loss function. Mixup training contributes to the development of a more varied training dataset, thereby improving the model's ability to generalize and increasing its robustness.

2.3.4 Certification Techniques

For certified adversarial robustness, on the same model and the same input sample, certification techniques aim to calculate a non-trivial certified robust radius, that is, to tighten the gap between the provided certified radius (the robustness guarantee) and the actual robust radius of the model (the true ability to resist all possible perturbations).

Randomized smoothing is the most popular technique that certifies the adversarial robustness of the model by introducing randomness into the input data before classification. During prediction, the smoothed classifier, built on the base classifier F, aims to produce the final predicted label for x by taking a majority vote from the predictions made by F on multiple noisy versions of x. During certification, by adding noise to the input x, this method computes a non-trivial certified robust radius R, providing a robustness guarantee of the model to potential perturbations within that radius.

Smoothed Classifier. The smoothed classifier is transformed from the base classifier F (defined in Definition 1) through randomized smoothing, as defined in Eq. (2-32). When queried at x, smoothed classifier F_{smooth} returns the class most likely predicted by F when x is perturbed by a large amount of noise sampling from a smoothing distribution.

Definition 7 (Smoothed Classifier)**.** *Given a base classifier F, an input x, a smoothing distribution $\mathcal{D}$ with the Probability Density Function (PDF) φ, the smoothed classifier F_{smooth} is defined as:*

$$F_{smooth}(x) = \arg\max_{c\in[C]} \mathbb{P}_{\eta\sim\mathcal{D}}(F(x+\eta)=c) = \arg\max_{c\in[C]} \int_{\eta\sim\mathcal{D}} \mathbb{I}[F(x+\eta)=c]\varphi(\eta)d\eta \tag{2-32}$$

where $\mathbb{P}$ is the probability function, $\mathbb{I}$ serves as the indicator function, yielding a value of 1 when the specified condition holds true, and $\varphi(\eta)$ denotes the probability density of the sampled noise data η.

Since the integral in the above equation cannot be solved exactly, the Monte-Carlo estimation defined in Eq. (2-33) is commonly used to approximate the exact solution.

$$F_{smooth}(x) = \arg\max_{c\in[C]} \frac{1}{n}\sum_{k=1}^{n} \mathbb{I}[F(x+\eta_k)=c] \tag{2-33}$$

where η represents the noise drawn from the smoothing distribution $\mathcal{D}$, and n indicates the quantity of noise data, and η_k denotes the k-th noise sample. The performance of the smoothed classifier F_{smooth} is determined by the base classifier F, the number of sampled noise n, and the smoothing distribution $\mathcal{D}$.

Certification with Zero-order Information Most randomized smoothing-based certification approaches only utilize the zero-order information of the smoothed classifier F_{smooth} to calculate the certified radius R, as defined in in Definition 8.

Definition 8 (Zero-order Information)**.** *Given a smoothed classifier F_{smooth}, an input x, the zero-order information refers to the basic characteristics of F_{smooth}, such as the statistical probability P_c that F_{smooth} predicts x as each class c in $[C]$, defined as:*

$$For\ all\ c\in[C],\ P_c = \mathbb{P}(F_{smooth}(x)=c) = \mathbb{P}_{\eta\sim\mathcal{D}}(F(x+\eta)=c) \tag{2-34}$$

Certification with First-order Information There are also a few approaches that attempt to introduce the high-order information of the smoothed classifier F_{smooth} to compute a tighter robustness guarantee, as defined in Definition 9.

Definition 9 (First-order Information)**.** *Given a smoothed classifier F_{smooth}, an input sample*

x, first-order information involves the derivative or gradient of F_{smooth}, defined as:

$$\begin{aligned} For\ all\ c \in [C],\ ||\nabla F^c_{smooth}(x)||_p &= ||\nabla \mathbb{P}(F_{smooth}(x) = c)||_p \\ &= ||\nabla \mathbb{P}_{\eta \sim \mathcal{D}}(F(x+\eta) = c)||_p = ||\frac{\partial(\mathbb{P}_{\eta \sim \mathcal{D}}(F(x+\eta) = c))}{\partial(x)}||_p \end{aligned} \tag{2-35}$$

where $F^c_{smooth}(x) = \mathbb{P}(F_{smooth}(x) = c)$ is the statistical probability that F_{smooth} predicts x as c, and $||\nabla F^c_{smooth}(x)||_p$ is the l_p-measured magnitude of the gradient of F^c_{smooth} at x.

2.3.5 Transfer Techniques

The main techniques for transferring adversarial robustness between DNNs include adversarial fine-tuning and adversarial distillation.

Adversarial fine-tuning (AFT) involves modifying the parameters of a pre-trained robust source model using a dataset from the target domain. This process aims to tailor the model to the new data domain while enhancing its robustness against adversarial attacks[83]. To distinguish between tunable and non-tunable layers, the set of the first k layers of the DNN is referred to as the representation learner f_{θ_r} and the layers closer to the output that are more pertinent to the specific classification task are referred to as the classification layers f_{θ_c}. Depending on whether fine-tuning is performed only on f_{θ_c} or on the entire neural network $f_\theta = f_{\theta_r} \circ f_{\theta_c}$, AFT approaches can be classified into Frozen Representation Learner-based and Entire Neural Network-based.

Frozen Representation Learner-based AFT. The parameters of classification layers f_{θ_c} are the only ones that will be modified according to the target-domain training set, as shown in Eq. (2-36):

$$\min_{\theta_c} \mathbb{E}_{(x,y_{true}) \sim D^T_{train}}[\mathcal{L}(f_{\theta_c}(f_{\theta_{r_0}}(x)), y_{true})] \tag{2-36}$$

where D^T_{train} represents the training set for the target domain, θ_{r_0} indicates the fixed parameters of the source representation learner, and θ_c signifies the parameters associated with the output layer. This method and its variants are used in AFT[83], FRFE[85], FFTL[87], RDT[84], AutoLoRa[90], and TWINS[88].

Entire Neural Network-based AFT. The parameters of all layers of the pre-trained model are updated based on the target-domain training set D^T_{train}. Furthermore, in order not to forget the representation learner of the robust source model, an additional regularization is performed on the representation learner parameter θ_r, as shown in Eq. (2-37):

$$\min_{\theta_c, \theta_r} \mathbb{E}_{(x,y_{true}) \sim D^T_{train}}[\mathcal{L}(f_{\theta_c}(f_{\theta_r}(x)), y_{true}) + \lambda||f_{\theta_r}(x) - f_{\theta_{r_0}}(x))||_2] \tag{2-37}$$

where $\lambda \in [0, 1]$ denotes the weight of the dissimilarity penalty between the representations of the target model and the source model. This method has also been further explored in FRFE[85], ADVCL[86], FFTL[87], and TWINS[88].

Adversarial distillation (AD) is an adversarial version of knowledge distillation that aims to compress the *source model* f^S (teacher) into a lightweight *target model* f^T (student) and retain the robustness of the source model on the target domain.

Knowledge Distillation. Conventional knowledge distillation (KD)[103] aims to train a smaller target model f^T by using the knowledge of a large source model f^S on clean samples, as shown in Eq. (2-38):

$$\min_{\theta} \mathbb{E}_{(x,y_{true})\sim D^T_{train}}[\mathbf{KL}(f^{S^\tau}(x)|f^{T^\tau}_\theta(x)) + \lambda \cdot \mathcal{L}(f^T_\theta(x), y_{true})] \tag{2-38}$$

where θ signifies the parameters of the entire target model, $\lambda \in [0, 1]$ denotes the weight, $\mathbf{KL}$ represents the KL divergence. $f^{T^\tau}_\theta(x)$ and $f^{S^\tau}(x)$ are the probability vectors of the target and source models on x, calculated from the logits $f^T_\theta(x)$ and $f^S(x)$ under the same temperature τ, as Eq. (2-39) specifies:

$$f^\tau_i(x) = \frac{\exp(\frac{f_i(x)}{\tau})}{\sum_{j=1}^{c} \exp(\frac{f_j(x)}{\tau})} \tag{2-39}$$

where $i, j \in [1, ..., c]$ denotes the index of the class.

Singular Sample-Guided AD. In addition to the regular classification loss, vanilla adversarial distillation (VAD)[91] minimizes the distillation loss between the distilled output of f^T for AEs $x + \delta$ and that of f^S for clean samples x, as Eq. (2-40) specifies. Note that only the soft labels $f^S(x)$ on the clean samples are used as the supervision information.

$$\min_{\theta} \mathbb{E}_{(x,y_{true})\sim D^T_{train}}[\lambda \cdot \tau^2 \mathbf{KL}(f^{S^\tau}(x)|f^{T^\tau}_\theta(x+\delta)) + (1-\lambda) \cdot \mathcal{L}(f^T_\theta(x), y_{true})] \tag{2-40}$$

where $\lambda \in [0, 1]$ denotes the weight of regular classification loss. The adversarial perturbation δ referenced in Eq. (2-40) is derived by maximizing the Cross-Entropy loss between $f^T_\theta(x + \delta)$ and y_{true}, as outlined in Eq. (2-18).

Dual Samples-Guided AD. This type of method, represented by Adaptive Adversarial Distillation (AAD)[97], aims to use soft labels $f^S(x)$ on clean samples and soft labels $f^S(x + \delta)$ on AEs simultaneously for supervision, as specified in Eq. (2-41):

$$\min_{\theta} \mathbb{E}_{(x,y_{true})\sim D^T_{train}}[\max_{||\delta||_p<\epsilon} \lambda \cdot \mathbf{KL}(f^{S^\tau}(x+\delta)|f^{T^\tau}_\theta(x+\delta)) + (1-\lambda) \cdot \mathbf{KL}(f^S(x)|f^T_\theta(x))]. \tag{2-41}$$

Chapter III Adversarial Robustness Generalization with Latent Representation Mixup

Deep neural networks excel at solving intuitive tasks that are hard to describe formally, such as classification, but are easily deceived by maliciously crafted samples, leading to misclassification. Recently, it has been observed that the attack-specific robustness of models obtained through adversarial training does not generalize well to novel or unseen attacks. While data augmentation through mixup in the input space has been shown to improve the generalization and robustness of models, there has been limited research progress on mixup in the latent space. Furthermore, almost no research on mixup has considered the robustness of models against emerging on-manifold adversarial attacks. In this chapter, we study the generalization method for adversarial robustness of DNNs and propose a robust training framework, Latent Representation Mixup (`LarepMixup`). First, we design a latent-space data augmentation approach based on dual-mode manifold interpolation, which allows for interpolating disentangled representations of source samples via convex mixing and binary mask mixing, to synthesize semantic samples. Then, we present a designed multi-label training algorithm using mixed samples and mixed labels to smooth the decision boundary, enhancing robustness against perturbations near the boundary. Finally, we show experimental results on various image classifiers and datasets, demonstrating that the proposed method delivers competitive generalization performance of adversarial robustness across various off-manifold and on-manifold adversarial examples in both white-box and black-box scenarios compared to leading mixup training techniques.

3.1 Overview

Deep neural networks (DNNs) have achieved outstanding success in deep learning (DL) tasks, including computer vision, speech recognition, and natural language processing. However, recent studies have demonstrated that DNNs are susceptible to adversarial examples (AEs), which are created using imperceptible perturbations to cause misclassification by the classifier[9,11-13]. Adversarial attacks can be categorized into off-manifold and on-manifold attacks based on the space where perturbations are generated[40]. The manifold is a geometric object representing the dataset's underlying distribution, capturing its latent factors. Off-manifold attacks, like FGSM[9], PGD[14], and AutoAttack[32], aim to manipulate input

features, while on-manifold attacks, such as OM-FGSM and OM-PGD, target representations in the latent space. Adversarial training (AT)[9] is a key proactive defense mechanism against adversarial attacks that integrates defender-generated AEs into the original training set. AT defenses are divided into off-manifold and on-manifold variants, aiming to construct respective AEs to enhance model robustness[40-41]. However, AT relies on prior knowledge of attacks, limiting its generalization against novel or unseen attacks.

Motivated by solving this challenge, we focus on generalizing model robustness to various potential adversarial attacks without training with AEs in advance. Previous efforts, such as InputMixup[42], AdaMix[47], AdvMix[43], CutMix[45], and PuzzleMixup[46], have used mixed examples and mixed labels to train neural networks for image classification, achieving enhanced robustness. Mixup has also been applied to text classification for improving generalization[104-105] and robustness[106]. Unlike AT, mixup training does not assume the defender's knowledge of the attack method. However, most research focuses on input-space mixup, synthesizing mixed examples by combining source samples in the input space. Consequently, the resulting samples may lack realistic semantics, negatively impacting the model's ability to learn meaningful representations. Additionally, such mixed samples might not effectively improve robustness against adversarial attacks, as they may not capture subtle differences between clean examples (CEs) and adversarial examples (AEs) exploited by attacks.

To synthesize mixed examples that satisfy the underlying feature structure of a given dataset, we consider mixing latent representations and mapping them to the input space. Limited work exists on latent-space mixup training besides ManifoldMixup[48] and PatchUp[49], which utilize mixed feature maps from a classifier's randomly selected hidden layer as extra training signals. These methods consider off-manifold adversarial attacks but neglect on-manifold adversarial attacks. More critically, the hidden layer of a classifier struggles to capture the full complexity of the underlying data manifold due to limited expressivity. Mixing entangled features may not correspond to real input samples and could disrupt boundary learning. Moreover, the necessary alterations to hidden layers architecture make it difficult to apply these methods to different models flexibly. To tackle these issues, we create mixed examples using interpolation on a manifold captured by an external generative model, which better represents the dataset.

We propose `LarepMixup`, a robust training framework that uses mixed examples to improve the generalization of robustness against off-manifold and on-manifold adversarial attacks. First, we extract an approximately exact data manifold coordinate system using a generative adversarial network, allowing training and test samples to be projected onto less

entangled latent representations. Second, we adopt a mixing mode like convex or binary mask mixing to synthesize on-manifold and off-manifold mixed examples by combining representations in the low-dimensional manifold. Lastly, we fine-tune all layers of the classifier using an augmented dataset containing mixed examples and original training examples with a softlabel-based cross-entropy loss function. We evaluate the performance of `LarepMixup` on various DNNs using CIFAR-10, SVHN, and ImageNet-Mixed10. Results demonstrate our method effectively boosts robustness against multiple attacks, including FGSM, PGD, AutoAttack, DeepFool, CW, OM-FGSM, OM-PGD, Fog, Snow, Elastic, and JPEG.

In this chapter, we make the following contributions:

- We design a flexible data augmentation strategy, dual-mode manifold interpolation, for synthesizing mixed examples using convex or binary mask mixing modes. We interpret the rationality of mixed examples in improving robustness in terms of their relative position to AE.
- We propose `LarepMixup`, the first mixup-based training framework addressing the threats from off/on-manifold adversarial attacks simultaneously. It boosts the model's robustness against perturbations in the input and latent spaces without relying on any prior knowledge of the adversary.
- We capture the approximate manifold of the data distribution $p(x, y|z)$ by learning the latent variable space $\mathcal{Z}$ of the StyleGAN-ADA model. The on-manifold datasets created by projecting high-dimensional inputs to disentangled low-dimensional representations are open-sourced.
- Extensive evaluations on different DNNs and datasets show that our method improves off/on-manifold robustness compared to previous mixup training methods. Notably, we are the first to focus on the performance of the mixup trained model regarding on-manifold attacks and perceptual attacks, which are recommended for evaluating the generalized robustness of DNNs on unseen regular and AE.

3.2 Problem Formulation

3.2.1 Threat Model

In this chapter, we focus on untargeted white-box adversarial attacks, which deceive deep learning models by introducing perturbations to the input data. First, attackers seek to mislead the model into making any incorrect prediction without a specific target class.

Furthermore, attackers have full access to the model's architecture and weights, making them the most challenging adversaries. Specifically, the attacks considered fall into two categories:

Off-manifold Adversarial Attacks. Off-manifold adversarial attacks aim to perturb input features directly in the high-dimensional input space, causing models to misclassify, as defined in Eq. (2-10). These attacks generate AEs by introducing imperceptible perturbations to input data, thereby shifting the inputs away from their original distribution. Common off-manifold adversarial attacks include methods such as FGSM, PGD, AutoAttack, DeepFool, and CW, which directly target input features and push them outside the data manifold.

On-manifold Adversarial Attacks. On-manifold adversarial attacks aim to perturb the latent space representations, targeting high-level feature abstractions learned by the model, as defined in Eq. (2-14). These attacks manipulate data points within the latent space, often remaining closer to the original data manifold, which makes them harder to detect. Typical on-manifold adversarial attacks include OM-FGSM and OM-PGD, which attack the model by distorting the learned representations without necessarily shifting the input outside of its original distribution.

3.2.2 Research Goal

This chapter aims to address three main problems to improve the generalization of the model's robustness to unknown types of off-manifolds and on-manifold adversarial attacks.

Problem 1. *Boosting off-manifold adversarial robustness.* The first goal is to enhance the robustness against off-manifold adversarial attacks, even those not previously seen during training. This involves improving the model' s ability to defend against input-space perturbations, ensuring that the decision boundary remains robust to AEs generated outside the data manifold.

Problem 2. *Boosting on-manifold adversarial robustness.* The second goal focuses on increasing the robustness against on-manifold adversarial attacks. Since these attacks occur in the latent space and often involve subtle manipulations, the challenge is to ensure that the robustness can generalize to such AEs, ensuring the learned feature representations are robust to latent space perturbations.

Problem 3. *Scaling across various DNN architectures.* The final goal is to develop a scalable solution that can be applied across various DNN architectures. This requires that the proposed solution is versatile and can enhance adversarial robustness in different models, whether convolutional networks, residual networks, or other model types used in modern image classification applications.

3.2.3 Key Challenge

The main challenges to addressing the problems outlined above are as follows.

Challenge 1. *Learning approximate exact data manifold.* One of the key challenges is to accurately capture the underlying data manifold. To defend against on-manifold adversarial attacks, the model must learn new samples that are augmented based on the approximately exact data manifold. This requires advanced techniques such as generative adversarial networks to map input data into a latent space with disentangled representations, which enables the synthesis of unseen semantic samples that follow the underlying distribution of the dataset or are near the decision boundary.

Challenge 2. *Generating diverse semantic mixed samples.* Semantic samples are data instances that contain meaningful and coherent information consistent with the underlying distribution of the original dataset. To improve off-manifold and on-manifold adversarial robustness, the training process must include semantic samples synthesized on demand in the latent space. A key challenge is augmenting the training set with diverse, meaningful mixed samples. One potential approach involves using convex and binary mask interpolation to generate realistic, varied samples, ensuring the augmented samples help the model against both off-manifold and on-manifold AEs.

Challenge 3. *Balancing Adversarial Robustness and Regular Predictive Performance.* Another critical challenge is achieving a balance between improving adversarial robustness and maintaining high performance on clean (non-adversarial) data. The model needs to be resilient to adversarial attacks without sacrificing its overall accuracy and generalization on regular data. This trade-off between robustness and prediction performance is a central issue in robust training, requiring careful design of loss functions and training processes.

3.3 Design of Method LarepMixup

To improve the generalization of adversarial robustness in DNNs, we propose a robust training framework called Latent Representation Mixup (`LarepMixup`). This section first introduces a dual-mode manifold interpolation data augmentation approach, which generates mixed samples either near the decision boundary or aligned with the underlying data distribution. Next, we present the framework of `LarepMixup` training, which smooths the DNN decision boundary by leveraging mixed semantic samples and corresponding mixed labels.

3.3.1 Dual-mode Manifold Interpolation Strategy

Manifold interpolation refers to the approach of constructing new representation points by combining disentangled representations on object manifolds embedded in the latent space. We design a data augmentation strategy, dual-mode manifold interpolation, to synthesize on-manifold and off-manifold mixed examples. To infer the manifold coordinate system consisting of degrees of freedom as accurately as possible, a generative adversarial network is utilized for projecting samples in the input space $x \in \mathcal{X}$ to the latent representation in the embedding space $z \in \mathcal{Z}$: $x = G(z)$ and synthesizing high-dimensional mixed samples from the low-dimensional mixed representation z_{mix}: $x_{mix} = G(z_{mix})$. We propose two kinds of mixing modes: convex combination and binary mask combination.

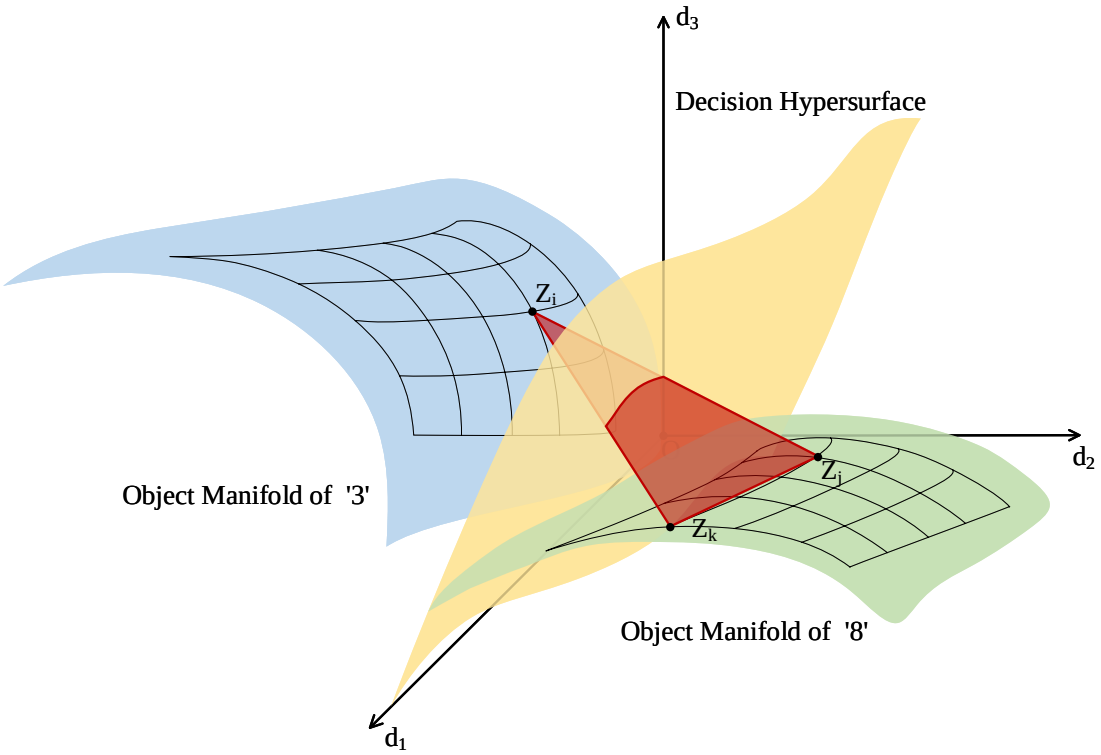


Figure 3.1 Convex combination-based manifold interpolation.

Mode 1. Convex Combination-based Manifold Interpolation. We first design a manifold interpolation mode based on the convex combination, which targets continuously creating mixed representation points along a certain direction in the latent feature space, as shown in Figure 3.1. For z_i, z_j, interpolations constructed by dual convex combination are located on the line segment between z_i and z_j. For z_i, z_j, z_k, interpolations constructed by ternary convex combination are located on the plane enclosed by the z_i, z_j, z_k.

Dual Convex Combination. For latent representations z_i, z_j corresponding to any two

samples x_i, x_j in the training set, the mixed latent representation (z_{mix}, y_{mix}) is created as Eq. (3-1):

$$\begin{aligned} z_{mix} &= \alpha z_i + (1-\alpha) z_j, \\ y_{mix} &= \alpha y_i + (1-\alpha) y_j, \end{aligned} \tag{3-1}$$

where the coefficient scalar $\alpha \in [0, 1]$ is randomly sampled from the Beta(β) distribution. We mix labels using the same coefficient, following the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated labels.

Multivariate Convex Combination. For latent representations z_1, ... , z_k corresponding to any k samples x_1, ... , x_k in the training set, the mixed latent representation (z_{mix}, y_{mix}) is created as Eq. (3-2):

$$\begin{aligned} z_{mix} &= \alpha_1 z_1 + ... + \alpha_k z_k, \\ y_{mix} &= \alpha_1 y_1 + ... + \alpha_k y_k, \end{aligned} \tag{3-2}$$

where the coefficient vector $\alpha = (\alpha_1, ..., \alpha_k) \in A \coloneqq \{\mathbb{R}^k : \alpha_i \in [0, 1], \sum_{i=1}^{k} \alpha_i = 1\}$ is sampled from the Dirichlet(γ) distribution with $dim(\gamma) = k$. $\mathbb{R}^k$ represents a vector space consisting of k real numbers. A is the k-dimensional probability simplex. α is a k-dimensional real-valued vector whose components are non-negative and sum to 1.

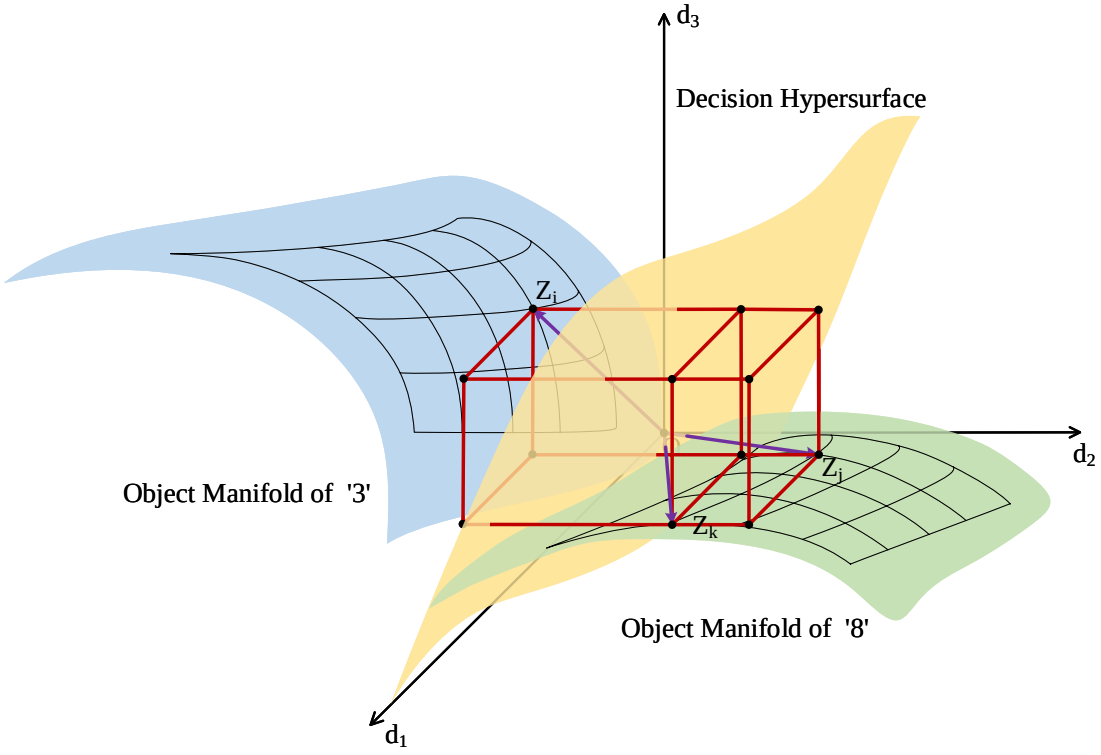


Figure 3.2 Binary mask combination-based manifold interpolation.

Mode 2. Binary Mask Combination-based Manifold Interpolation We further design binary mask combination-based manifold interpolation, which targets recombining the components of source representation vectors to synthesize the mixed samples, as shown in Figure 3.2. For z_i, z_j, interpolations constructed by dual binary mask combination are located on the vertices of the polyhedrons formed by the components of z_i, z_j. For z_i, z_j, z_k, interpolations constructed by ternary binary mask combination are located on the vertices of the polyhedrons formed by the components of z_i, z_j, z_k.

Dual Binary Mask Combination. For n-dimensional latent representations z_i, z_j corresponding to any two samples x_i, x_j in the training set, the mixed representation (z_{mix}, y_{mix}) is created as Eq. (3-3):

$$\begin{aligned} z_{mix} &= m \odot z_i + (1_B - m) \odot z_j, \\ y_{mix} &= \lambda y_i + (1-\lambda) y_j, \end{aligned} \tag{3-3}$$

where the coefficient vector $m \in B \coloneqq \{0,1\}^n$ is randomly sampled from the n-fold Bernoulli(p) distribution, the coefficient scalar $\lambda = \frac{n_{m_i=1}}{n}$ is worked out according to the proportion of the number of non-zero elements $n_{m_i=1}$ in the binary coefficient vector m to the dimension n of itself, 1_B is a binary mask filled with ones, and $\odot$ denotes the element-wise multiplication.

Multivariate Binary Mask Combination. For n-dimensional representations z_1, ... , z_k corresponding to k samples x_1, ... , x_k in the training set, the mixed representation (z_{mix}, y_{mix}) is created as Eq. (3-4):

$$\begin{aligned} z_{mix} &= m_1 \odot z_1 + ... + m_k \odot z_k, \\ y_{mix} &= \lambda_1 y_1 + ... + \lambda_k y_k, \end{aligned} \tag{3-4}$$

where the coefficient vectors $m_i \in B \coloneqq \{0,1\}^n$ and $\sum_{i=1}^{k} m_i = 1_B$. m_1 is firstly sampled from the n-fold Bernoulli(p) distribution, and then q non-zero elements in the vector $1_B - m_1$ are replaced with binary values sampled from the q-fold Bernoulli distribution, to obtain the vector m_2. Subsequent m_i is sampled in the same way.

Interpretation of Mixed Examples in Improving Off/On-manifold Robustness. The impact of mixed examples on the generalization of adversarial robustness of deep neural networks can be interpreted from two perspectives: robustness against on-manifold adversarial attacks and robustness against off-manifold adversarial attacks.

Off-Manifold Adversarial Robustness. For source representations located on different object manifolds, that is, when the source samples belong to different categories, the mixed sample formed by the manifold interpolation strategy will leave all source object manifolds

and be closer to the decision boundary than at least one of the source samples. Since conventional adversarial attacks essentially generate off-manifold AEs[40-41], the mixed samples augmented based on the proposed interpolation strategy can cover some off-manifold AEs in the consistent feasible region. Thus, by learning from the off-manifold mixed samples and corresponding mixed softlabels, the decision boundary of the classifier will be encouraged to yield lower champion confidences for points lying in regions between the object manifolds, presenting smoother. Particularly, the area covered by interpolation points is not restricted to specific attacks, so the robustness improvement can be generalized to some unseen attacks.

On-Manifold Adversarial Robustness. For source representations within the same object manifold, that is, when the source samples belong to the same category, the mixed sample formed by the manifold interpolation strategy will be close to or lie within the source object manifold, which can be regarded as the unseen samples meeting the underlying data distribution, such as the on-manifold AEs[40-41,102]. Thus, by training on on-manifold mixed examples, the classifier will be encouraged to learn an approximate manifold that is closer to the underlying manifold of the dataset. In other words, hidden layers of the model will be encouraged to learn high-level representations closer to the real latent variables that support the underlying data distribution of the given dataset. On-manifold adversarial robustness is essentially the generalization of a model to unseen samples within a manifold. Thus, fine-tuning with on-manifold mixed examples can be beneficial in boosting the on-manifold robustness.

3.3.2 Robust Training Framework

A geometric illustration of the Latent Representation Mixup (`LarepMixup`)-based robust training framework is shown in Figure 3.3. Raw samples (x_i, x_j) are projected into latent representations (z_i, z_j) at first. Then, source representations and labels are separately combined in the interpolation module using a mixup function with optional mixing modes. Finally, the target model F is fine-tuned using softlabel-based cross-entropy loss on mixed labels y_{mix} and samples x_{mix}, which are synthesized from mixed representation z_{mix}.

Stage 1. Low-dimensional Manifold Embedding. In our work, the StyleGAN2-ADA network[107] is adopted to project images into the latent space, which excels at learning disentangled variance factors to represent the latent space of complex training datasets[108]. We use 1000 iterations of gradient descent to find the disentangled latent code z, which is mapped from the randomly sampled code z_{random} through the mapping network F_{map} in the StyleGAN. The low-dimensional manifold embedding method in `LarepMixup` is summarized in the Algorithm 3.1. The loss term for optimizing the representation is de-

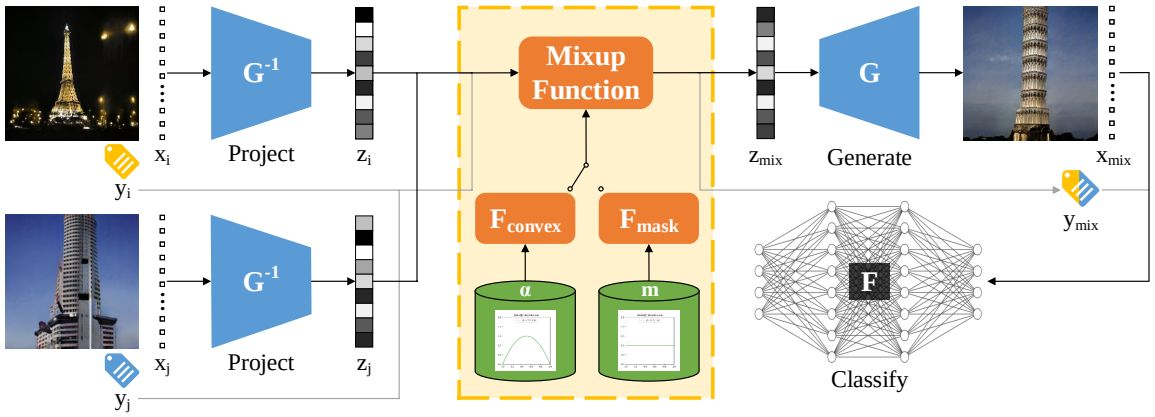


Figure 3.3 Framework of latent representation mixup (`LarepMixup`).
`LarepMixup` consists of three main stages: Low-dimensional Manifold Embedding (left), Latent Representations Mixup (middle), and Softlabel-based Multi-Label Training (right).

fined as the combination of the image quality term and regularization term $\mathcal{L}_{total}(G(z), x) = \mathcal{L}_{image} + \mathcal{L}_{noise}$, following the definition in the original work, where $\mathcal{L}_{image}$ signifies the LPIPIS distance between x and $G(z)$, and $\mathcal{L}_{noise}$ indicates the sum of squares of the noise map resolution autocorrelation coefficients. In `LarepMixup`, on-manifold dataset is denoted as $D_M = \{G(z_i), y_i))\}_{i=1}^{N}$, where N is the number of samples selected from training set D_{train}, $z_i = G^{-1}(x_i)$ is the result of projecting $x_i \in D_{train}$ into the latent space via synthesis network G, and y_i is the ground truth label corresponding to x_i.

Stage 2. Latent Representations Mixup. We implemented dual mixup and the ternary mixup interfaces, each of which supports both convex and binary mask mixing modes. To enhance off/on-manifold adversarial robustness concurrently, we mix source samples from different and identical classes.

Dual Latent Representations Mixup. The dual latent representations mixup method is presented in Algorithm 3.2. For a batch of representations with the batch size of $batchsize$, it combines with its shuffled version, enabling a mixing space of $batchsize^2$. The mixing mode is specified by the enumerated parameter e.

Ternary Latent Representations Mixup. The ternary latent representations mixup method is given in Algorithm 3.3. A batch of representations will be combined with the objects obtained by shuffling itself twice, so the mixing space can reach $batchsize^3$. Relative to dual mixup, ternary mixup spans a broader area, like the triangle in Figure 3.1. Moreover, k-source latent representations mixup expands the mixing space to a larger volume of $batchsize^k$.

Stage 3. Softlabel-based Multi-Label Training The vanilla classifier, trained on normal samples, is designed to be fine-tuned on an augmented dataset containing mixed ex-

Algorithm 3.1 Low-dimensional Manifold Embedding

Input: examples $(x, y_{true}) \in D_{train}$, optimization iteration number T, learning rate η.
Output: representations $(z, y_{true}) \in Z_{train}$, on-manifold examples $(G(z), y_{true}) \in D_M$.

pretrain G on D_{train}
for $i = 1$ to N **do**
  $t \leftarrow 0$
  sample $z_{random,i} \sim \text{Normal}(0, 1)$
  $z_{i,t} \leftarrow F_{map}(z_{random,i})$
  while $t < T$ **do**
    generate $G(z_{i,t})$
    $z_{i,t+1} \leftarrow z_{i,t} - \eta(\nabla_{z(i,t)} \mathcal{L}_{total}(G(z_{i,t}), x_i))$
    $t \leftarrow t + 1$
  end while
  $z_i \leftarrow z_{i,t+1}$
  add (z_i, y_i) to Z_{train}
  add $(G(z_i), y_i)$ to D_M
  $i \leftarrow i + 1$
end for

amples $(x_{mix}, y_{mix}) \in D_{mix}$ and original examples $(x_{ori}, y_{ori}) \in D_{train}$ to learn a robust decision boundary while avoiding overfitting to the mixed examples and knowledge loss on the original examples. For the augmented example x_{mix} with the soft mixed label y_{mix} (label vectors having two or three non-zero elements summing to 1), cross-entropy loss based on the one-hot label is inapplicable. Instead, we separately calculate the cross-entropy loss for mixed examples on multiple target labels and combine them with the same coefficient α used for the sample mixing. The objective of the softlabel-based training is formalized as Eq. (3-5).

$$\min_{\theta} \mathbb{E}_{(x, y_{true}) \sim D_{train} \cup D_{mix}} \mathcal{L}_{soft}(f_\theta(x), y_{true}). \tag{3-5}$$

`LarepMixup` is proposed as an implicit regularization technique based on data augmentation, making it broadly applicable to common DNNs without requiring any modification of the model structure. While our framework is illustrated with images, it can be extended to other input domains by replacing the StyleGAN-based manifold embedding designed for images with appropriate representation encoding algorithms suitable for other input features, such as Autoencoder for network traffic features and BERT for text features.

Algorithm 3.2 Dual Latent Representations Mixup

Input: batch of n-dim representations (Z_{ori}, Y_{ori}), mixing mode e, index shuffle function F_{shu}, mixing coefficient transform function F_{tra}.

Output: batch of mixed examples (X_{mix}, Y_{mix}).

$(Z_{shu}, Y_{shu}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
if $e = ConvexMixup$ **then**
  sample $\alpha \sim$ Beta(β)
  $Z_{mix} \leftarrow \alpha Z_{ori} + (1-\alpha) Y_{ori}$
  $Y_{mix} \leftarrow \alpha Y_{ori} + (1-\alpha) Y_{shu}$
end if
if $e = MaskMixup$ **then**
  sample $p \sim$ Uniform$(0, 1)$
  sample $m \sim n$-fold Bernoulli(p)
  $Z_{mix} \leftarrow m \odot Z_{ori} + (1_B - m) \odot Z_{shu}$
  $\lambda \leftarrow F_{tra}(m)$
  $Y_{mix} \leftarrow \lambda Y_{ori} + (1-\lambda) Y_{shu}$
end if
$X_{mix} \leftarrow G(Z_{mix})$
output (X_{mix}, Y_{mix})

3.4 Experimental Setup

3.4.1 Testbed

We developed the project using PyTorch 1.8.1[109]. Experiments were conducted on an NVIDIA GV102 GPU with CUDA V11.1.74. Off-manifold adversarial attacks were implemented with the Adversarial Robustness Toolbox[110], while on-manifold adversarial attacks were achieved by aggregating StyleGAN and advertorch[111]. Source code and on-manifold dataset are available: https://github.com/LarepMixup.

3.4.2 Model Architectures

To analyze the universality of `LarepMixup` on different classifier architectures, we used a series of base models implemented in the Torchvision library[109], including convolutional block-based DNNs (such as AlexNet[112] and VGG-19[113]), residual block-based DNNs (such as ResNet-18/34/50[114] and DenseNet-169[115]), and inception block-based DNN (such as GoogLeNet[116]). To maintain fairness when comparing various DNNs on the

Algorithm 3.3 Ternary Latent Representations Mixup

Input: batch of n-dim representations (Z_{ori}, Y_{ori}), mixing mode e, index shuffle function F_{shu}, nonzero element counting function $F_{nonzero}$, function $F_{rep}(a, b)$ to replace nonzero elements in a with b.

Output: batch of mixed examples (X_{mix}, Y_{mix}).

$(Z_{shu1}, Y_{shu1}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
$(Z_{shu2}, Y_{shu2}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
if $e = ConvexMixup$ **then**
  sample $\alpha = (\alpha_1, \alpha_2, \alpha_3) \sim$ Dirichlet(γ)
  $Z_{mix} \leftarrow \alpha_1 Z_{ori} + \alpha_2 Z_{shu1} + \alpha_3 Z_{shu2}$
  $Y_{mix} \leftarrow \alpha_1 Y_{ori} + \alpha_2, Y_{shu1} + \alpha_3 Y_{shu2}$
end if
if $e = MaskMixup$ **then**
  $n_1 \leftarrow n$
  sample $p_1 \sim$ Uniform$(0, 1)$
  sample $m_1 \sim n_1$-fold Bernoulli(p_1)
  $num_{nonzero} \leftarrow F_{nonzero}(1_B - m_1)$
  $n_2 \leftarrow num_{nonzero}$
  sample $p_2 \sim$ Uniform$(0, 1)$
  sample $temp \sim n_2$-fold Bernoulli(p_2)
  $m_2 \leftarrow F_{rep}(1_B - m_1, temp)$
  $m_3 \leftarrow 1_B - m_1 - m_2$
  $z_{mix} \leftarrow m_1 \odot Z_{ori} + m_2 \odot Z_{shu1} + m_3 \odot Z_{shu2}$
  $\lambda_1, \lambda_2, \lambda_3 \leftarrow F_{tra}(m_1, m_2, m_3)$
  $y_{mix} \leftarrow \lambda_1 Y_{ori} + \lambda_2 Y_{shu1} + \lambda_3 Y_{shu2}$
end if
$X_{mix} \leftarrow G(Z_{mix})$
output (X_{mix}, Y_{mix})

same dataset or different datasets on a single model, we did not modify any base architecture and used uniform parameters during dataset preprocessing. Additionally, we conducted experiments on the PreActResNet-18/34/50[117] and WideResNet-28-10[118] adopted in the compared mixup training schemes.

3.4.3 Datasets

In this chapter, three color-channel datasets are used for experimental evaluation.

- The CIFAR-10 dataset[119] contains $3\times32\times32$ samples from 10 categories: $\{airplane$, $automobile$, $bird$, cat, $deer$, dog, $frog$, $horse$, $ship$, $truck\}$. Each category contains $50,000$ training samples and $10,000$ test samples.
- The SVHN dataset[120] contains $3\times32\times32$ samples from 10 categories: $\{1$, 2, 3, 4, 5, 6, 7, 8, 9, $0\}$, including a total of $73,257$ training samples and $26,032$ testing samples.
- ImageNet-Mixed10 dataset[41] contains $3\times256\times256$ samples from 10 categories: $\{dog$, $bird$, $insect$, $monkey$, $feline$, $truck$, $fruit$, $horse$, $fungus$, $boat\}$, selected from the ImageNet dataset[112], including $77,237$ training samples and $3,000$ testing samples.

3.4.4 Attack Configuration

Parameters in Adversarial Attacks. We use two categories of adversarial attack methods: off-manifold and on-manifold. We normalize input sample ranges across datasets to the $[-1,1]$ interval, which is passed as a clip parameter to attack interfaces. Except for the ablation study of perturbation strength, the attack parameters of other experiments are given in Table 3.1. ϵ and η represent norm bounds for off-manifold and on-manifold perturbations, respectively, in p-norm bounded attacks. ϵ_s and η_s indicate single-step upper bounds for ϵ and η. n_i refers to the maximum iteration rounds. We use the default confidence of 0. For the CW attack based on optimization, there is no configuration parameter about the perturbation threshold, but the confidence parameter k needs to be configured. All CW attacks used in this work adopt the default confidence of 0. These attack parameters are chosen because they relatively balance the perceptibility of adversarial perturbations with the attack success rate on the vanilla model.

Parameters in Perceptual Attacks. The perceptual attack methods we use can be divided into three categories: weather conditions (Fog, Snow), elastic transformation, and digital compression (JPEG), as shown in Table 3.2. The reason for using perceptual attacks is to evaluate the generalization ability of the robust model to unseen attacks, which is adopted in[41,121-123]. Our parameter configuration mainly refers to the perceptual attack parameters used in the DMAT[41]. All these attacks use 200 Gradient Descent iterations.

Table 3.1 Parameters in adversarial attacks

Dataset	Perturbation Space	Name	Norm	Configuration			
				$\epsilon(\eta)$	$\epsilon_s(\eta_s)$	n_i	k
CIFAR-10	Off-Manifold	FGSM	l_∞	0.05	-	-	-
		PGD	l_∞	0.05	0.10	100	-
		AutoAttack	l_∞	0.05	0.10	-	-
		DeepFool	l_2	0.02	-	100	-
		CW	l_2	-	-	-	0
	On-Manifold	OM-FGSM	l_∞	0.05	-	-	-
		OM-PGD	l_∞	0.05	0.01	40	-
SVHN	Off-Manifold	FGSM	l_∞	0.10	0.10	-	-
		PGD	l_∞	0.10	0.10	100	-
		AutoAttack	l_∞	0.10	0.10	-	-
		DeepFool	l_2	0.10	-	100	-
		CW	l_2	-	-	-	0
	On-Manifold	OM-FGSM	l_∞	0.10	-	-	-
		OM-PGD	l_∞	0.10	0.01	40	-
ImageNet-Mixed10	Off-Manifold	FGSM	l_∞	0.02	0.10	-	-
		PGD	l_∞	0.02	0.10	100	-
		AutoAttack	l_∞	0.02	0.10	-	-
		DeepFool	l_2	0.02	-	100	-
		CW	l_2	-	-	-	0
	On-Manifold	OM-FGSM	l_∞	0.02	-	-	-
		OM-PGD	l_∞	0.02	0.01	40	-

3.4.5 Defense Configuration

Parameters in Standard Training. The models we employ primarily take two forms: one originates from the Torchvision library without any structural modifications, which includes AlexNet, ResNet-18/34/50, DenseNet-169, VGG-19, and GoogleNet; the other is independently implemented, including PreActResNet-18/34/50. Furthermore, preprocessing across all datasets is consistent. The input range of samples for all datasets is normalized to the $[-1, 1]$ interval using a normalization function with mean and standard deviation of 0.5. During standard training, the initial learning rate, 0.01, is reduced to one-tenth of the original every 10 epochs.

Parameters in Mixup Training. We evaluate mixup training methods, including input-space mixup (InputMixup, CutMix, PuzzleMixup) that directly mix input samples, and latent-space mixup (ManifoldMixup, PatchUp, `LarepMixup`) that mix latent samples. The maximum number of epochs is set to 40 for all mixup training methods. The initial learning rate is 0.01, reduced by a factor of 10 every 10 epochs. All augmented datasets used for

Table 3.2 Parameters in perceptual attacks

Dataset	Perturbation Space	Name	Norm	Configuration		
				ϵ (in pixel)	ϵ_s	n_i
CIFAR-10	Off-Manifold	Fog	l_∞	128	0.002	200
		Snow		0.0625	0.002	200
		Elastic		0.5	0.035	200
		JPEG		32	2.250	200
SVHN	Off-Manifold	Fog	l_∞	128	0.002	200
		Snow		0.0625	0.002	200
		Elastic		0.5	0.035	200
		JPEG		32	2.250	200

mixup training consisted of mixed examples and an equal number of clean training examples. The number of mixed examples is consistent with the length of the training set. A dual-convex mixing mode is adopted in all experiments except for experiments evaluating the effect of mixing modes on the robustness performance. For InputMixup, ManifoldMixup, PuzzleMixup, and CutMix, the sampling distribution is set to Beta(1.0,1.0). For PatchUp, the sampling distribution is set to the Bernoulli distribution. The parameters of `LarepMixup` training are shown in Table 3.3. Since we choose to use a 512-dimensional vector to describe the latent representation, 512 Bernoulli trials are performed to determine whether the mask value of each dimension of the latent representation is 1 or 0 for binary mask mixing mode. The probability that the mask value of each dimension takes a value of 1 in each Bernoulli trial follows a uniform distribution. It is worth mentioning that the Bernoulli3$(512, p)$ distribution used for ternary mask mixing refers to conducting Bernoulli$(512, p)$ sampling of three source samples successively.

Table 3.3 Parameters in `LarepMixup` training

Dataset	Defense					
	Epochs	BatchSize	Lr	MixedNum	Mixup Mode	Sampling Distribution
CIFAR-10	40	256	0.01	50,000	Dual Convex	Beta(1.0, 1.0)
					Dual Mask	Bernoulli$(512, p)$, $p\sim$U(0,1)
SVHN	40	256	0.01	73,257	Dual Convex	Beta(1.0, 1.0)
					Dual Mask	Bernoulli$(512, p)$, $p\sim$U(0,1)
ImageNet-Mixed10	40	32	0.01	77,237	Dual Convex	Beta(1.0, 1.0)
					Dual Mask	Bernoulli$(512, p)$, $p\sim$U(0,1)
					Ternary Convex	Dirichlet(1.0, 1.0, 1.0)
					Ternary Mask	Bernoulli3$(512, p)$, $p\sim$U(0,1)

Parameters in Adversarial Training. In adversarial training (AT) that is also based on data augmentation, we ensure fairness in performance comparison by setting the same number of epochs, batch size, learning rate, and augmented examples as mixup training. The augmented examples used in AT consist of AEs and an equal number of clean training examples. We generate on-manifold AEs and off-manifold AEs, each with half the number of original training samples, to ensure that the total number of augmented examples is consistent, whether in mixup training or AT. The properties of the evaluated robust training methods are compared in Table 3.4.

Table 3.4 Comparison of defense methods attributes

Method	Faced Attack Surfaces	Knowledge of Attacker	Augmentation Space
PGD-AT[14]	Off-manifold	Known	Input Space
PGD-DMAT[41]	Off-manifold & On-manifold	Known	Input & Latent Space
InputMixup[42]	Off-manifold	Unknown	Input Space
CutMix[45]	Off-manifold	Unknown	Input Space
PuzzleMixup[46]	Off-manifold	Unknown	Input Space
ManifoldMixup[48]	Off-manifold	Unknown	Latent Space
PatchUp[49]	Off-manifold	Unknown	Latent Space
`LarepMixup`(Ours)	Off-manifold & On-manifold	Unknown	Latent Space

3.4.6 Evaluation Metrics

This chapter primarily uses two metrics, clean accuracy and robust accuracy, to empirically evaluate the regular predictive performance and adversarial robustness of the model.

Clean Accuracy. Clean accuracy reflects the regular predictive performance of a model when evaluated on clean examples (CEs) from the test set. It measures the proportion of correctly classified examples from a clean test set, as defined in Eq. (3-6):

$$Clean\ Accuracy = \frac{Number\ of\ Correct\ Predictions\ on\ CEs}{Total\ Number\ of\ Tested\ CEs} = \frac{N_{(F(x)=y_{true})}}{N_{TotalTest}}. \quad (3\text{-}6)$$

Robust Accuracy. Robust accuracy evaluates the robustness performance of the model against adversarial attacks. It measures the proportion of correctly classified instances from a set of adversarial examples (AEs), as defined in Eq. (2-27):

3.5 Horizontal Experimental Results and Analysis

In this section, we compare `LarepMixup` with SOTA defense methods against adversarial attacks. This includes mixup training under the same defensive capability assumptions as ours and adversarial training (AT) (Section 3.5.1), which operates under stronger defensive capability assumptions (Section 3.5.2).

3.5.1 Comparison with SOTA Mixup Training Methods

We tested the performance of our proposed method in improving general robustness compared to SOTA mixup training methods. For each attack, we ran each robust training method six times with the same settings and averaged the results. The initial learning rate, epochs, and batch size were 0.01, 256, and 40. All mixup training methods based on beta distribution sampling had parameters (1.0, 1.0) of the beta distribution. Experiments were conducted on PreActResNet models, with CIFAR-10 results in Table 3.5 and SVHN results in Table 3.6. We bolded the best prediction accuracy and underlined the runner-up for each column.

We evaluate off-manifold adversarial robustness using five off-manifold adversarial attacks: FGSM, PGD, AutoAttack, DeepFool, and CW. FGSM and DeepFool are single-step attacks, PGD is multi-step, and AutoAttack is an integrated version of multiple attacks with PGD as the primary one. For CIFAR-10, budgets of DeepFool, FGSM, PGD, OM-FGSM, and OM-PGD are 0.02, 0.05, 0.05, 0.05, 0.05, respectively. For SVHN, all budgets are set to 0.1. Refer to Table 3.1 for more details on attack parameters. As seen in Table 3.5, our method achieves excellent defense results in most cases on the CIFAR-10 dataset. Convex-`LarepMixup` and ManifoldMixup use the linear interpolation strategy, while Mask-`LarepMixup` and PatchUp employ a binary mask mixing strategy. On the SVHN dataset, we observe from Table 3.6 that `LarepMixup` and ManifoldMixup have their own areas of expertise. ManifoldMixup is competitive in PGD-related attacks, while `LarepMixup` has a stable advantage in FGSM, DeepFool, CW, and OM-FGSM.

3.5.2 Comparison with SOTA Adversarial Training Methods

To further compare the difference between the improved robustness based on the proposed method and the improved robustness based on adversarial training (AT), we compared `LarepMixup` with two powerful AT methods, PGD-AT[14] and PGD-DMAT[41], on the CIFAR-10 and SVHN datasets. In PGD-AT, the defender generates the same number of white-box PGD examples as the original training samples for training. In PGD-DMAT, the

Table 3.5 Accuracy of CIFAR-10 classification models on off/on-manifold AEs

$KnoA$ denotes that prior KNOwledge about adversarial Attackers is required during training.
$ModN$ denotes that Network architecture must be MODified during training.

PreActResNet-18

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**87.37±0.00**	32.07±0.00	28.93±0.00	7.59±0.00	10.36±0.00	2.60±0.00	51.02±0.00	21.68±0.00		
InputMixup [42]	84.48±1.45	63.58±3.36	68.12±3.46	56.63±10.20	37.97±2.58	41.11±2.10	58.53±0.43	44.11±1.34	✗	✗
CutMix [45]	82.14±3.00	65.51±1.03	69.67±1.34	64.41±3.55	36.79±2.60	39.74±3.10	57.59±0.31	43.50±1.71	✗	✗
PuzzleMixup [46]	83.11±1.64	65.73±2.46	70.35±2.60	64.03±6.06	38.86±1.53	41.83±1.74	57.80±0.77	43.68±2.19	✗	✗
ManifoldMixup [48]	71.10±4.17	49.26±1.34	52.49±1.91	44.08±1.60	25.33±2.76	27.19±2.53	50.16±1.66	38.64±0.80	✗	✓
PatchUp [49]	72.02±4.10	51.35±2.13	55.91±2.29	44.61±2.56	28.81±3.35	30.94±3.13	52.22±2.32	41.33±1.24	✗	✓
Ours-Convex	84.02±1.77	**68.86±2.88**	**72.65±3.59**	**66.98±5.93**	39.03±2.16	42.03±2.31	**60.02±0.91**	**46.72±1.52**	✗	✗
Ours-Mask	84.60±1.27	66.56±1.50	71.22±1.93	63.69±4.61	**39.27±2.97**	**42.54±2.74**	58.36±0.60	44.80±0.73	✗	✗

PreActResNet-34

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**83.57±0.00**	31.37±0.00	25.71±0.00	5.27±0.00	12.27±0.00	1.89±0.00	49.23±0.00	17.05±0.00		
InputMixup [42]	68.42±7.38	62.19±4.22	63.84±4.98	63.79±4.99	26.36±4.07	29.77±4.16	54.68±3.84	47.18±2.29	✗	✗
CutMix [45]	71.21±6.16	62.45±2.71	64.61±3.50	64.30±3.16	28.88±2.07	32.12±2.38	55.65±2.56	46.40±0.99	✗	✗
PuzzleMixup [46]	67.06±7.62	60.89±4.99	62.55±5.76	62.66±5.84	25.89±2.98	28.96±3.37	54.04±3.87	46.31±2.05	✗	✗
ManifoldMixup [48]	73.69±1.78	49.65±1.94	52.24±2.08	43.75±2.04	31.09±3.13	32.81±3.18	52.99±0.24	39.47±1.34	✗	✓
PatchUp [49]	72.71±2.96	49.53±1.44	52.76±2.80	42.31±1.80	32.35±3.66	34.10±3.45	53.03±2.37	39.38±1.63	✗	✓
Ours-Convex	78.44±1.60	**67.81±1.04**	**71.12±1.08**	**70.60±1.30**	**33.98±1.04**	**37.42±1.03**	**58.96±0.67**	**47.99±1.16**	✗	✗
Ours-Mask	77.13±3.17	66.16±1.58	68.90±1.62	68.40±2.16	32.95±2.26	36.38±2.23	58.31±0.96	47.30±1.06	✗	✗

PreActResNet-50

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**84.74±0.00**	35.27±0.00	26.89±0.00	5.43±0.00	12.03±0.00	1.13±0.00	50.26±0.00	19.13±0.00		
InputMixup [42]	74.93±2.22	65.28±1.42	67.42±1.67	67.57±2.04	31.35±2.59	34.39±2.40	58.92±1.28	49.81±1.23	✗	✗
CutMix [45]	75.27±3.01	64.68±2.18	66.98±2.46	66.97±2.30	30.70±2.77	33.84±2.78	58.40±1.00	48.61±1.24	✗	✗
PuzzleMixup [46]	67.35±5.41	59.96±2.92	61.18±3.36	61.41±2.79	26.84±2.06	29.58±2.03	55.72±1.91	48.46±1.39	✗	✗
ManifoldMixup [48]	76.17±3.03	54.54±2.59	56.76±2.88	47.64±6.91	29.97±4.24	32.81±4.04	55.26±0.93	40.93±2.34	✗	✓
PatchUp [49]	74.26±2.86	54.32±1.67	56.16±1.73	46.87±6.63	28.96±2.96	31.40±2.82	55.63±1.17	42.30±2.87	✗	✓
Ours-Convex	76.54±2.08	**66.99±1.87**	**69.73±1.92**	**69.69±1.49**	31.75±2.06	35.06±1.91	**59.73±1.00**	**50.04±0.52**	✗	✗
Ours-Mask	76.10±4.38	66.04±1.89	68.29±2.20	67.76±2.28	**34.21±3.45**	**37.32±3.67**	59.22±1.04	49.82±1.10	✗	✗

defender generates PGD and OM-PGD AEs, each with half the number of original training samples for training. Attack budgets of AEs used for AT are all set to 0.05. We used three kinds of the PreActeResNet models.

As shown in Figure 3.4, `LarepMixup` achieves higher robustness improvements than AT for most adversarial attacks on CIFAR-10, whether using convex mixing or binary mask mixing. Notably, `LarepMixup` also maintains higher accuracy on clean samples, which is very close to the clean accuracy. Figure 3.5 shows that `LarepMixup` still maintains good clean accuracy in SVHN, especially compared to the DMAT work. For off-manifold attacks, robustness from PGD-AT is greater, but when faced with on-manifold adversarial attacks, `LarepMixup` regains its advantage. Overall, `LarepMixup` achieves comparable robust performance to AT without actively generating AEs for training.

Table 3.6 Accuracy of SVHN classification models on off/on-manifold AEs

$KnoA$ denotes that prior KNOwledge about adversarial Attackers is required during training.
$ModN$ denotes that Network architecture must be MODified during training.

PreActResNet-18

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**95.97±0.00**	57.29±0.00	34.57±0.00	29.21±0.00	22.51±0.00	21.54±0.00	41.04±0.00	6.78±0.00		
InputMixup [42]	94.39±0.79	68.77±2.03	58.81±2.34	51.25±2.22	**60.50±3.33**	64.42±2.16	44.58±0.86	18.48±1.04	✗	✗
CutMix [45]	94.19±1.07	68.78±2.01	59.52±3.28	52.50±3.64	57.45±3.26	63.62±1.52	44.31±1.02	17.87±0.91	✗	✗
PuzzleMixup [46]	94.54±0.66	67.55±1.79	58.79±3.34	51.65±3.48	55.87±2.22	63.42±1.51	43.63±0.62	16.00±1.15	✗	✗
ManifoldMixup [48]	89.15±4.22	67.21±1.85	60.32±1.94	53.60±3.21	52.95±3.15	60.57±1.97	43.32±1.52	**22.19±2.01**	✗	✓
PatchUp [49]	89.87±1.78	66.44±0.78	58.96±1.90	52.36±2.82	54.68±2.69	61.54±1.68	43.40±0.91	21.51±1.05	✗	✓
Ours-Convex	94.38±0.61	**70.62±1.35**	**63.35±0.67**	**56.66±1.22**	58.14±0.75	**64.45±0.54**	45.24±0.44	19.59±0.57	✗	✗
Ours-Mask	94.42±0.93	70.22±1.30	60.02±1.72	53.34±2.02	57.98±2.44	64.36±1.08	**45.26±0.54**	19.90±0.71	✗	✗

PreActResNet-34

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**95.75±0.00**	57.11±0.00	35.57±0.00	29.80±0.00	19.94±0.00	25.62±0.00	36.62±0.00	5.01±0.00		
InputMixup [42]	93.41±1.85	66.14±0.85	60.42±6.52	52.82±7.44	49.76±3.32	62.47±1.10	39.97±0.97	17.07±0.85	✗	✗
CutMix [45]	93.36±2.74	65.71±0.56	60.09±7.25	53.39±8.66	49.26±2.00	61.83±1.35	39.81±1.09	16.25±0.88	✗	✗
PuzzleMixup [46]	92.53±4.79	65.12±0.82	61.06±7.05	54.17±8.54	48.65±3.22	61.63±2.37	39.24±1.89	15.89±2.15	✗	✗
ManifoldMixup [48]	81.27±2.68	61.63±2.07	**63.61±3.10**	**59.19±1.94**	44.88±4.40	56.29±3.92	36.11±1.07	21.68±1.26	✗	✓
PatchUp [49]	68.39±9.86	51.94±4.91	55.01±6.31	52.17±5.91	36.07±2.41	47.47±5.47	31.81±2.20	**22.19±2.72**	✗	✓
Ours-Convex	94.94±0.31	**68.37±0.76**	61.75±3.65	53.55±4.05	**52.21±1.67**	**64.61±1.27**	**41.13±0.41**	16.88±0.38	✗	✗
Ours-Mask	93.63±1.13	67.69±0.52	63.21±5.39	55.74±5.69	52.10±2.75	64.27±1.30	40.70±0.60	17.01±0.47	✗	✗

PreActResNet-50

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	$KnoA$	$ModN$
Vanilla	**95.76±0.00**	60.02±0.00	35.61±0.00	29.94±0.00	23.72±0.00	27.09±0.00	40.01±0.00	6.73±0.00		
InputMixup [42]	94.45±0.29	66.80±1.09	58.42±2.83	50.07±3.26	49.19±1.79	60.38±1.30	42.04±0.37	17.17±0.58	✗	✗
CutMix [45]	94.48±0.31	65.83±1.79	57.87±1.85	49.99±1.21	45.26±1.06	58.98±0.44	41.73±0.46	15.37±0.75	✗	✗
PuzzleMixup [46]	94.13±1.63	66.82±0.82	61.97±4.94	54.42±5.66	47.50±1.59	61.34±0.72	41.46±0.96	15.63±0.98	✗	✗
ManifoldMixup [48]	77.84±9.38	61.22±5.05	63.16±4.59	**60.21±3.74**	44.15±7.24	54.42±6.05	36.84±3.29	**22.97±1.79**	✗	✓
PatchUp [49]	78.36±1.65	58.62±3.41	60.46±3.63	57.70±3.59	43.14±3.84	54.24±2.05	36.25±1.74	21.95±1.75	✗	✓
Ours-Convex	93.53±1.96	**69.02±0.70**	**66.33±5.88**	59.78±7.54	49.39±2.03	61.59±1.07	**42.27±0.87**	17.57±1.23	✗	✗
Ours-Mask	94.16±1.73	68.15±0.46	60.39±2.93	52.62±3.15	**53.83±2.38**	**63.21±1.32**	41.45±3.86	19.34±3.09	✗	✗

3.6 Vertical Experimental Results and Analysis

We first present our constructed on-manifold dataset and the perception of mixed samples in Section 3.6.1. Then, we analyze the effect of varying perturbation budgets on robustness improvements in Section 3.6.2. Next, we demonstrate the adaptability of the proposed method on higher-dimensional datasets based on ImageNet and analyze the impact of different mixing modes on improving model robustness in Section 3.6.3. Next, we evaluate the robustness generalized to perceptual attacks in Section 3.6.4. Finally, we presented the time of the proposed method in Section 3.6.5.

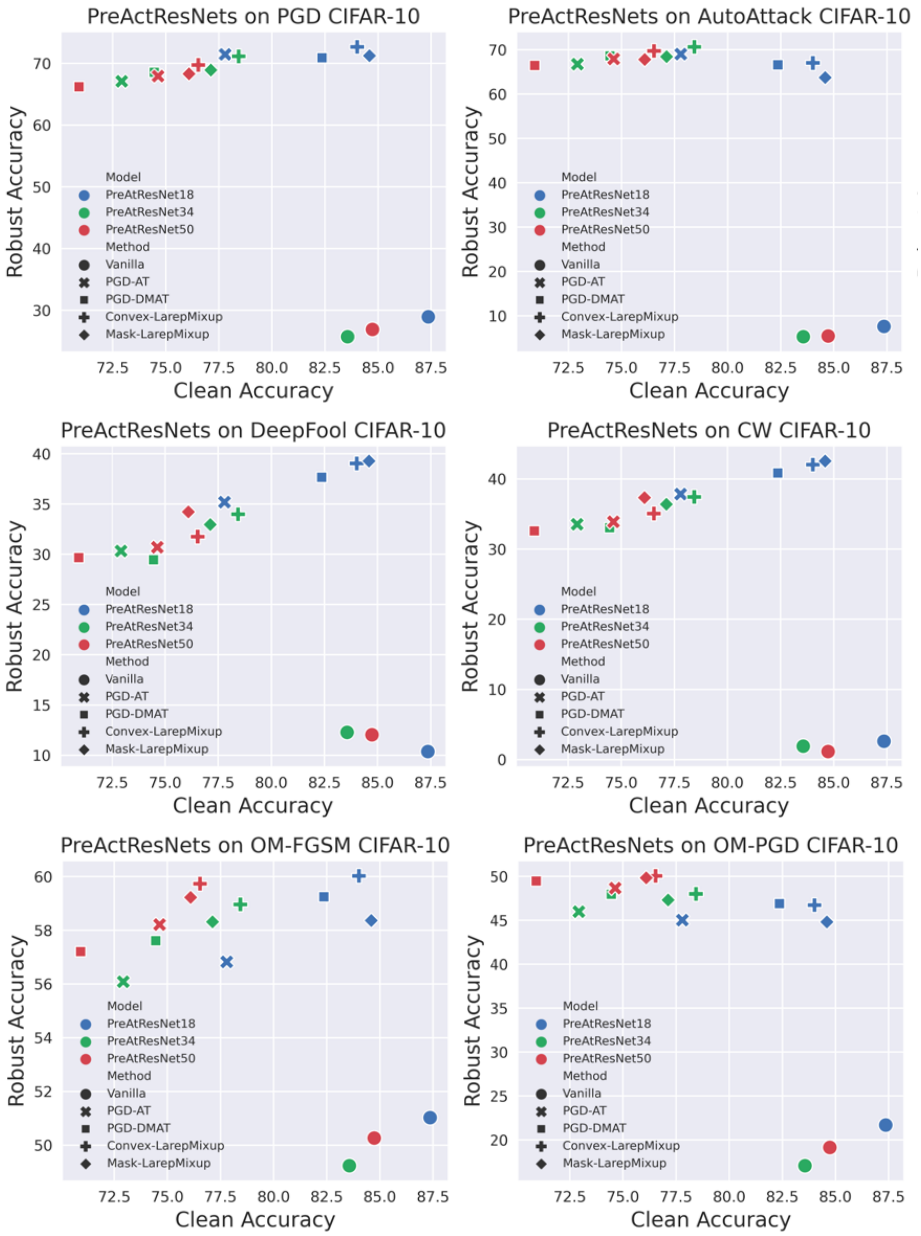


Figure 3.4 Accuracy of robust trained PreActResNets on various attacks (CIFAR-10).

3.6.1 Perception Analysis

Realistic perception is an essential requirement for augmented examples generated by mixup methods because unnatural semantic information in mixed examples can mislead the classifier and weaken the generalization of the model[45-46]. Experimental results show that the manifold learned by `LarepMixup` is almost identical to the underlying data manifold of the CIFAR-10 dataset, and the mixed examples synthesized by `LarepMixup` have meaningful semantics.

On-manifold Dataset. We train a StyleGAN2-ADA network with a 512-dimensional

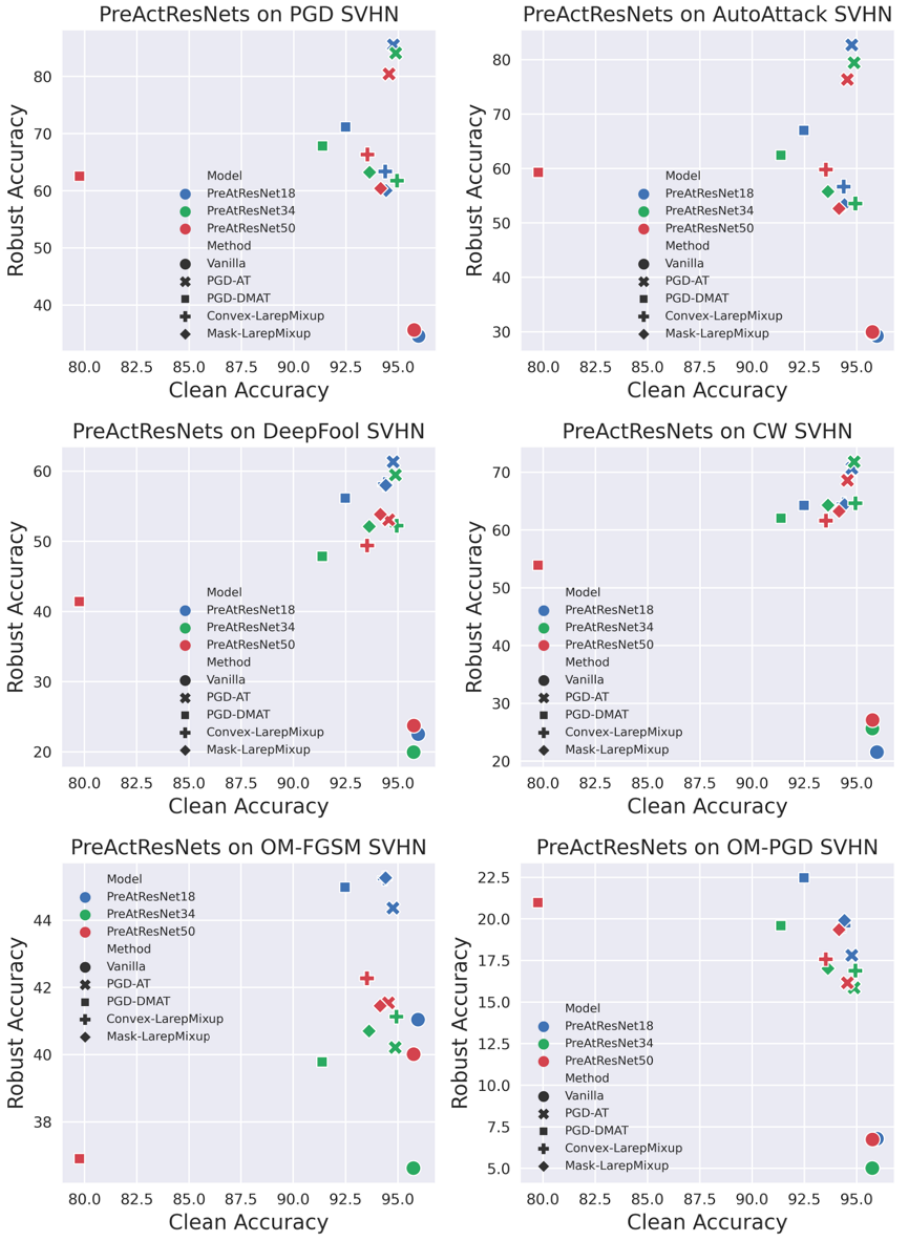
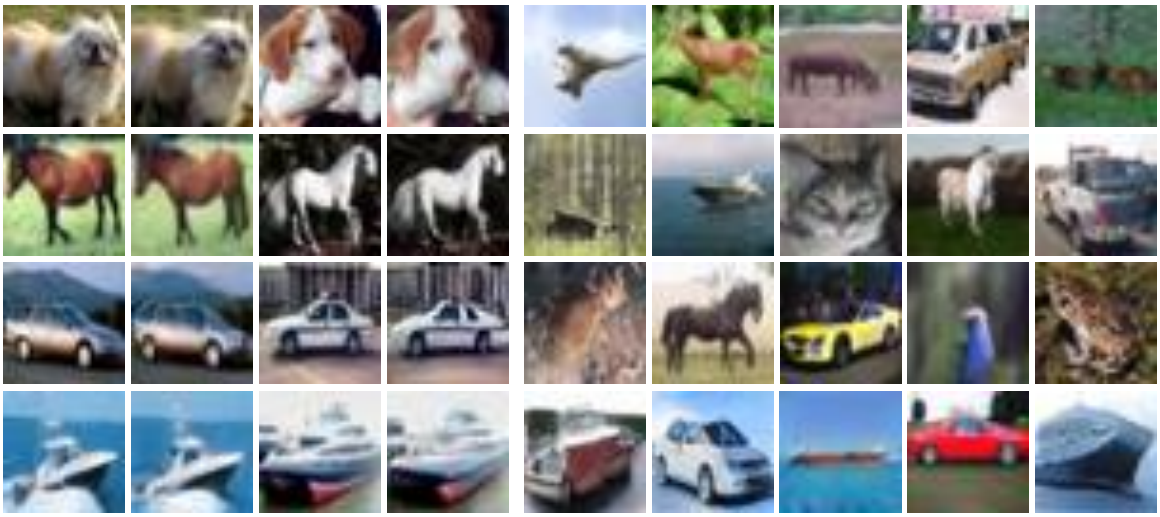
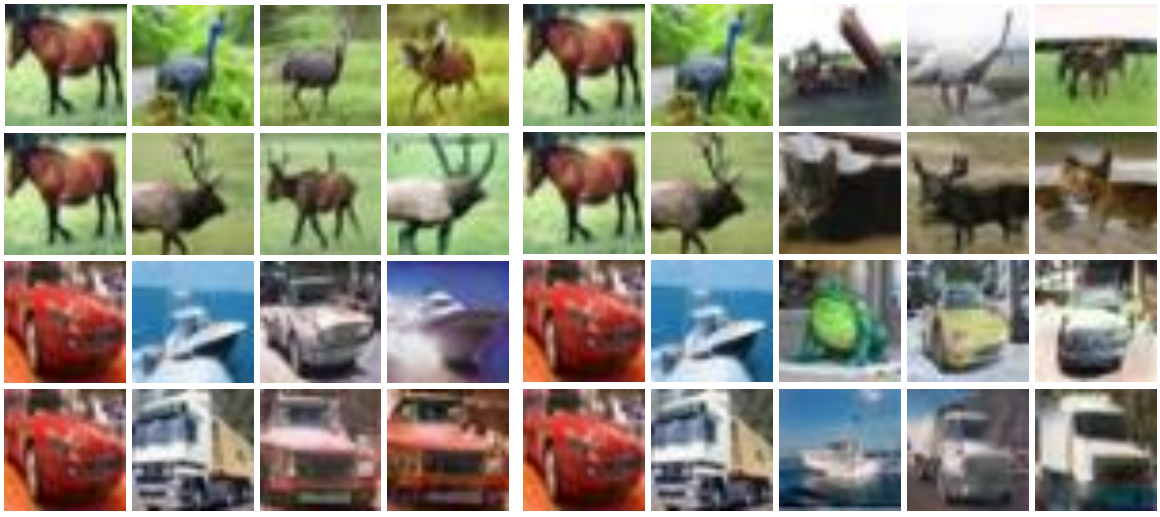


Figure 3.5 Accuracy of robust trained PreActResNets on various attacks (SVHN).

latent space on CIFAR-10. We then optimize a 512-dimensional latent representation vector for each training and testing sample to build a manifold representation set for CIFAR-10. Similarly, we also build respective on-manifold representation sets for SVHN and ImageNet-Mixed10, respectively. Taking CIFRA-10 as an example, it can be seen from Figure 3.6 (a) that when the testing samples are projected into the latent space learned on the training set, the reconstructed samples from latent representations are almost the same as the original test samples. This indicates that the data distribution supported by our learned manifold is close to the true data distribution. Moreover, we generate unknown on-manifold samples by randomly sampling representations in the manifold embedding space, as shown in Figure 3.6



(a) CIFAR-10 dataset projection. (b) On-manifold CIFAR-10 sampling.



(c) Dual `LarepMixup`. (d) Ternary `LarepMixup`.

Figure 3.6 On-manifold dataset and mixed samples.

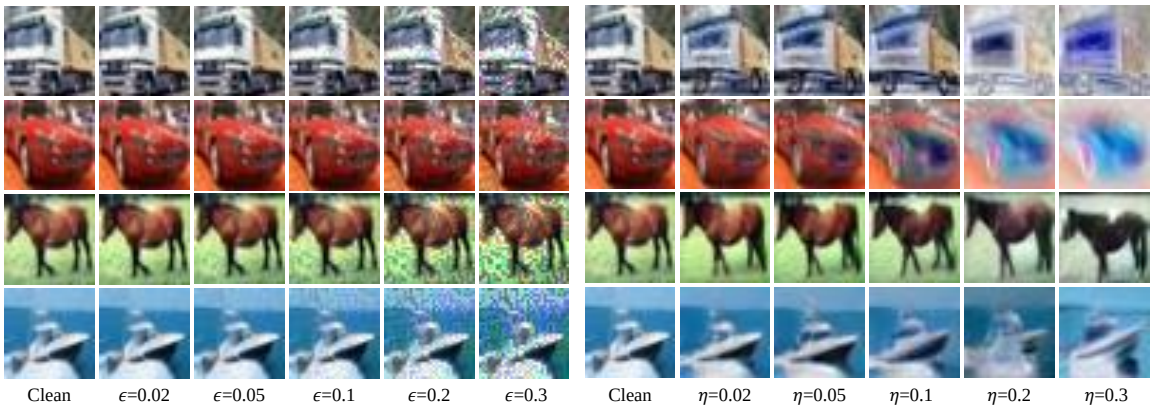
(b). The natural semantics of synthesized samples also prove that the manifold we constructed approximates the underlying data manifold.

Mixed Examples. The perception of convex mixed samples and binary mask mixed samples are shown in Figure 3.6 (c) and (d), respectively. For convex mixing, the synthesized examples show more smooth mixed characteristics between source samples, like luma, color, and contour, since the combination coefficient α can take a value from the continuous range, $[0, 1]$. Each specific feature in the convex mixed image that corresponds to a dimension of the latent representation will show the merged value of the scaled features of the source samples with a high probability. For binary mask mixing, the synthesized examples show fewer transitions between source features because the combination coefficient m is discrete

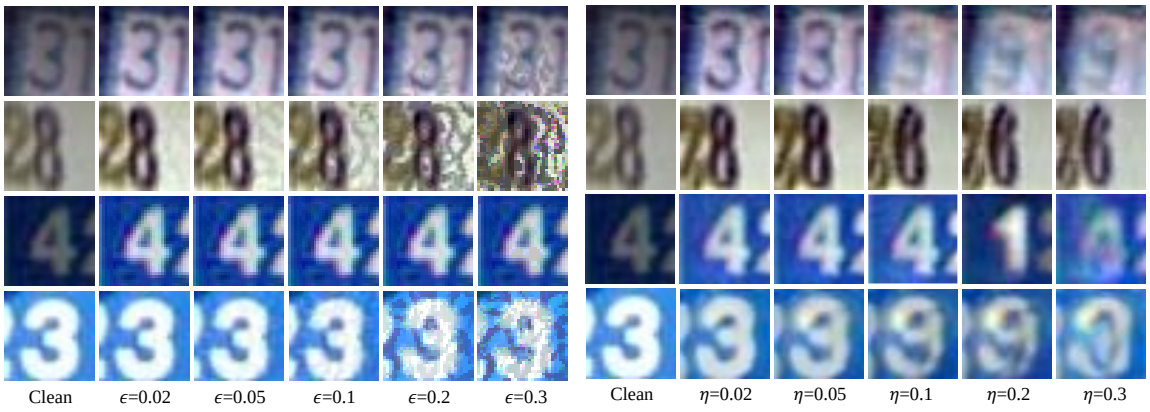
and can only be taken from the binary set $\{0,1\}^n$. Each specific feature in the binary mask mixed image preserves either the feature of one source sample or the other.

3.6.2 Evaluation under Different Attack Budgets

To demonstrate the effectiveness of `LarepMixup` in improving the off-manifold and on-manifold adversarial robustness of the model under different adversary attack strengths, we evaluate the top-1 accuracy of the classifier on PGD and OM-PGD AEs with different budgets of l_∞-bounded perturbations. For the PGD attack, the single-step budget is 0.02. For the OM-PGD attack, it is 0.005.



(a) PGD on CIFAR-10 (b) OM-PGD on CIFAR-10



(c) PGD on SVHN (d) OM-PGD on SVHN

Figure 3.7 Visualization of PGD and OM-PGD adversarial examples.

The perception of PGD and OM-PGD adversarial examples (AEs) under varying perturbation strengths is shown in Figure 3.7. In Figure 3.7 (a) and (c), off-manifold samples (PGD) display granular noise, which is caused by the adversarial perturbation directly superimposed on the pixels. In Figure 3.7 (b) and (d), on-manifold samples (OM-PGD) exhibit smooth noise due to perturbations on low-dimensional latent representations, affecting high-level features like direction and style. As the perturbation budget increases, semantic changes

become more drastic. However, as emphasized in[40], care needs to be taken that when implementing on-manifold attacks, label invariance should be considered. Careful control of the perturbation budget is needed to avoid changing the original class manifold, which would create invalid on-manifold AEs, as shown in the last two columns of Figure 3.7 (d).

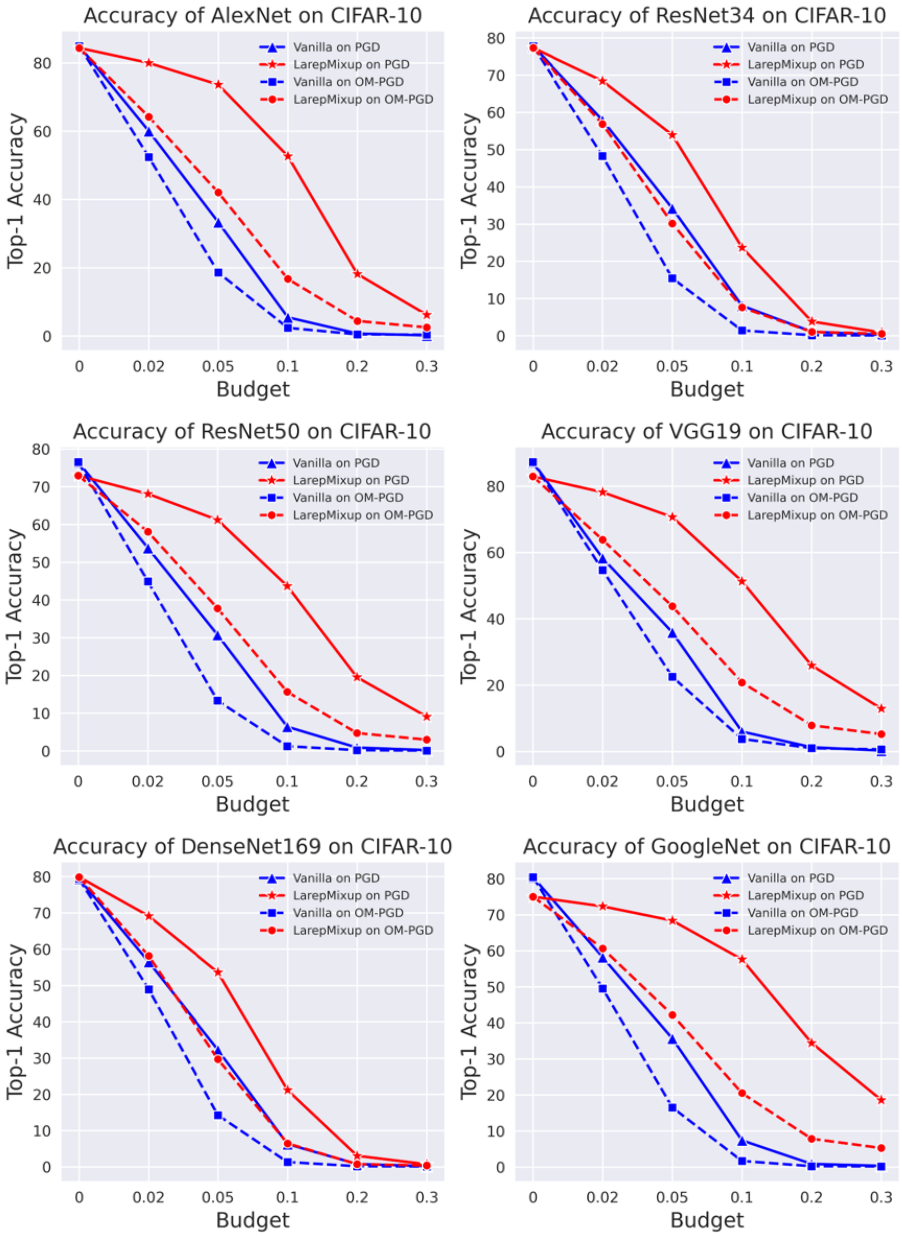


Figure 3.8 Accuracy of various models under different attack budgets (CIFAR-10).
PGD budget ϵ and OM-PGD budget η are set sequentially as $\{0.02, 0.05, 0.1, 0.2, 0.3\}$.

It can be seen from Figure 3.8 and Figure 3.9 that `LarepMixup` training notably enhances robustness against different off/on-manifold adversarial attack strengths on the CIFAR-10 and SVHN datasets. For PGD on CIFAR-10 with ϵ set to 0.02, 0.05, 0.1, 0.2, 0.3, the average accuracy of the seven models improves by 14.54%, 28.36%, 32.32%, 14.57%, 6.78%,

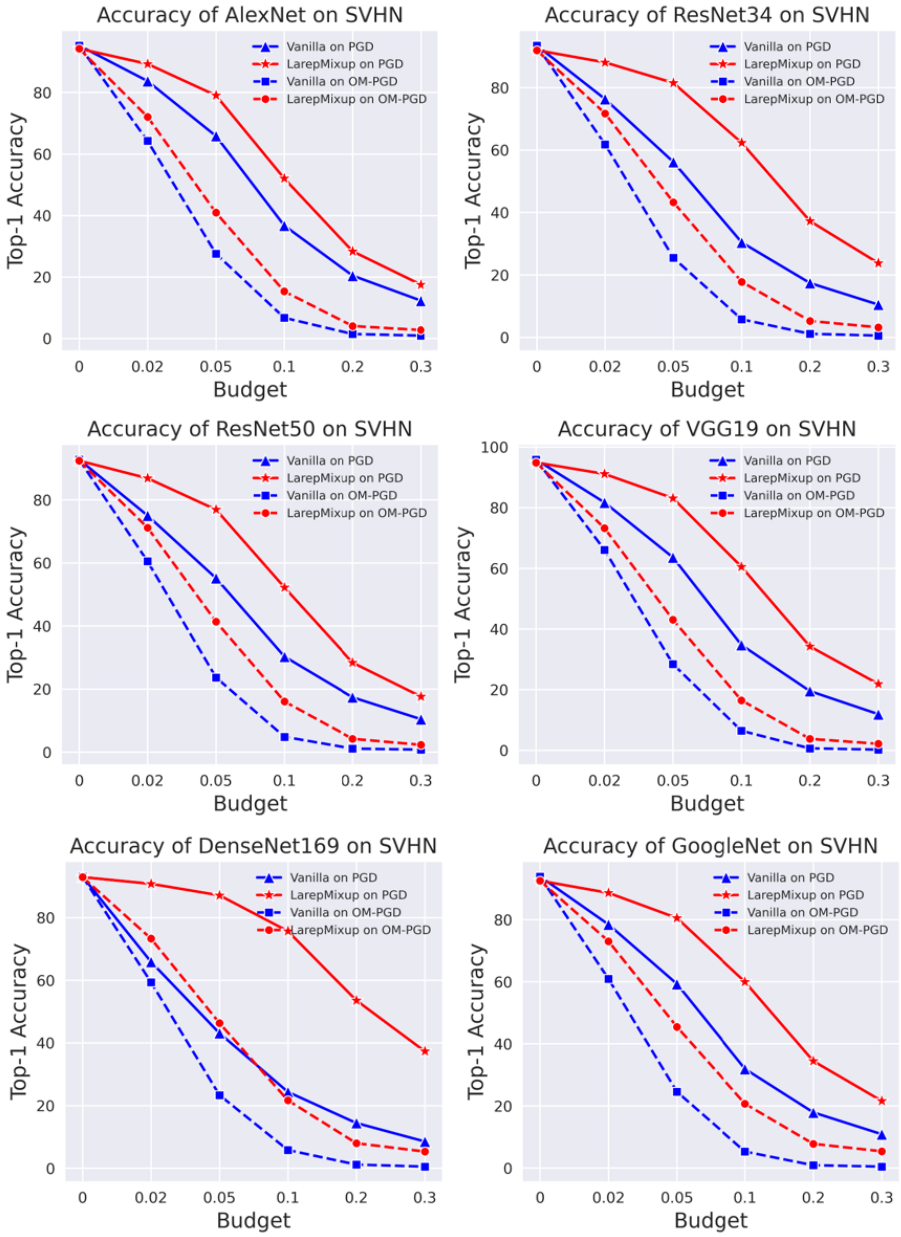


Figure 3.9 Accuracy of various models under different attack budgets (SVHN).
PGD budget ϵ and OM-PGD budget η are set sequentially as $\{0.02, 0.05, 0.1, 0.2, 0.3\}$.

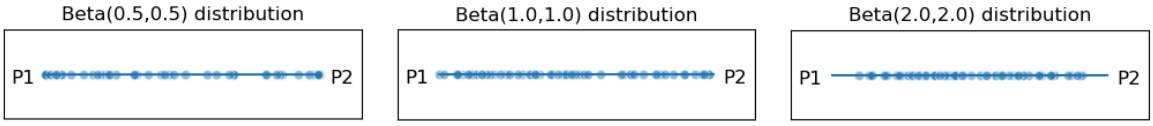
respectively. For OM-PGD on CIFAR-10 with η set to 0.02, 0.05, 0.1, 0.2, 0.3, the average accuracy of the seven models improves by 10.18%, 19.93%, 11.64%, 3.60%, 2.26%, respectively. Under the same settings, for PGD on SVHN, the average classification accuracy of the seven models improves by 12.51%, 24.67%, 29.27%, 18.13%, 12.38%, respectively. For OM-PGD on SVHN, the average classification accuracy of the seven models improves by 10.27%, 17.93%, 12.21%, 4.53%, 3.05%, respectively.

A remarkable observation is that when the budget η is too large, e.g., exceeds 0.1, the improvement in robust accuracy for on-manifold attacks diminishes. In conjunction with Fig-

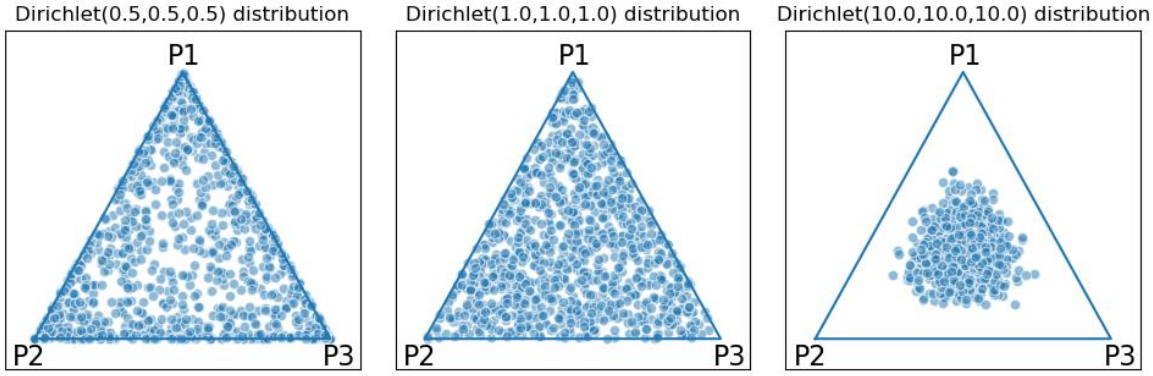
ure 3.7, we deduce that this occurs because an excessive attack budget generates some invalid OM-PGD attack samples. Consequently, in our following experiments, we employed on-manifold AEs with a 0.1 budget, under which invalid OM-FGSM and OM-PGD samples are seldom observed in CIFAR-10 and SVHN datasets.

3.6.3 Evaluation under Different Mixing Modes

To evaluate the efficacy of `LarepMixup` in improving adversarial robustness across different mixing modes, we alternate between dual/ternary convex mixing and dual/ternary mask mixing. We conduct experiments using the ImageNet-Mixed10 dataset to verify the proposed method's applicability to high-dimensional datasets. For each adversarial attack, we conduct proposed training three times under the same settings and take the average as the final result. The initial learning rate, epoch number, and batch size are 0.01, 40, and 32, respectively. The adversarial perturbation budget is 0.02. The parameters β of the Beta(β) distribution and γ of the Dirichlet(γ) distribution are hyperparameters, set by default to (1.0, 1.0) and (1.0, 1.0, 1.0), respectively. The positional relationship between source data points and mixed data points constructed using different coefficients is illustrated in Figure 3.10.



(a) Two source data points P_1, P_2 and 60 mixed data points.



(b) Three source data points P_1, P_2, P_3 and 4000 mixed data points

Figure 3.10 Effect of sampling distribution on the position of interpolation points.

Experimental results evaluated in different modes are shown in Table 3.7. For off-manifold adversarial attacks, the robustness improvement from `LarepMixup` is not much different in convex mixing and mask mixing. However, for on-manifold adversarial attacks, the advantages of convex mixing are obvious. Regarding the number of mixed source samples, $\{Dual, Ternary\}$, there is little difference in accuracy improvement between them. In

general, for FGSM, PGD, AutoAttack, DeepFool, CW, OM-FGSM, and OM-PGD attacks, the accuracy rate of the four mixing modes increased by 2.90%, 3.15%, 3.67%, 75.67%, 83.71%, 16.62%, 22.27%, respectively.

Table 3.7 Robust accuracy of PreActResNet-18 under different mixing modes (ImageNet-Mixed10)

Method	Vanilla	Dual-LarepMixup		Ternary-LarepMixup	
		Convex	Mask	Convex	Mask
Clean	90.47	90.57±0.55	**90.89±0.35**	90.67±0.21	90.24±1.25
FGSM	13.93	17.09±0.29	16.21±0.14	16.71±0.34	**17.29±0.94**
PGD	2.00	5.38±0.81	4.68±0.45	4.73±0.69	**5.81±1.32**
AutoAttack	0.00	**3.74±0.19**	3.68±0.29	3.60±0.18	3.66±0.04
DeepFool	8.87	**85.38±0.19**	83.98±0.42	84.89±0.18	83.93±1.00
CW	0.10	**84.61±0.30**	83.16±0.52	84.19±0.47	83.28±0.62
OM-FGSM	26.90	**59.91±1.30**	28.61±5.58	57.36±1.89	28.21±0.98
OM-PGD	20.43	**58.76±1.30**	27.99±5.92	56.59±1.87	27.47±1.44

3.6.4 Evaluation with Perceptual Attack Examples

In addition to specific adversarial attacks, perceptual attacks have been identified as a means to evaluate model robustness against potential attacks[41,121]. These attacks primarily use global color shifts and image filtering on normal images to create perturbed images. We consider four perceptual attacks: Fog, Snow, Elastic, and JPEG. For each perceptual attack, we conduct LarepMixup training thrice with the same settings and average the results. The initial learning rate, epochs number, batch size, and beta distribution parameters are 0.01, 40, 256, (1.0, 1.0), respectively. We conduct experiments on seven models using the CIFAR-10 and SVHN datasets. Taking the AlexNet as an example, the perception of four types of perceptual attack examples on CIFAR-10 are shown in Figure 3.11.

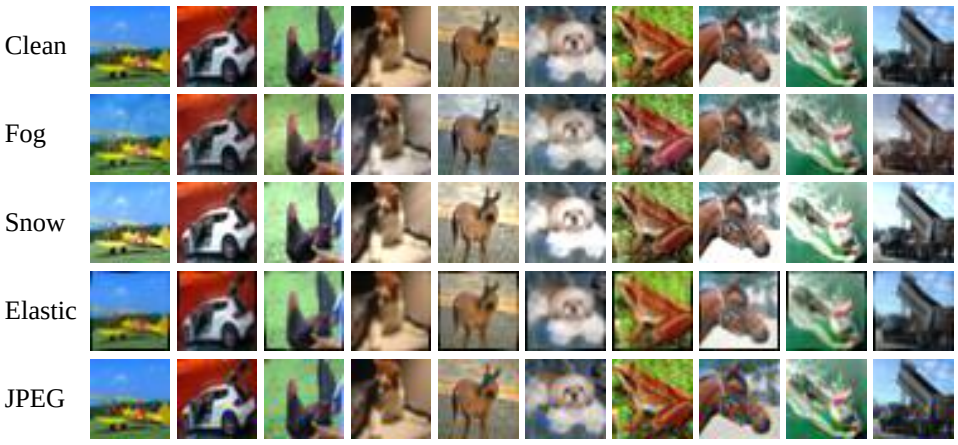


Figure 3.11 Percaptual attack examples.

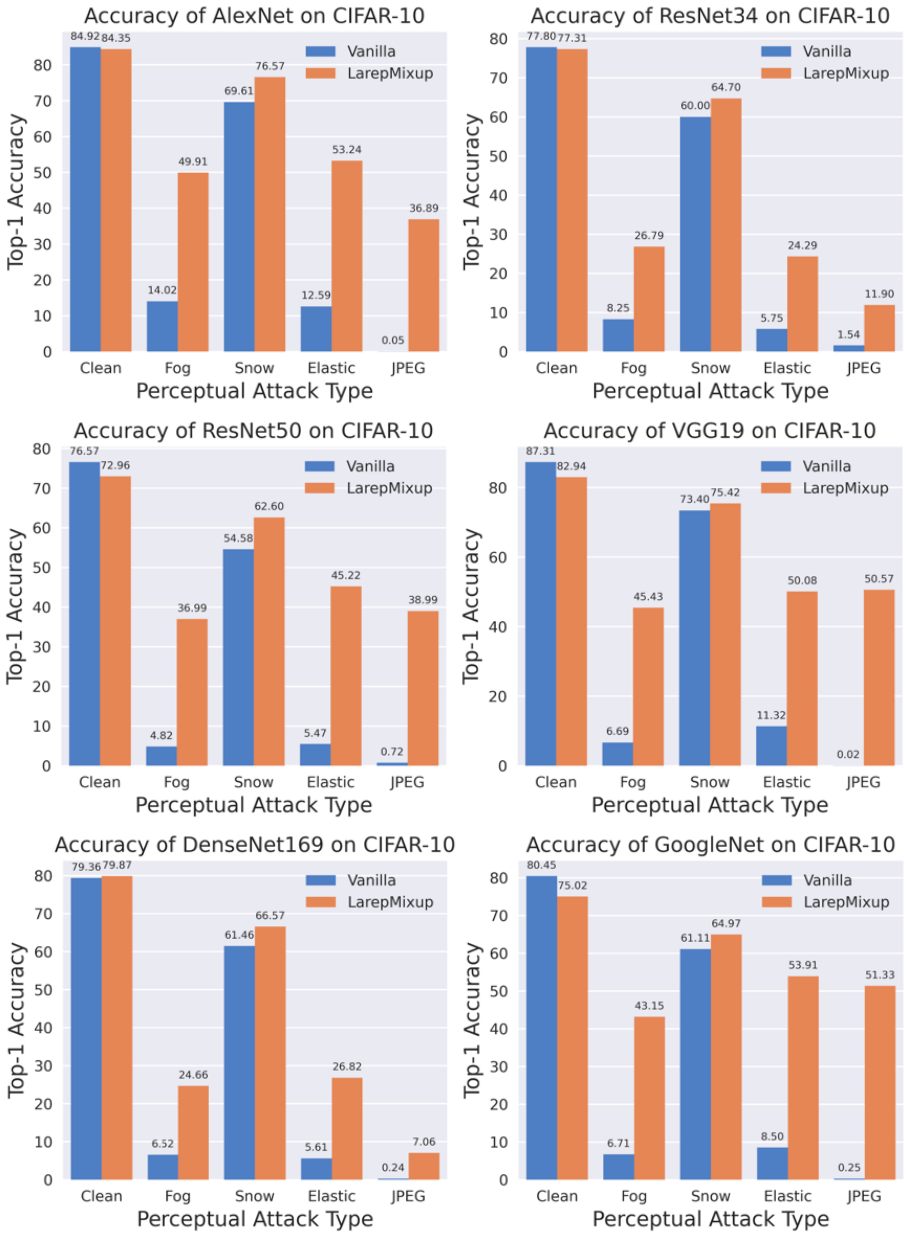


Figure 3.12 Accuracy of various models on perceptual attacks (CIFAR-10).

According to Figure 3.12, the accuracy of the classifiers on CIFAR-10 perceptual attacks has been greatly improved with `LarepMixup`, with the average accuracy of seven classifiers on Fog, Snow, Elastic, and JPEG samples increased by 28.17%, 5.19%, 31.79%, and 29.53%, respectively. At the same time, the accuracy of the seven classifiers on the clean test set dropped slightly, with an average reduction of 2.11%. Additionally, Figure 3.13 shows the improvement of the robustness of the classifiers on SVHN perceptual attacks, with the average accuracy of seven classifiers on Fog, Snow, Elastic, and JPEG samples increased by 17.89%, 14.42%, 35.87%, and 47.10%, respectively. Since natural samples are constructed by superimposing perturbations on feature vectors in input space, it is reasonable to regard

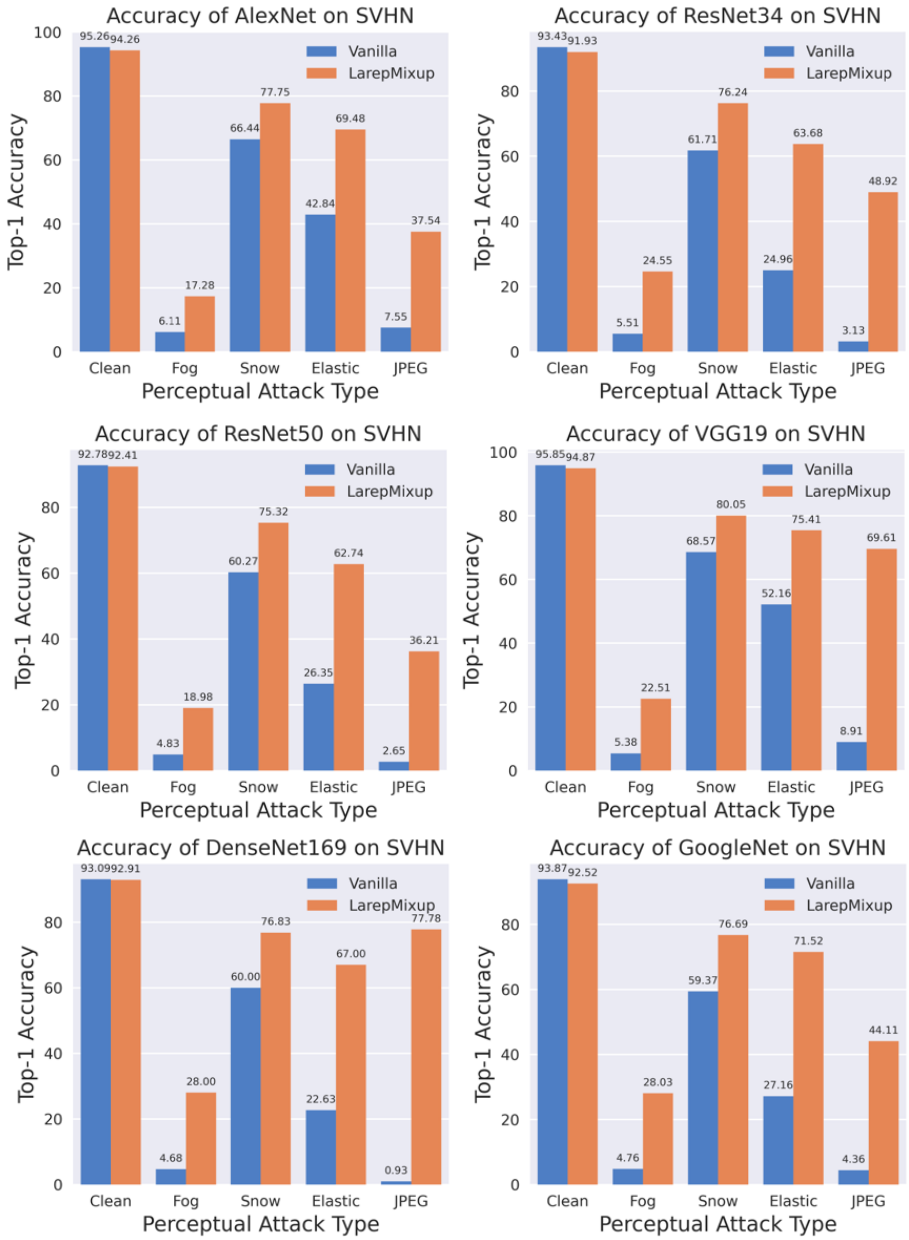


Figure 3.13 Accuracy of various models on perceptual attacks (SVHN).

them as unseen attack samples outside the manifold. It can be seen that the model trained by `LarepMixup` achieves generalized robustness to unseen off-manifold attacks.

3.6.5 Evaluation of Time Cost

In our scheme, the training of the StyleGAN model is separated from the training of the robust classifier. Once a StyleGAN model has been trained well on a given dataset, thereafter, it will only be used as a mapping function from low-dimensional representation to high-dimensional input to participating in the `LarepMixup` training of any target network. Taking the CIFAR-10 dataset as an example, we trained the StyleGAN model for 280 epochs

on the CIFAR-10 training set, each epoch took about 218 seconds. We spent a total of 16.9 hours training a StyleGAN model, realizing the final effect that a latent variable randomly sampled in the hidden space of StyleGAN can be mapped to a sample with real semantics in the input space. Then, we constructed the on-manifold CIFAR-10 datasets consisting of latent representations z and corresponding labels y. From the perspective of training a robust network, the above process can be regarded as a preprocessing process. After constructing the latent representation dataset of CIFAR-10, we then use it to build various robust networks on CIFAR-10. Taking the WideResNet-28-10 as an example, the time cost of `LarepMixup` training was almost 700 seconds per epoch. We trained the robust WideResNet-28-10 model for 40 epochs, a total of almost 7.7 hours.

3.7 Summary

In this chapter, we investigate the generalization of the adversarial robustness of DNNs on off-manifold and on-manifold adversarial attacks. The main idea of our work is to mix latent representations lying on the low-dimensional manifold of the training set to synthesize mixed samples that capture latent variation factors in the dataset, and use them as augmented examples to train a model that can stably recognize data points adjacent to the decision boundary. Extensive evaluations show that even without any adversary information, our method `LarepMixup` can significantly alleviate the sensitivity of the model to multiple attacks in the input space and latent space. While our framework is illustrated with images, it can be extended to other input domains by replacing the StyleGAN-based manifold embedding designed for images with appropriate representation encoding algorithms suitable for other input features, such as Autoencoder for network traffic features and BERT for text features.

Chapter IV Adversarial Robustness Certification with Multi-order Adaptive Randomized Smoothing

Deep neural networks (DNNs) are increasingly utilized in network intrusion detection (NID) due to their high detection accuracy and adaptability to evolving cyber threats. However, these DNNs-based NID systems face a similar challenge in other DNN-based classifiers: their vulnerability to adversarial attacks (a.k.a evasion attacks) designed to evade detection. To ensure adversarial robustness against various perturbations, there is a growing focus on certified defenses that can provide robustness guarantees for all potential perturbed inputs within the l_p-bounded region. Unfortunately, unlike existing methods focusing on homogeneous image feature spaces, the progress in adversarial robustness certification for the network traffic domain, characterized by heterogeneous features, has been limited.

To address such a gap, in this chapter, we study the certification technique for adversarial robustness of DNNs and propose a certified defense framework, Multi-order Adaptive Randomized Smoothing (`MARS`), designed to be applicable to heterogeneous input features, such as network traffic data. First, we introduce a designed adaptive randomized smoothing algorithm that leverages zero-order and first-order information to calculate robust radii, providing tighter lower bounds of robustness than existing methods. Then, we present a proposed dimension-wise robust radius calculation algorithm based on the sensitivity of each feature dimension, enabling fine-grained robustness certification for heterogeneous input features.

`MARS` has three important characteristics: (i) It dynamically expands the certified robust region toward high confidence by using the zero-order output and first-order gradient information of the smoothed classifier. (ii) It adaptively samples random noise according to the decision boundary by optimizing the parameters of a smoothing distribution specific to each traffic feature dimension. (iii) It supports multiple l_p norm-bounded robustness guarantees according to customized requirements. Finally, we present experimental results on various network intrusion detectors and datasets, demonstrating that the proposed method enhances certification tightness. It provides l_p certified radii that constrain larger perturbation regions than the leading robustness certification approach for network intrusion detectors. Moreover, it improves the empirical adversarial robustness against diverse adversarial attacks.

4.1 Overview

Deep learning (DL)-based network intrusion detectors (NIDs) excel in detecting complex and evolving cyber threats by leveraging the capability of deep neural networks (DNNs) to analyze large-scale and diverse traffic data[5-6]. However, previous work has shown that network traffic classifiers based on DNNs[18-19] are as vulnerable to adversarial attacks using adversarial examples (AEs) as text[124], image[16-17], speech[125], and video classifiers[126-127]. An attacker can transform an otherwise correctly classified clean input into an adversarial example by subtly adding perturbations[14-15], resulting in the victim classifier misclassifying these AEs. Adversarially modified malicious traffic usually mimics normal traffic patterns with constrained changes, thereby easily evading detection and keeping the features as similar as possible to clean samples[20].

Empirical defense methods designed to enhance the robustness of DNNs, such as adversarial training[9,14], feature denoising[23-24], and model ensembling[128-129], have been fully verified in the DL field, and their applicability has also been explored for network intrusion detection[21]. However, a common problem in various domains is that the robustness achieved by these heuristic strategy-driven empirical defenses is likely to be bypassed by new adversarial attack approaches[79,82]. Such a shortcoming allows attackers to easily evade the "supposedly robust" model through adaptive attacks[57], leading to an endless arms race of adversarial attacks and defenses. Moreover, in high-risk applications like autonomous driving, healthcare, and network intrusion detection, it is difficult to establish a high level of trust in the output results of the model based on empirical defenses.

Recognizing these shortcomings, research on the robustness of security-sensitive DL models has gradually shifted to certified defense[58]. Such a defense aims to calculate a *certified radius* for each input, to indicate that the model's predictions remain consistent for any variant of the current input within the region bounded by this radius, see Figure 2.4. The certified radius is provided as a robustness guarantee along with the input's predictions. For the same model and input sample, a larger certified radius obtained indicates a tighter robustness guarantee provided by the certified defense method, as defined in Definition 6. Incomplete certification, which aims to compute the *lower bound* of the exact robust radius of the model as the certified radius, avoids the NP-complete challenge of computing the exact robust radius of a DNN in complete certification[58,69]. However, a notable drawback of incomplete certification is that the robustness guarantee provided is loose; that is, the calculated lower bound is far from the exact robust radius.

To compute the non-trivial certified radius for DNNs, many approaches have been pro-

posed to upgrade incomplete certification algorithms for image classifiers, including deterministic certification, such as activation polytope[70-74], interval bound propagation[130-132], relaxation[133-136], neuron branching and bounding[61,64-66], and probabilistic certification, such as differential privacy[77-78] and randomized smoothing[69,79-81,137]. Given that an ideal certified defense against adversarial attacks should be model agnostic, that is, it should be applicable to various types of DL models without modifying or being limited by the specific internal structures of these models, randomized smoothing-based approaches have been proven to be the most competitive in terms of tighter and architecturally scalable certification in the field of image[58], text[138], and graph[139] classification.

Motivations. However, certified defense efforts for network intrusion detection have been minimal. The main challenges arise from the heterogeneity of network traffic features, where different dimensions carry varying semantics and characteristics. In contrast to image features representing pixel values, network traffic feature dimensions involve protocol types, destination network services, timestamps, data packet counts, flow-byte rates, and more. Additionally, the diversity of NID architectures also introduces difficulties to certified defenses. In binary/multi-class classification or known/unknown anomaly detection tasks, the detection principles employed lead to the utilization of various DNN models, such as CADE (Contrastive Autoencoder for Drifting detection and Explanation)[6], ACID (Adaptive Clustering-based Intrusion Detection)[5], etc. This imposes strict demands on the scalability of certification methods across diverse model architectures.

Until now, only one approach, BARS (Boundary-Adaptive Randomized Smoothing)[82], has been proposed to certify the robustness of network traffic classifiers. It obtained larger l_2 certified radii than neuron branching and bounding methods by utilizing the zero-order information-based randomized smoothing. Unfortunately, its l_2 robustness guarantee is proven relatively loose, and it lacks certification for other l_p norms-bounded robustness guarantees. Providing multiple l_p-measured certified radii can help in deeply analyzing model vulnerabilities and boosting general robustness against adversarial attacks using diverse norms like l_1 or l_∞ in different contexts. Moreover, BARS only handles adversarial attacks, neglecting some natural corruptions, such as Latency and PacketLoss, that may be induced by random noise in the network environment.

Our Method. To address the above shortcomings, we propose `MARS`, a novel framework that certifies the robustness of DNN-based NIDs using **M**ulti-order **A**daptive **R**andomized **S**moothing, as illustrated in Figure 4.1. The capabilities of `MARS` are demonstrated in three main aspects.

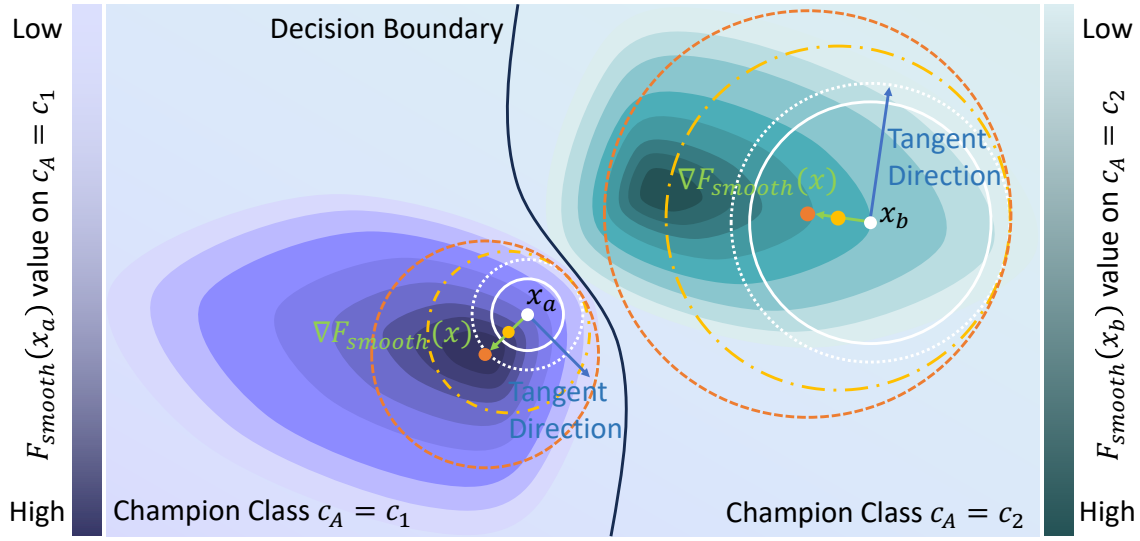


Figure 4.1 Certified radius obtained through MARS.

c_A denotes the class with the highest confidence returned by the smoothed classifier F_{smooth} on the input. x_a and x_b are two inputs predicted as different champion classes, c_1 and c_2. The solid white line outlines the certified region relying solely on zero-order output $F^{c_A}_{smooth}(x)$. The blue arrow indicates the tangent direction of $F^{c_A}_{smooth}(x)$ at x. The green arrow represents the gradient vector $\nabla F^{c_A}_{smooth}(x)$. The dotted white line outlines the certified region utilizing zero-order and a gradient with zero magnitude. The yellow and orange dotted lines outline the certified regions employing zero-order and a gradient with non-zero magnitude, where orange corresponds to a larger gradient magnitude.

- First, to adapt to the heterogeneity of d-dimensional network traffic features, we design a dimension-wise certified radius calculation method by expanding the real value of the certified radius into a certified radius vector for the network traffic domain, and quantifying the radius contribution of each dimension based on the dimensional feature sensitivity analysis.

- Then, to tighten the robustness guarantee, we adopt a two-step strategy: (i) We optimize the dimensional parameters of the multivariate smoothing distribution so that the certification algorithm can adaptively sample dense noised samples near the boundary for probability statistics. (ii) We iteratively move the symmetry center of the l_p certified robust area along the gradient direction of the smoothed classifier, that is, the direction in which the confidence score of the output class increases, and use binary search to estimate the upper and lower bounds of the interval of certified radius, so that the certified radius can be further improved.

- Finally, to provide diverse l_p certificates, we construct a smoothing distribution set consisting of Gaussian, Laplacian, and Uniform distributions, and implement specific parameter optimization and first-order gradient estimation for each distribution.

We compare MARS with SOTA traffic-specific certification BARS [82], and image-specific

methods Vanilla Randomized Smoothing (VRS)[79] and First Order-based Randomized Smoothing (FRS)[81], in terms of the certified radius, certified accuracy, robust accuracy, clean accuracy, and time overhead.

In this chapter, we make the following contributions:

- We propose MARS — a robustness certification framework, to certify the robust radius of DNN-based NIDs without requiring any modification to the model structure. It achieves a tighter l_2 robustness guarantee (12.23% average increase compared to BARS) and extends certification from l_2 to l_1 and l_∞ guarantees compared to other advanced methods.

- We are the first to utilize the high-order information of the smoothed classifier to guide the expansion of the certified region obtained based on the zero-order output information in network traffic classification. Our approach demonstrates improved tightness in various l_p robustness guarantees.

- We are the first to introduce a threat model of random noise-based natural corruptions in addition to the threat of adversarial attacks in NID robustness certification. Our experimental results confirm that MARS significantly enhances robustness against adversarial attacks (33.93% higher on l_∞-PGD, 13.79% higher on l_2-PGD, 10.01% higher on l_1-EAD) and natural corruptions (16.87% higher on Latency, 19.85% higher on PacketLoss) compared to the base detection model.

4.2 Problem Formulation

4.2.1 Threat Model

We consider two robustness threats faced by DNNs: (i) adversarial attacks — deliberately launched by attackers using adversarial examples (AEs), and (ii) natural corruptions — unintentionally caused distribution shift by random noise in the network environment.

Adversarial Attacks. We first focus on white-box adversarial attacks. The adversary creates strong l_p AEs based on complete knowledge of the victim network traffic classifier f_θ. By assuming that the attacker possesses full knowledge of the model, we aim to simulate a most powerful threat in the adversarial scenario where the adversary has maximum visibility into the model internals. This enables the evaluation of the robustness of the model against sophisticated attacks, leveraging the model's inner workings while also revealing potential vulnerabilities of the target model. Although adversarial attacks can target clean samples

belonging to any category, in network intrusion detection, especially in multi-classification scenarios, the more realistic situation is that the evasion goal only includes causing the originally malicious traffic to be classified as benign, but does not include causing originally benign traffic to be classified as malicious or causing malicious traffic to be classified as another attack type. Thus, we assume that *the adversary will launch adversarial* attacks only on originally malicious traffic, where the target label y_{target} in Definition 5 is set to benign.

Natural Corruptions We also consider the robustness of the classifier to distribution shifts arising from natural variations in datasets. Natural corruptions result from uncontrollable environmental factors, such as lighting changes in images or recording device alterations in speech. As the first work to consider natural corruptions in robustness certification in the traffic domain, we focus on the distribution shifts caused by random noise added to time-related and quantity-related traffic features. By assuming noise background in temporal and spatial characteristics, we aim to mimic a scenario where natural corruptions like *Latency* and *PacketLoss* arise from network congestion or electromagnetic interference. Unlike adversarial attacks, these corruptions are typically unintentional; thus *both clean* benign and malicious traffic can be corrupted.

With these considerations, the robustness of network traffic classifiers against adversarial and natural perturbations on the input can be certified and empirically demonstrated, ensuring effective operation amid unexpected network disturbances.

4.2.2 Research Goal

Our design goal is to provide the traffic classifier prediction with a robustness guarantee that reflects the tight lower bound of the robustness of the model on any unknown perturbations. We need to address the following three issues.

Problem 1. *Formally define a certified radius as the l_p robustness guarantee that can constrain heterogeneous network traffic features.* Homogeneous image feature vectors typically share semantics and value ranges across dimensions, resulting in a single real-value certified radius R, constraining all dimensions concurrently. Conversely, heterogeneous traffic feature vectors exhibit varied semantics, value ranges, and significance in traffic analysis and noise tolerance. Thus, a network traffic-specific certified radius form must be designed to reflect robust regions within heterogeneous feature dimensions.

Problem 2. *Tighten the l_2 certified radius to provide a stricter l_2 robustness guarantee than the only existing certification approach for NIDs.* A larger certified radius signifies higher prediction credibility of the model, which is essential in applications demanding high-

confidence results. For NIDs, detection combined with a certified radius can minimize false positives and false negatives. For example, if it is required that only predictions with certified radii higher than the robust radius threshold on the predicted malicious category trigger anomaly warnings, alert fatigue can be alleviated.

Problem 3. *The l_1 and l_∞ robustness guarantees of the model on a given input must also be provided.* While l_2 attacks are common, adversaries may employ other norms (e.g., l_1, l_∞) for escape purposes. It is vital to have a comprehensive approach for calculating diverse l_p measured certified radii to assess the overall robustness. Especially when confronting unknown types of adversarial attacks, comparing various l_p certified radii is particularly valuable for thoroughly analyzing the model weaknesses and enhancing l_p robustness.

4.2.3 Key Challenge

The approaches and challenges to address these problems are considered below.

Approach Direction to Problem 1. The solution we consider is to extend the real-value certified radius R to a vector $(R_1, ..., R_d) \in \mathbb{R}^d$, where R_i denotes the dimension-wise robustness guarantee for the i-th feature x_i of the input x. Yet, computing the certified radius vector R poses a challenge.

Challenge 1. *Efficiently calculate the dimension-wise certified radius R_i for each dimension of the input $x = (x_1, ..., x_d) \in \mathbb{R}^d$.* An easy way to calculate R_i is employing randomized smoothing separately for each dimension by adding noise to feature x_i while keeping other dimensions constant. However, since we assume that network traffic feature dimensions are not independent, this approach has two issues: (i) The certified radius vector composed of the independently calculated dimension-wise certified radius represents the upper bound of the exact certified radius vector rather than the lower bound, because it does not account for the correlation between different feature dimensions of network traffic, such as x_i and x_j. (ii) Performing randomized smoothing independently for each feature dimension increases the time cost significantly by a factor of d, making it impractical for network traffic classification tasks with real-time requirements.

Approach Direction to Problem 2. To tighten the l_2 robustness guarantee, the solution we envision is to improve the randomized smoothing-based certification algorithm by introducing high-order information about the smoothed classifier. Yet, this solution requires addressing the following challenge.

Challenge 2. *Characterize the correlation between the* first-order information of a smoothed classifier and the certified radius and derive a tighter robustness guarantee by ex-

ploiting it. The first-order gradient information of the classification function on the input reveals how subtle changes in the input affect predictions. This valuable insight could determine robust regions in smoothed classifiers. However, this area remains unexplored in NIDs and is in the early stage of image classification. The challenge is the lack of analysis connecting this information to the certified radius, impeding its use as a supplementary condition for certified radius calculations.

Approach Direction to Problem 3. The solution we consider is to select distinct smoothing distributions for various l_p guarantees to align the sampling area of the noise samples used for smoothing with the l_p-measured surroundings of the input x. Nevertheless, the challenge is as follows.

Challenge 3. *For various l_p perturbations, choose suitable smoothing distribution types and parameter settings to obtain non-trivial tight robustness lower bounds.* While Gaussian distributions have proven effective for l_2 perturbations when combined with zero-order information, they might not always be optimal for l_1 and l_∞ attacks, especially when first-order information is also used. Hence, it is essential to integrate prior knowledge to generate candidates for appropriate smoothing distributions and to experimentally ascertain whether the zero-order or first-order information under these distributions is appropriate for various l_p robustness certifications.

4.3 Design of Method MARS

This section introduces the design of the proposed certified defense method, Multi-order Adaptive Randomized Smoothing (`MARS`), to provide non-trivial tight l_p norm-bounded robustness guarantees for heterogeneous input features, such as network traffic data. First, we introduce the basic architecture of the smoothed network traffic classifier. Then, we detail the certification algorithm that leverages zero-order and first-order information to calculate robust radii. Next, Finally, we present a dimension-wise robust radius calculation algorithm based on the sensitivity of each feature dimension.

4.3.1 Architecture of the Smoothed Classifier

The basic architecture of a smoothed network traffic classifier (e.g., a Network Intrusion Detector (NID)) is shown in Figure 4.2. The main difference between the smoothed classier F_{smooth} and the base classier F is that the predicted label of F_{smooth} on the input x is the champion class c_A, which is the most often predicted class by the base classifier $F(x)$ across a set of noised samples $x + \eta$.

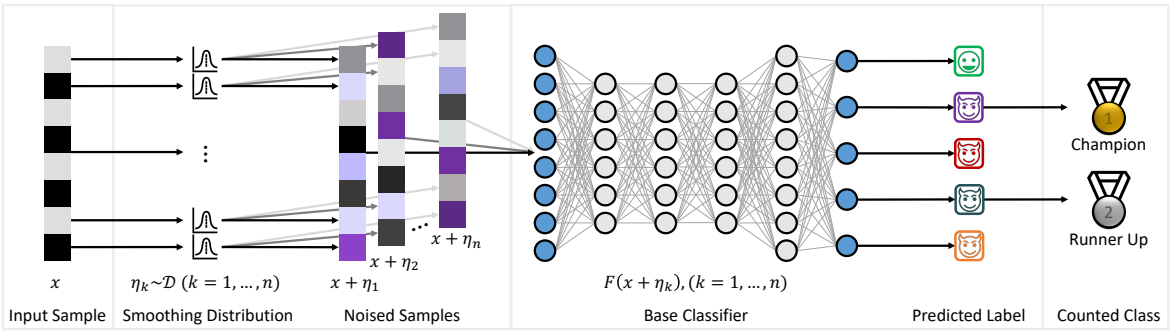


Figure 4.2 Architecture of the smoothed network traffic classifier.

Prediction Procedure: $n = n_{small}$, identify the champion class c_A that is predicted most times among n noised samples.

Certification Procedure: $n = n_{large}$, count the number of noised samples predicted as c_A to estimate $P_A = \mathbb{P}(F_{smooth}(x) = c_A)$ and $P_B = \mathbb{P}(F_{smooth}(x) = c_B)$.

During the inference phase, the operation of the smoothed classifier involves two processes: prediction and verification.

Prediction Procedure. This procedure aims to determine the class by the smoothed classifier for the input x. It begins by choosing a smoothing distribution $\mathcal{D}$ with mean 0. Then, n_{small} (defaults to 100) noise vectors η are sampled and added to x to obtain n_{small} noised samples. The base classifier predicts them and identifies the champion class c_A and runner-up class c_B.

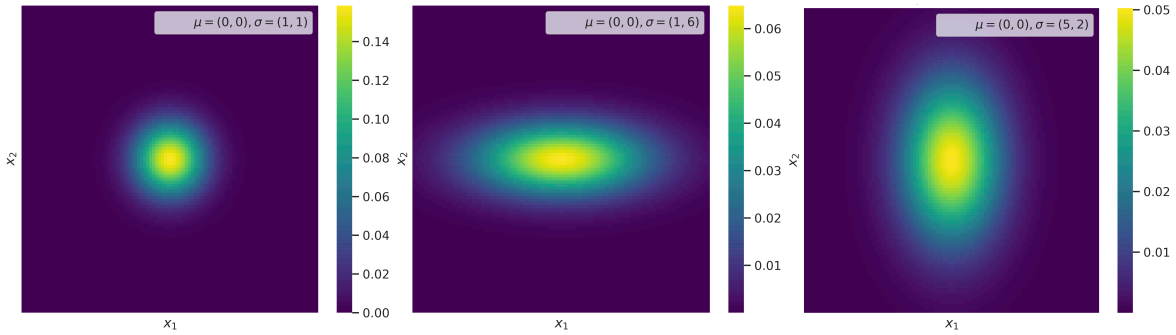
Certification Procedure. This procedure aims to calculate a l_p-measured certified radius R. First, n_{large} (defaults to $10,000$) noises are randomly sampled from the smoothing distribution $\mathcal{D}$ and sequentially added to the input x to obtain n_{large} noised samples. Then, the number of these samples predicted as the champion class c_A is recorded as $n_A = \sum_{k=1}^{n_{large}} \mathbb{I}[F(x + \eta_k) = c_A]$. With n_{large} and n_A, the certified radius is then calculated using the zero-order output and first-order gradient of the smoothed classifier.

4.3.2 Multi-order Adaptive Robustness Certification

To achieve tight robustness guarantees for DNN models on heterogeneous inputs, our proposed adaptive robustness certification method based on multi-order information involves two stages: (i) Smoothing Distribution Parameter Optimization and (ii) Gradient-based Certified Radius Calculation.

Stage 1: Smoothing Distribution Parameters Optimization. During the training phase, we optimize the parameters of the smoothing distribution used for sampling noise η, making noised samples $x + \eta$ closer to the decision boundary.

The difference between a smoothed network traffic classifier and a smoothed image classifier is that the noise values η_i in each dimension of the noise vector η are sampled from



(a) Same σ across dimensions (b) Different σ across dimensions (c) Different σ across dimensions

Figure 4.3 PDF of binary Gaussian distribution $\mathcal{N}(\mu, \sigma)$.

a dimension-specific optimized distribution, where all dimensions of heterogeneous x are matched to the optimal smoothing distribution parameters ϑ. Take the Gaussian distribution as an example. Before the optimization, the d-dimensional noise vector η is sampled from the multivariate standard Gaussian distribution $\mathcal{D}_{std} = \mathcal{N}(\mu = O, \sigma^2 = I)$, where O and I denote the d-dimensional full-zero vector and the full-one vector, respectively. After the optimization, the noise vector η is sampled from the optimized multivariate Gaussian distribution $\mathcal{D}_\vartheta = \mathcal{N}(\mu = O, \sigma^2 = \vartheta I)$. Parameters ϑ optimized dimensionally for the multivariate distribution facilitate an adaptive approach to the classification boundary in feature space. Figure 4.3 shows the Probability Density Function (PDF) of the smoothing distribution with the same or different σ values across dimensions.

The optimization of $\vartheta = \vartheta_{shape} \times \vartheta_{scale}$ involves two steps: distribution shape optimization and distribution scale optimization.

Distribution Shape Optimization. The aim is to optimize the vector parameter ϑ_{shape} in a multivariate distribution, keeping $\vartheta_{scale} = 1$. This encourages the sampling region of noised samples $x + \eta$ to be close to the decision boundary of the class predicted by the classifier F for x by optimizing Eq. (4-1).

$$\begin{aligned} &\min_{\vartheta} \mathbb{E}_{x \sim D_{train}} \mathbb{I}[F(x + \eta_\vartheta) \neq F(x)] L(f(x + \eta_\vartheta), F(x)) \\ &-\mathbb{I}[F(x + \eta_\vartheta) = F(x)] L(f(x + \eta_\vartheta), F(x)), \end{aligned} \tag{4-1}$$

where $\vartheta = \vartheta_{shape} \times 1$ and $\mathbb{I}$ is the indicator function.

Distribution Scale Optimization. The optimization goal of the distribution scale, as defined in Eq. (4-2), is to expand the coverage of the sampling area by adjusting the scalar parameter ϑ_{scale} of the multivariate distribution while maintaining the contour shape of the sampling area by fixing ϑ_{shape} to the optimized value ϑ^*_{shape}, so that the certified radius R

can be as large as possible.

$$\max_{\vartheta} R = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B})) = \max_{\vartheta} \frac{\vartheta}{2}(\Phi^{-1}(\mathbb{P}_{\eta\sim\mathcal{D}_{std}} (F(x+\vartheta\eta) = c_A)) - \Phi^{-1}(\mathbb{P}_{\eta\sim\mathcal{D}_{std}}(F(x+\vartheta\eta) = c_B))) \tag{4-2}$$

Stage 2: Gradient-based Certified Radius Calculation. During the inference phase, we focus on calculating the certified radius using the zero-order and first-order information together. we calculate the magnitude of the first-order gradient $||\nabla F^{c}_{smooth}(x)||_p$ of the smoothed classifier w.r.t x and expand the certified robust region along the direction in which the confidence score F^{c}_{smooth} increases. The zero-order information we use is the statistical probability $P_A = \mathbb{P}(F_{smooth}(x) = c_A) = F^{c_A}_{smooth}(x)$ of the smoothed classifier when predicting x as the champion class c_A. The first-order information we use is the gradient magnitude $||\nabla F^{c_A}_{smooth}||_p$ of the F_{smooth}. The overall calculation process is detailed in Algorithm 4.1, which can be divided into two steps: Probability-based Radius Calculation and Gradient-based Radius Extension.

Step 1: Probability-based Radius Calculation. This step is to calculate the lower bound of the perturbation radius R that the smoothed classifier can tolerate on x based on $F^{c_A}_{smooth}(x)$, which is the estimated probabilities of the smoothed classifier predicting x as the champion class. Suppose the most probable class c_A is returned by F_{smooth} with probability $P_A = \mathbb{P}_{\eta\sim\mathcal{D}}(F(x+\eta) = c_A)$, and the runner-up class c_B is returned with probability $P_B = \mathbb{P}_{\eta\sim\mathcal{D}}(F(x+\eta) = c_B)$. We need to estimate the $\underline{P_A}$ and $\overline{P_B}$, which represent the lower bound of P_A and the upper bound of P_B, respectively. $\underline{P_A}$ is estimated like[79], using `LowerConfidenceBound`(n_{large}, n_A, α), which first calculates the interval $[\underline{P_A}, \overline{P_A}]$ where P_A holds with a probability of at least $(1-\alpha)$ for k-fold Binomial(n_{large}, P_A) sampling and then returns the left boundary of the interval. Then simply take $\overline{P_B} = 1 - \underline{P_A}$. Like[79], the certified radius R_{zero} based only on the zero-order information is calculated according to Eq. (4-3):

$$R_{zero} = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B})) \tag{4-3}$$

where Φ^{-1} is the inverse of the cumulative distribution function (CDF) of the standard Gaussian Distribution $\mathcal{N}(O, I)$. The size of R_{zero} is shown in the white solid line surrounding the input sample. x_a or input sample x_b in Figure 4.1.

Step 2: Gradient-based Radius Extension. The goal of this step is to move and expand the certified robust region with radius R_{zero} along the gradient direction $\nabla F^{c_A}_{smooth}$ at x. We can see from Figure 4.1 that the gradient-based certification breaks the symmetry

Algorithm 4.1 Multi-order Robustness Certification

Input: test samples $(x, y) \in D_{test}$, base classifier F with the classification function f_θ, smoothed classifier F_{smooth}

Output: overall certified radius R for each test sample

for $j = 1$ to $len(D_{test})$ **do**
  load one test example x_j, y_j;
  count champion class c_A according to Eq. (2-33);
  sample n_{large} noise samples $\{\eta_1, ..., \eta_{n_{large}}\}$ from smoothing distribution $\mathcal{D}$;
  count $n_A \leftarrow \sum_{k=1}^{n_{large}} \mathbb{I}[F_\theta(x + \eta_k) = c_A]$;
  estimate $\underline{P_A}$ by LowerConfidenceBound(n_{large}, n_A, α) that estimates the interval $[\underline{P_A}, \overline{P_A}]$ where P_A holds;
  $\overline{P_B} \leftarrow 1 - \underline{P_A}$;
  if $\underline{P_A} < 0.5$ **then**
    output abstain certification;
  else if $\underline{P_A} \geq 0.5$ **then**
    calculate the certified radius R_{zero};
    estimate gradient of the smoothed classifier in the champion class dimension c_A on x: $\nabla F^{c_A}_{smooth}(x)$;
    calculate the magnitude of the gradient;
    if $||\nabla F^{c_A}_{smooth}(x)||_p \geq \varphi(R_{zero})$ **then**
      certified radius $R = R_{zero}$;
    else if $||\nabla F^{c_A}(x)||_p < \varphi(R_{zero})$ **then**
      certified radius $R = R_{first}$ calculated according to Eq. (4-4);
    end if
    output R for test sample x.
  end if
end for

of the certified region centered at x and admits non-isotropic certified radius bounds. As the gradient direction reflects the area where the prediction confidence $F^{c_A}_{smooth}$ is higher than at the current data point x, moving the center x of the certified region along the gradient direction and exploring a larger radius is conducive to further expanding the original certified region. Take a l_2 robust radius as an example. Refer to[81], we obtain the final certified radius R by solving the system of simultaneous equations shown in Eq. (4-4). Specifically, our goal is to reduce the length of the interval $[R_{low}, R_{high}]$ with an initial value of $[R_{low_0} = R_{zero},\ R_{high_0} = \varphi(\frac{(1+\underline{P_A})}{2})]$ by binary searching, where φ is the PDF of the smoothing distribution. To this end, we continuously increase R_{low}, reduce R_{high}, and take the $r = \frac{(R_{low}+R_{high})}{2}$ as the certified radius which matches the requirement in Eq. (4-4), where

z_1 and z_2 are fixed. The search stops when $R_{high} - R_{low} \leq 0$.

$$\begin{aligned} \Phi(z_1 - R) - \Phi(z_2 - R) &= 0.5 \\ \Phi(z_1) - \Phi(z_2) &\leq F_{smooth}(x) = P_A \\ \varphi(z_2) - \varphi(z_1) &\geq \sigma ||\nabla F^{c_A}_{smooth}(x)||_2 \end{aligned} \tag{4-4}$$

In this way, we achieve a real-value overall certified radius R using the multi-order information-based randomized smoothing, which provides an equal certified size for all dimensions.

4.3.3 Dimensional Radius Weight Calculation

To calculate the certified radius vector $(R_1, ..., R_d)$ for an input $x = (x_1, ..., x_d)$ with heterogeneous features while considering correlations between feature dimensions, we weigh the overall certified radius R previously achieved based on the robustness contribution of each feature dimension. The dimension-wise certified radius is then given by $R_i = w_i \times R$, where w_i represents the weight of the contribution for each feature dimension. The certified radius weight w_i is obtained through two steps: Dimensional Feature Sensitivity Analysis and Dimensional Radius Contribution Quantification. Algorithm 4.2 details the calculation.

Step 1: Dimensional Feature Sensitivity Analysis. In this step, we quantify the sensitivity of each dimension of the input feature vector x to the prediction score on the output class. For all dimensions, the more sensitive features are more likely to change the output results. Therefore, sensitive features are also important features for NID. We calculate a sensitivity score s_i for each dimension of the input sample x belonging to class c according to $s_i = \frac{d(f^c_\theta(x))}{d(x_i)}$, where i denotes the i-th dimension. Since our goal is to obtain a sensitivity score vector $s = (s_1, ..., s_d)$ corresponding to a specific category, we average the sensitivity scores of all samples belonging to the same category and denote the result as $\bar{s} = (\bar{s_1}, ..., \bar{s_d})$.

Step 2: Dimensional Radius Contribution Quantification. In this step, we convert the average feature sensitivity score $\bar{s}$ into the contribution of the robustness of each dimension to the overall certified radius R of x, thereby proportionally allocating the overall certified radius to each dimension of the input vector. We first normalize the sensitivity score vector $\bar{s}$ to $\tilde{s} = (\tilde{s_1}, ..., \tilde{s_d}) = (\frac{e^{\bar{s_1}}}{\sum_{i=1}^{d} e^{\bar{s_i}}}, ..., \frac{e^{\bar{s_d}}}{\sum_{i=1}^{d} e^{\bar{s_i}}})$ whose components sum to 1. Then, dimensional robust radius contribution weight w_i is calculated according to Eq. (4-5).

$$R_i = w_i \times R, w_i = \frac{R_i}{R} = \frac{1}{d\tilde{s_i}}, \tag{4-5}$$

where d is the number of dimensions in the input feature vector x, $\frac{1}{d}$ and R respectively de-

Algorithm 4.2 Dimensional Radius Weight Calculation

Input: d-dimensional training samples $(x, y) \in D_{train}$, base C-class classifier F with the classification function f_θ, smoothed classifier F_{smooth}.

Output: d-dim weight vector w_c for each class $c \in \{1, .., C\}$

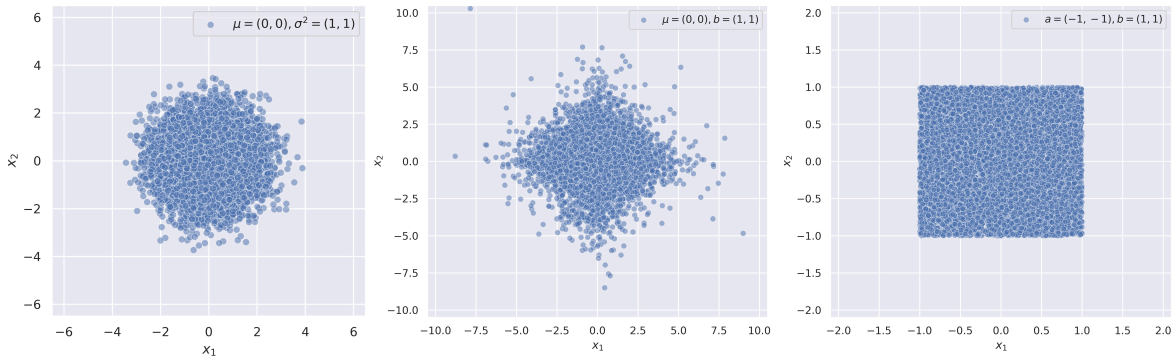
for $j = 1$ to $len(D_{train})$ **do**
  load one training example x_j, y_j;
  for $c = 1$ to C **do**
    $n_c \leftarrow 0$
    class c-specific dataset $D_{train,c} \leftarrow \emptyset$
    if $y_j = c$ **then**
      add (x_j, y_j) to $D_{train,c}$
      $n_c \leftarrow n_c + 1$
      sensitivity score $s_j = ({s_j}^1, {s_j}^2, ..., {s_j}^d) \leftarrow \frac{d(f_\theta^c(x_j))}{d(x_j)}$
    end if
  end for
end for
radius weight vector set $D_{weight} \leftarrow \emptyset$
for $c = 1$ to C **do**
  class c-specific average sensitivity score $\bar{s_c} = (\bar{s_c}^1, \bar{s_c}^2, ..., \bar{s_c}^d) \leftarrow \frac{\sum_{j=1}^{n_c} s_j}{n_c}$
  class c-specific unit sensitivity score $\tilde{s_c} \leftarrow (\frac{e^{\bar{s_c}^1}}{\sum_{k=1}^{d} e^{\bar{s_c}^k}}, \frac{e^{\bar{s_c}^2}}{\sum_{k=1}^{d} e^{\bar{s_c}^k}}, ..., \frac{e^{\bar{s_c}^d}}{\sum_{k=1}^{d} e^{\bar{s_c}^k}})$
  class c-specific radius weight $w_c \leftarrow \frac{1}{d\tilde{s_i}}$
  add (w_c, c) to D_{weight}
end for

note the normalized sensitivity of a single dimension and the overall certified radius when assuming equal sensitivity across dimensions. Sensitivity and robustness proportions generally have an inverse relationship: higher sensitivity correlates with lower robustness.

4.3.4 Smoothing Distribution Alignment

To provide guarantee calculations for robust regions under various types of l_p norm measures, selecting the appropriate distribution whose sampling region aligns with the l_p-bounded robust region is essential. We explore the question of which probability distribution yields the most random noises into the corresponding l_p-measured certificate region. After simulating the sampling areas of various probability distributions, the area formed by $10,000$ noise samples sampled from various distributions can be seen in Figure 4.4.

In the l_2-bounded robust region, noised samples from a Gaussian distribution form areas resembling a circle in 2D feature space, aligning with the l_2 norm. In the l_1-bounded region, samples from a Laplacian distribution form diamond-shaped areas, indicating higher central



(a) Gaussian $\mathcal{N}(\mu, \sigma)$ (b) Laplacian $\mathcal{L}(\mu, b)$ (c) Uniform $\mathcal{U}(a, b)$

Figure 4.4 Smoothing Distribution Sampling Area.

$10,000$ noised samples drawn from different smoothing distributions respectively. (a) Gaussian distribution aligns the l_2-bounded region. (b) Laplacian distribution aligns the l_1-bounded region. (c) Uniform distribution aligns the l_∞-bounded region.

probability density and consistent with the l_1 norm. In the l_∞-bounded region, samples from a Uniform distribution take on shapes akin to a square, catering to extreme variations and matching the l_∞ norm.

4.4 Experimental Setup

4.4.1 Testbed

We implemented the method using PyTorch 2.0.1 and SciPy V1.11.2[140]. Each experiment ran three times with varied random seeds on an NVIDIA GeForce 3090 GPU with CUDA V11.7, and the averages were shown. The code for `MARS` has been open-sourced at https://github.com/CertNID/MARS.

4.4.2 Model Architectures

We evaluated two SOTA DNN-based network intrusion detectors (NIDs), CADE[6] and ACID[5], with different architectures and specialties to evaluate the certified defense performance of MARS and other methods.

Contrastive Autoencoder for Drifting detection and Explanation (CADE). CADE is a concept drift model trained on $n-1$ classes and tested on n classes to detect unknown samples. It employs an encoder-decoder model architecture with a monitoring system to analyze the relationship between input data and training data. It is well-suited for scenarios where the nature and attributes of observed samples may change compared to the knowledge acquired during training.

ACID (Adaptive Clustering-based Intrusion Detection). ACID is a multi-classification

NID model that integrates unsupervised and supervised learning. It identifies input from n categories through the learning of n classes during the training phase. ACID's strength lies in its utilization of traffic features obtained via a clustering-based representation learning approach. This enables the expression of more comprehensive sample information compared to manually defined traffic features, particularly for high-dimensional data.

4.4.3 Datasets

Following and extending the dataset settings in the leading work BARS[82], we evaluated the performance of MARS using two sub-datasets: CSE-CIC-IDS-2018-CADE and CSE-CIC-IDS-2018-ACID (see Table 4.1), both derived from the CSE-CIC-IDS-2018 dataset[141]. For the construction of each sub-dataset, we not only preprocessed the data in the original CSE-CIC-IDS-2018 dataset to filter out duplicate samples, samples with invalid timestamps, and samples with infinite values, but also *digitized* and *normalized* the raw feature values, including one-hot encoded categorical features. This process mapped the network traffic vector, which includes both discrete and continuous features, into a space that can be measured by the l_p norm. For each dataset, the split ratio of training samples and test samples is $8:2$.

Table 4.1 Information on network intrusion detection datasets used for evaluation.

Dataset	CSE-CIC-IDS-2018-CADE				CSE-CIC-IDS-2018-ACID			
	DoS-Hulk-Drift Dataset		Infiltration-Drift Dataset		Diverse-Intrusions Dataset		Similar-Intrusions Dataset	
	Class	Number	Class	Number	Class	Number	Class	Number
Training	Benign	52996	Benign	52996	Benign	52996	Benign	52996
	SSH-Bruteforce	9385	SSH-Bruteforce	9385	FTP-Bruteforce	12590	DoS-GoldenEye	26565
	Infiltration	7390	DoS-Hulk	34789	DDoS-HOIC	53476	DoS-SlowHTTPTest	11191
	-	-	-	-	Bot	22584	DDoS-LOIC-HTTP	46095
Test	Benign	13249	Benign	13249	Benign	13249	Benign	13249
	SSH-Bruteforce	2346	SSH-Bruteforce	2346	FTP-Bruteforce	3148	DoS-GoldenEye	6641
	Infiltration	1894	DoS-Hulk	8697	DDoS-HOIC	13369	DoS-SlowHTTPTest	2798
	DoS-Hulk	43486	Infiltration	9327	Bot	5646	DDoS-LOIC-HTTP	11524

CSE-CIC-IDS-2018-CADE: For CADE, we used samples in the CSE-CIC-IDS-2018 dataset belonging to the Benign class and 3 malicious categories (including SSH-Bruteforce, DoS-Hulk, and Infiltration). Specifically, we used one day's traffic of Benign (02/14), SSH-Bruteforce (02/14), DoS-Hulk (02/16), and Infiltration (03/01) to populate the dataset. Due to the amount of samples in the original dataset, for each class, we collected 10% of the samples in the original dataset. Since CADE is a NID that supports concept drift detection, the test set must contain at least one unseen category in addition to the sample categories that the CADE model has seen during the training phase. To this end, according to different offset

categories, the CSE-CIC-IDS-2018-CADE dataset is further divided into the DoS-Hulk-Drift dataset and the Infiltration-Drift dataset. In DoS-Hulk-Drift, DoS-Hulk appears only in the test set. In Infiltration-Drift, Infiltration appears exclusively in the test set.

CSE-CIC-IDS-2018-ACID: For ACID, we used samples in the CSE-CIC-IDS-2018 dataset belonging to the Benign class and 6 malicious classes, including FTP-Bruteforce, DDoS-HOIC, Bot, DoS-GoldenEye, DoS-SlowHTTPTest, and DDoS-attacks-LOIC-HTTP. These data were divided into two datasets: Diverse-Intrusions dataset with diverse intrusion types, and Similar-Intrusions dataset with similar intrusion types (See Table 4.1), used for conventional and fine-grained multi-class detection evaluations, respectively. Specifically, we used one day's traffic of Benign (02/14), FTP-Bruteforce (02/14), Bot (03/02), and DDoS-HOIC (02/21) to populate the Diverse-Intrusions dataset. For the FTP-Bruteforce class, we collected 40% of the samples in the original dataset. For each other class, we collected 10% of the samples in the original dataset. For the Similar-Intrusions dataset, we have Benign (02/14), DoS-GoldenEye (02/15), DoS-SlowHTTPTest (02/16), and DDoS-attacks-LOIC-HTTP (02/20). For the DoS-GoldenEye class, we used 80% of the samples in the original dataset. For each other class, we collected 10% in the original dataset.

One-hot Encoding of Categorial Features. For the categorial feature dimensions, we have one-hot encoded them. Destination Port features have been encoded to 0, 1, and 2, with 0 for the low-frequency port (< 1000), 1 for medium ($1000 \sim 10000$), and 2 for high (> 10000). For the Protocol feature, we encoded '0' to 0, '6-TCP' to 1, and '17-UDP' to 2. Note that 0, 1, and 2 means index where the 1 occupies rather than a number. For example, 0 for $[1, 0, 0]$. Furthermore, timestamp features were converted to UNIX seconds format before normalization and MinMax scaling.

4.4.4 Attack Configuration

Parameters in Adversarial Attack. We use two types of white-box adversarial attacks: Projected Gradient Descent (PGD)[14] and Elastic-Net Attack to DNN (EAD)[15] For l_2-PGD, l_1-PGD, and l_1-EAD attacks, the perturbation budget ϵ is set to 1.0, allowing for a maximum adversarial perturbation that can significantly alter the input. Additionally, the per-step perturbation budget ϵ_s is configured to 0.75, which restricts the perturbation at each iteration to ensure controlled modifications of the input. For the l_∞-PGD attack, the parameters are adjusted to accommodate its unique characteristics: the perturbation budget ϵ is set to 0.2, and the per-step perturbation budget ϵ_s is limited to 0.1. l_p-PGD is typically most powerful under the L_∞ norm constraint, so the perturbation budget required to achieve evasion is smaller.

In all cases, the maximum number of iterations, $N_{\text{iteration}}$, is uniformly set to 20, providing a consistent framework for evaluating the effectiveness of each attack method.

Parameters in Natural Corruption Perturbable features under two natural corruption threats are shown in Figure 4.5.

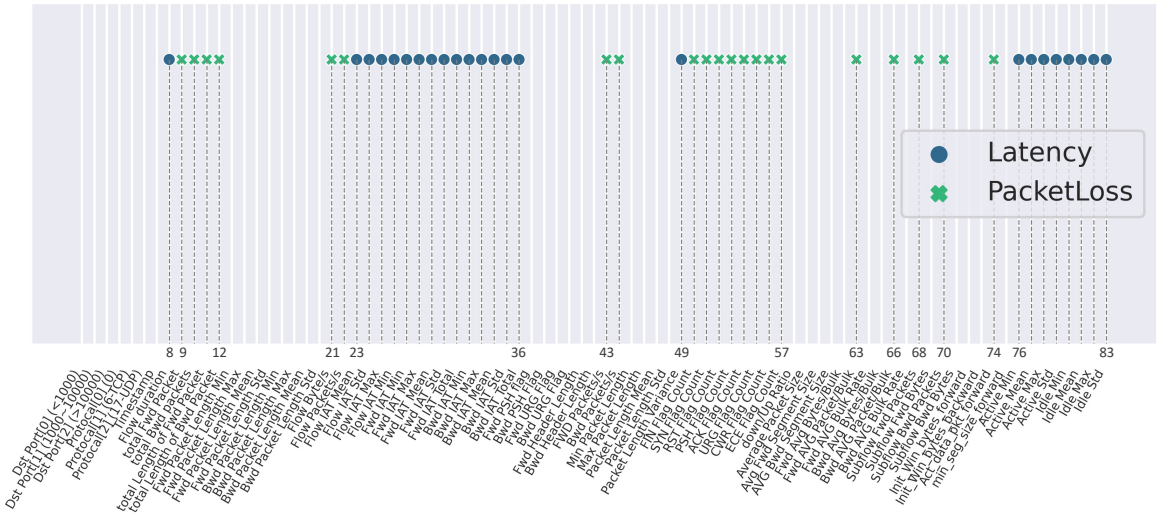


Figure 4.5 Features perturbed under natural corruptions.

Latency. We used a Gaussian distribution with a standard deviation of 1 and a mean of 0 to add random noise to the time-related traffic features, such as the time between two flows, time between two packets, time a flow was active before becoming idle, time a flow was idle before becoming active, etc.

PacketLoss. We used a Gaussian distribution with a standard deviation of 1 and a mean of 0 to add random noise to packet quantity-related features and partial length-related features correlated with the packet number, such as the number of total packets, number of packets transferred per second, number of packets bulk rate, number of packets in a sub-flow, etc.

4.4.5 Defense Configuration

To ensure a fair comparison, we selected three SOTA robustness certification methods known for their good architecture-level scalability as certified defense baselines. These methods rely on randomized smoothing, allowing for applicability across various NID structures. Table 4.2 illustrates their properties.

Vanilla Randomized Smoothing (VRS). VRS[79] is the first randomized smoothing-based robustness certification technique designed for image classification. It uses random Gaussian noise and the Monte Carlo sampling method to obtain the l_2-measured radius only through the zero-order information.

First Order-based Randomized Smoothing (FRS). Since zero order-based methods

Table 4.2 Comparison of certified defense methods

Method	Heterogeneity	Universality	Robustness Guarantee Diversity			Adversarial Attacks			Natural Corruptions	
			l_2 Radius	l_1 Radius	l_∞ Radius	l_2 Attack	l_1 Attack	l_∞ Attack	Latency	Loss
VRS[79]	○	●	●	○	○	○	○	○	○	○
FRS[81]	○	●	●	●	●	○	○	○	○	○
BARS[82]	●	●	●	○	○	○	○	●	○	○
MARS	●	●	●	●	●	●	●	●	●	●

so far still have some gaps between the actual robust radius and the available certified radius, FRS[81] (2020) was proposed to provide certified robustness for image classifiers by using the first-order gradient of the smoothed classifier, which further tightens the l_p radius.

Boundary-Adaptive Randomized Smoothing (BARS). BARS[82] is the only existing NID robustness certification framework, which is built on top of VRS. It focuses on zero-order information and adapts to the network traffic features with dimension-wise optimized smoothing distribution but only provides l_2 robustness guarantees.

4.4.6 Evaluation Metrics

We evaluate certified robustness against any potential perturbated inputs, empirical robustness against specific adversarial inputs and corrupted inputs, and regular predictive performance on clean inputs using various metrics.

Certified Robustness. We use Mean Certified Radius and Certified Accuracy to evaluate the certified robustness of the model.

Mean Certified Radius (MCR). The average certified radius is calculated on a per-class basis, as shown in Eq. (4-6):

$$MCR = \frac{\sum_{i=1}^{N} R_i}{N}, \tag{4-6}$$

where N represents the total number of test samples belonging to the same class, R_i denotes the certified radius for each test sample. We count MCR based on all samples in the same class to observe the certified robustness of the classifier across different categories. This approach allows us to evaluate the model's certified robustness against targeted adversarial attacks on different classes. A larger value of MCR indicates a tighter lower bound for the robust radius, indicating that the model can maintain its performance with a greater margin of safety when faced with adversarial perturbations.

Certified Accuracy. Given a certified radius threshold R_{given}, the certified accuracy measures the proportion of test samples that are correctly predicted by the certified defended

classifier F_{smooth} with a certified radius R greater than R_{given}, as shown in Eq. (4-7).

$$Certified\ Accuracy = \frac{N_{(F_{smooth}(x)=y_{true})\&(R\geq R_{given})}}{N_{TotalCertTest}} \tag{4-7}$$

The numerator, $N_{(F_{smooth}(x)=y_{true})\&(R\geq R_{given})}$, counts only those correctly predicted samples that also satisfy the condition $R \geq R_{given}$. The denominator, $N_{TotalCertTest}$, indicates the total number of test samples for certification evaluation. A higher certified accuracy reflects a greater number of samples passing the robustness certification under the specified threshold R_{given}. By tracking certified accuracy, we can assess the effectiveness of a DNN model in maintaining correct predictions while satisfying certified robustness.

Empirical Robustness. We use Robust Accuracy to assess the model's empirical robustness against adversarial attacks and natural corruptions. For natural corruptions, we also evaluate metrics such as Recall, Precision, F1-score, False Positive Rate (FPR), and False Negative Rate (FNR) on the corrupted test data.

Robust Accuracy. Robust Accuracy reflects the proportion of perturbed test samples (e.g., adversarial examples or corrupted examples) that the model predicts correctly in all perturbed test samples, as expressed in Eq. (4-8):

$$Robust\ Accuracy = \frac{N_{(F_{smooth}(x^*)=y_{true})}}{N_{TotalPertTest}} = \frac{TP+TN}{TP+TN+FP+FN} \tag{4-8}$$

where, $N_{(F_{smooth}(x^*)=y_{true})}$ denotes the number of perturbed samples correctly predicted, and $N_{TotalPertTest}$ denotes the total number of perturbed test samples for evaluation. Specifically, TP is True Positives (malicious samples correctly classified), TN is True Negatives (benign samples correctly classified), FP is False Positives (benign samples incorrectly classified), and FN is False Negatives (malicious samples incorrectly classified). We use Robust Accuracy to evaluate the empirical robustness of NIDs on four specifical adversarial attacks (l_2-PGD, l_∞-PGD, l_1-PGD, l_1-EAD) and two natural corruptions (PacketLoss and Latency).

When evaluating the robust accuracy on adversarial examples, Robust Accuracy calculated by Eq. (4-9) is equal to the Recall calculated by Eq. (4-10) because the adversarial test set contains only adversarial malicious (AdvMal) traffic (described in Section 4.2.1), so that TN and FP are always 0.

$$Robust\ Accuracy\ (on\ adversarial\ examples) = \frac{N_{(F_{smooth}(x^*)=y_{true})}}{N_{TotalAdvMalTest}} \tag{4-9}$$

$$Recall = \frac{TP}{TP+FN} \tag{4-10}$$

When evaluating the robust accuracy on natural corruption examples, since the cor-

rupted test set contains corrupted benign and corrupted malicious traffic (described in Section 4.2.1), robust accuracy is calculated as the ratio of correctly predicted samples among all test perturbated samples, as expressed in Eq. (4-8).

Precision. It represents the proportion of TP among all positive predictions, given by Eq. (4-11):

$$Precision = \frac{TP}{TP + FP}. \tag{4-11}$$

F1-score. It is the harmonic mean of Precision and Recall, balancing the two to provide a single metric for performance when both FP and FN are important. It is calculated as Eq. (4-12):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{4-12}$$

False Positive Rate (FPR). It is also known as False Alarm Rate, measures the proportion of negative test samples (such as benign) that are incorrectly classified as positive samples (such as malicious) in all negative test samples, calculated as Eq. (4-13):

$$FPR = \frac{FP}{FP + TN}. \tag{4-13}$$

False Negative Rate (FNR). It quantifies the proportion of positive test samples that are incorrectly classified as negative in all positive test samples, defined as Eq. (4-14):

$$FNR = \frac{FN}{FN + TP}. \tag{4-14}$$

Regular Performance We use Clean Accuracy to evaluate the regular predictive performance of the model on clean data without perturbation. We also evaluate metrics such as Recall, Precision, F1-score, FPR, and FNR on the clean test data.

Clean Accuracy. It is the ratio of correctly predicted clean test samples among all clean test samples, as defined in Eq. (4-15):

$$CleanAcc = \frac{N_{(F_{smooth}(x)=y_{true})}}{N_{TotalCleanTest}}. \tag{4-15}$$

4.5 Horizontal Experimental Results and Analysis

In this section, we compare the performance of `MARS` with state-of-the-art (SOTA) certified defenses based on randomized smoothing (RS) across several aspects. These include comparisons of the tightness of l_2 certified robustness guarantees (Section 4.5.1), l_1 and l_∞ robustness guarantees tightness (Section 4.5.2), empirical robustness against adversarial at-

tacks (Section 4.5.3), empirical robustness against natural corruptions (Section 4.5.4), and certified and empirical robustness in fine-grained intrusion detection (Section 4.5.5).

4.5.1 Comparison of l_2 Robustness Guarantee with SOTA RS Methods

We first compare the tightness of the l_2 robustness guarantees provided by MARS with VRS[79] and FRS[81] for the image domain, and BARS[82] for the network traffic domain. For a fair comparison, we used the same dataset and settings as BARS. We calculated the MCR and certified accuracy (defined in Section 4.4.6) for each category on ACID and CADE.

Setup. To be comparable with VRS and FRS, which do not consider dimension-wise radius, the object we compare is the overall certified radius R of the smoothed model. For the smoothed classifier, n_{small} and n_{large} are set to 100 and $10,000$. The parameters for the smoothed classifier are: n_{small} =100, n_{large} =10,000. The learning rate for optimizing the noise shape is set to 0.01. The maximum number of training epochs is limited to 10. The failure probability α for radius calculation is set to 0.001. To be consistent with the evaluation setup in BARS, MCR and certified accuracy are measured by category for each test set.

Results. The results of MCR (see Figure 4.6) and certified accuracy (see Figure 4.7) show that MARS always outperforms the SOTA randomized smoothing methods. Especially on the CADE-Infiltration-Drift dataset and the CSE-CIC-IDS-2018-ACID dataset, MARS shows significant advantages over BARS when both VRS and FRS failed certification in many categories. Also, for the SSH-Bruteforce category in the CADE-DoS-Hulk-Drift dataset, we observe that the certified radius obtained by all methods is always zero, which indicates that CADE itself is very sensitive and vulnerable to the SSH-Bruteforce attack in the CADE-DoS-Hulk-Drift dataset, leading to failure to certify.

Detailed MCR results are available in Table 4.3, which shows the certified robustness performance measured by the mean certified radius in each category for two CADE models and an ACID model. A certified radius equal to 0 indicates that the detection model is vulnerable to the specific attack category. Even if the clean sample is perturbated very weakly, that is, the noise budget is very small, it is easy for the model to lose confidence in the prediction results, leading to certification failure, that is, getting a radius equal to 0. As we see from the table, for the CADE models trained with VRS or FRS on the CADE-DoS-Hulk-Drift dataset, SSH-Bruteforce and Infiltration are attacks that are relatively difficult to detect. For the CADE trained on the CADE-Infiltration-Drift dataset, benign is a vulnerable category.

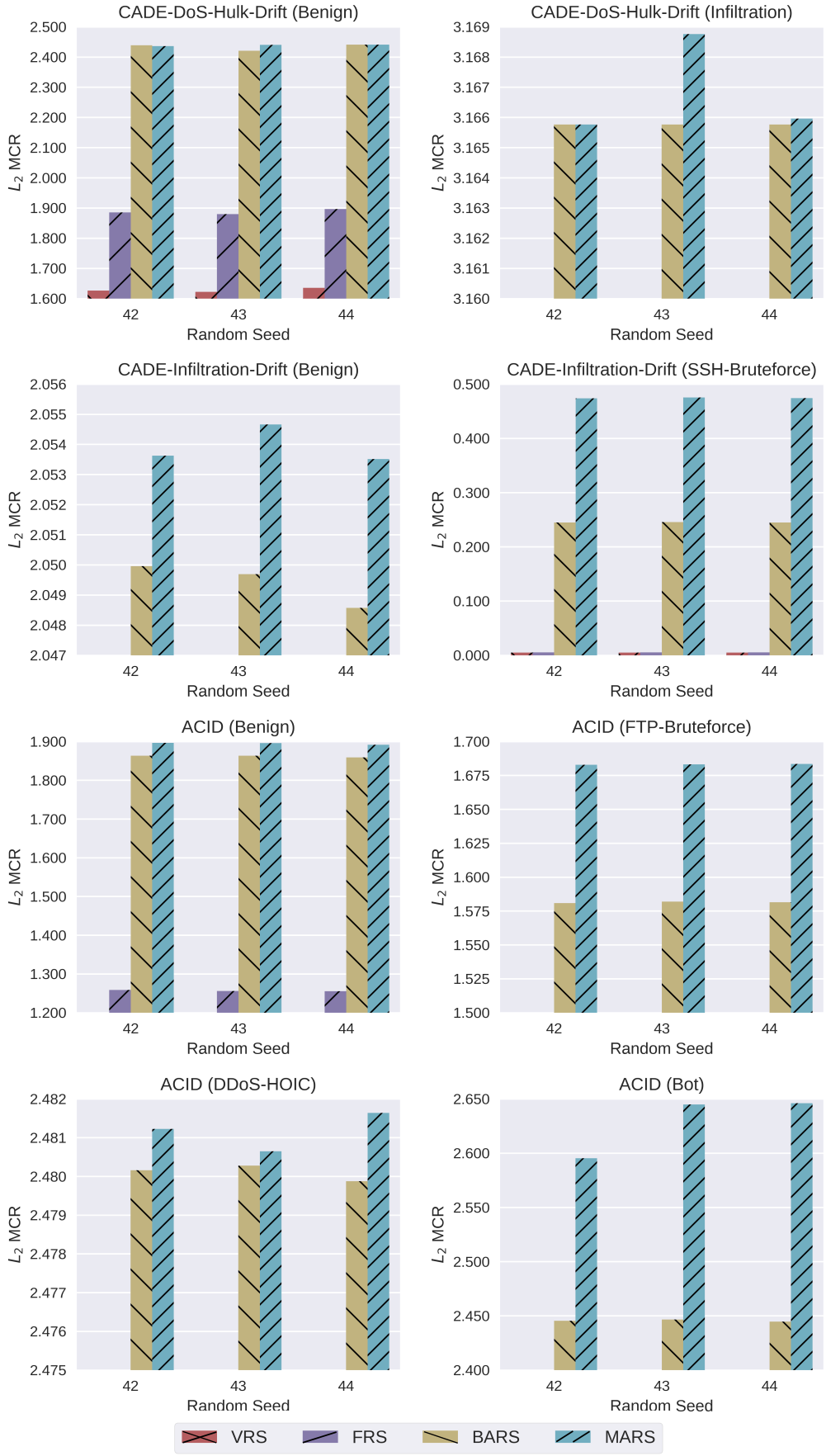


Figure 4.6 Comparison of l_2 Mean Certified Radius (MCR).

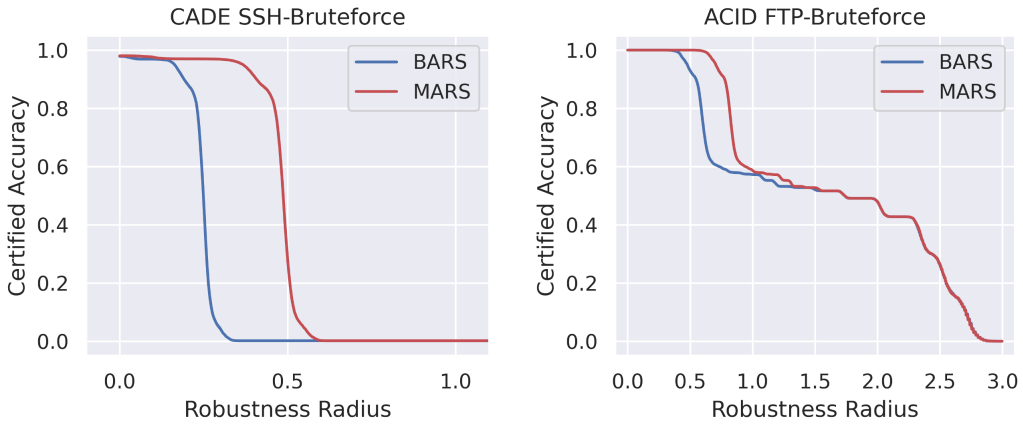


Figure 4.7 Comparison of certified accuracy of l_2 robustness guarantee.

For the ACID, the detection of FTP-Bruteforce, DDoS-HOIC, and Bot attacks is unstable. Fortunately, BARS and MARS can not only enhance the certified robustness against those attacks model is originally robust againt, but also increase the certified radius in these vulnerable classes, and the improvement brought by MARS has always been greater than BARS. However, we observed that CADE trained based on the CADE-Infiltration-Drift dataset has reached the consistent conclusion that it is difficult to pass certification in the SSH-Bruteforce category with all certification methods. This is due to the inherent vulnerability of the CADE model obtained under this training setting, because CADE trained on another dataset performs normally on the same class.

Table 4.3 Comparison of l_2 Mean Certified Radius (MCR)

Method	Seed	CADE-DoS-Hulk-Drift Dataset			CADE-Infiltration-Drift Dataset			ACID-CIC-IDS-2018 Dataset			
		Benign	SSH-Bruteforce	Infiltration	Benign	SSH-Bruteforce	DoS-Hulk	Benign	FTP-Bruteforce	DDoS-HOIC	Bot
VRS [79]	42	1.6260	0.0000	0.0000	0.0000	0.0049	1.4110	0.0012	0.0000	0.0000	0.0000
	43	1.6219	0.0000	0.0000	0.0000	0.0049	1.4077	0.0012	0.0000	0.0000	0.0000
	44	1.6356	0.0000	0.0000	0.0000	0.0048	1.4060	0.0011	0.0000	0.0000	0.0000
FRS [81]	42	1.8850	0.0000	0.0000	0.0000	0.0054	1.6350	1.2585	0.0000	0.0000	0.0000
	43	1.8796	0.0000	0.0000	0.0000	0.0054	1.6315	1.2558	0.0000	0.0000	0.0000
	44	1.8965	0.0000	0.0000	0.0000	0.0054	1.6283	1.2554	0.0000	0.0000	0.0000
BARS [82]	42	2.4391	0.0000	3.1658	2.0500	0.2448	1.6406	1.8636	1.5808	2.4802	2.4455
	43	2.4204	0.0000	3.1658	2.0497	0.2456	1.6371	1.8636	1.5819	2.4803	2.4465
	44	2.4407	0.0000	3.1658	2.0486	0.2446	1.6340	1.8590	1.5815	2.4799	2.4447
MARS	42	2.4361	0.0000	3.1658	2.0536	0.4738	1.6406	1.8964	1.6828	2.4812	2.5953
	43	2.4404	0.0000	3.1688	2.0547	0.4755	1.6471	1.8958	1.6832	2.4806	2.6450
	44	2.4407	0.0000	3.1660	2.0535	0.4741	1.6340	1.8918	1.6835	2.4816	2.6461

4.5.2 Comparison of l_p Robustness Guarantee with SOTA RS Methods

To assess the tightness of l_p robustness guarantees across different norms, we first compare the sizes of l_2, l_1, and l_∞ certified radii with the leading method FRS [81], since neither

VRS nor BARS supports l_1 and l_∞ robustness certification. FRS incorporates l_2, l_1, and l_∞ guarantees but relies exclusively on standard Gaussian distribution for smoothing. For a fair comparison, we compare `MARS` with FRS using Gaussian smoothing distribution across all l_p norms, without distribution alignment.

Results. The results of MCR (see Figure 4.8) show that `MARS` consistently provides tighter l_p robustness guarantees compared to FRS. Especially when FRS fails certification on many classes (with radius 0) due to its nature of smoothing all traffic feature dimensions indiscriminately, `MARS` still outputs non-trivial l_2, l_1 and l_∞ radii. Furthermore, the results of certified accuracy (see Figure 4.9) show that l_2 and l_1 radii are close, but the l_∞ radius remains the smallest, as l_∞ is the most difficult to capture by the Gaussian distribution.

Detailed MCR results are available in Table 4.4. We can see that if there is a lack of customized certification design for network traffic data, even the FRS that also uses first-order information will have a hard time effectively certifying the robust radius of the attack samples for the NIDs. Meanwhile, we also observed that compared with l_2 and l_1 certification, it is more difficult to obtain a tight l_∞ robustness guarantee, that is, a large l_∞ certified radius. This is reasonable because under the same budget $||\delta||_p < R$, the l_∞-measured certified region will be larger in volume than l_2 and l_1, thus it is easier for samples to evade detection.

4.5.3 Comparison of Empirical Robustness against Different l_p Adversarial Attacks with SOTA RS Methods

Considering the diversity of adversary changes, unlike BARS which only tests against l_∞ adversarial attacks, we use various l_p norms to enrich the threat model. We employ Projected Gradient Descent (PGD)[14] and Elastic-Net Attack to DNN (EAD)[15] as threat models of white-box adversarial attacks to generate adversarial examples $x^* = x + \delta$.

Setup. Following the BARS settings, we selected one of the malicious categories, Bot, as a representative to test whether the ACID model with certified defense can correctly identify adversarial Bot samples, and calculated the robust accuracy and clean accuracy (defined in Section 4.4.6). For l_2-PGD, l_1-PGD, and l_1-EAD, perturbation budget ϵ that determines the maximum adversarial perturbation is set to 1.0 and per-step perturbation budget ϵ_s that

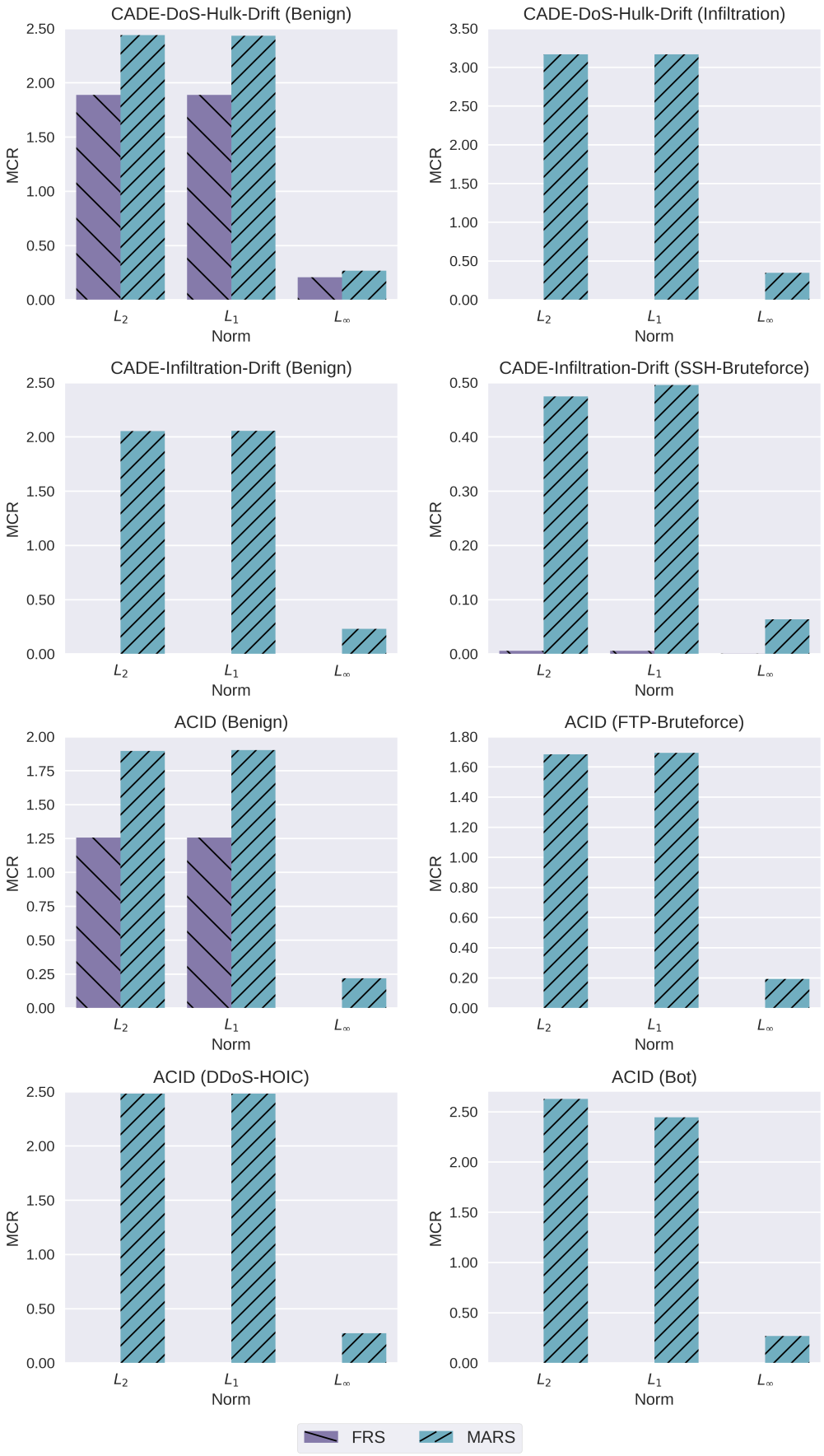


Figure 4.8 Comparison of l_p Mean Certified Radius (MCR) under the same smoothing distribution.

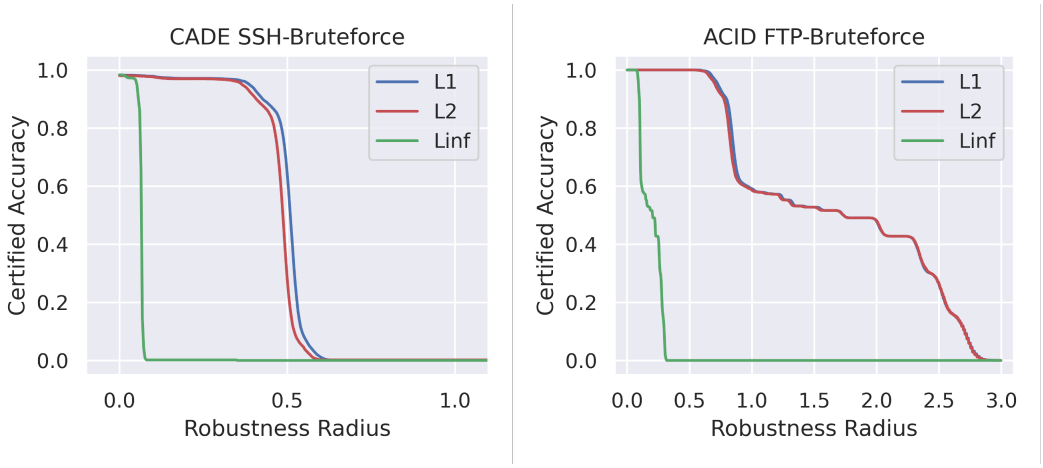


Figure 4.9 Comparison of l_p certified accuracy under the same smoothing distribution.

Table 4.4 Comparison of l_p Mean Certified Radius (MCR) under the same smoothing distribution.

Norm	Method	Seed	CADE-DoS-Hulk-Drift Dataset			CADE-Infiltration-Drift Dataset			ACID-CIC-IDS-2018 Dataset			
			Benign	SSH-Bruteforce	Infiltration	Benign	SSH-Bruteforce	DoS-Hulk	Benign	FTP-Bruteforce	DDoS-HOIC	Bot
l_2	FRS [81]	42	1.8850	0.0000	0.0000	0.0000	0.0054	1.6350	1.2585	0.0000	0.0000	0.0000
		43	1.8796	0.0000	0.0000	0.0000	0.0054	1.6315	1.2558	0.0000	0.0000	0.0000
		44	1.8965	0.0000	0.0000	0.0000	0.0054	1.6283	1.2554	0.0000	0.0000	0.0000
	MARS	42	2.4361	0.0000	3.1658	2.0536	0.4738	1.6406	1.8964	1.6828	2.4812	2.5953
		43	2.4404	0.0000	3.1688	2.0547	0.4755	1.6471	1.8958	1.6832	2.4806	2.6450
		44	2.4407	0.0000	3.1660	2.0535	0.4741	1.6340	1.8918	1.6835	2.4816	2.6461
l_1	FRS [81]	42	1.8850	0.0000	0.0000	0.0000	0.0054	1.6350	1.2585	0.0000	0.0000	0.0000
		43	1.8796	0.0000	0.0000	0.0000	0.0054	1.6315	1.2558	0.0000	0.0000	0.0000
		44	1.8965	0.0000	0.0000	0.0000	0.0054	1.6283	1.2554	0.0000	0.0000	0.0000
	MARS	42	2.4391	0.0000	3.1658	2.0556	0.4950	1.6406	1.9028	1.6933	2.4812	2.4453
		43	2.4204	0.0000	3.1658	2.0567	0.4968	1.6371	1.9028	1.6936	2.4806	2.4450
		44	2.4407	0.0000	3.1658	2.0556	0.4953	1.6340	1.8990	1.6939	2.4816	2.4461
l_∞	FRS [81]	42	0.2069	0.0000	0.0000	0.0000	0.0006	0.1795	0.0000	0.0000	0.0000	0.0000
		43	0.2063	0.0000	0.0000	0.0000	0.0006	0.1791	0.0000	0.0000	0.0000	0.0000
		44	0.2082	0.0000	0.0000	0.0000	0.0006	0.1787	0.0000	0.0000	0.0000	0.0000
	MARS	42	0.2663	0.0000	0.3475	0.2299	0.0633	0.1801	0.2182	0.1920	0.2723	0.2684
		43	0.2657	0.0000	0.3475	0.2300	0.0635	0.1797	0.2181	0.1920	0.2723	0.2684
		44	0.2679	0.0000	0.3475	0.2299	0.0633	0.1794	0.2177	0.1920	0.2724	0.2685

determines the maximum allowed perturbation at each iteration is set to 0.75. For l_∞-PGD, ϵ is 0.2 and ϵ_s is 0.1. The maximum number of iterations $N_{iteration}$ is set to 20 for all attacks. Reported averages are based on results from random seeds (42, 43, 44).

Results. Since we only measure the Robust Accuracy of the model against adversarial malicious examples, robust accuracy here is equivalent to the Recall. The results of Robust Accuracy on adversarial examples (see Table 4.5) show that compared to BARS, our defense boosts the robust accuracy against adversarial attacks by 1.70% for l_2-PGD, 7.17% for l_∞-PGD, and 10.11% for l_1-EAD. Compared to the base NID without any certified defense (noted as Vanilla), robust accuracy increases by 13.79% for l_2-PGD, 33.94% for l_∞-PGD,

and 10.01% for l_1-EAD. Notably, we also observe that the base ACID detector itself is already very robust to l_1-PGD attacks, and both BARS and MARS preserve this robustness. Thus, we tested the more powerful l_1-EAD, essentially a linear mixture of l_1 and l_2 penalty functions. Neither Vanilla nor BARS can resist l_1-EAD, and only MARS enhances model robustness against it. Also, it can be seen from the table that both BARS and MARS well retain the detection accuracy of the vanilla ACID model on clean Bot samples, reaching 100%.

Table 4.5 Comparison of empirical robustness of ACID against different adversarial attacks

Metric		Clean Accuracy	Robust Accuracy/Recall on Adversarial Examples			
Method	Seed	Clean	l_2-PGD	l_∞-PGD	l_1-PGD	l_1-EAD
Vanilla	42	1.0000	0.8395	0.5501	1.0000	0.0032
	43	1.0000	0.8395	0.5502	1.0000	0.0016
	44	1.0000	0.8395	0.5502	1.0000	0.0032
	mean±std	1.0000±0.0000	0.8395±0.0000	0.5502±0.0001	1.0000±0.0000	0.0027±0.0009
BARS [82]	42	1.0000	0.9601	0.8154	1.0000	0.0016
	43	1.0000	0.9601	0.8190	1.0000	0.0017
	44	1.0000	0.9610	0.8190	1.0000	0.0016
	mean±std	1.0000±0.0000	0.9604±0.0005	0.8178±0.0020	1.0000±0.0000	0.0016±0.0001
MARS	42	1.0000	0.9779	0.8925	1.0000	0.1031
	43	1.0000	0.9784	0.8863	1.0000	0.1021
	44	1.0000	0.9759	0.8898	1.0000	0.1031
	mean±std	1.0000±0.0000	**0.9774±0.0013**	**0.8895±0.0031**	1.0000±0.0000	**0.1028±0.0006**

4.5.4 Comparison of Empirical Robustness against Different Natural Corruptions with SOTA RS Methods

Besides adversarial examples, natural corruptions from changes in the cyber environment can also lead to model misclassification. We generate naturally corrupted samples from clean benign and malicious inputs using Latency and PacketLoss, as defined in Section 4.2.1 and implemented in Section 4.4.4. Since natural corruption noise generally does not follow specific patterns, using an empirical distribution would make more sense. In our study, distribution shifts in the feature dimensions related to packet arrival time and packet number are simulated using random noise following a Gaussian distribution with mean 0. We adjust the standard deviation σ in $\{0.5, 1.0, 1.5\}$ to mimic the different corruption strengths.

Results. The results of Robust Accuracy on corrupted samples (see Figure 4.10) show that we can see that MARS consistently outperforms BARS and Vanillia across various corruption intensities and certified classes. Also, under the same corruption strength, both vanilla and certified defended models show higher resilience to PacketLoss than to Latency for benign and malicious traffic. This suggests that ACID is more sensitive to perturbations in time-related features and more robust in quantity-related features.

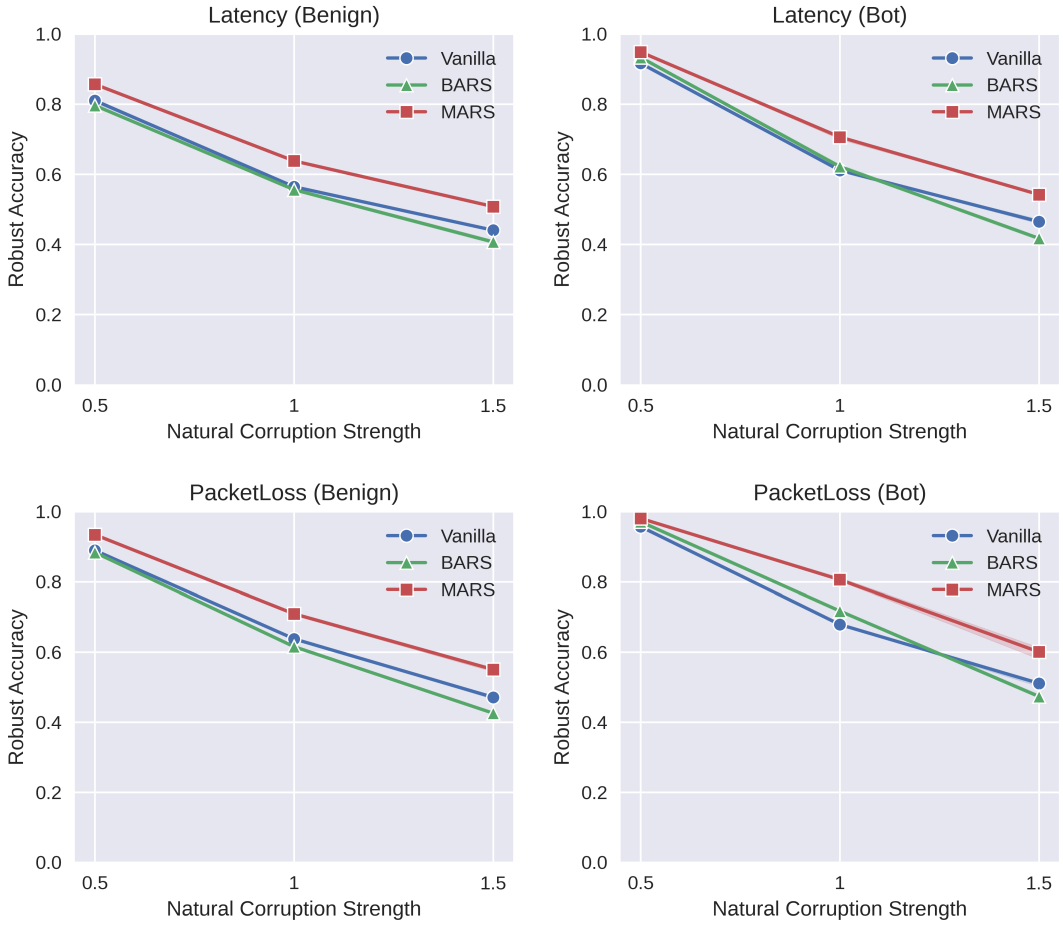


Figure 4.10 Comparison of empirical robustness of ACID against varied natural corruptions.

Table 4.6 Comparison of empirical robustness of ACID against varied natural corruptions

Class		Benign						Bot					
Metric		CleanAcc	Robust Accuracy					CleanAcc	Robust Accuracy				
Method	Seed	Clean	Latency (σ)			PacketLoss (σ)			Clean	Latency (σ)			PacketLoss (σ)	
			0.5	1.0	1.5	0.5	1.0	1.5		0.5	1.0	1.5	0.5	1.0	1.5
Vanilla	42	1.0000	0.8038	0.5643	0.4409	0.8902	0.6377	0.4760	1.0000	0.9152	0.6109	0.4644	0.9541	0.6784	0.5158
	43	1.0000	0.8112	0.5643	0.4391	0.8897	0.6377	0.4656	1.0000	0.9198	0.6108	0.4710	0.9616	0.6782	0.5019
	44	1.0000	0.8153	0.5643	0.4409	0.8881	0.6377	0.4718	1.0000	0.9137	0.6114	0.4589	0.9548	0.6790	0.5151
BARS [82]	42	1.0000	0.7939	0.5608	0.4123	0.8827	0.6202	0.4248	1.0000	0.9306	0.6178	0.4201	0.9722	0.7132	0.4793
	43	1.0000	0.7956	0.5554	0.4019	0.8866	0.6155	0.4257	1.0000	0.9323	0.6194	0.4194	0.9731	0.7212	0.4704
	44	1.0000	0.7986	0.5517	0.4064	0.8806	0.6108	0.4267	1.0000	0.9362	0.6278	0.4118	0.9674	0.7164	0.4702
MARS	42	1.0000	0.8564	0.6354	0.5066	0.9354	0.7139	0.5435	1.0000	0.9495	0.6993	0.5448	0.9803	0.8018	0.5802
	43	1.0000	0.8564	0.6414	0.5042	0.9337	0.7050	0.5525	1.0000	0.9486	0.7095	0.5409	0.9814	0.8118	0.6130
	44	1.0000	0.8569	0.6370	0.5125	0.9328	0.7079	0.5549	1.0000	0.9470	0.7096	0.5400	0.9796	0.8039	0.6080

The results in Table 4.6 show the detailed Robust Accuracy of the ACID against two natural corruptions: Latency and PacketLoss. It can be seen from the consistent results that `MARS` always has the highest robust accuracy for increasing natural disturbances. Compared with the enhanced robustness from `MARS`, BARS weakens the detector's ability to identify natural corruptions.

4.5.5 Comparison of Adversarial Robustness in Fine-grained Intrusion Detection with SOTA RS Methods

Even with similar intrusion purposes, attackers often employ varied techniques and tools. Considering the differences among attackers, we assessed the fine-grained detection performance of MARS on datasets containing similar network intrusion types.

Setup. This evaluation includes various DoS-related attacks to extend performance assessment in fine-grained NID scenarios. Using the Similar-Intrusions Dataset involving benign (defined in Table 4.1), we trained a four-class ACID model with 99% accuracy, 100% precision, and an F1-score of 1 on clean test set in 312 seconds. Various l_p certified radii and robust accuracy were tested in fine-grained intrusion detection.

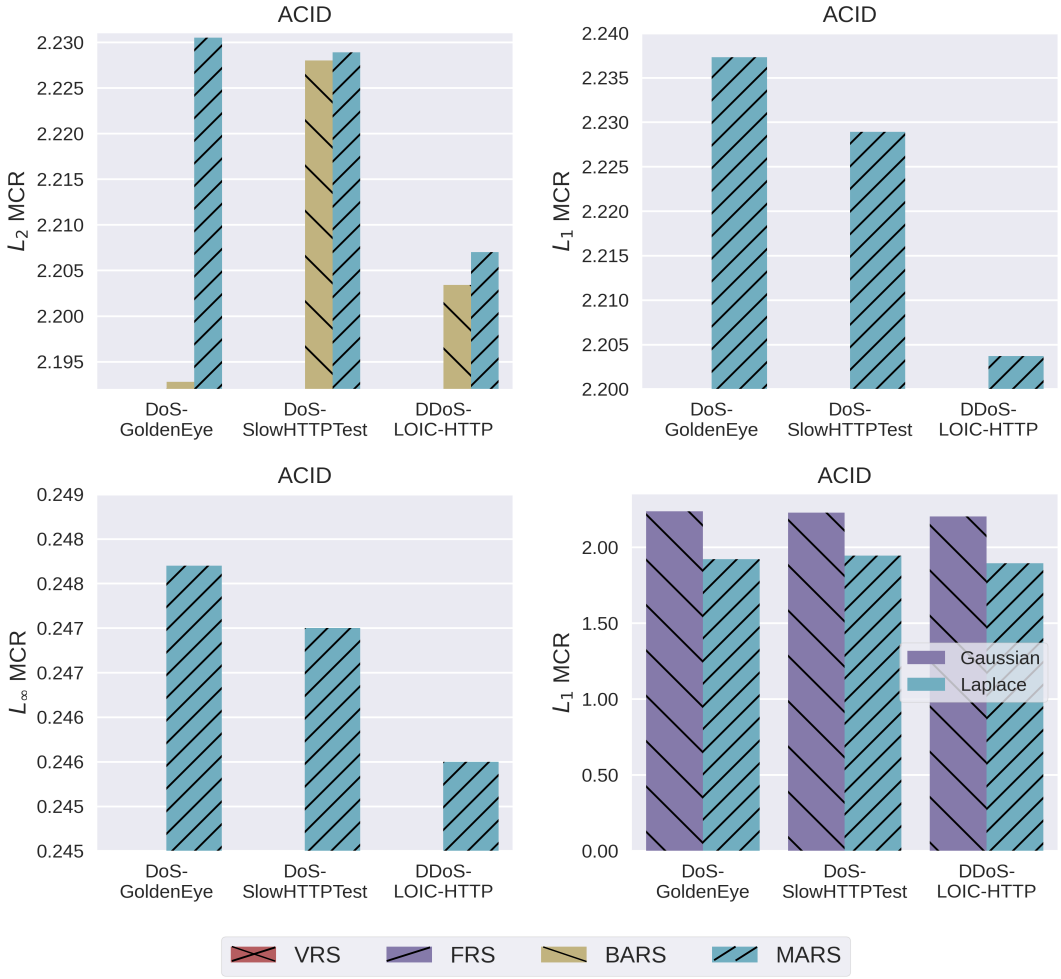


Figure 4.11 Comparison of l_p Mean Certified Radius (MCR) in fine-grained intrusion detection.

Results. We compared l_2, l_1, and l_∞ radius with other randomized smoothing methods. Analogous to the previous cases, MARS achieved the largest certified radii in all three similar intrusions chosen (DoS-GoldenEye, DoS-SlowHTTPTest, and DDoS-attacks-LOIC-HTTP).

Table 4.7 Comparison of l_2 MCR and Average Certification Time (ACT) (per sample/sec) in fine-grained detection of similar intrusions.

Method	Benign	DoS-GoldenEye	DoS-SlowHTTPTest	DDoS-LOIC-HTTP
VRS [79]	0.1950 (0.0028)	0.0001 (0.0039)	0.0000 (0.0053)	0.0000 (0.0033)
FRS [81]	1.2077 (0.0238)	0.0024 (0.0309)	0.0014 (0.0459)	0.0000 (0.0269)
BARS [82]	1.9036 (0.0029)	2.1928 (0.0040)	2.2280 (0.0051)	2.2034 (0.0033)
MARS	**1.9260 (0.0253)**	**2.2305 (0.0331)**	**2.2289 (0.0361)**	**2.2070 (0.0279)**

Table 4.8 Comparison of l_p MCR and Average Certification Time (ACT) (per sample/sec) in fine-grained detection of similar intrusions under the same smoothing distribution.

Norm	Method	Benign	DoS-GoldenEye	DoS-SlowHTTPTest	DDoS-LOIC-HTTP
l_2	FRS [81]	1.2077 (0.0238)	0.0024 (0.0309)	0.0014 (0.0459)	0.0000 (0.0269)
	MARS	**1.9260 (0.0253)**	**2.2305 (0.0331)**	**2.2289 (0.0361)**	**2.2070 (0.0279)**
l_1	FRS [81]	1.2077 (0.0236)	0.0000 (0.0311)	0.0000 (0.0365)	0.0000 (0.0272)
	MARS	**1.9317 (0.0256)**	**2.2373 (0.0334)**	**2.2289 (0.0378)**	**2.2037 (0.0283)**
l_∞	FRS [81]	0.0000 (0.0240)	0.0000 (0.0321)	0.0000 (0.0377)	0.0000 (0.0279)
	MARS	**0.2196 (0.0263)**	**0.2482 (0.0336)**	**0.2475 (0.0383)**	**0.2460 (0.0293)**

The results of MCR (see Figure 4.11, Table 4.7,and Table 4.8) show that MARS achieves the largest MCR across all l_p norms: l_2 radius increased by 9×10^{-4} to 3.77×10^{-2} compared to BARS, l_1 by 7.24×10^{-1} to 2.2373 and l_∞ by 2.20×10^{-1} to 2.48×10^{-1} compared to FRS. The time for MARS to certify a single sample is comparable to that of FRS, which is around 0.02 to 0.03 seconds. Although it is higher than BARS and VRS based on zeroth-order information, it is still an efficient millisecond-level overhead. MARS certified 6.6K GoldenEye, 2.8K SlowHTTPTest, and 11.5K LOICHTTP samples in 219.82, 100.99, and 321.19 sec, respectively. Also, we noticed that using Laplace or Uniform may not certainly lead to better results, this could be due to the nature of classifiers and dataset. Different distributions may work better with different datasets.

Furthermore, we observe that MARS outperforms BARS in defending the vanilla classifier against Latency and PacketLoss corruption, improved accuracy by 6.12% under Latency and PacketLoss corruptions (See Figure 4.12 and Table 4.9). False Positive Rate (FPR) on corrupted Benign samples decreased by 4.27%, and False Negative Rate (FNR) on corrupted GoldenEye decreased by 6.35%. With MARS defense, the classifier achieves higher precision with corruption on both Latency and PacketLoss. This improvement is consistent across classes, as seen in Benign and DoS-GoldenEye examples.

Table 4.9 Comparison of empirical robustness of ACID against natural corruptions in fine-grained intrusion detection.

Metric	Test Class	Method	Clean	Latency	PacketLoss
FPR	Benign	Vanilla	0.0001	0.1991	0.1798
		BARS [82]	0.0001	0.1425	0.1362
		MARS	0.0001	**0.0969**	**0.0965**
FNR	DoS-GoldenEye	Vanilla	**0.0002**	0.5884	0.6060
		BARS [82]	0.0704	0.3594	0.4125
		MARS	0.0136	**0.2486**	**0.3964**
Accuracy	All	Vanilla	**0.9900**	0.5603	0.5893
		BARS [82]	0.7580	0.5595	0.5764
		MARS	0.8092	**0.6206**	**0.6378**
Precision	All	Vanilla	**1.0000**	0.8474	0.8576
		BARS [82]	0.9999	0.8919	0.8917
		MARS	0.9999	**0.9119**	**0.9147**

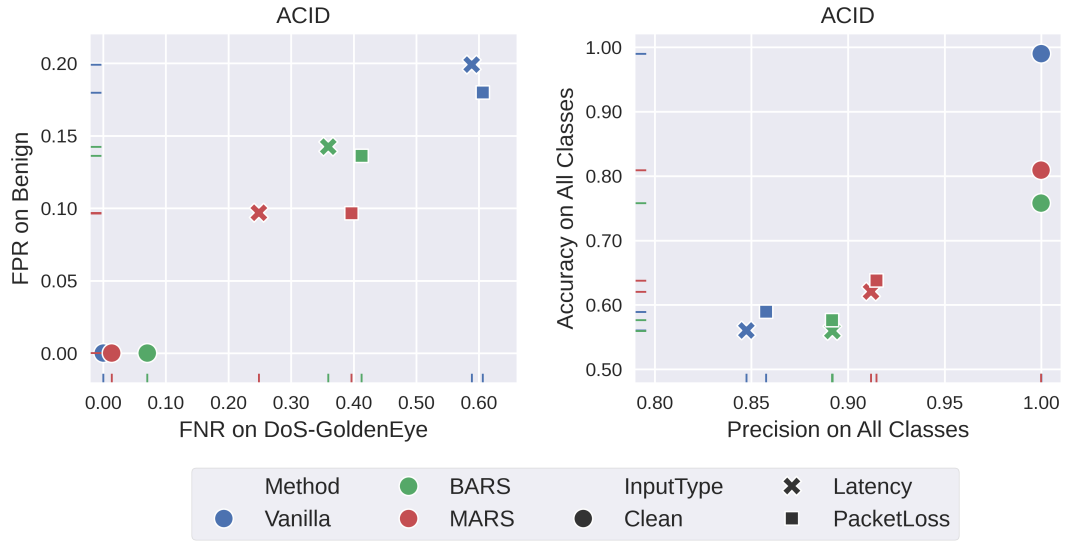


Figure 4.12 Comparison of empirical robustness of ACID against natural corruptions in fine-grained detection of similar intrusions.

4.6 Vertical Experimental Results and Analysis

In this section, we first present the analysis of dimension-wise certified robustness (Section 4.6.1). Then, we demonstrate the tightness of l_1 and l_∞ certified robustness guarantees with different smoothing distributions (Section 4.6.2). Later, we presented the time cost of the proposed method (Section 4.6.3).

4.6.1 Evaluation of Dimension-wise Certified Robustness

The certified radius presented in the previous sections is the overall radius, which is derived from a weighted average of dimension-wise certified radii. Analyzing the dimensional radius R_i can help intrusion detection participants identify sensitive (important) and insen-

sitive (robust) features, informing model feature-specific robustness. We evaluated MARS-defend ACID's l_2 R_i on the GoldenEye class.

Results. A smaller radius indicates greater sensitivity and importance to the NID, as perturbations to such features are more likely to alter the model's predictions. The results of dimensional MCR on all features shown in Figure 4.13) and the top-5 and bottom-5 radius rankings listed in Table 4.10 reflect the ACID model's sensitivity to InterArrivalTime (IAT)-related features and its relative robustness to quantity-related features in the DoS-GoldenEye attack, which is consistent with previous findings of increased FNR for ACID on Latency-corrupted DoS-GoldenEye samples in Section 4.5.5.

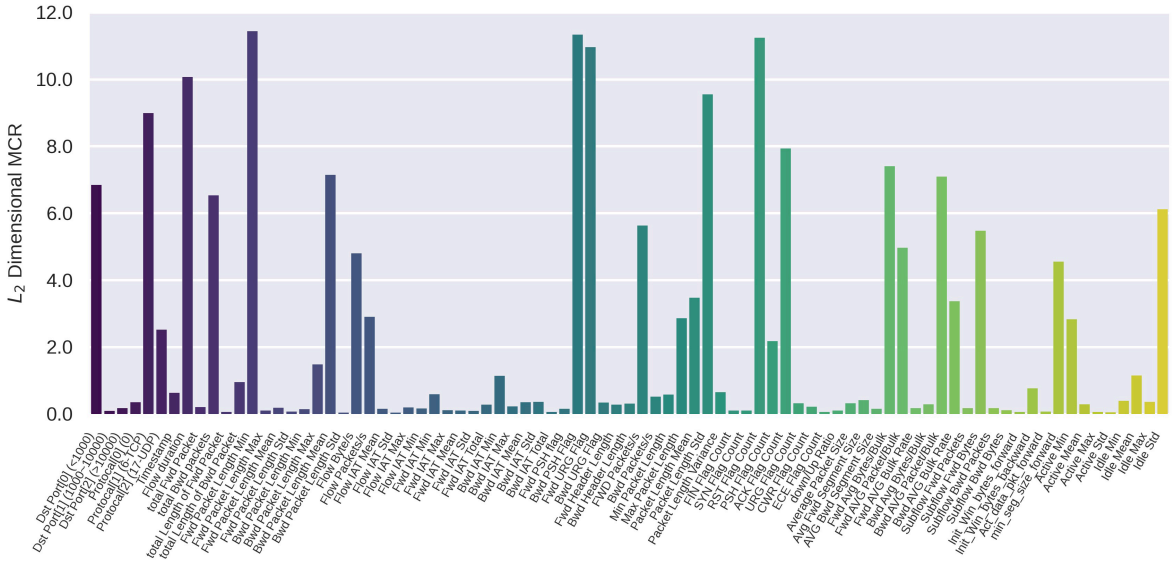


Figure 4.13 l_2 Dimensional MCR of ACID on DoS-GoldenEye.

4.6.2 Evaluation of l_p Robustness under Different Smoothing Distributions

VRS, FRS, and BARS all use the Gaussian distribution as the smoothing distribution; only MARS considers distribution alignment. We evaluate the certified radii under the smoothing distribution alignment setting. We sequentially used Gaussian, Laplacian, and Uniform distributions as smoothing distributions for l_2, l_1, and l_∞ guarantees calculation.

Setup. The Gaussian distribution $\mathcal{N}(\mu, \sigma)$ and the Laplacian distribution $\mathcal{L}(\mu, b)$ have the same mean 0 and standard deviation 1, lower bound a and upper bound b of the Uniform distribution $\mathcal{U}(a, b)$ are $-\sqrt{3}$ and $\sqrt{3}$ respectively. For l_2 robustness guarantees shown in Section 4.5.1, Gaussian is the default distribution type under our settings. Thus, we only show the tightness of l_1 and l_∞ robustness guarantees obtained through MARS under different smoothing distribution settings here.

Table 4.10 Sensitive and robust features on DoS-GoldenEye

No	Radius	FeatureName	Description
24	0.0426	Flow_IAT_Std	Standard deviation time two flows.
20	0.0433	Bwd_Packet_Length_Std	Standard deviation size of packet in backward direction.
79	0.0488	Active_Std	Standard deviation time a flow was active before becoming idle.
72	0.0569	Init_Win_bytes_forward	Number of bytes sent in initial window in the forward direction.
78	0.0576	Active_Max	Maximum time a flow was active before becoming idle.
8	10.0741	Flow_Duration	Flow duration.
39	10.9644	Fwd_URG_Flag	Number of times URG flag was set in packets travelling in the forward direction (0 for UDP).
52	11.2367	RST_Flag_Count	Number of packets with RST.
38	11.3300	Bwd_PSH_Flag	Number of times PSH flag was set in packets travelling in the backward direction (0 for UDP).
13	11.4358	Fwd_Packet_Length_Min	Minimum size of packet in forward direction.
All	2.2305	MCR	Mean certified radius per class.

Results. Results of l_1 MCR (see Figure 4.14) show that Gaussian and Laplacian distributions each excel in different classes, indicating that simply using a single distribution may miss a tighter certified radius, leading to a necessity of analyzing different distributions. Results of l_∞ MCR can be seen in Table 4.11, which enlightens us that for the l_1 robustness guarantee, the Gaussian distribution consistently used by the compared methods is not necessarily the best in various categories. Although the smoothing area of the uniform distribution is closer to l_∞, as it has upper and lower bounds, the choice of distribution parameters greatly limits the addition of noise. However, for the l_∞ certified radius, Gaussian distribution indeed has advantages. Although the l_∞-measured certified area best matches the Uniform distribution in shape, the smoothing region that the Uniform Distribution can cover is strictly controlled by the upper and lower bound parameters, and sharp truncation areas will appear. Therefore, the certification performance will be largely affected by the hyperparameters.

4.6.3 Evaluation of Time Cost

We calculated the Average Certification Time (ACT) per sample for detection models (CADE and ACID) across various network traffic categories.

Results. The results of ACT (see Table 4.12) show that, compared to BARS, `MARS`

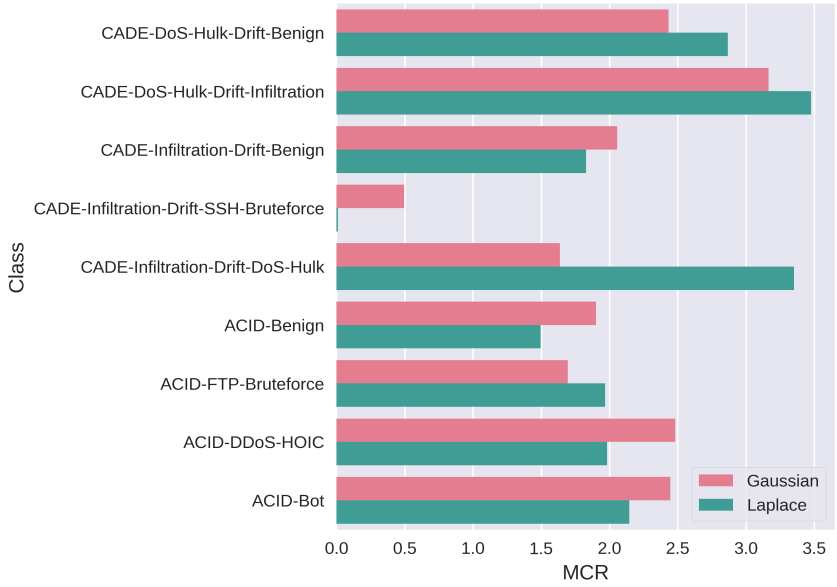


Figure 4.14 l_1 Mean Certified Radius (MCR) under different smoothing distributions

Table 4.11 l_p Mean Certified Radius (MCR) under different smoothing distributions

Norm	Distribution	Seed	CADE-DoS-Hulk-Drift Dataset			CADE-Infiltration-Drift Dataset			ACID-CIC-IDS-2018 Dataset			
			Benign	SSH-Bruteforce	Infiltration	Benign	SSH-Bruteforce	DoS-Hulk	Benign	FTP-Bruteforce	DDoS-HOIC	Bot
l_1	Gaussian	42	2.4391	0.0000	3.1658	2.0556	0.4950	1.6406	1.9028	1.6933	2.4812	2.4453
		43	2.4204	0.0000	3.1658	2.0567	0.4968	1.6371	1.9028	1.6936	2.4806	2.4450
		44	2.4407	0.0000	3.1658	2.0556	0.4953	1.6340	1.8990	1.6939	2.4816	2.4461
	Laplacian	42	2.8599	0.0000	3.4763	1.8303	0.0110	3.3576	1.4865	1.9654	1.9812	2.1438
		43	2.8801	0.0000	3.4762	1.8297	0.0110	3.3504	1.5023	1.9665	1.9825	2.1452
		44	2.8600	0.0000	3.4762	1.8273	0.0110	3.3439	1.4950	1.9692	1.9814	2.1440
l_∞	Gaussian	42	0.2663	0.0000	0.3475	0.2299	0.0633	0.1801	0.2182	0.1920	0.2723	0.2684
		43	0.2657	0.0000	0.3475	0.2300	0.0635	0.1797	0.2181	0.1920	0.2723	0.2684
		44	0.2679	0.0000	0.3475	0.2299	0.0633	0.1794	0.2177	0.1920	0.2724	0.2685
	Uniform	42	0.1449	0.0000	0.1897	0.0211	0.0003	0.0984	0.1581	0.1605	0.1404	0.1214
		43	0.1452	0.0000	0.1897	0.0198	0.0003	0.0982	0.1581	0.1605	0.1404	0.1214
		44	0.1464	0.0000	0.1898	0.0192	0.0003	0.0980	0.1580	0.1605	0.1404	0.1215

achieves the highest MCR in all competitions but increases ACT per sample by 21.1 milliseconds (ms) on average. Despite this minor increase, the certification time cost of MARS remains reasonable: ACT of CADE ranges from 8.0 to 14.6 ms, and ACT of ACID from 24.4 to 36.1 ms. ACT per class set was also assessed. MARS-defend CADE requires about 28 seconds (sec) to certify 2K Infiltration or SSH-Bruteforce samples and approximately 125 sec for 13K benign or 10K DoS-Hulk samples. This trade-off between slightly longer processing times and enhanced reliability is favorable in security-critical NID applications.

Table 4.12 Comparison of l_2 certification time overhead of certified defense methods.

ACT (Seconds) /sample	CADE				ACID						
	Benign	SSH-Bruteforce	Infiltration	DoS-Hulk	Benign	FTP-Bruteforce	DDoS-HOIC	Bot	DoS-GoldenEye	DoS-SlowHTTPTest	DDoS-LOIC-HTTP
VRS [79]	0.0008	0.0015	0.0017	0.0012	0.0028	0.0045	0.0028	0.0040	0.0039	0.0053	0.0033
FRS [81]	0.0089	0.0125	0.0122	0.0134	0.0238	0.0323	0.0235	0.0565	0.0309	0.0459	0.0269
BARS [82]	0.0004	0.0011	0.0013	0.0007	0.0029	0.0046	0.0029	0.0041	0.0040	0.0051	0.0033
MARS	0.0080	0.0131	0.0131	0.0146	0.0253	0.0346	0.0244	0.0327	0.0331	0.0361	0.0279

4.7 Summary

In this chapter, we studied the certification of adversarial robustness of DNNs and introduced MARS, a framework to certify the adversarial robustness of the model with heterogeneous input features, such as network traffic data. Experiments demonstrate that MARS has field-leading performance in terms of certified adversarial robustness evaluated by certified radius and certified accuracy, as well as empirical adversarial robustness evaluated by robust accuracy, false alarm rate, etc. (i) Our innovative use of first-order information offers tighter bounds (12.23% average increase in MCR) and higher certified accuracy across all evaluated classifiers compared to the leading BARS. (ii) Our certified defense not only boosts detection accuracy on adversarial attacks (by 7.17% for l_∞-PGD and 10.11% for l_1-EAD compared to BARS) but also exhibits better detection accuracy on natural corruptions (16.65% higher on Latency and 18.23% higher on PacketLoss) , which is a notable achievement as smoothed classifiers often exhibit reduced robustness to natural corruption. (iii) The ability to support diverse l_p certifications enhances our flexibility. In future work, we plan to investigate automatic distribution detection for unknown adversarial attacks and design non-l_p robustness certification against structural perturbations.

Chapter V Adversarial Robustness Transfer with Contrastive Adversarial Representation Distillation

Adversarially robust deep neural networks (DNNs) are crucial for creating security-sensitive applications that are robust against adversarial attacks (a.k.a evasion attacks) using adversarial examples, such as network intrusion detection systems. However, training accurate and robust detection models from scratch typically demands extensive labeled data and costly adversarial training. Existing transfer learning (TL) approaches help to mitigate this issue, but most prioritize the prediction accuracy on clean data and ignore the robustness against adversarial examples. Our empirical studies reveal that conventional TL techniques generally yield accurate but not robust models, while the few existing adversarial TL works that consider robustness are less effective and usually degrade accuracy. In this chapter, we study the transfer technique for adversarial robustness of DNNs and propose a robustness-preserving transfer learning framework, Contrastive Adversarial Representation Distillation (`CARD`), to produce a target model for a given domain with limited data from a robust source model adversarially trained in another domain.

`CARD` tackles two issues: scarcity of target-domain training data and robustness of the target model against adversarial attacks. It has three main characteristics: (i) Adaptive dimension alignment — it adaptively aligns the input feature dimensions in the source domain and the target domain and aligns the hidden representation dimensions of the target model and the source model, thereby flexibly supporting transfer learning across data domains with different feature dimensions and across deep neural networks with different structures. (ii) Contrastive hidden representation distillation — it guides the target model in convergent with the target-domain distribution with only a few samples by contrastively learning the latent domain-invariant information in the source model representation space. (iii) Dual-view robustness capture — it strengthens the attention on domain-invariant robustness by using two positive views in contrastive learning, including the adversarial manipulation view and the natural corruption view. Finally, we show experimental results on various network intrusion detectors and datasets, demonstrating that the proposed method enhances the transferability of adversarial robustness across data domains and model structures. It achieves leading adversarial robustness and regular predictive performance for lightweight models with limited training data compared to SOTA adversarial fine-tuning and distillation methods.

5.1 Overview

Deep learning (DL) techniques are increasingly attractive for developing network intrusion detection (NID) systems[2-3]. This is due to the exceptional capability of neural models to discover hidden representations in large amounts of data and enhance classification accuracy, as evidenced in fields such as computer vision[1], speech recognition[142], and networking[4]. In this work, we focus on deep neural network (DNN)-based network intrusion detection models because of their superior capability to process high-dimensional features[8], efficient scalability with large datasets[143], and flexible adaptability to changing network threats[144].

Learning a well-performing network traffic classifier from scratch can be challenging[145]. Complex attack patterns and privacy concerns in accessing real traffic make collecting and labeling sufficient traffic samples difficult, while the evolving nature of network intrusions requires the model to adapt quickly to new traffic data distributions with limited training samples. Also, new NID application domains, like the Internet of Things (IoT) and mobile edge computing systems, often require lightweight models[146-147], which may sacrifice the representation learning ability of neural models[148]. Moreover, the robustness of network traffic classifiers to adversarial attacks using adversarial examples (AEs)[19,21-22], where malicious parties manipulate inputs to cause misclassification, is crucial. These models should also be resilient to natural noise from changes in the network environment.

A category of approaches for addressing the problem of data scarcity is based on transfer learning (TL)[149-150]. TL leverages a model trained in a given domain, referred to as the *source domain*, to train a model in a different but related domain, referred to as the *target domain*, in which very few labeled data are available. Previous work on TL for network traffic classifiers has shown that knowledge transfer across related domains not only enhances the adaptability of the models to changing network threats[151-152], but also makes target-domain lightweight models suitable for deployment on devices with limited computing and storage resources[153],[154]. However, such TL approaches do not address the pressing problem of robustness against adversarial attacks and specifically do not address the following crucial problem: *Let f^S be a source NID model, which has been trained to be robust against adversarial attacks, and let f^T be a target NID model derived from f^S using TL, does the property of robustness against adversarial attacks automatically transfer to f^T?* Especially, our assumed threat model is adaptive adversarial attackers, meaning the target model's robustness is evaluated on AEs adaptively crafted by the adversary toward the target model.

Robustness-preserving TL ensures that the network traffic classifier trained via TL from a robust source model is accurate and robust without requiring any additional training for ro-

bustness. Furthermore, this concept is also pivotal in the broader area of foundation models (FMs)[155], which are large models trained on massive datasets. Rather than creating models from scratch, one can use an FM model as a starting point to generate models specialized/adapted for different tasks, model architectures, and data domains. In the context of FMs, TL methods that enable derived models to inherit the robustness of the source model are crucial for many application areas. Especially in NID, which is characterized by frequent changes in traffic data distribution caused by evolving network threats, robustness-preserving TL is critical. It would yield accurate target models with respect to (w.r.t.) both benign and malicious traffic while being able to withstand adversarial attacks and traffic changes.

In this chapter, we demonstrate that conventional TL methods, such as standard finetuning and knowledge distillation, fail to transfer robustness to the target NID model. However, it is possible to design a TL approach that preserves the adversarial robustness of the source NID model. We introduce Contrastive Adversarial Representation Distillation (`CARD`), a novel TL framework that utilizes a robust source model trained in a related domain to generate an accurate and robust model for the target domain. It addresses two issues simultaneously: the scarcity of target-domain data and the robustness of the target model. `CARD` is tailored for DNN-based network traffic classifiers and is designed to adapt to diverse TL scenarios (see Figure 5.1), including cases where the source and target models differ only in their data domains, differ only in their architectures, or differ in both data domains and architectures.

Our design of `CARD` is built on three key technical elements: ① To support TL between domains with different input feature dimensions and TL between DNN models with different structures, we design a dual adaptive dimension alignment mechanism. Such a mechanism uses three optimized embedding networks to not only align the input feature dimensions of the target domain with the source domain but also align the hidden representation dimensions of the target model and the source model — thereby obtaining embedded target-domain samples that can be fed to the input layer of the source model and embedded hidden representations that can be used for similarity calculations between different models. ② To learn the real data distribution from a small number of target-domain training samples, we use Contrastive Representation Distillation (CRD)[156] to extract domain-invariant latent features learned by the source model in the source domain with sufficient training data — thereby guiding the target model to converge to the target data domain with limited labeled training data after combining supervision of ground-truth labels. ③ To fully capture robustness-relevant information from the source model, we design two robustness-aware contrastive views, namely

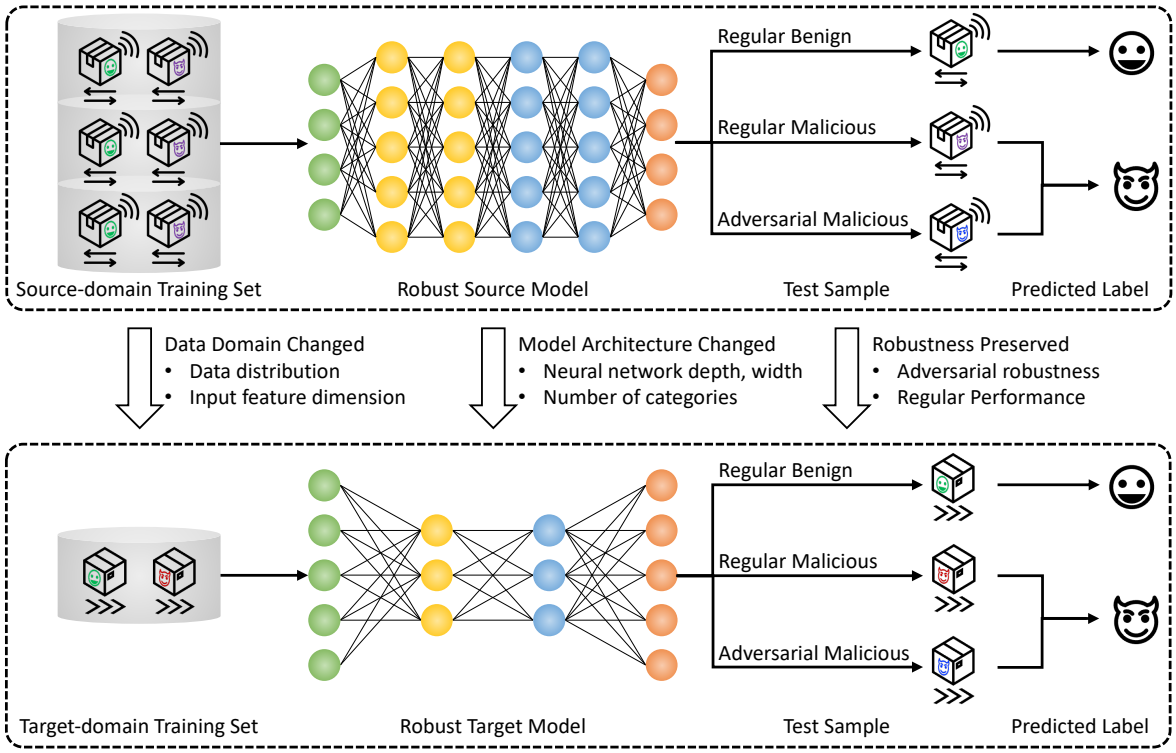


Figure 5.1 Transfer learning scenarios where CARD can be applied to.
The target task and the source task differ in data domains, model architectures, or both.

the adversarial manipulation view and the natural corruption view, to force the target-domain model clustering these two types of samples with the anchor sample in the embedding space and separating them from negative samples — thereby generating a target-domain model that is robust to adversarial perturbations and natural corruptions after combining adversarial distillation and adversarial training.

We conduct extensive evaluations of CARD on multiple TL scenarios: cross-domain TL (only data domain changes), cross-model TL (only model architecture changes), and more challenging TL where both the data domain and model change. We also compare CARD with baseline TL methods such as NID-specific standard fine-tuning[150] and knowledge distillation[153], which do not consider robustness, as well as state-of-the-art (SOTA) image-specific robustness-preserving TL methods, including adversarial fine-tuning (FRFE[85], TWINS[88]) and adversarial distillation (VAD[91], AAD[97]) in binary and multi-classification tasks.

In this chapter, we make the following contributions:

- We propose the first robustness-preserving TL framework for NID, CARD, to learn a robust target-domain network traffic classifier without requiring large amounts of labeled training samples by contrastively distilling a robust source model. Owing to the dual adaptive dimension alignment mechanism we design, this is the first solution

that can flexibly adapt to a variety of TL scenarios.

- We introduce the first notion of dula robustness-aware positive views, including the adversarial manipulation view and the natural corruption view, in contrastive representation distillation to capture the robust domain-invariant information from the representation space of the source model.
- We extensively evaluate the robustness and generalization of the target model, repeating each experiment with three random seeds. Results show that `CARD` achieves an average 13% improvement in robust accuracy on adversarial samples and an 11% improvement on corrupted samples, with a 6% higher clean accuracy on regular samples compared to SOTA adversarial fine-tuning and distillation methods.

5.2 Problem Formulation

5.2.1 Threat Model

We focus on two robustness threats faced by DNNs: (i) adversarial attacks — deliberately launched by attackers using adversarial examples (AEs), and (ii) natural corruptions — unintentionally caused distribution shifts by changes in the network environment.

Adversarial Attacks. We assume that the adversarial attacker is adaptive, meaning that the robustness of the target model is assessed on AEs adaptively crafted by the adversary against the target model rather than the source model. Additionally, we consider white-box adversarial attacks. The adversary creates strong AEs based on complete knowledge of the victim network traffic classifier. By assuming that the attacker possesses full knowledge of the model, we aim to simulate a threat scenario where the adversary has maximum visibility into the model internals. This enables the evaluation of the robustness of the model against sophisticated attacks, leveraging knowledge of the model's inner workings while also revealing potential vulnerabilities of the target model.

Natural Corruptions. We also consider the robustness of the target model to distribution shifts arising from natural variations in datasets. Natural corruptions result from uncontrollable environmental factors, such as lighting changes in images or recording device alterations in speech. Unlike adversarial attacks, these corruptions are typically unintentional. Specifically for NID, we focus on the distribution shift caused by random noise added to time-related, quantity-related, and length-related traffic features. By assuming noise background in temporal and spatial characteristics, we aim to mimic a scenario where natural corruptions arise from network congestion or electromagnetic interference.

With these considerations, the robustness of network traffic classifiers against adversarial and natural perturbations on the input can be empirically demonstrated, ensuring effective operation amid unexpected network disturbances.

5.2.2 Research Goal

Our goal is to design a robustness-preserving transfer learning (TL) technique that, starting from a robust source model f^S and a target-domain training set D^T_{train} with a small number of labeled samples, ensures the robustness and accuracy of the target model f^T for different TL tasks shown Figure 5.2. To achieve this, we need to address three key issues.

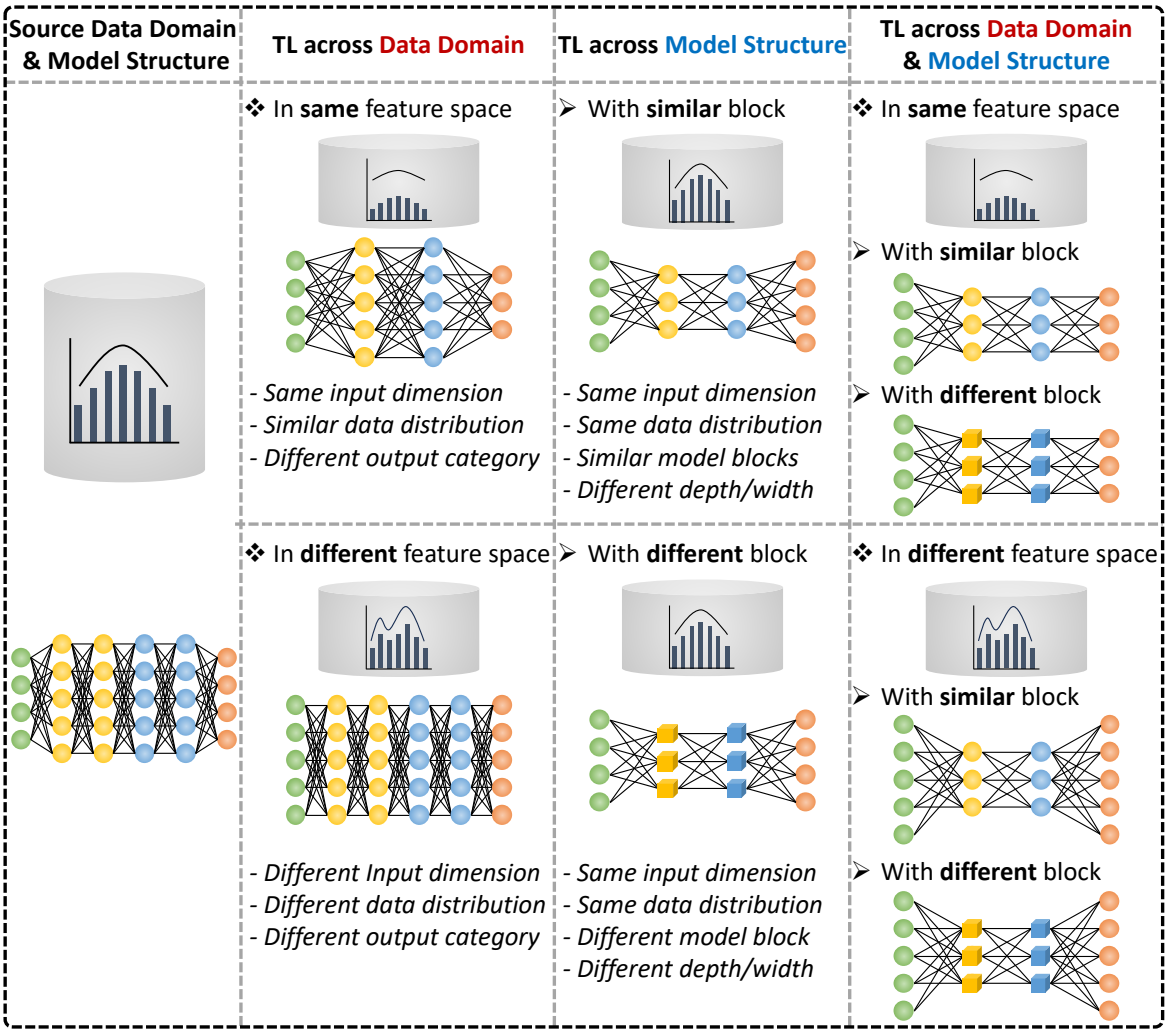


Figure 5.2 Differences in datasets and models between the target domain and the source domain in the transfer learning tasks targeted by CARD.

Problem 1. *Use TL to learn an accurate and robustness-preserving target model when the source and target tasks differ in data domains w.r.t data distribution, number of features, values of some features, or number of categories.* For fine-tuning TL, the target and the

source model may share the model architecture but have differences in the data domains. With respect to NID, because concept drift often occurs in network environments, even when the number and meaning of the feature dimensions of network traffic do not change, that is, *the same input feature space*, the data distribution can change over time, thus resulting in a new target data domain different from the source data domain. Furthermore, since the target-domain data may involve different network protocols or services than the source domain, and the tools for processing network traffic features may also be different, it is common that the feature dimensions and meanings of the target-domain data may be different from the ones of source-domain data, that is, *different input feature spaces*, resulting in greater differences between the source-domain data and the target-domain data.

Problem 2. *Use TL to learn an accurate and robustness-preserving target model when the source and target tasks differ in model structures w.r.t depth, width, and layer structure.* In some scenarios, the source and target model are applied to the same data domain; however, the goal is to obtain a compressed target model with a smaller number of parameters. In several application domains, NID models run in resource-constrained environments, such as embedded devices, edge computing nodes, or mobile devices, so the actual target-domain models need to have small model sizes and fewer parameters to run with low latency, resulting in differences of architectures between the target and source models. Previous work has shown that it is difficult to adversarially train robust low-capacity DNNs from scratch compared to high-capacity DNNs, especially with limited training samples. Thus, it is critical to be able to use TL to train robust smaller DNNs.

Problem 3. *Use TL to train an accurate and robustness-preserving target DNN model from a robust source DNN model when the source and target tasks differ in both data domains and model structures.* This is a more complex situation as both the data domain and model structure of the source and target models differ. On the one hand, large-scale source models trained with rich data need to be adapted to train a smaller model, that is, with a smaller number of parameters, for use in a target domain in which the amount of training data is limited. On the other hand, there could also be differences in the number of output categories. Therefore, we need a TL that is able to balance the predictive performance of the target model on clean data with its robustness against adversarial and corrupted samples.

5.2.3 Key Challenge

The approaches and challenges to address these problems are considered below.

Approach Direction to Problem 1. We use contrastive learning (CL)[157] to direct the

target model to reference the source model, aiming to generate similar representations for positive pairs, consisting of target-domain samples with the same label, and dissimilar representations for negative pairs, consisting of target-domain samples with different labels. Such a strategy forces the target model to learn class-invariant representations based on a small number of target-domain samples to enhance robustness. However, such a strategy requires addressing several challenges.

Challenge 1. *Effectively selecting positive and negative pairs and similarity metrics for inducing the target model to learn robustness-relevant representations.* Appropriate and diverse contrastive pairs can help the model better capture the correct connections among the anchor sample x, positive samples x^+, and negative samples x^-, facilitating the learning of domain-invariant representations. Yet, there is still a lack of specific designs for contrastive pairs to convey robust representation.

Challenge 2. *Transforming the target-domain samples to fit the input layer of the source model when the input feature dimensions of the target and source domains are different.* Since the source model is pre-trained, it is necessary to adapt the target-domain input samples to the source model through an increase or decrease in dimensionality. Thus, we need an adaptive input feature transformation method to align the input feature space dimensions between the target and source domains.

Approach Direction to Problem 2. We use hidden-layer and output-layer distillation to transfer robustness, with the hidden representations and soft labels as knowledge carriers. This allows the target model to learn the output probability distribution from the source model even with different model architectures. Also, similarity matching between representations helps the target model learn high-level abstract information in the source model, particularly robust underlying feature representations. Yet, this approach faces the following challenges.

Challenge 3. *Efficiently extracting and matching hidden representations between source and target models that may come from different representation spaces.* When the source and target models have different architectures, their hidden layer embeddings may differ in dimensions and abstract meaning. Thus, effective representation alignment is needed to compare robustness-relevant representations between the models.

Approach Direction to Problem 3. We leverage the dual adaptive dimension alignment mechanism used in the approaches for Tasks 1 and 2 to address the dimensional differences caused by differences in data domain and model structure. Also, to balance the regular predictive performance and robustness of the target model, we use a weighted comprehensive loss that combines contrastive loss, distillation loss, and classification loss. However, using

the output of the source model as supervisory information presents the following challenges.

Challenge 4. *Mitigating erroneous supervision caused by incorrect predictions by the source model on the target domain.* The robustness and accuracy of the target-domain model are not only affected by the specific TL algorithm but also by the inherent generalization ability with respect to the robustness and accuracy of the source model obtained by adversarial training on the source domain. When the target model uses the predictions of the source model as supervision information to transfer robustness knowledge from the source domain, the source model may produce incorrect supervision on the target domain due to issues such as overfitting the source-domain distribution. Thus, we need a mechanism to alleviate the misleading caused by incorrect predictions of the source model.

5.3 Design of Method CARD

This section introduces the design of the proposed robustness-preserving transfer learning method, Contrastive Adversarial Representation Distillation (`CARD`), to transfer the adversarial robustness of DNNs across different tasks. The core idea of `CARD` is to produce a robust target model with limited target-domain training data by contrastively imitating the hidden representations of a robust source model. Each sample in the target-domain training set D^T_{train} serves as an anchor. The target model learns hidden representations by comparing the input anchor sample x with positive samples x^+ (transformed samples similar to x) and negative samples x^- (samples in D^T_{train} dissimilar to x), bringing positive samples closer and pushing negative samples further in the latent space. The hidden representations of the source model on adversarial variants and corrupted variants of anchor samples are adopted as carriers of robustness knowledge for transfer learning. The architecture of `CARD` comprises three components: Robustness-aware View Construction, Adaptive Dimension Alignment, and Contrastive Distillation Learning (see Figure 5.3).

5.3.1 Robustness-aware View Construction

In contrastive learning, contrastive views (including positive views x^+ and negative views x^-) refer to different augmented versions of the anchor data. Contrastive pairs involve positive pairs (x, x^+) consisting of instances with similar features and negative pairs (x, x^-) consisting of dissimilar instances. The target model should be directed to minimize the distance between samples in positive pairs (x, x^+), and to maximize the distance between samples in negative pairs (x, x^-). Here, we detail the construction process of these pairs in `CARD` (see the left side of Figure 5.3).

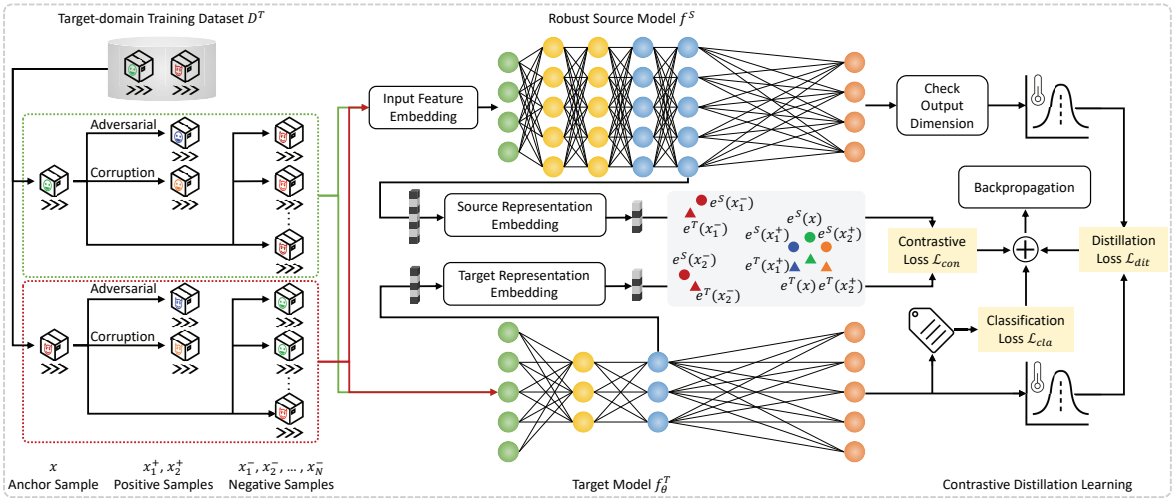


Figure 5.3 Architecture of CARD.

Left: contrastive view construction with green for clean benign samples, red for clean malicious samples (different reds for different categories of anomalies), blue for adversarial examples, and orange for naturally corrupted examples. Middle: dual adaptive dimension alignment for input features and hidden representations. Right: learning of target model based on TL losses ($\mathcal{L}_{con}$, $\mathcal{L}_{dit}$) and local classification loss ($\mathcal{L}_{cla}$).

Positive Pairs. Unlike standard contrastive learning (CL)[157] in the image domain, where a positive pair (x, x^+) usually consists of the anchor sample x and its transformation x^+ (e.g., rotations, flips), we propose two robustness-aware contrastive views for network traffic data: the adversarial manipulation view x^*, and the natural corruption view $\tilde{x}$. The positive pairs for each anchor sample x include $(x, x_1^+) = (x, x^*)$ and $(x, x_2^+) = (x, \tilde{x})$.

Adversarial Manipulation View. The goal of minimizing the representation distance between positive samples in CL is consistent with the goal of the model learning representations robust to adversarial perturbations. Adversarial and clean samples should have similar representations in the latent space. Thus, we select the adversarial variant x^* of the anchor x as one positive view. The adversarial-view positive sample $x^* = x + \delta$ is generated according to Eq. (2-18) for x in the target-domain training set.

Natural Corruption View. Corrupted sample simulate the network traffic that is naturally disturbed in the network environment. The goal of minimizing the representation distance between positive samples is also consistent with the goal of the model learning representations that are robust to natural corruptions. Thus, we select the corrupted version $\tilde{x}$ of the anchor x as another positive view. To generate corruption-view positive sample $\tilde{x} = x + \zeta$, time, quantity, and length-related features of the network packets are modified by random perturbations ζ sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$.

Negative Pairs. The negative sample should encourage the model to learn the difference between normal traffic and abnormal traffic. Especially for NID in multi-class classification

scenarios, the model must not only learn the coarse-grained differences between benign and malicious traffic patterns but also learn the fine-grained differences between different malicious traffic categories. The negative pairs set consists of $\{(x, x_1^-), ..., (x, x_N^-)\}$. When the anchor x is a benign sample, we select N malicious samples from different attack types as negative samples. We prioritize abnormal samples that are significantly different in features, such as communication protocols, ports, and number of connection attempts. When the anchor is a malicious sample, we select a total of N negative samples, including benign samples and other types of abnormal samples that are different from the anchor. Afterward, each anchor sample will be simultaneously inputted into the source and target models together with the positive and negative samples.

5.3.2 Adaptive Dimension Alignment

To support TL tasks across data domains, especially across different input feature spaces, we design an Input Feature Dimension Alignment module to deal with the issue that the number of feature dimensions d^S of the source-domain sample $x \in D^S$ is inconsistent with d^T of the target-domain sample $x \in D^T$. To adapt TL across model structures, especially across different hidden representation spaces, we design two Hidden Representation Embedding modules to align the number of the captured hidden representation dimension d_e^S of the source model f^S and d_e^T of the target model f_θ^T. The process is shown in the middle of Figure 5.3.

Input Feature Dimension Alignment. In the process of learning the target model, we need to input target-domain samples $x \in D_{train}^T$ into the pre-trained source model f^S to obtain contrast information and distillation information. When d^T is different from d^S, the source model cannot directly use the target domain as input. To this end, we apply the following adaptive alignment measures in the input space. We align the d^T-dimensional target-domain input x with the dimension d^S of the source-domain input space using an Input Dimension Alignment module E_{ϑ^I}. It consists of a linear transformation layer $h_{\vartheta^I}(x)$ with optimizable parameters ϑ^I and an l_2 normalization function $m(h_{\vartheta^I}(x)) = \frac{h_{\vartheta^I}(x)}{||h_{\vartheta^I}(x)||_2}$. The two hyperparameters of the linear transformation layer, input dimension and output dimension, are automatically set to d^T and d^S, respectively. Such a way makes it possible for CARD to adaptively align the input dimensions for cross-domain TL. Since the performance of the input dimension alignment module affects the hidden representation and the class probability output by the source model, which in turn affects the contrastive loss and distillation loss, we optimize parameters ϑ^I of the linear transformation layer along with the parameters of f_θ^T.

Hidden Representation Embedding. The output of the specific hidden layer of the

model is termed as the hidden representation e. We calculate the contrastive loss based on the representations, e^S and e^T, from the last hidden layers of the source and target models, respectively. To accommodate TL across various architectures with potentially inconsistent representation dimensions, we introduce two hidden representation embedding modules: Source Representation Embedding E_{ϑ^S}, and Target Representation Embedding E_{ϑ^T}. These modules map the original representations e^S and e^T, which may have different dimensions, to a common dimension d_e, facilitating the contrasting of representation characteristics. These embedding modules resemble the Input Dimension Alignment module but with differences in the input dimensions of the linear transformation layers (e^S and e^T) and the output dimension (a predefined constant d_e typically smaller than the input dimension). As the performance of these modules affects the contrastive loss, the parameters of their linear transformation layers, ϑ^S and ϑ^T, are jointly optimized along with the target model parameters during training. Afterward, aligned hidden representations from the source and target models are used for CL. Moreover, the original representations are also given as input to the output layers of both models to produce logit values for distillation and classification learning.

5.3.3 Contrastive Distillation Learning

Considering the diversity of TL goals, such as boosting adversarial robustness while balancing the regular predictive performance of the target model, we design a contrastive adversarial distillation learning algorithm (see the right side of Figure 5.3) to address the requirements of different TL tasks. The detailed learning process is shown in Algorithm 5.1.

For each anchor sample x, as well as its positive samples x^+ and negative samples x^-, we obtain the corresponding hidden representations of the same dimension from the source and target models and output logit values. Then, we jointly optimize the parameters of the target model f_θ^T, input dimension alignment module E_{ϑ^I}, and hidden representation embedding modules E_{ϑ^S}, E_{ϑ^T}, by minimizing a weighted loss $\mathcal{L}_{card}$ consisting of three terms: contrastive loss $\mathcal{L}_{con}$, distillation loss $\mathcal{L}_{dit}$, and classification loss $\mathcal{L}_{cla}$, as shown in Eq. (5-1).

$$\min_{\theta,\vartheta^I,\vartheta^S,\vartheta^T} \mathcal{L}_{card} = \alpha \cdot \mathcal{L}_{con} + \beta \cdot \mathcal{L}_{dit} + \gamma \cdot \mathcal{L}_{cla}, \tag{5-1}$$

where α, β, $\gamma \in$ [0,1].

Algorithm 5.1 Contrastive Adversarial Representation Distillation

Input: d^T-dimensional target-domain training set D^T_{train}, source model f^S with input dimension d^S and output dimension c^S, predefined hidden representation dimension d_e, input dimension alignment module E_{ϑ^I}, hidden representation embedding modules E_{ϑ^S} and E_{ϑ^T}, adversarial sample generator G_{adv}, natural corrupted example generator G_{cor}.

Output: robust target model f^T_θ with output dimension c^T.

for $i = 1$ to $len(D^T_{train})$ **do**
  select anchor sample $x \leftarrow x_i \in D^T_{train}$
  generate adversarial sample $x + \delta \leftarrow G_{adv}(x, f^T_\theta)$; generate corrupted sample $x + \zeta \leftarrow G_{cor}(x)$
  select N samples $\{x^-_j\}^N_{j=1}$ from D^T_{train} with label $\neq y$
  construct positive pairs $X_p = (x, x^+_1, x^+_2) \leftarrow (x, x + \delta, x + \zeta)$
  construct negative pairs $X_n = (x, x^-_1, ..., x^-_N)$
  if $d^T \neq d^S$ **then**
    $\hat{X}_p, \hat{X}_n \leftarrow$ apply InputProjection $E_{\vartheta^I}(d^T, d^S)$ on X_p, X_n; $X_p \leftarrow \hat{X}_p$, $X_n \leftarrow \hat{X}_n$
  end if
  feed anchor x to source model f^S and target model f^T_θ
  extract representations $e^S \leftarrow f^{r^S}(x)$,$e^T \leftarrow f^{r^T}_\theta(x)$;
  $e^S \leftarrow e^{S^E} \leftarrow$ apply $E_{\vartheta^S}(e^S, d_e)$, $e^T \leftarrow e^{T^E} \leftarrow$ apply $E_{\vartheta^T}(e^T, d_e)$
  output logit value $o^S \leftarrow f^S(x)$, $o^T \leftarrow f^T_\theta(x)$
  for $j = 1$ to 2 **do**
    feed x^+_j to f^S and f^T_θ; extract hidden representations $e^{+^S}_j \leftarrow f^{r^S}_\theta(x^+_j)$,$e^{+^T}_j \leftarrow f^{r^T}_\theta(x^+_j)$
    $e^{+^S}_j \leftarrow e^{+^{S^E}}_j \leftarrow$ apply $E_{\vartheta^S}(e^{+^S}_j, d_e)$; $e^{+^T}_j \leftarrow e^{+^{T^E}}_j \leftarrow$ apply $E_{\vartheta^T}(e^{+^T}_j, d_e)$
    if $j = 1$ **then**
      logit from target model $o^T_{adv} \leftarrow f^T_\theta(x + \delta) = f^T_\theta(x^+_j)$
    end if
  end for
  for $j = 1$ to N **do**
    feed x^-_j to f^S and f^T_θ; extract hidden representations $e^{-^S}_j \leftarrow f^{r^S}_\theta(x^-_j)$, $e^{-^T}_j \leftarrow f^{r^T}_\theta(x^-_j)$
    $e^{-^S}_j \leftarrow e^{-^{S^E}}_j \leftarrow$ apply $E_{\vartheta^S}(e^{-^S}_j, d_e)$; $e^{-^T}_j \leftarrow e^{-^{T^E}}_j \leftarrow$ apply $E_{\vartheta^T}(e^{-^T}_j, d_e)$
  end for
  compute loss $\mathcal{L}^T_{con}(e^S, e^T, \{e^{+^T}_j\}^2_{j=1}, \{e^{-^T}_j\}^N_{j=1})$; $\mathcal{L}^S_{con}(e^T, e^S, \{e^{+^S}_j\}^2_{j=1}, \{e^{-^S}_j\}^N_{j=1})$;
  compute contrastive loss $\mathcal{L}_{con} = \mathcal{L}^T_{con} + \mathcal{L}^S_{con}$; compute classification loss $\mathcal{L}_{cla}(o^T, y_{true})$
  if $c^S = c^T$ **then**
    compute distillation loss $\mathcal{L}_{dit}(o^T, o^T_{adv}, o^S)$
    compute weighted loss $\mathcal{L}_{card} = \alpha \cdot \mathcal{L}_{con} + \beta \cdot \mathcal{L}_{dit} + \gamma \cdot \mathcal{L}_{cla}$
  else
    compute weighted loss $\mathcal{L}_{card} = \alpha \cdot \mathcal{L}_{con} + \gamma \cdot \mathcal{L}_{cla}$
  end if
  update parameters of f^T_θ, E_{ϑ^S}, E_{ϑ^T}, E_{ϑ^I}
end for

Contrastive Loss. This enables the target model f_θ^T to learn hidden representations that are invariant to certain data augmentations or data variations, such as representations that are robust to adversarial perturbations or natural corruptions. We define the contrastive loss $\mathcal{L}_{con}$ referring to CRD[156] and extend it to incorporate two new positive views. $\mathcal{L}_{con}$ consists of two parts: (i) the contrastive loss $\mathcal{L}_{con}^T$ between the representation e^S of the anchor sample x on the source model and the representations of positive and negative samples on the target model, (ii) and the contrastive loss $\mathcal{L}_{con}^S$ between the representation e^T of the anchor sample x on the target model and the representations of positive and negative samples on the source model, as shown in Eq. (5-2).

$$\mathcal{L}_{con} = \mathcal{L}_{con}^T(e^S, e^T, e^{+T}, e^{-T}) + \mathcal{L}_{con}^S(e^T, e^S, e^{+S}, e^{-S}) \tag{5-2}$$

where e denotes the embedded hidden representation of the anchor input x, e^+ and e^- denote the representation of the positive samples x^+ and negative samples x^- respectively, S and T are used to mark representations from source and target models respectively. Specifically, $\mathcal{L}_{con}^T$ and $\mathcal{L}_{con}^S$ are calculated according to (5-3) and (5-4), respectively, where e^S and e^T are successively regarded as the anchor representation.

$$\begin{aligned}\mathcal{L}_{con}^T(e^S, e^T, \{e_j^{+T}\}_{j=1}^2, \{e_j^{-T}\}_{j=1}^N) = -[&\log h(e^S, e^T) \\ &+ \sum_{j=1}^{2} \log h(e^S, e_j^{+T}) + \sum_{j=1}^{N} \log(1 - h(e^S, e_j^{-T}))],\end{aligned} \tag{5-3}$$

$$\begin{aligned}\mathcal{L}_{con}^S(e^T, e^S, \{e_j^{+S}\}_{j=1}^2, \{e_j^{-S}\}_{j=1}^N) = -[&\log h(e^T, e^S) \\ &+ \sum_{j=1}^{2} \log h(e^T, e_j^{+S}) + \sum_{j=1}^{N} \log(1 - h(e^T, e_j^{-S}))].\end{aligned} \tag{5-4}$$

N denotes the number of negative samples, e_j^+ and e_j^- denote the embedded representations of the j-th positive sample x_j^+ and the j-th negative sample x_j^-, respectively. The contrastive score of any contrastive pair (a, b) is calculated as in (5-5)[156], with higher scores indicating greater similarity.

$$h(a, b) = \frac{score(a, b)}{score(a, b) + N \cdot P_n} = \frac{\exp(\frac{a \cdot b}{\tau})}{\exp(\frac{a \cdot b}{\tau}) + \frac{N}{M}}, \tag{5-5}$$

where M is the training dataset D^T size, P_n is the noise probability in negative samples, τ is the temperature parameter.

Distillation Loss. To guide f_θ^T in imitating the output probability distributions, o^S and o_{adv}^S, from f^S on both clean input x and adversarial input $x_1^+ = x^* = x + \delta^*$, we incorporate

a logit-based distillation loss, as defined in (5-6).

$$\begin{aligned}
&\mathcal{L}_{dit}(o^T, o^S, o^T_{adv}, o^S_{adv}) = \mathrm{KL}(o^{S^\tau}|o^{T^\tau}) + \mathrm{KL}(o^{S^\tau}|o^{T^\tau}_{adv}) + \mathrm{KL}(o^{S^\tau}_{adv}|o^{T^\tau}_{adv})) \\
&= \mathrm{KL}(f^S(x)^\tau|f^T_\theta(x)^\tau) + \mathrm{KL}(f^S_\theta(x)^\tau|f^T_\theta(x^*)^\tau) + \mathrm{KL}(f^S_\theta(x^*)^\tau|f^T_\theta(x^*)^\tau) \\
&= \mathrm{KL}(S(\frac{f^S(x)}{\tau})|S(\frac{f^T_\theta(x)}{\tau}) + \mathrm{KL}(S(\frac{f^S(x)}{\tau})|S(\frac{f^T_\theta(x^*)}{\tau}) + \mathrm{KL}(S(\frac{f^S(x^*)}{\tau}|S(\frac{f^T_\theta(x^*)}{\tau}),
\end{aligned} \tag{5-6}$$

where o denotes the output probability distribution from the softmax layer S, τ is a temperature factor to scale the logits. We avoid enforcing output dimension alignment to prevent the prediction errors of f^S on D^T from misleading f^T_θ in cross-domain TL involving different label spaces. Thus, we apply $\mathcal{L}_{dit}$ only when the label spaces of f^S and f^T_θ are consistent.

Classification Loss. To balance clean performance and robustness of f^T_θ in TL, we use a CrossEntropy-based classification loss defined in (5-7), integrating standard and adversarial training. In a data scarcity setting, weighted $\mathcal{L}_{cla}$ enables f^T to learn clean example x and adversarial example x^* with truth label y from its own domain, rather than relying solely on possibly imprecise signals from f^S.

$$\mathcal{L}_{cla}(o^T, y) = \mathcal{L}_{CE}(o^T, y) + \mathcal{L}_{CE}(o^T_{adv}, y) = \mathcal{L}_{CE}(f^T_\theta(x), y) + \mathcal{L}_{CE}(f^T_\theta(x^*), y). \tag{5-7}$$

5.4 Experimental Setup

5.4.1 Testbed

We implemented the method using PyTorch 2.2.0 and Python 3.9.18. Each experiment ran three times with varied random seeds on an NVIDIA GeForce RTX 3090 GPU with Torch Cuda 12.1. The code is available at https://github.com/RobTransfer/CARD.

5.4.2 Model Architectures

We used the WideResNet-34-10[118] as the source model and used the popular ResNet-18[114] and lightweight MobileNet[158] as the target models to evaluate robustness-preserving transfer learning (TL) across model architectures (see Table 5.1.) The source models were trained by PGD-AT[14] (defined in Eq. (2-30)) with l_∞-PGD adversarial examples (AEs) generated with $\epsilon = 0.1$, $\epsilon_{step} = 0.05$, and 20 iterations.

5.4.3 Datasets

We evaluated TL scenarios between four datasets in the network intrusion field, divided into source-domain and target-domain datasets. Detailed information is shown in Table 5.2.

Table 5.1 Model architecture and parameter information

Role	Architecture	Block Type	Total Params	Forward Size	Params Size	Total Size
Source Model	WideResNet-34-10	Residual	46,159,545	7.38M	176.08M	183.47M
Target Model	ResNet-18	Residual	11,172,297	1.29M	42.62M	43.91M
Target Model	MobileNet	Separable	3,215,625	1.69M	12.27M	13.95M

Table 5.2 Dataset information

Role	Name	ClassNum	Percentage of Target Train	Num of Limited Train-Benign	Num of Limited Train-Malicious	Num of Limited Train
Target Dataset	UNSW-NB15 (WithExploit)	2	5% 10%∼50%	2595 5189∼25945	978 1956∼9776	3573 7145∼35721
Target Dataset	UNSW-NB15 (All)	10	5% 10%∼50%	2595 5189∼25945	2704 5408∼27040	5299 10597∼52985
Source Dataset	UNSW-NB15 (NoExploit)	9	5% 10%∼50%	2595 5190∼25950	1727 3454∼17270	4322 8644∼43220
Source Dataset	NSL-KDD (All)	5	5%	3368	2931	6299

Source-domain dataset. The two datasets used as source-domain datasets in TL are UNSW-NB15 (NoExploit) and NSL-KDD (All).

- ***UNSW-NB15 (NoExploit).*** This dataset is derived from the UNSW-NB15[159] dataset, containing both *benign* traffic and 8 distinct types of network attacks. It excludes any samples from the *exploit* attack category, making it a focused subset for learning purposes without *exploit* data.

- ***NSL-KDD (All).*** This dataset is the complete version of NSL-KDD[160], which addresses the redundancy and imbalance issues in the original KDD'99[161] dataset. It includes *benign* traffic and all four major attack types: *DoS*, *Probe*, *R2L*, and *U2R*. The NSL-KDD dataset is widely used in NID research, offering a more balanced and representative distribution of attack and normal traffic samples, making it suitable for training models to detect a wide range of network intrusions.

Target-domain dataset. The two datasets used as target-domain datasets in TL are UNSW-NB15 (WithExploit) and UNSW-NB15 (All).

- ***UNSW-NB15 (WithExploit).*** This dataset is also derived from the UNSW-NB15[159] dataset, attack samples from the *exploit* attack category. It focuses specifically on

exploit-based intrusions, which involve attackers taking advantage of vulnerabilities in software or systems.

- ***UNSW-NB15 (All).*** This dataset is the complete version of UNSW-NB15[159] dataset, which includes *benign* traffic and all 9 types of attack samples, providing a comprehensive view of the dataset's attack landscape. It encompasses various categories of network attacks, including *exploit*, *fuzzers*, *backdoor*, *DoS*, *reconnaissance*, *generic*, *worms*, *shellcode*, and *analysis*. It is suitable for evaluating a model's performance across diverse and complex threat scenarios.

5.4.4 Attack Configuration

Parameters in Adversarial Attack. To evaluate the adversarial robustness of the target model, we use the l_∞-PGD algorithm to generate adversarial examples (AEs) in a white-box setting. The attack is configured with a maximum perturbation $\epsilon = 0.1$, step size $\epsilon_{step} = 0.05$, and runs for 20 iterations. This ensures the adversarial perturbations remain within the l_∞ norm constraint while effectively altering the input to fool the model.

Parameters in Natural Corruption. Random noise is applied to specific traffic features related to time, quantity, and length in the clean input. This noise follows a Gaussian distribution with a mean of 0 and a standard deviation of 1, simulating random environmental noise that can occur in a real-world network.

5.4.5 Defense Configuration

Robustness-preserving TL Baselines. We compared `CARD` with a baseline where the target model is standardly trained from scratch (StdTrain), conventional fine-tuning (FT), conventional knowledge distillation (KD), SOTA adversarial fine-tuning (such as FRFE and TWINS), and SOTA adversarial knowledge distillation (such as VAD and AAD) methods with open source code (see Table 5.3).

Parameters in Training with `CARD`. During the contrastive distillation learning with `CARD`, the coefficients α for contrastive loss function $\mathcal{L}_{con}$ (defined in Eq. (5-2)) and β for distillation loss function $\mathcal{L}_{dit}$ (defined in Eq. (5-6)) are set at 0.33, and γ for classification loss function $\mathcal{L}_{cla}$ (defined in Eq. (5-7)) is set to 0.5. $\mathcal{L}_{con}$ used a temperature τ of 0.1 with $N =$ 4,096 negative samples. The models were trained for 50 epochs, including an early stopping mechanism.

Testing Pipeline. We tested `CARD` in TL scenarios involving different model architectures and data domains according to Figure 5.2. For TL across data domains, we created

Table 5.3 Comparison of robustness-preserving transfer learning (TL) methods

Transfer Method	Robustness Transferring	Designed for NID	Evaluate Binary Classification	Evaluate Multi-Class Classification	Differences between Source and Target Tasks: Same Model Different Domains	Different Models Same Domain	Different Models Different Domains
FT [150]	○	●	●	○	●	○	○
KD [153]	○	●	●	○	○	●	○
FRFE [85]	●	○	○	●	●	○	○
TWINS [88]	●	○	○	●	●	○	○
VAD [91]	●	○	○	●	○	●	○
AAD [97]	●	○	○	●	○	●	○
CARD	●	●	●	●	●	●	●

two scenarios based on the similarity of the input feature spaces between the target-domain dataset and source-domain dataset.

- ***TL in the Same Input Feature Space.*** In this scenario the target NID dataset has the same input feature space as the source NID dataset, that is, the input features and number of dimensions are the same. This occurs when the target dataset is generated using the same type of network, protocols, and network data collection tools as the source dataset. For experiments in this scenario, we use the training samples in UNSW-NB15 (NoExploit) as source-domain training dataset, and the training samples in UNSW-NB15 (WithExploit) as target-domain training dataset.

- ***TL in Different Input Feature Spaces.*** In this scenario, the source NID dataset differs from the target NID dataset in the input feature space. This typically occurs when datasets are generated for distinct network types using different protocols, or when advances in monitoring tools lead to capturing more meaningful features, altering the feature space. Also, attack distributions may change due to older attacks becoming irrelevant from patching and entirely new attack families being introduced. For this scenario, we use the training samples in NSL-KDD (All) as source-domain training dataset and training samples in UNSW-NB15 (All) as the target-domain training set.

Particularly for network intrusion detection, we assessed two detection granularities for each transfer learning case: multi-class classification and binary classification.

- ***TL in Multi-class Classification.*** For *TL in the Same Input Feature Space*, we learn a 2-class detector on UNSW-NB15 (WithExploit) from a source 9-class detector adversarially trained on UNSW-NB15 (NoExploit). For *TL in Different Input Feature*

Spaces, we learn a 10-class detector on UNSW-NB15 (All) from a source 5-class detector adversarially trained on NSL-KDD (All).

- ***TL in Binary Classification.*** For *TL in the Same Input Feature Space*, we learn a binary classifier on UNSW-NB15 (WithExploit) from a source binary classifier adversarially trained on UNSW-NB15 (NoExploit). For *TL in Different Input Feature Spaces*, we learn a binary classifier on UNSW-NB15 (All) from a source binary classifier adversarially trained on NSL-KDD (All).

5.4.6 Evaluation Metrics

For each transfer learning case, we reported the average evaluation results of the target model across multiple runs with three random seeds {42, 43, 44}. We evaluated the performance of the target model on different versions of the target domain test samples using the following metrics: Accuracy (Acc), F1-score (F1), Recall, Precision (Precis), False Positive Rate (FPR), and False Negative Rate (FNR). Results on clean test samples (e.g CleAcc), adversarial test samples (e.g AdvAcc), and corrupted test samples (e.g CorAcc) reveal the clean performance, adversarial robustness, and natural robustness of the target model, respectively.

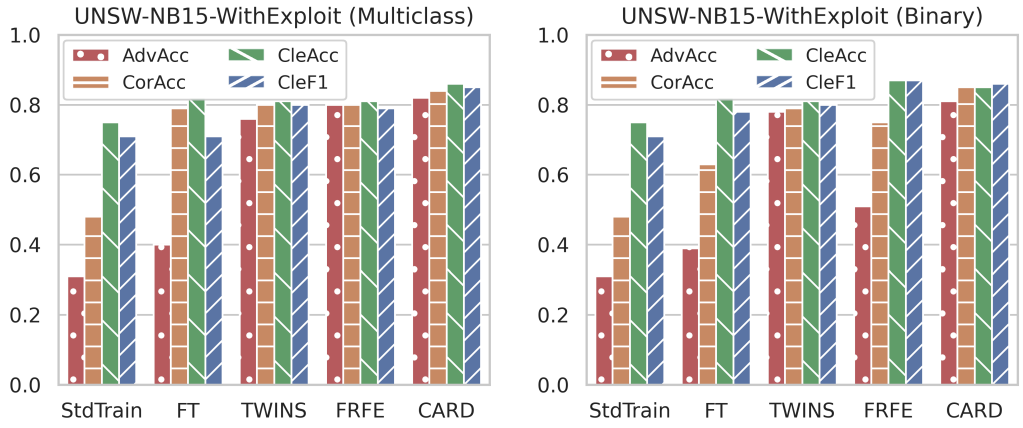
5.5 Horizontal Experimental Results and Analysis

In this section, we first evaluate the proposed method using limited target-domain training data against SOTA adversarial fine-tuning and distillation methods approaches in various TL scenarios: transfer across data domains (Section 5.5.1), transfer across model architectures (Section 5.5.2), and transfer across both domains and models (Section 5.5.3).

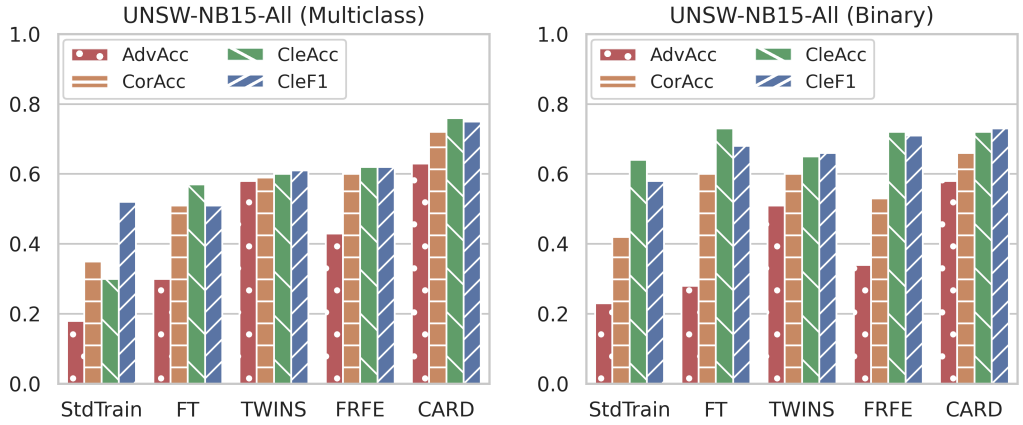
5.5.1 Comparison on Cross-domain TL Tasks with SOTA Fine-Tuning Methods Using Scarce Target Domain Training Data

In these experiments, we examine adapting a pre-trained model to build a target-domain NID model with limited target-domain training data. We compare `CARD` with a baseline in which the target model is trained from scratch (referred to as StdTrain), basic fine-tuning (FT[150]), and SOTA robustness-preserving fine-tuning methods (TWINS[88] and FRFE[85]). We use the WideResNet-34-10 source model as the target-domain classifier, adjusting only the output layer dimensions. We assess two cross-domain TL tasks: within the same input feature space and across different spaces.

Task 1. *Cross-domain TL in the Same Input Feature Space.* We trained robust multi-



(a) TL across domains with the same input space.

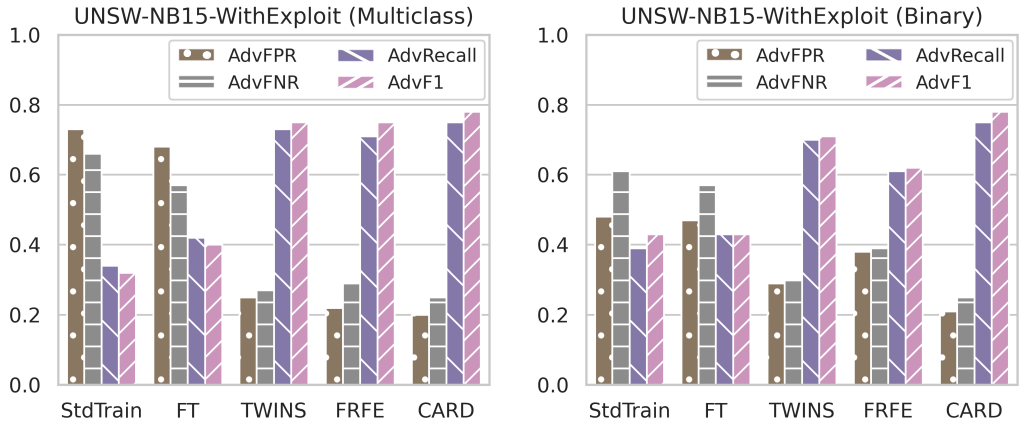


(b) TL across domains with different input spaces.

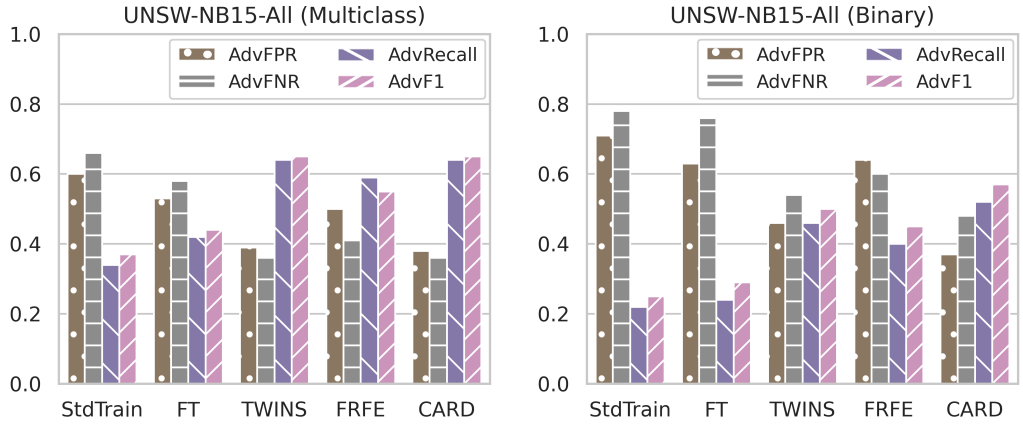
Figure 5.4 Comparison with SOTA fine-tuning methods on cross-domain TL.
(a) Generate UNSW-NB15 (WithExploit) detectors by fine-tuning robust UNSW-NB15 (NoExploit) detectors.
(b) Generate UNSW-NB15 (All) detectors by fine-tuning robust NSL-KDD (All) detectors.

class and binary NID models using PGD-AT on UNSW-NB15 (NoExploit) with 100% labeled data, and used them as source models to learn target-domain models on UNSW-NB15 (WithExploit) with only 5% (3573) labeled data.

Results. See Figure 5.4 (a) and Figure 5.5 (a) for results. `CARD` shows major improvements over StdTrain in both robustness and regular predictive performance. For multi-class classification, AdvAcc and CorAcc increase by 51.14% and 46.33%, while StdAcc and F1 scores rise by 10.63% and 14.25%, respectively. For binary detection, `CARD` achieves improvements for the AdvAcc (50.14%), CorAcc (36.87%), StdAcc (9.96%), and F1 (14.81%) metrics. In contrast, the baseline FT[150] falls short in preserving robustness, yielding comparatively modest gains in AdvAcc (9.08%) and StdAcc (7.94%). Compared to adversarial fine-tuning methods, in multi-class classification, `CARD` achieves 6.33% and 2.33% higher AdvAcc compared to TWINS[88] and FRFE[85], respectively, and 3.67% and 4.33% higher



(a) TL across domains with the same input space.



(b) TL across domains with different input spaces.

Figure 5.5 Adversarial comparison with SOTA fine-tuning methods on cross-data domain TL.
The experimental setup is the same as Figure 5.4. Using finer-grained metrics for adversarial robustness.

CorAcc. This advantage is more pronounced in binary classification, where CARD outperforms TWINS and FRFE by 3.33% and 30% in AdvAcc, and 5.33% and 9.67% in CorAcc, respectively.

Task 2. *Cross-domain TL in Different Input Feature Spaces.* We use the same settings as the previous task, but replace the source-domain dataset with NSL-KDD (All) and the target-domain dataset with UNSW-NB15 (All), using only 5% (5299) of the labeled target-domain training data.

Results. Results are presented in Figure 5.4 (b) and Figure 5.5 (b). CARD significantly outperforms StdTrain, boosting AdvAcc and CorAcc by 39.19% and 37.52% in multi-class classification, and showing notable advancements in binary detection under various metrics. Also, CARD improves over FT, with an increase in AdvAcc (27.5% and 29.65% in multi-class and binary classification), and superior regular performance, exhibiting 18.76% higher

StdAcc and 23.97% higher F1 in multi-classification and comparable results in binary classification. Compared with TWINS and FRFE, CARD also has superior performance. In multi-class classification, AdvAcc is equivalent to TWINS and notably higher than FRFE, while CorAcc is 13.33% and 12% higher, respectively. In the binary case, AdvAcc is 7.33% and 24% higher, and CorAcc is 5.33% and 13% higher, respectively.

Summary. *These results show that CARD outperforms baseline FT and SOTA adversarial FT methods in generating robust and accurate target models with limited training data. The advantages are also demonstrated in other robustness evaluation metrics on adversarial samples, like advRecall, advFNR, etc.*

5.5.2 Comparison on Cross-model TL Tasks with SOTA Distillation Methods Using Scarce Target Domain Training Data

We explore TL scenarios where a lightweight target model (student) but with a compact target-domain task is trained to inherit the robustness of a larger source model (teacher). Both source and target models are trained on the same dataset, with the source model using 100% of the training data and the target model distilled with only 5%. We compare CARD against StdTrain, basic distillation (KD[153]), and SOTA robustness-preserving methods (VAD[91], AAD[97]) on two datasets, UNSW-NB15 (NoExploit) and NSL-KDD (All). The source model is distilled into ResNet-18 and MobileNet (outlined in Table 5.1).

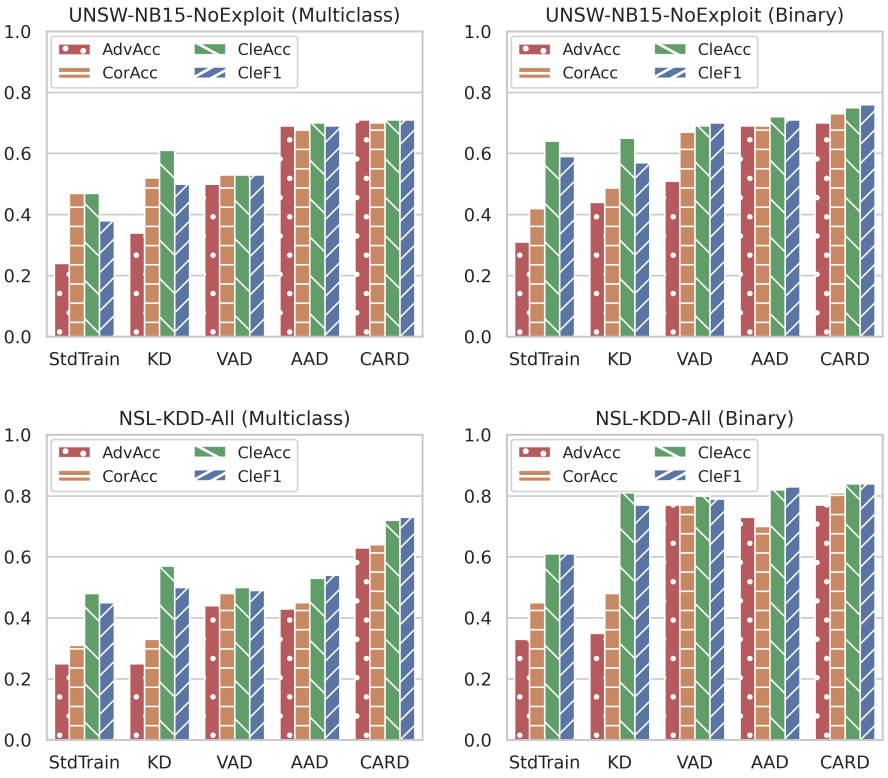
Task 1. *TL across Models with Similar Building Blocks.* We trained separate robust multi-class and binary NID models using PGD-AT on UNSW-NB15 (NoExploit) with 100% training data. These models served as sources to distill both multi-class and binary NID models based on ResNet-18, using only 5% (4322) of the training data from UNSW-NB15 (NoExploit) or 5% (6299) of the training data from NSL-KDD-All.

Results. Figure 5.6 (a) and Figure 5.7 (a) show the results. CARD surpasses other methods in multi-class and binary classification scenarios. Compared to KD, AdvAcc and CorAcc increase by 37.81% and 18.00% in the multi-class case, and by 26.67% and 24.00% in the binary case. This suggests that a target model derived from a robust source model via standard KD-based TL does not inherit robustness. Compared to AD methods, CARD improves AdvAcc and CorAcc by 11.67% and 10.00% on average in the multi-class case, and by 10.00% and 4.67% in the binary case. This shows CARD surpasses SOTA in robustness transfer across architectures.

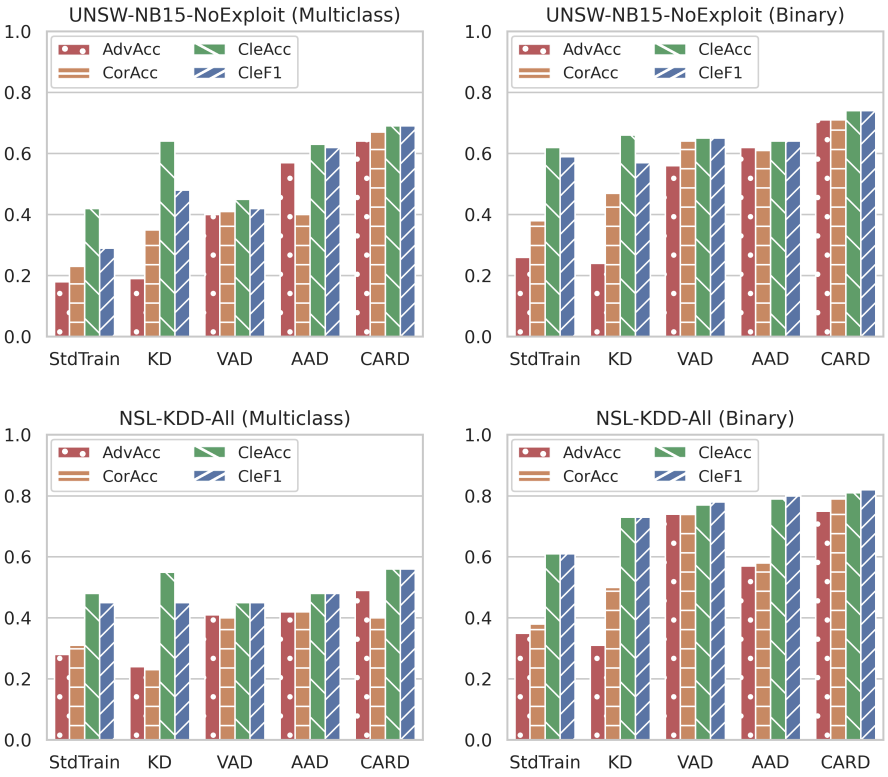
Task 2. *TL across Models with Different Building Blocks.* We use the same settings as the previous task, except for replacing the target domain model with MobileNet.

Results. Results are presented in Figure 5.6 (b) and Figure 5.7 (b). In comparing StdTrain and CARD on MobileNet, CARD consistently outperforms StdTrain in both robustness and regular performance. Particularly in binary classification, the performance gap is more evident than in multi-class classification. Furthermore, compared to the distillation methods, CARD shows superior robustness preservation over KD, VAD, and AAD. Compared to KD, AdvAcc and CorAcc increase by 44.67% and 31.33% in the multi-class case, and by 47.33% and 24.00% in the binary case. Compared to AD methods, CARD boosts AdvAcc and CorAcc by 15.17% and 26.00% on average in the multi-class case, and by 12.00% and 8.33% in the binary case.

Summary. *CARD proves its effectiveness in maintaining robustness while preserving generalization in lightweight NID models with limited training data, surpassing baseline KD and SOTA methods like VAD and AAD. Its robustness strengths extend to other metrics on adversarial traffic, like advF1, advFPR, etc.*



(a) TL across models with similar building blocks.

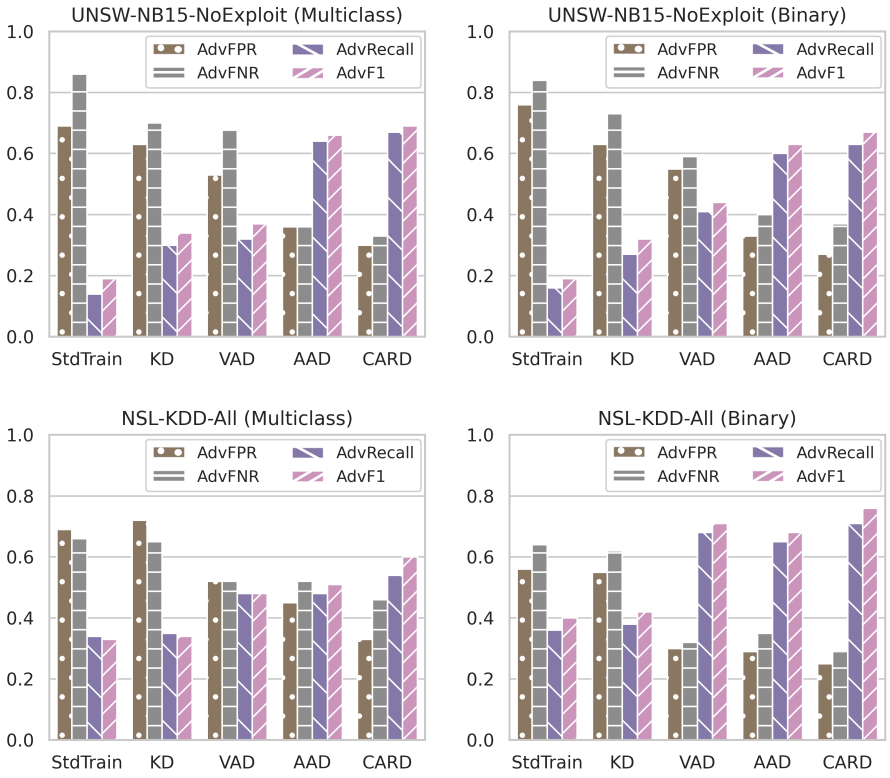


(b) TL across models with different building blocks.

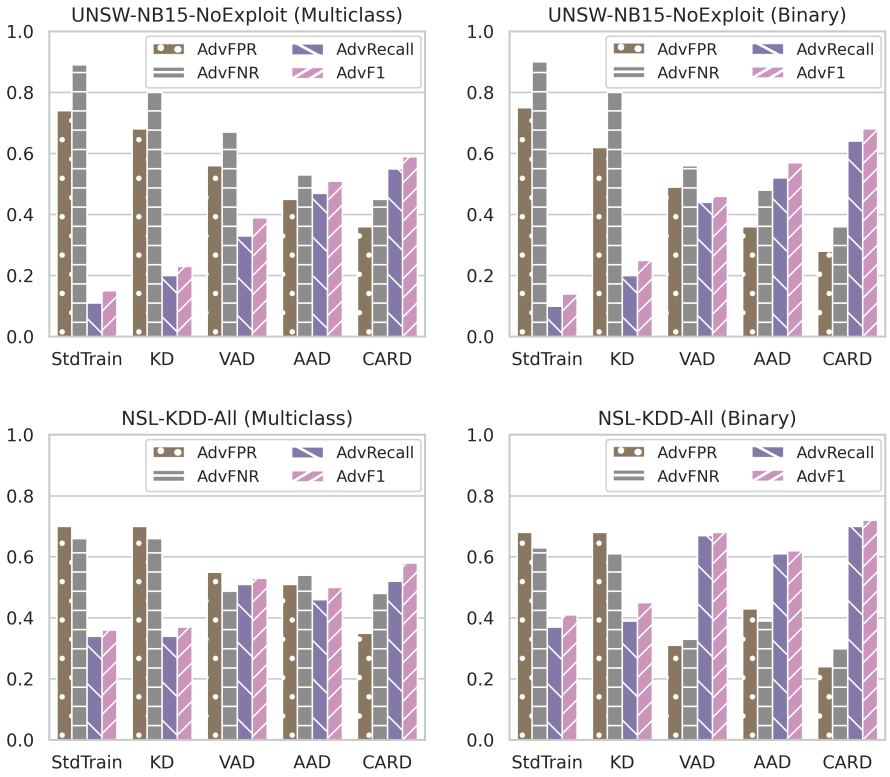
Figure 5.6 Comparison with SOTA distillation methods on cross-model TL.

(a) Distill robust WideResNet-34-10-based UNSW-NB15-NoExploit and NSL-KDD-All detectors to ResNet-18.

(b) Distill robust WideResNet-34-10-based UNSW-NB15-NoExploit and NSL-KDD-All detectors to MobileNet.



(a) TL across models with similar building blocks.



(b) TL across models with different building blocks.

Figure 5.7 Adversarial comparison with SOTA distillation methods on cross-model architecture TL.
The experimental setup is the same as Figure 5.6. Using finer-grained metrics for adversarial robustness.

5.5.3 Comparison on Cross-domain-and-model TL Tasks with Training from Scratch Using Scarce Target Domain Training Data

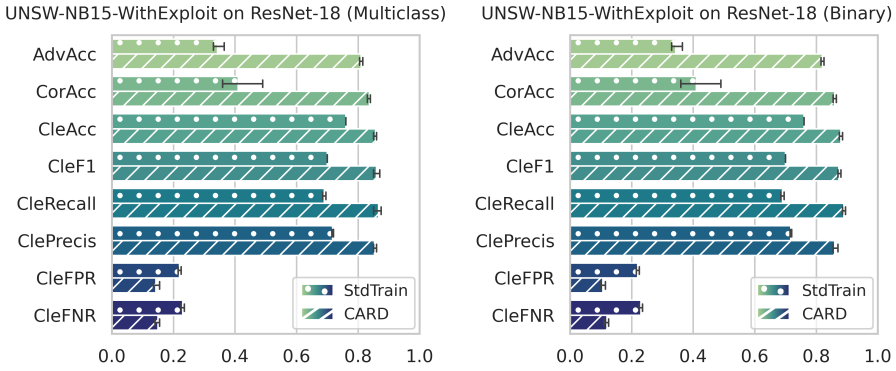
These experiments address TL tasks across data domains and model architectures, which are crucial for real-world NID due to dynamic networks, emerging threats, and resource constraints. Given the absence of tailored robustness-preserving TL methods, we evaluated CARD's performance on two target-domain models, ResNet-18 and MobileNet, for TL tasks within the same input feature space and across different input feature spaces.

Task 1. *Cross-domain-and-model TL in the Same Input Feature Space.* We use the same setup introduced in Section 5.5.1, but replace the target model with ResNet-18 and MobileNet, yielding four TL tasks across domains and models, including TL from the multi-class NID model on UNSW-NB15 (NoExploit) to ResNet-18 and MobileNet on UNSW-NB15 (WithExploit), and their binary detection versions.

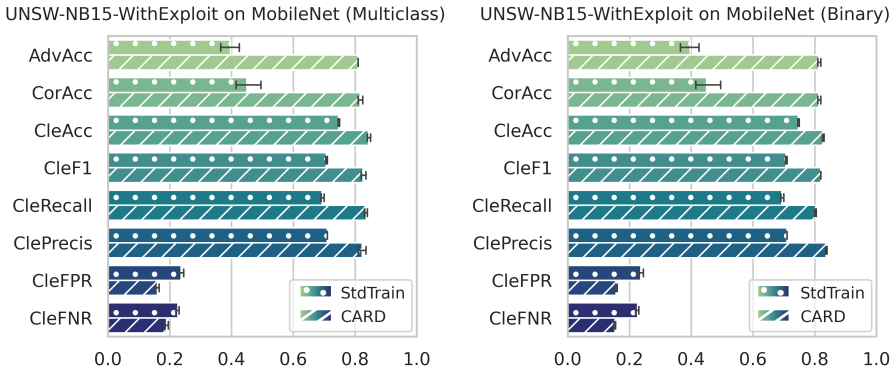
Results. See Figure 5.8 (a) (b) and Figure 5.9 (a) (b) for results. CARD demonstrates remarkable advantages in robustness compared to StdTrain. In multi-class classification on ResNet-18, CARD achieves AdvAcc and CorAcc rates of 81% and 84%, surpassing StdTrain by significant margins. On MobileNet, although the advantage slightly decreases due to the larger structural difference, CARD still outperforms StdTrain by 41.45% and 36.42% in AdvAcc and CorAcc, respectively. Moreover, CARD shows superior generalization on clean samples, with recall rates reaching 87% on ResNet-18 and 84% on MobileNet, substantially higher than StdTrain. CARD's FPR on both models is notably lower than StdTrain.

Task 2. *Cross-domain-and-model TL in Different Input Feature Spaces.* We use the setup introduced in Section 5.5.1, except for replacing the target-domain model with ResNet-18 and MobileNet successively, creating four TL tasks across different data domains and model architectures. This includes TL from the multi-class NID model on NSL-KDD (All) to ResNet-18 and MobileNet on UNSW-NB15 (All), along with their binary classification.

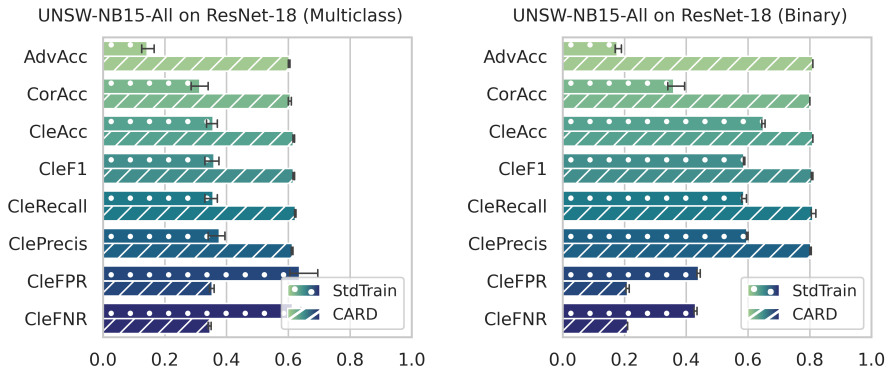
Results. Results are presented in Figure 5.8 (c) (d) and Figure 5.9 (c) (d). The performance of the target model in this scenario decreases compared to TL in the same input feature space. The reason is that NSL-KDD (All) has a wider range of categories than UNSW-NB15 (WithExploit), making training more complex. Nevertheless, CARD outperforms StdTrain on all metrics with lower standard deviations. Robustness metrics highlight our method's superiority, with AdvAcc and CorAcc reaching 60% and 61% respectively on ResNet-18, and 60% and 54% respectively on MobileNet, which are substantially higher than StdTrain in multi-class detection. In the binary detection, CARD's AdvAcc surges to 81% on ResNet-18 and MobileNet, indicating a substantial lead of 63% and 57% over StdTrain, respectively.



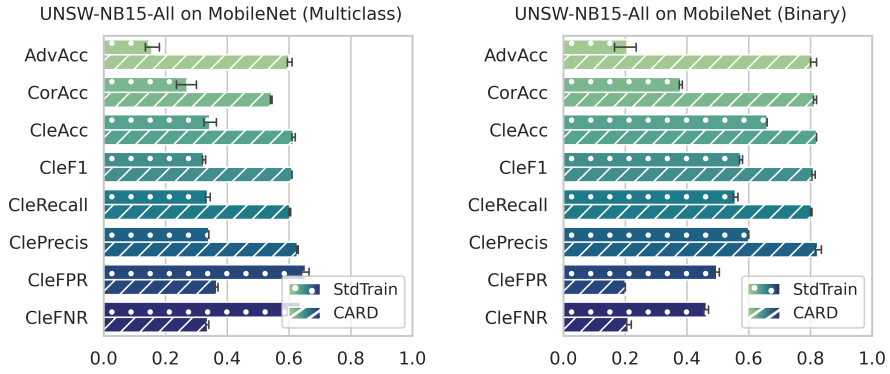
(a) TL across domains with same input space and models with similar blocks.



(b) TL across domains with same input space and models with different blocks.

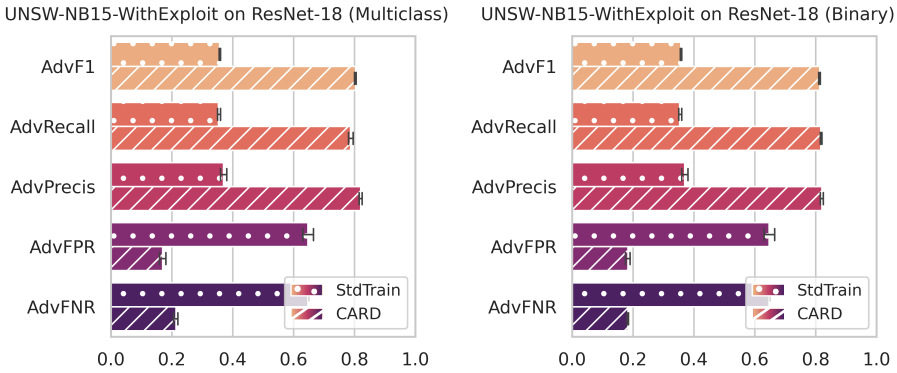


(c) TL across domains with different input spaces and models with similar blocks.

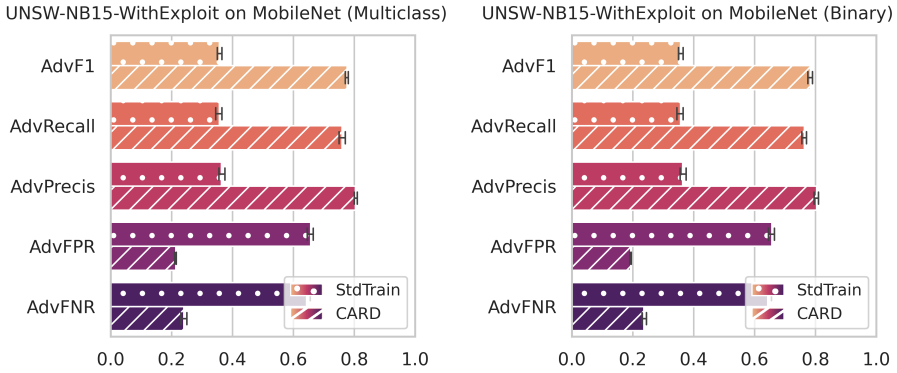


(d) TL across domains with different input spaces and models with different blocks.

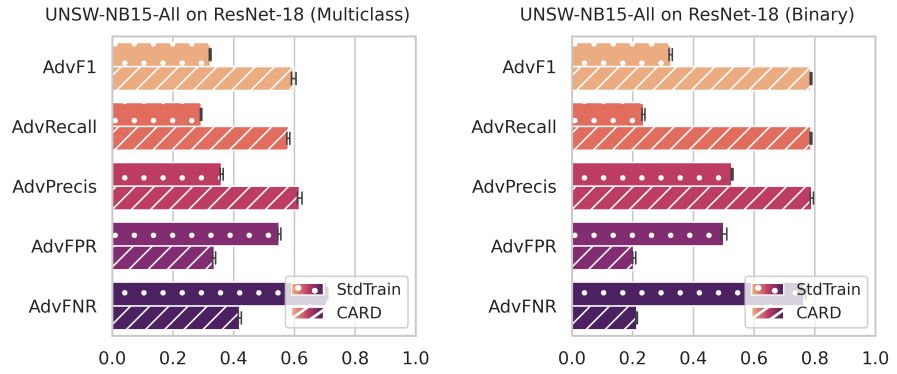
Figure 5.8 Comparison with standard training from scratch on cross-domain&model TL.



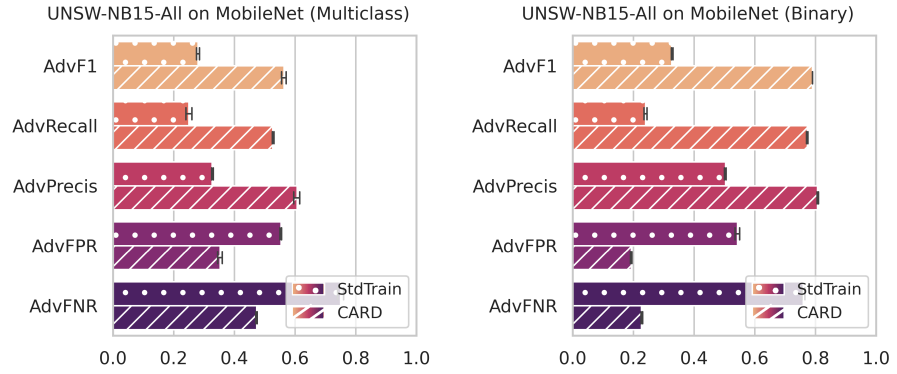
(a) TL across domains with same input space and models with similar blocks.



(b) TL across domains with same input space and models with different blocks.



(c) TL across domains with different input spaces and models with similar blocks.



(d) TL across domains with different input spaces and models with different blocks.

Figure 5.9 Adversarial comparison with standard training from scratch on cross-domain&model TL.

Moreover, CARD also balances regular predictive performance well. Compared with StdTrain, CARD reduces the FNR of malicious traffic and the FPR of benign traffic by 28.2% and 26.92% on ResNet-18, and by 28.87% and 29.39% on MobileNet in multi-class detection.

Summary. *These results highlight CARD's efficacy and flexibility in preserving robustness when learning target models with differing data domains and lighter architectures compared to the source model, significantly outperforming training from scratch.*

5.6 Vertical Experimental Results and Analysis

In prior experiments, we assessed TL under limited training conditions by utilizing only 5% of the target-domain training samples. In this section, we mainly investigate how the number of target-domain training samples affects the model's adversarial robustness against adversarial examples and its regular predictive performance on clean examples. We examine the impact of target-domain training data number on TL tasks across data domains (Section 5.6.1), model architectures (Section 5.6.2), and both(Section 5.6.3), respectively using the setups introduced in Section 5.5.1, Section 5.5.2, and Section 5.5.3. We evaluate CARD's performance as the number of training samples increases, incrementally varying the percentages to $\{5\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$ (see Table 5.2). We compare CARD with StdTrain, SOTA adversarial fine-tuning, and SOTA adversarial distillation methods under the most challenging TL tasks, with results averaged from three random seeds. Finally, we conduct ablation studies on the proposed loss function.

5.6.1 Impact of Target Domain Training Data Number on Cross-domain TL Tasks

We compare CARD against other methods under various amounts of data on the UNSW-NB15 (All) multi-class task from an NSL-KDD (All)-pretrained model because this is the most challenging cross-domain TL task we have performed in Section 5.5.1. We will see how different methods change and how CARD compares with StdTrain, basic fine-tuning (FT[150]), and SOTA robustness-preserving fine-tuning methods (TWINS[88] and FRFE[85]) as data increases.

Results. As shown in Figure 5.10, StdTrain shows the most significant improvement, heavily relying on ground truth data for effective generalization. While other TL methods also improve with more data, their progress is slower than StdTrain due to their dependence on both the data and the source model. Among the TL methods, CARD stands out, consistently outperforming as more data is added. However, the benefits of TL diminish as data

availability increases, with StdTrain eventually matching or even surpassing some TL methods. At 5% data, CARD outperformed StdTrain by 23.61% in F1 and 37.92% in CorrAcc, but as more data became available, the differences narrowed to 13.33% and 34.51%, respectively. In terms of FPR, we see a reduction across the board, with StdTrain benefiting the most from increased exposure to malicious data. Its FNR dropped from 45% to 38%, marking the greatest improvement. As more data was introduced, StdTrain overtook VAD in Precision, surpassing it after 40% of the data was available. Nonetheless, according to the results of AdvAcc, robust-preserving TL techniques are still very valuable against adversarial robustness, of which CARD is the most significant, as StdTrain has limited gains in this regard even with more training data.

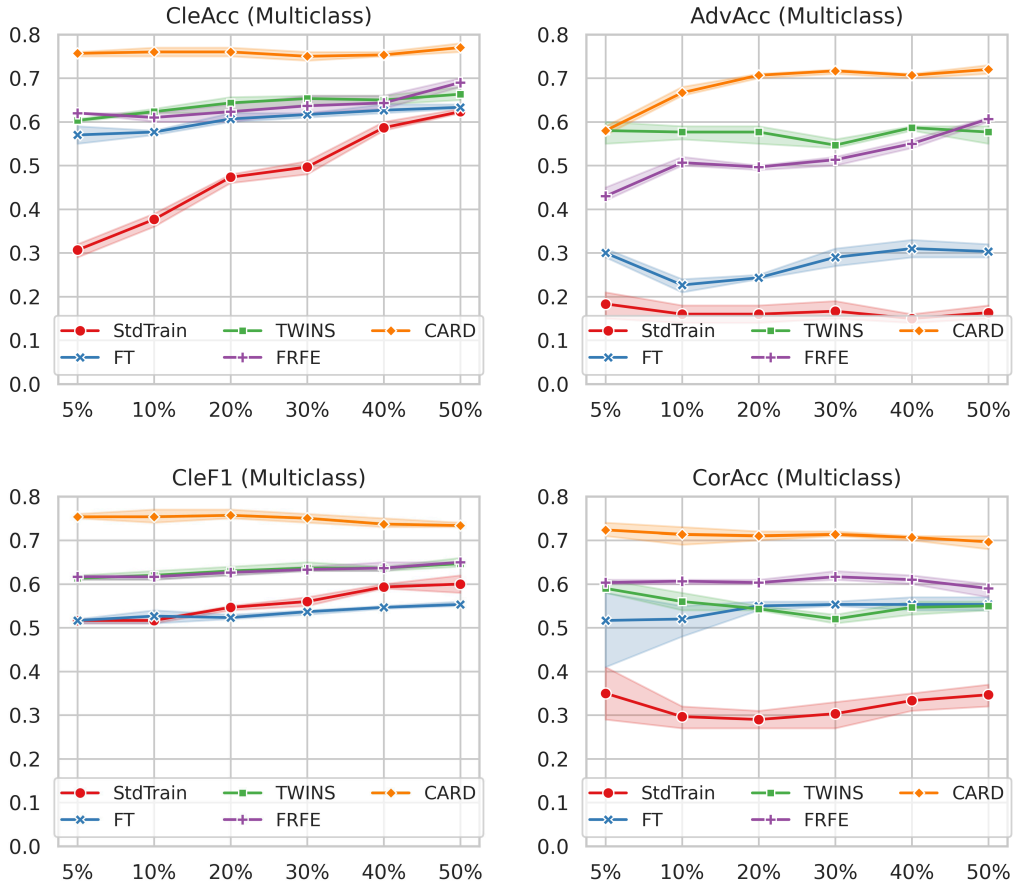


Figure 5.10 Impact of target-domain training data amount on cross-domain TL.
Generate a 10-class UNSW-NB15 (All) detector by fine-tuning a robust 5-class NSL-KDD (All) detector.

5.6.2 Impact of Target Domain Training Data Number on Cross-model TL Tasks

Likewise, we compare CARD against other methods under various amounts of data on the NSL-KDD (All) multi-class task to MobileNet from the source model, as this is the most challenging cross-model TL task we have performed in Section 5.5.2. We aim to compress a multi-class classifier into a lightweight one with different basic model blocks. We will see how different methods change and how CARD stands against StdTrain, basic distillation (KD[153]), and SOTA robustness-preserving distillation methods (VAD[91] and AAD[97]) as training data size increases.

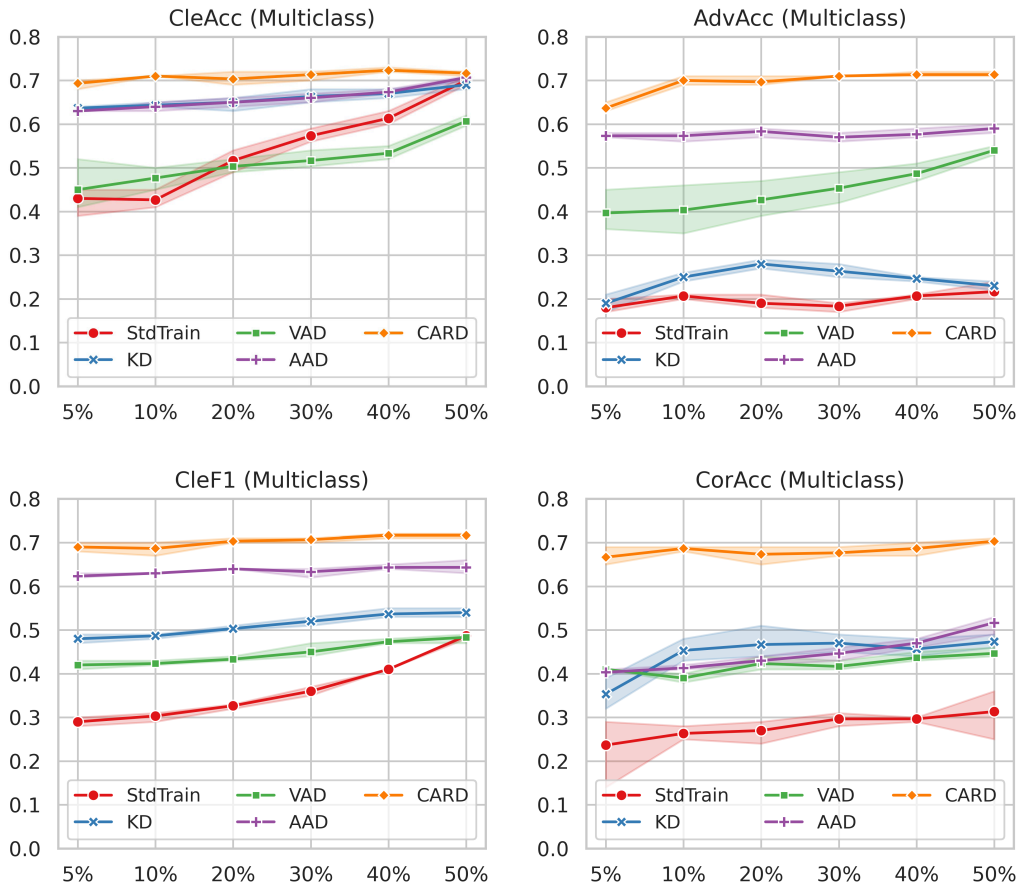


Figure 5.11 Impact of target-domain training data amount on cross-model TL.
Distill a robust WideResNet-34-10-based 9-class UNSW-NB15 (NoExploit) detector to MobileNet.

Results. As shown in Figure 5.11, the results demonstrate steady improvement for CARD as training data increases. TL methods remain advantageous in limited data scenarios, highlighting the value of TL when little data is available. For CorAcc, the gap between CARD and StdTrain narrows from 43.34% at 5% of training data to 38.87% at 50%, showing only slight improvement. In terms of FPR, CARD outperforms StdTrain by 21.33% at 50% data,

compared to a 21.18% gap at 5%. Across various metrics, we observe that StdTrain, lacking a robustness mechanism, sees limited benefit from more data. TL methods, on the other hand, while slowly losing advantage as more data becomes available, they remain strong. Among them, CARD continues to excel, maintaining a 9.19% lead over the best SOTA method (AAD) across multiple metrics.

5.6.3 Impact of Target Domain Training Data Number on Cross-domain-and-model TL Tasks

We compared CARD with StdTrain across different amounts of training data in a challenging cross-domain and cross-model TL task. In this TL scenario, the target model, MobileNet, is more lightweight than the source model, WideResNet-34-10, and the input feature space shifts from the 5-class NSL-KDD (All) to the 10-class UNSW-NB15 (All).

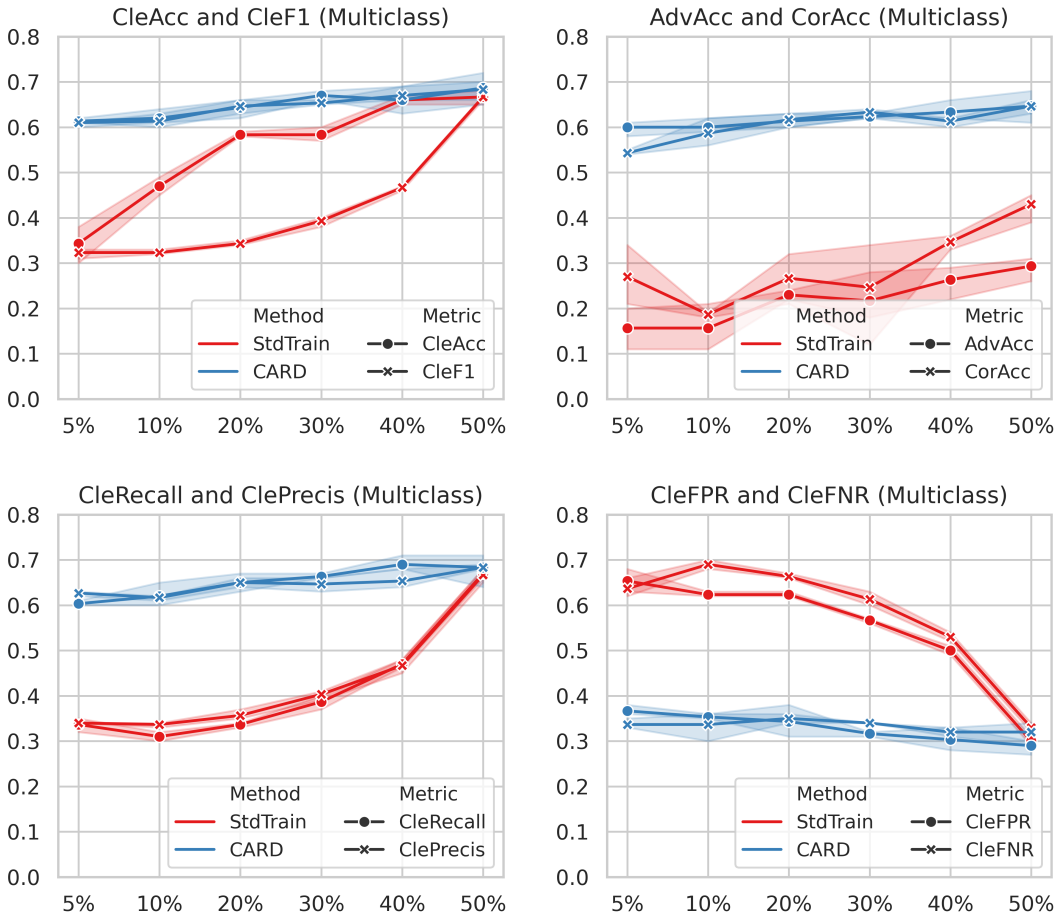


Figure 5.12 Impact of target-domain training data amount on cross-domain-and-model TL. Generate a MobileNet-based 10-class UNSW-NB15 (All) detector from a WideResNet-34-10-based robust 5-class NSL-KDD (All) detector.

Results. Results in Figure 5.12 show that StdTrain requires more training data for reasonable performance than CARD. We can see that The performance increase from CARD is

gentle, with no significant gap between using little data and abundant data. With 50% data, CARD performs better on StdAcc with lower FPR and FNR, but the improvement is not much higher than with 5% data. This indicates that CARD effectively transfers robustness and predictive knowledge from a pre-trained source model even with limited target-domain data. When the performance gained through TL from source models is analogous to having abundant data, TL becomes meaningful and is worth doing. Additionally, increased training data does not compromise the robustness advantage of CARD.

Summary. *Even with more target-domain training data, CARD consistently excels in adversarial robustness and regular predictive performance compared to StdTrain and SOTA TL methods across various challenge TL tasks. Moreover, as training data increases, CARD's advantage in generalization is gradually caught up, but its robustness lead remains difficult to narrow by others.*

5.6.4 Ablation Study on Loss Function

Loss of CARD is composed of three components: contrastive loss $\mathcal{L}_{con}$, distillation loss $\mathcal{L}_{dit}$, and classification loss $\mathcal{L}_{cla}$. In this subsection, we analyze how the different components of loss $\mathcal{L}_{card} = \alpha \cdot \mathcal{L}_{con} + \beta \cdot \mathcal{L}_{dit} + \gamma \cdot \mathcal{L}_{cla}$ (described in Eq. (5-1)) can work independently or in combination.

We use cross-model architecture TL tasks to assess the performance of CARD under various influences because in these tasks, all three components are used. When the source and target-domain datasets have very different properties, such as distinct output categories and different numbers of classes, the distillation loss $\mathcal{L}_{dit}$ based on KL divergence fails to accommodate output logical value vectors with differing dimensions for distance measurement. Given that forcing the output dimensions of the source model to be aligned with the target model can potentially introduce significant noise, to avoid the source model misleading the target model, we opt not to use distillation loss in the cross-domain transfer task.

We assess the performance of CARD on two datasets, UNSW-NB15 (NoExploit) and NSL-KDD (All). We test the role each loss term plays individually or in combination. We use the same dataset for both the source and target domains, training the source model with 100% of the data and distilling the target model with only 5%. This allows us to assess the effectiveness of CARD in handling data scarcity. The average results across three random seeds (41, 42, 43) are reported.

Task 1. *Cross-model TL on Various Losses using UNSW-NB15 (NoExploit).* When distilling WideResNet-34-10-based robust UNSW-NB15 (NoExploit) detectors to ResNet-

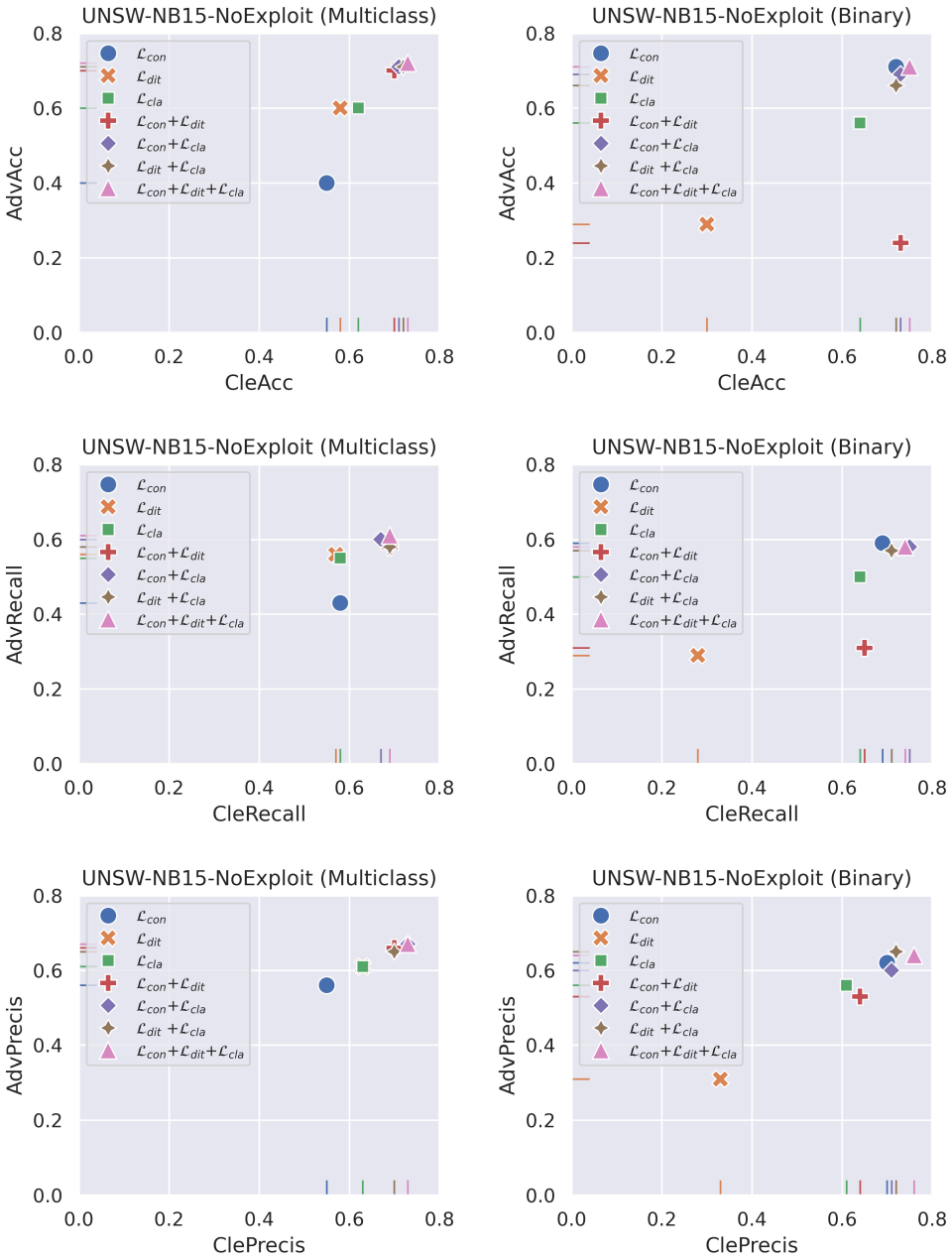


Figure 5.13 Ablation study on loss function in cross-model TL tasks on UNSW-NB15 (NoExploit).
Distill robust UNSW-NB15 (NoExploit) detectors based on WideResNet-34-10 to ResNet-18.

18, which includes multi-class and binary classification cases, we observe that employing all three losses simultaneously consistently yields the optimal results.

Results. See Figure 5.13 for results. When the $\mathcal{L}_{cla}$ is used with $\mathcal{L}_{dit}$, the results are comparable to that of the $\mathcal{L}_{cla}$ combined with the $\mathcal{L}_{con}$. This is reasonable because $\mathcal{L}_{con}$ attempts to achieve a similar knowledge distillation goal as $\mathcal{L}_{dit}$ at the hidden representation level, which makes it free from the constraint that the source and target models have to have the same number of categories and thus can be used on any TL task, including cross-domain TL tasks. Through our experiments, we have found that the $\mathcal{L}_{dit}$ tends to be more effective

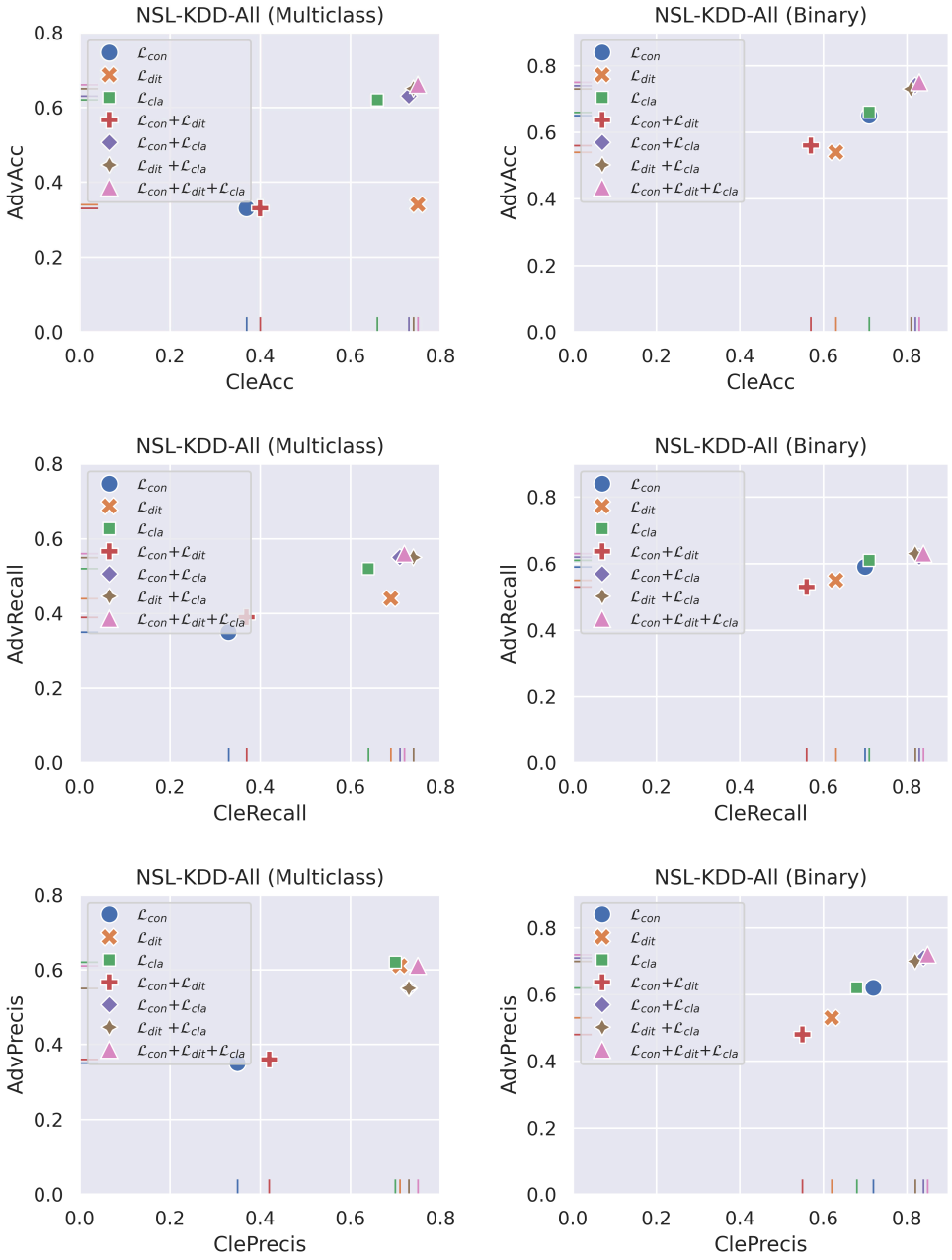


Figure 5.14 Ablation study on loss function in cross-model TL tasks on NSL-KDD (All).
Distill robust NSL-KDD (All) detectors based on WideResNet-34-10 to ResNet-18.

for training binary classification models while $\mathcal{L}_{con}$ tends to be more effective for multi-class NID models. And classification loss $\mathcal{L}_{cla}$ is crucial in improving the regular predictive prediction performance measured by StdAcc. Overall, integrating these losses can improve the predictive performance and robustness of the model generally.

Task 2. *Cross-model TL on Various Losses using NSL-KDD (All).* In these experiments, we distill robust WideResNet-34-10-based NSL-KDD (All) detectors to ResNet-18 for both multi-class and binary classification cases. We observed analogous trends and performance as previous task, indicating a stable behavior of our loss function across different datasets.

From Figure 5.14, it is evident that across various TL tasks, whether multi-class classification or binary classification on the target-domain dataset, CARD achieves the best results when all three losses are combined. Also, we observe that combining $\mathcal{L}_{dit}$ and $\mathcal{L}_{con}$ yields sub-optimal outcomes. This combination may hinder effective optimization as each loss pulls parameter updates in divergent directions. For instance, distillation loss might enforce a distribution shape incongruent with that enforced by $\mathcal{L}_{con}$. However, incorporating $\mathcal{L}_{cla}$ mitigates this issue by introducing ground truth, thereby directing optimization toward accurate class predictions. Such a way ensures the target model can stably learn the underlying representation, thereby better balancing the regular performance and robustness of the model.

5.6.5 Evaluation on Adaptive Attack

In robustness-preserving TL, the hybrid and implicit nature of the target model's robustness origins make it difficult for an attacker, even with access to the white-box target model's parameters, to infer the complete defense algorithm, such as the robust source model's parameters and TL strategy. Nevertheless, considering sophisticated threats that may arise in the real world, we enhanced the adversary's capabilities and conducted adaptive attacks against the target model for further robustness evaluation. Adaptive attacks, unlike static evasion attacks, dynamically adjust attack strategies to counter defense mechanisms[57], posing a greater threat. We designed two adaptive strategies (see Fig. 5.15) and evaluated models trained with 30 epochs in the multi-class case (see Fig. 5.16, Fig. 5.17, and Fig. 5.18).

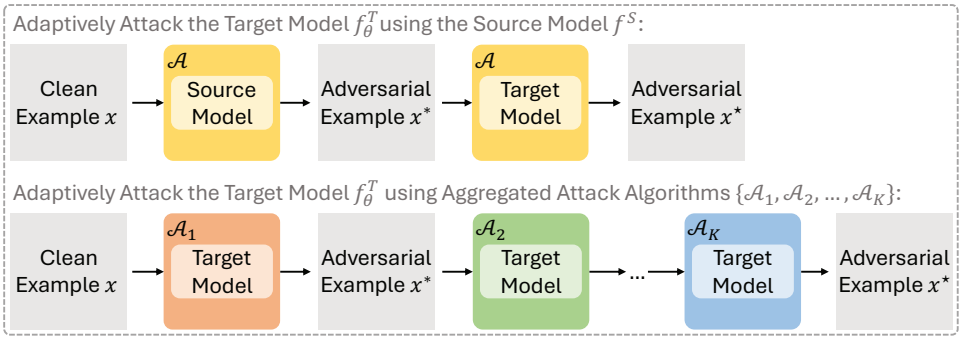


Figure 5.15 Adaptive attack strategies.

Task 1. *Source Model-based Adaptive Attack (SM-Adapt).* This adaptive strategy assumes the attacker has white-box access to f^S, f_θ^T, training dataset D^T, and knowledge of the perturbation algorithm $\mathcal{A}$ (e.g., PGD) used in the adversarial pretraining of f^S. ① Attacker generates adversarial examples x^* for target domain using f^S and $\mathcal{A}$ with 20 iterations. ② x^* undergoes further perturbation via f_θ^T and $\mathcal{A}$, producing adaptive adversarial examples $x^\star$. ③ In cross-domain TL, due to mismatched input features or label spaces, the attacker must

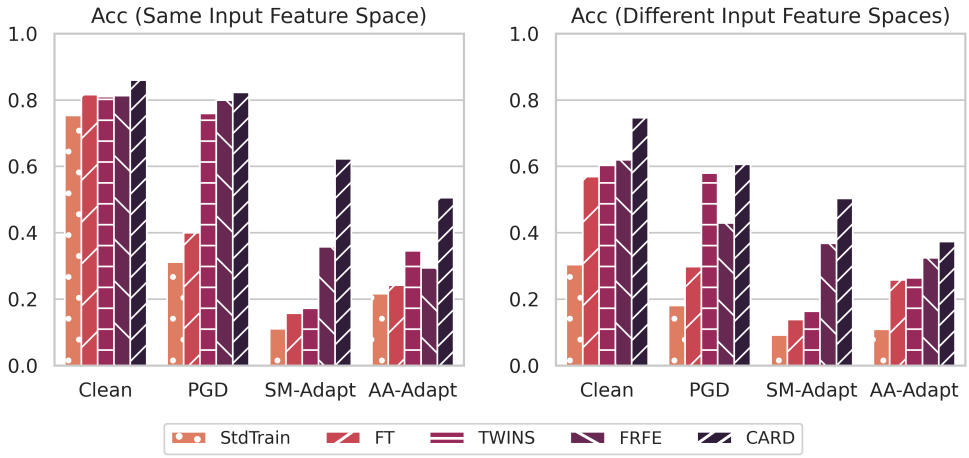


Figure 5.16 Robustness against adaptive attacks in cross-domain TL.

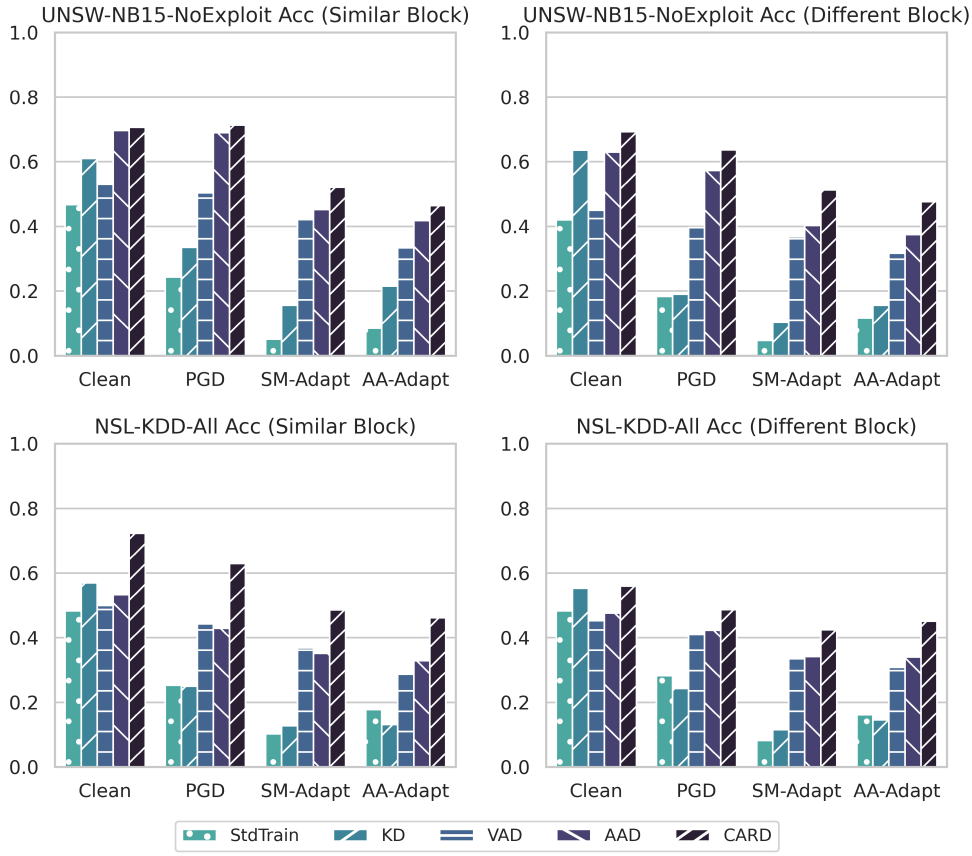


Figure 5.17 Robustness against adaptive attacks in cross-model TL.

first fine-tune f^S with D^T before generating perturbations.

Task 2. *Aggregated Attacks-based Adaptive Attack (AA-Adapt).* This strategy assumes that the attacker has white-box access to f_θ^T and knows that its robustness partly stems from $\mathcal{A}$ (e.g., PGD). The attacker seeks to enhance the attack by sequentially applying attacks $\{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$ on x using f_θ^T. We employed the AutoAttack[32] framework. In our implementation of $K = 2$ setting, ① attacker first generates x^* using AutoPGD[32] alongside

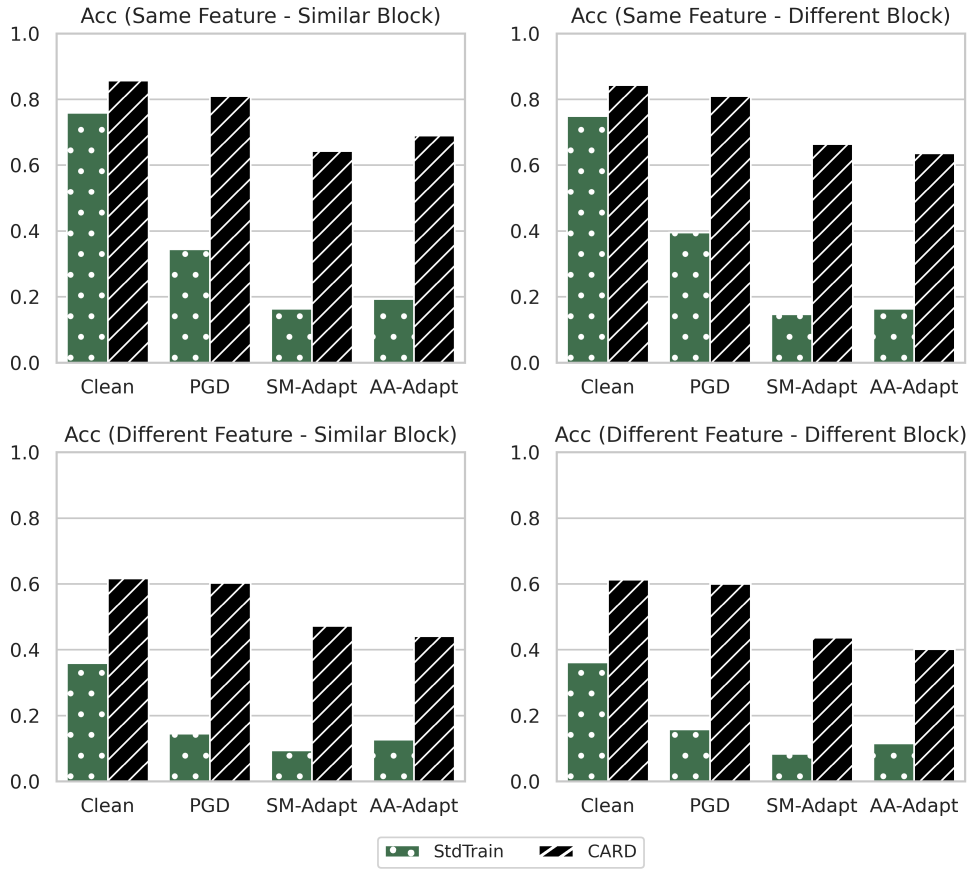


Figure 5.18 Robustness against adaptive attacks in cross-domain-and-model TL.

iterations 100. ② x^* are then perturbed using DeepFool[12] with 100 iterations, finally producing $x^\star$. The results show that CARD consistently outperforms baselines in robustness to adaptive attacks across various TL tasks.

5.6.6 Evaluation of Time Cost

We compared the training time of 50 epochs of CARD and baselines in the data scarcity setting for multi-class detection.

Results. Compared to robustness-preserving TL baselines, CARD trains faster than AAD[97] (2023) and TWINS[88] (2023) but slower than FRFE[85] (2020) and VAD[91] (2020).

- Cross-domain TL: Average per-epoch time for StdTrain, FT, TWINS, FRFE, and CARD is 1.13 seconds (s), 1.11s, 40.01s, 1.14s, and 20.58s, respectively.
- Cross-model TL: Average per-epoch time for StdTrain, KD, VAD, AAD, and CARD is 0.35s, 0.76s, 3.14s, 15.68s, and 4.40s, respectively.
- Cross-domain-and-model TL: Average per-epoch training time for StdTrain and CARD is 0.27s and 3.15s, respectively.

Moreover, we found the lighter the target model, the lower the overhead introduced by CARD. Using ResNet or MobileNet as target model reduces time by an average of 80.89% compared to retaining WideResNet from the source model. MobileNet, being the lighter option, trains 20.39% faster on average than ResNet. In summary, CARD outperforms baselines in robustness and clean performance, with a slight time cost over standard TL but greater efficiency than TWINS and AAD.

5.7 Summary

In this chapter, we introduce Contrastive Adversarial Representation Distillation (CARD), the first robustness-preserving transfer learning method tailored for DNN-based network intrusion detection. CARD produces an accurate and robust target model by leveraging a robust source model, addressing two key challenges: limited target-domain training data and the robustness of the target model against adversarial attacks and natural noise. The contrastive transfer learning algorithm, based on robustness-aware views, captures domain-invariant robustness through adversarial and natural corruption views. Additionally, the adaptive dimension alignment mechanism enables the flexible transfer of both adversarial and regular knowledge across different data domains and model architectures. Extensive experiments demonstrate the effectiveness of CARD in transferring adversarial robustness across data domains (e.g., fine-tuning) and model structures (e.g., distillation), as well as in more complex tasks involving lightweight target models with limited training data. In the future, we will explore its application in transferring robustness from larger foundation models.

Chapter VI Conclusion and Future Work

6.1 Conclusion

Recent advancements in deep learning (DL) have transformed fields like computer vision and cybersecurity. Deep neural networks (DNNs), with their multi-layered architectures, are adept at capturing complex patterns from large datasets, making them integral to modern AI systems. However, the opacity in DNN decision-making has led to significant security risks, particularly adversarial attacks. Subtle adversarial manipulations on the data can severely degrade model performance while evading detection, which poses a serious threat to applications such as autonomous driving and intrusion detection. This dissertation focuses on enhancing DNN adversarial robustness against adversarial attacks, addressing three key issues: generalizing adversarial robustness to unknown attack types, providing tight guarantees for adversarial robustness, and effectively transferring adversarial robustness across different tasks.

Our primary contributions are as follows:

- To enhance the generalization of adversarial robustness against unknown attacks, Latent Representation Mixup (`LarepMixup`) is proposed as a robust training framework utilizing on-manifold and off-manifold mixed examples. This approach includes a data augmentation technique based on multi-mode manifold interpolation to generate mixed samples near the decision boundary using convex and binary mask mixing. Additionally, a multi-label training algorithm leverages mixed semantic samples and labels to smooth the decision boundary. Experiments demonstrate that `LarepMixup` improves both pixel-level and representation-level robustness in white-box and black-box scenarios, enhancing generalization across various input and latent space perturbations.

- To tighten the certification of adversarial robustness in DNNs, Multi-order Adaptive Randomized Smoothing (`MARS`) is proposed as a certified defense framework that leverages high-order information to extend the certified robust region. This framework includes an adaptive randomized smoothing algorithm that uses zero-order and first-order information to calculate robust radii, achieving tighter lower bounds of robustness than existing methods. Additionally, a dimension-wise robust radius calculation algorithm based on feature sensitivity allows for fine-grained robustness certification across heterogeneous input features. Experiments demonstrate that `MARS` effectively

certifies adversarial robustness in larger l_p norm-bounded perturbation regions, enhancing certified robustness against various adversarial attacks and natural corruption.

- To facilitate the transfer of adversarial robustness in DNNs across different tasks, Contrastive Adversarial Representation Distillation (`CARD`) is proposed as a robustness-preserving transfer learning framework utilizing robustness-aware contrastive views. It features a distillation strategy based on adaptive dimensional alignment, enabling knowledge transfer amid changes in data domains and models. Additionally, a contrastive transfer learning algorithm leverages adversarial manipulation and natural corruption views to capture domain-invariant robustness. Experiments demonstrate that `CARD` enhances the transferability of adversarial robustness across data domains and model structures, achieving superior adversarial robustness and regular predictive performance in lightweight target-domain models with limited training data.

In conclusion, this dissertation contributes to improving the adversarial robustness of DNNs by addressing critical challenges in generalization, certification, and transferability. The proposed methods not only advance the understanding of adversarial robustness but also provide practical solutions for deploying secure DNN-based systems in dynamic and high-stakes environments such as autonomous driving and cybersecurity。

6.2 Future Work

This dissertation emphasizes the development of innovative defense techniques to counter adversarial attacks within deep neural networks. The rapid advancement of DL and foundation models (FMs) introduces both challenges and opportunities for future research. Below are several promising directions for further investigation.

Firstly, while robust training based on latent mixup strategies has been effectively demonstrated using images, there remains substantial potential for extending these applications to other input domains. Future research could explore more advanced generative models, such as Variational Autoencoder (VAE), diffusion models, Transformer, and Generative Pre-trained Transformer (GPT), to study interpolation strategies applicable to diverse data types, including network traffic, text, and video. For these specific data domains, the focus will be on learning latent representations and creating mixed samples that possess realistic characteristics. Applying latent mixup in non-image domains may necessitate the design of customized methods to capture unique disentangled representations on the data manifold. However, the potential for enhancing adversarial robustness through mixup training is significant, mainly

as this defense strategy does not rely on any adversarial knowledge. Optimization of mixup training methods across various domains is expected to lead to more DL models with improved adversarial robustness and generalization capabilities.

Secondly, as FMs increasingly integrate into various applications, ensuring their adversarial robustness against adversarial attacks is of paramount importance. Future work could focus on extending robustness certification methods to larger vision-text FMs, which present unique challenges due to their multimodal nature. Current certification techniques often struggle to assess robustness across multiple data modalities jointly. Developing a novel framework to evaluate the adversarial robustness of these models against adversarial attacks will be crucial, particularly by considering the interactions between visual and textual inputs. This endeavor will involve integrating advanced certification strategies that account for the heterogeneous nature of text data and the homogeneous nature of image data, ensuring that robustness guarantees are comprehensive and contextually relevant. Furthermore, exploring methods that leverage shared representations between modalities could enable a unified approach to certification, enhancing the overall adversarial robustness of vision-text FMs.

Finally, exploring techniques for transferring adversarial robustness from FMs to downstream models is promising. Achieving adversarial robustness often requires large-scale training data and significant computational resources. Complex and resource-intensive FMs, like the Contrastive Language–Image Pretraining (CLIP) model and GPT-4, can provide valuable representation learning capabilities for relatively lightweight downstream models. However, matching category spaces and accommodating diverse input data modalities will pose challenges in this transfer process. Developing effective strategies for such situations can encourage efficient exploitation of the robustness advantages gained from large FMs in practical applications where computational efficiency is critical. Research in this area will help advance the deployment of secure DL models across various industries, ensuring they remain efficient and robust against adversarial attacks.

Bibliography

[1] RAWAT W, WANG Z. Deep convolutional neural networks for image classification: A comprehensive review[J]. Neural Computation, 2017, 29(9): 2352-2449.

[2] MIRSKY Y, DOITSHMAN T, ELOVICI Y, et al. Kitsune: An ensemble of autoencoders for online network intrusion detection[C]//Network and Distributed Systems Security (NDSS) Symposium. 2018: 1-15.

[3] LIU H, LANG B. Machine learning and deep learning methods for intrusion detection systems: A survey[J]. Applied Sciences, 2019, 9(20): 4396.

[4] ZHANG C, PATRAS P, HADDADI H. Deep learning in mobile and wireless networking: A survey [J]. IEEE Communications Surveys & Tutorials, 2019, 21(3): 2224-2287.

[5] DIALLO A F, PATRAS P. Adaptive clustering-based malicious traffic classification at the network edge[C]//IEEE Conference on Computer Communications (INFOCOM). 2021: 1-10.

[6] YANG L, GUO W, HAO Q, et al. Cade: Detecting and explaining concept drift samples for security applications[C]//USENIX Security Symposium (USENIX). 2021: 2327-2344.

[7] 张玉清, 董颖, 柳彩云, 等. 深度学习应用于网络空间安全的现状, 趋势与展望[J]. 计算机研究与发展, 2018, 55(6): 1117-1142.

[8] KHAN R U, ZHANG X, ALAZAB M, et al. An improved convolutional neural network model for intrusion detection in networks[C]//IEEE Cybersecurity and Cyberforensics Conference (CCC). 2019: 74-77.

[9] GOODFELLOW I J, SHLENS J, SZEGEDY C. Explaining and harnessing adversarial examples [C]//International Conference on Learning Representations (ICLR). 2015: 1-11.

[10] 张思思, 左信, 刘建伟. 深度学习中的对抗样本问题[J]. 计算机学报, 2019, 42(8): 1886-1904.

[11] PAPERNOT N, MCDANIEL P, JHA S, et al. The limitations of deep learning in adversarial settings [C]//IEEE European Symposium on Security and Privacy (EuroS&P). 2016: 372-387.

[12] MOOSAVI-DEZFOOLI S M, FAWZI A, FROSSARD P. Deepfool: A simple and accurate method to fool deep neural networks[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 2574-2582.

[13] CARLINI N, WAGNER D. Towards evaluating the robustness of neural networks[C]//IEEE Symposium on Security and Privacy (S&P). 2017: 39-57.

[14] MADRY A, MAKELOV A, SCHMIDT L, et al. Towards deep learning models resistant to adversarial attacks[C]//International Conference on Learning Representations (ICLR). 2018: 1-23.

[15] CHEN P Y, SHARMA Y, ZHANG H, et al. Ead: Elastic-net attacks to deep neural networks via

adversarial examples[C]//AAAI Conference on Artificial Intelligence (AAAI). 2018: 1-8.

[16] JIA W, LU Z, ZHANG H, et al. Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems[C]//Network and Distributed Systems Security (NDSS) Symposium. 2022: 1-17.

[17] YANG Y, GAO R, LI Y, et al. What you see is not what the network infers: Detecting adversarial examples based on semantic contradiction[C]//Network and Distributed Systems Security (NDSS) Symposium. 2022: 1-20.

[18] HAN D, WANG Z, ZHONG Y, et al. Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors[J]. IEEE Journal on Selected Areas in Communications (JSAC), 2021, 39(8): 2632-2647.

[19] WANG N, CHEN Y, HU Y, et al. Manda: On adversarial example detection for network intrusion detection system[C]//IEEE Conference on Computer Communications (INFOCOM). 2021: 1-10.

[20] NASR M, BAHRAMALI A, HOUMANSADR A. Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations[C]//USENIX Security Symposium (USENIX). 2021: 2705-2722.

[21] ZHANG C, COSTA-PEREZ X, PATRAS P. Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms[J]. IEEE/ACM Transactions on Networking (TON), 2022, 30(3): 1294-1311.

[22] YAN H, LI X, ZHANG W, et al. Automatic evasion of machine learning-based network intrusion detection systems[J]. IEEE Transactions on Dependable and Secure Computing (TDSC), 2023, 21 (1): 153-167.

[23] LIAO F, LIANG M, DONG Y, et al. Defense against adversarial attacks using high-level representation guided denoiser[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2018: 1778-1787.

[24] XIE C, WU Y, MAATEN L V D, et al. Feature denoising for improving adversarial robustness[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 501-509.

[25] 郭敏, 曾颖明, 于然, 等. 基于对抗训练和 VAE 样本修复的对抗攻击防御技术研究[J]. 信息网络安全, 2019, 9: 66-70.

[26] SZEGEDY C, ZAREMBA W, SUTSKEVER I, et al. Intriguing properties of neural networks[C]// International Conference on Learning Representations (ICLR). 2014: 1-10.

[27] KURAKIN A, GOODFELLOW I, BENGIO S, et al. Adversarial examples in the physical world [C]//International Conference on Learning Representations (ICLR). 2017: 1-13.

[28] MOOSAVI-DEZFOOLI S M, FAWZI A, FAWZI O, et al. Universal adversarial perturbations[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 1765-1773.

[29] BALUJA S, FISCHER I. Adversarial transformation networks: Learning to generate adversarial examples[A]. 2017: 1-13.

[30] SARKAR S, BANSAL A, MAHBUB U, et al. Upset and angri: Breaking high performance image classifiers[A]. 2017: 1-9.

[31] SU J, VARGAS D V, SAKURAI K. One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation (TEVC), 2019, 23(5): 828-841.

[32] CROCE F, HEIN M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks[C]//International Conference on Machine Learning (ICML). 2020: 2206-2216.

[33] CROCE F, HEIN M. Minimally distorted adversarial examples with a fast adaptive boundary attack [C]//International Conference on Machine Learning (ICML). 2020: 2196-2205.

[34] ANDRIUSHCHENKO M, CROCE F, FLAMMARION N, et al. Square attack: A query-efficient black-box adversarial attack via random search[C]//European Conference on Computer Vision (ECCV). 2020: 486-501.

[35] CHALUPKA K, PERONA P, EBERHARDT F. Visual causal feature learning[C]//The Conference on Uncertainty in Artificial Intelligence (UAI). 2015: 181-190.

[36] HUANG R, XU B, SCHUURMANS C, Dale anFGSMd Szepesvári. Learning with a strong adversary[A]. 2015: 1-12.

[37] SHAHAM U, YAMADA Y, NEGAHBAN S. Understanding adversarial training: Increasing local stability of neural nets through robust optimization[A]. 2015: 1-12.

[38] KURAKIN A, GOODFELLOW I, BENGIO S. Adversarial machine learning at scale[C]// International Conference on Learning Representations (ICLR). 2017: 1-17.

[39] ZHANG H, YU Y, JIAO J, et al. Theoretically principled trade-off between robustness and accuracy [C]//International Conference on Machine Learning (ICML). 2019: 7472-7482.

[40] STUTZ D, HEIN M, SCHIELE B. Disentangling adversarial robustness and generalization[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 6976-6987.

[41] LIN W A, LAU C P, LEVINE A, et al. Dual manifold adversarial robustness: Defense against lp and non-lp adversarial attacks[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 1-12.

[42] ZHANG H, CISSE M, DAUPHIN Y N, et al. Mixup: Beyond empirical risk minimization[C]// International Conference on Learning Representations (ICLR). 2018: 1-13.

[43] LEE S, LEE H, YOON S. Adversarial vertex mixup: Toward better adversarially robust generalization[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020: 272-281.

[44] PANG T, XU K, ZHU J. Mixup inference: Better exploiting mixup to defend adversarial attacks [C]//International Conference on Learning Representations (ICLR). 2020: 1-14.

[45] YUN S, HAN D, OH S J, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features[C]//IEEE/CVF International Conference on Computer Vision (ICCV). 2019: 6023-6032.

[46] KIM J H, CHOO W, SONG H O. Puzzle mix: Exploiting saliency and local statistics for optimal mixup[C]//International Conference on Machine Learning (ICML). 2020: 5275-5285.

[47] GUO H, MAO Y, ZHANG R. Mixup as locally linear out-of-manifold regularization[C]//AAAI Conference on Artificial Intelligence (AAAI). 2019: 3714-3722.

[48] VERMA V, LAMB A, BECKHAM C, et al. Manifold mixup: Better representations by interpolating hidden states[C]//International Conference on Machine Learning (ICML). 2019: 6438-6447.

[49] FARAMARZI M, AMINI M, BADRINAARAAYANAN A, et al. Patchup: A feature-space block-level regularization technique for convolutional neural networks[C]//AAAI Conference on Artificial Intelligence (AAAI). 2022: 1-19.

[50] LIU X, ZOU Y, KONG L, et al. Data augmentation via latent space interpolation for image classification[C]//IEEE International Conference on Pattern Recognition (ICPR). 2018: 728-733.

[51] MAKHZANI A, SHLENS J, JAITLY N, et al. Adversarial autoencoders[A]. 2015: 1-10.

[52] SAINBURG T, THIELK M, THEILMAN B, et al. Generative adversarial interpolative autoencoding: Adversarial training on latent space interpolations encourage convex latent distributions[A]. 2018: 1-15.

[53] BERTHELOT D, RAFFEL C, ROY A, et al. Understanding and improving interpolation in autoencoders via an adversarial regularizer[C]//International Conference on Learning Representations (ICLR). 2019: 1-19.

[54] CEMGIL T, GHAISAS S, DVIJOTHAM K D, et al. Adversarially robust representations with smooth encoders[C]//International Conference on Learning Representations (ICLR). 2020: 1-20.

[55] KINGMA D P, WELLING M. Auto-encoding variational bayes[C]//International Conference on Learning Representations (ICLR). 2014: 1-14.

[56] BECKHAM C, HONARI S, VERMA V, et al. On adversarial mixup resynthesis[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-12.

[57] TRAMER F, CARLINI N, BRENDEL W, et al. On adaptive attacks to adversarial example defenses [C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 1633-1645.

[58] LI L, XIE T, LI B. Sok: Certified robustness for deep neural networks[C]//IEEE Symposium on Security and Privacy (S&P). 2022: 94-115.

[59] CHENG C H, NÜHRENBERG G, RUESS H. Maximum resilience of artificial neural networks

[C]//International Symposium on Automated Technology for Verification and Analysis (ATVA). 2017: 251-268.

[60] TJENG V, XIAO K Y, TEDRAKE R. Evaluating robustness of neural networks with mixed integer programming[C]//International Conference on Learning Representations (ICLR). 2019: 1-21.

[61] ZHANG H, WENG T W, CHEN P Y, et al. Efficient neural network robustness certification with general activation functions[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2018: 1-10.

[62] GOWAL S, DVIJOTHAM K D, STANFORTH R, et al. Scalable verified training for provably robust image classification[C]//IEEE/CVF International Conference on Computer Vision (ICCV). 2019: 4842-4851.

[63] ZHANG H, CHEN H, XIAO C, et al. Towards stable and efficient training of verifiably robust neural networks[C]//International Conference on Learning Representations (ICLR). 2019: 1-25.

[64] XU K, SHI Z, ZHANG H, et al. Automatic perturbation analysis for scalable certified robustness and beyond[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 1129-1141.

[65] LYU Z, GUO M, WU T, et al. Towards evaluating and training verifiably robust neural networks [C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021: 4308-4317.

[66] WANG S, ZHANG H, XU K, et al. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2021: 29909-29921.

[67] ZHANG H, WANG S, XU K, et al. General cutting planes for bound-propagation-based neural network verification[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2022: 1656-1670.

[68] ZHANG R, SUN J. Certified robust accuracy of neural networks are bounded due to bayes errors [C]//International Conference on Computer Aided Verification. Springer, 2024: 352-376.

[69] LEE G H, YUAN Y, CHANG S, et al. Tight certificates of adversarial robustness for randomly smoothed classifiers[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-12.

[70] WONG E, KOLTER Z. Provable defenses against adversarial examples via the convex outer adversarial polytope[C]//International Conference on Machine Learning (ICML). 2018: 5286-5295.

[71] JORDAN M, LEWIS J, DIMAKIS A G. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-23.

[72] FROMHERZ A, LEINO K, FREDRIKSON M, et al. Fast geometric projections for local robustness certification[C]//International Conference on Learning Representations (ICLR). 2020: 1-15.

[73] LIM C H, URTASUN R, YUMER E. Hierarchical verification for adversarial robustness[C]// International Conference on Machine Learning (ICML). 2020: 6072-6082.

[74] VORÁČEK V, HEIN M. Provably adversarially robust nearest prototype classifiers[C]// International Conference on Machine Learning (ICML). 2022: 22361-22383.

[75] LEE S, LEE J, PARK S. Lipschitz-certifiable training with a tight outer bound[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 16891-16902.

[76] FAZLYAB M, ROBEY A, HASSANI H, et al. Efficient and accurate estimation of lipschitz constants for deep neural networks[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-12.

[77] LECUYER M, ATLIDAKIS V, GEAMBASU R, et al. Certified robustness to adversarial examples with differential privacy[C]//IEEE Symposium on Security and Privacy (S&P). 2019: 656-672.

[78] PHAN H, THAI M T, HU H, et al. Scalable differential privacy with certified robustness in adversarial learning[C]//International Conference on Machine Learning (ICML). 2020: 7683-7694.

[79] COHEN J, ROSENFELD E, KOLTER Z. Certified adversarial robustness via randomized smoothing[C]//International Conference on Machine Learning (ICML). 2019: 1310-1320.

[80] HAO Z, YING C, DONG Y, et al. Gsmooth: Certified robustness against semantic transformations via generalized randomized smoothing[C]//International Conference on Machine Learning (ICML). 2022: 8465-8483.

[81] MOHAPATRA J, KO C Y, WENG T W, et al. Higher-order certification for randomized smoothing [C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 4501-4511.

[82] WANG K, WANG Z, HAN D, et al. Bars: Local robustness certification for deep learning based traffic analysis systems[C]//Network and Distributed Systems Security (NDSS) Symposium. 2023: 1-18.

[83] HENDRYCKS D, LEE K, MAZEIKA M. Using pre-training can improve model robustness and uncertainty[C]//International Conference on Machine Learning (ICML). 2019: 2712-2721.

[84] XU X, ZHANG J Y, MA E, et al. Adversarially robust models may not transfer better: Sufficient conditions for domain transferability from the view of regularization[C]//International Conference on Machine Learning (ICML). 2022: 24770-24802.

[85] SHAFAHI A, SAADATPANAH P, ZHU C, et al. Adversarially robust transfer learning[C]// International Conference on Learning Representations (ICLR). 2020: 1-14.

[86] FAN L, LIU S, CHEN P Y, et al. When does contrastive learning preserve adversarial robustness from pretraining to finetuning?[C]//Annual Conference on Neural Information Processing Systems

(NeurIPS). 2021: 21480-21492.

[87] YAMADA Y, OTANI M. Does robustness on imagenet transfer to downstream tasks?[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022: 9215-9224.

[88] LIU Z, XU Y, JI X, et al. Twins: A fine-tuning framework for improved transferability of adversarial robustness and generalization[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023: 16436-16446.

[89] XIE C, TAN M, GONG B, et al. Adversarial examples improve image recognition[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020: 819-828.

[90] XU X, ZHANG J, KANKANHALLI M. Autolora: A parameter-free automated robust fine-tuning framework[A]. 2023: 1-13.

[91] GOLDBLUM M, FOWL L, FEIZI S, et al. Adversarially robust distillation[C]//AAAI Conference on Artificial Intelligence (AAAI): Vol. 34. 2020: 3996-4003.

[92] MUHAMMAD A, ZHOU F, XIE C, et al. Mixacm: Mixup-based robustness transfer via distillation of activated channel maps[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2021: 4555-4569.

[93] ZHU J, YAO J, HAN B, et al. Reliable adversarial distillation with unreliable teachers[C]// International Conference on Learning Representations (ICLR). 2021: 1-15.

[94] ZI B, ZHAO S, MA X, et al. Revisiting adversarial robustness distillation: Robust soft labels make student better[C]//IEEE/CVF International Conference on Computer Vision (ICCV). 2021: 16443-16452.

[95] MAROTO J, ORTIZ-JIMÉNEZ G, FROSSARD P. On the benefits of knowledge distillation for adversarial robustness[A]. 2022: 1-31.

[96] BAI T, ZHAO J, WEN B. Guided adversarial contrastive distillation for robust students[J]. IEEE Transactions on Information Forensics and Security (TIFS), 2023, 43: 9643-9655.

[97] HUANG B, CHEN M, WANG Y, et al. Boosting accuracy and robustness of student models via adaptive adversarial distillation[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023: 24668-24677.

[98] HINTON G E, SALAKHUTDINOV R R. Reducing the dimensionality of data with neural networks [J]. Science, 2006, 313(5786): 504-507.

[99] GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning[M]. MIT press, 2016.

[100] COHEN U, CHUNG S, LEE D D, et al. Separability and geometry of object manifolds in deep neural networks[J]. Nature Communications, 2020, 11(1): 1-13.

[101] SEUNG H S, LEE D D. The manifold ways of perception[J]. Science, 2000, 290(5500): 2268-2269.

[102] JALAL A, ILYAS A, DASKALAKIS C, et al. The robust manifold defense: Adversarial training

using generative models[A]. 2017: 1-19.

[103] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[A]. 2015: 1-9.

[104] GUO H. Nonlinear mixup: Out-of-manifold data augmentation for text classification[C]//AAAI Conference on Artificial Intelligence (AAAI). 2020: 4044-4051.

[105] CHEN J, YANG Z, YANG D. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification[C]//Annual Meeting of the Association for Computational Linguistics (ACL). 2020: 2147-2157.

[106] ZHAO J, WEI P, MAO W. Robust neural text classification and entailment via mixup regularized adversarial training[C]//International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). 2021: 1778-1782.

[107] KARRAS T, AITTALA M, HELLSTEN J, et al. Training generative adversarial networks with limited data[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 12104-12114.

[108] KARRAS T, LAINE S, AILA T. A style-based generator architecture for generative adversarial networks[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 4401-4410.

[109] PASZKE A, GROSS S, MASSA F, et al. Pytorch: An imperative style, high-performance deep learning library[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-12.

[110] NICOLAE M I, SINN M, TRAN M N, et al. Adversarial robustness toolbox v1.0.0[A]. 2018: 1-34.

[111] DING G W, WANG L, JIN X. Advertorch v0.1: An adversarial robustness toolbox based on pytorch [A]. 2019: 1-6.

[112] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017: 1-19.

[113] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]//International Conference on Learning Representations (ICLR). 2015: 1-14.

[114] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 770-778.

[115] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 4700-4708.

[116] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2015: 1-9.

[117] HE K, ZHANG X, REN S, et al. Identity mappings in deep residual networks[C]//European Conference on Computer Vision (ECCV). 2016: 630-645.

[118] ZAGORUYKO S, KOMODAKIS N. Wide residual networks[A]. 2017: 1-15.

[119] KRIZHEVSKY A, HINTON G, et al. Learning multiple layers of features from tiny images[M]. Toronto, ON, Canada, 2009.

[120] NETZER Y, WANG T, COATES A, et al. Reading digits in natural images with unsupervised feature learning[C]//Annual Conference on Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning and Unsupervised Feature Learning. 2011: 4-4.

[121] KANG D, SUN Y, HENDRYCKS D, et al. Testing robustness against unforeseen adversaries[A]. 2019: 1-23.

[122] GULSHAD S, METZEN J H, SMEULDERS A. Adversarial and natural perturbations for general robustness[A]. 2020: 1-13.

[123] TAORI R, DAVE A, SHANKAR V, et al. Measuring robustness to natural distribution shifts in image classification[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 18583-18599.

[124] LI J, JI S, DU T, et al. Textbugger: Generating adversarial text against real-world applications[C]// Network and Distributed Systems Security (NDSS) Symposium. 2019: 1-15.

[125] SCHONHERR L, KOHLS K, ZEILER S, et al. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding[C]//Network and Distributed Systems Security (NDSS) Symposium. 2019: 1-28.

[126] LI S, NEUPANE A, PAUL S, et al. Stealthy adversarial perturbations against real-time video classification systems[C]//Network and Distributed Systems Security (NDSS) Symposium. 2019: 1-15.

[127] CHANG J W, JAVAHERIPI M, HIDANO S, et al. Rovisq: Reduction of video service quality via adversarial attacks on deep learning-based video compression[C]//Network and Distributed Systems Security (NDSS) Symposium. 2023: 1-19.

[128] LIU X, CHENG M, ZHANG H, et al. Towards robust neural networks via random self-ensemble [C]//European Conference on Computer Vision (ECCV). 2018: 369-385.

[129] PANG T, XU K, DU C, et al. Improving adversarial robustness via promoting ensemble diversity [C]//International Conference on Machine Learning (ICML). 2019: 4970-4979.

[130] GOWAL S, DVIJOTHAM K, STANFORTH R, et al. On the effectiveness of interval bound propagation for training verifiably robust models[A]. 2018: 1-16.

[131] BITTERWOLF J, MEINKE A, HEIN M. Certifiably adversarially robust detection of out-of-distribution data[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020: 16085-16095.

[132] SHI Z, WANG Y, ZHANG H, et al. Fast certified robust training with short warmup[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2021: 18335-18349.

[133] RAGHUNATHAN A, STEINHARDT J, LIANG P. Certified defenses against adversarial examples [A]. 2018: 1-15.

[134] SALMAN H, YANG G, ZHANG H, et al. A convex relaxation barrier to tight robustness verification of neural networks[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2019: 1-12.

[135] RAGHUNATHAN A, STEINHARDT J, LIANG P S. Semidefinite relaxations for certifying robustness to adversarial examples[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2018: 1-11.

[136] XU K, ZHANG H, WANG S, et al. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers[A]. 2020: 1-15.

[137] FISCHER M, BAADER M, VECHEV M. Scalable certified segmentation via randomized smoothing[C]//International Conference on Machine Learning (ICML). 2021: 3340-3351.

[138] MOON H C, JOTY S, ZHAO R, et al. Randomized smoothing with masked inference for adversarially robust text classifications[C]//Annual Meeting of the Association for Computational Linguistics (ACL). 2023: 5145-5165.

[139] WANG B, JIA J, CAO X, et al. Certified robustness of graph neural networks against adversarial structural perturbation[C]//ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD). 2021: 1645-1653.

[140] VIRTANEN P, GOMMERS R, OLIPHANT T E, et al. Scipy 1.0: Fundamental algorithms for scientific computing in python[J]. Nature Methods, 2020, 17(3): 261-272.

[141] UNB. Cse-cic-ids2018[EB/OL]. 2018. https://www.unb.ca/cic/datasets/ids-2018.html.

[142] HINTON G, DENG L, YU D, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups[J]. IEEE Signal Processing Magazine, 2012, 29 (6): 82-97.

[143] HASSAN M M, GUMAEI A, ALSANAD A, et al. A hybrid deep learning model for efficient intrusion detection in big data environment[J]. Information Sciences, 2020, 513: 386-396.

[144] AHMAD Z, SHAHID KHAN A, WAI SHIANG C, et al. Network intrusion detection system: A systematic study of machine learning and deep learning approaches[J]. Transactions on Emerging Telecommunications Technologies (ETT), 2021, 32(1): 1-29.

[145] MASUM M, SHAHRIAR H. A transfer learning with deep neural network approach for network intrusion detection[J]. International Journal of Intelligent Computing Research (IJICR), 2021, 12 (1): 1-9.

[146] WEI C, XIE G, DIAO Z. A lightweight deep learning framework for botnet detecting at the iot edge[J]. Computers & Security, 2023, 129: 1-9.

[147] HAN D, ZHOU H, WENG T H, et al. Lmca: A lightweight anomaly network traffic detection model integrating adjusted mobilenet and coordinate attention mechanism for iot[J]. Telecommunication Systems, 2023, 84(4): 549-564.

[148] SU Z, ZHANG J, WANG L, et al. Lightweight pixel difference networks for efficient visual representation learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2023, 45(12): 14956-14974.

[149] PAN S J, YANG Q. A survey on transfer learning[J]. IEEE Transactions on Knowledge and Data Engineering (TKDE), 2009, 22(10): 1345-1359.

[150] SINGLA A, BERTINO E, VERMA D. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning[C]//IEEE International Conference on Smart Computing (SMARTCOMP). 2019: 69-74.

[151] SINGLA A, BERTINO E, VERMA D. Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation[C]//ACM Asia Conference on Computer and Communications Securit (AsiaCCS). 2020: 127-140.

[152] PAWLICKI M, KOZIK R, CHORAŚ M. Towards deployment shift inhibition through transfer learning in network intrusion detection[C]//International Conference on Availability, Reliability and Security (ARES). 2022: 1-6.

[153] ZHAO R, CHEN Y, WANG Y, et al. An efficient and lightweight approach for intrusion detection based on knowledge distillation[C]//IEEE International Conference on Communications (ICC). 2021: 1-6.

[154] WANG Z, LI Z, HE D, et al. A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning[J]. Expert Systems with Applications, 2022, 206: 1-17.

[155] BOMMASANI R, HUDSON D A, ADELI E, et al. On the opportunities and risks of foundation models[A]. 2021: 1-214.

[156] TIAN Y, KRISHNAN D, ISOLA P. Contrastive representation distillation[C]//International Conference on Learning Representations (ICLR). 2019: 1-19.

[157] CHEN T, KORNBLITH S, NOROUZI M, et al. A simple framework for contrastive learning of visual representations[C]//International Conference on Machine Learning (ICML). 2020: 1-20.

[158] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[A]. 2017: 1-9.

[159] UNSW. Unsw-nb15[EB/OL]. 2015. https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[160] UNB. Iscx nsl-kdd[EB/OL]. 2009. https://www.unb.ca/cic/datasets/nsl.html.

[161] UCI. Kddcup-99[EB/OL]. 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Acknowledgements

I would like to sincerely thank everyone who supported me during my PhD journey at Xidian University and Purdue University. This is the perfect moment to express my gratitude to my teachers, lab members, friends, and family for their invaluable support and assistance. Without their encouragement, these achievements would not have been possible.

Firstly, I extend my deep appreciation for my advisor, Professor **Xiaofeng Chen**, a distinguished scholar in cryptography, cloud computing, blockchain, and AI security. It is both a privilege and an honor to be his doctoral student. Throughout my studies, Professor Chen has provided invaluable guidance and support, significantly shaping my academic and personal life. I am particularly grateful for his insights regarding my research direction and career path. Without his mentorship, I would not have had the numerous opportunities to explore, study, and engage within the field of AI security. I would also like to express my heartfelt thanks to Associate Professor Meixia Miao, the wife of Professor Chen. Their unwavering support and harmonious family life have profoundly inspired me. The balance and fulfillment they embody have left a lasting impression, for which I am truly thankful.

Secondly, I express heartfelt thanks to my foreign advisor, Professor **Elisa Bertino**, Samuel Conte Distinguished Professor of Computer Science at Purdue University. During the last three years of my PhD, she provided meticulous guidance and unwavering support. Her thoughtful academic advice, combined with genuine care and assistance in everyday life, has been invaluable. Professor Bertino exemplifies a rigorous attitude and tireless dedication to scientific research, serving as a powerful inspiration for my scholarly pursuits. Her profound insight into critical issues and complex problems fosters a continuous flow of inspiration. In addition, the harmonious and cohesive atmosphere within her team has brought immense comfort and enrichment to my doctoral experience, creating a collaborative environment where ideas flourish and friendships blossom.

Moreover, I am grateful to Professor Willy Susilo from the University of Wollongong, Australia; Professor Jin Li and Professor Changyu Dong from Guangzhou University; Professor Zheli Liu from Nankai University; Professor Ninghui Li and Associate Professor Bruno Ribeiro from Purdue University, US; Assistant Professor Hyunwoo Lee from Korea Institute of Energy Technology, Korea; Ashish Kundu from Cisco Research, US; Professor Miroslaw Kutylowski from Wroclaw University of Technology, Poland; Senior Lecturer Siqi Ma from University of New South Wales, Australia; Professor Fangguo Zhang from Sun Yat-sen Uni-

versity; Professor Bo Yang and Professor Yong Yu from Shaanxi Normal University, Professor Debiao He from Wuhan University; Professor Zhibo Wang from Zhejiang University; and Professor Xinyi Huang from Jinan University. Their guidance and support in my research deepened my understanding of scientific inquiry.

Furthermore, I am truly appreciative of Professor Jianfeng Wang, Associate Professor Shichong Tan, Associate Professor Xiaoyu Zhang, and Lecturer Guohua Tian from Xidian University; Associate Professor Jianghong Wei from PLA Information Engineering University; Associate Professor Yunling Wang and Lecturer Yanmei Cao from Xi'an University of Posts and Telecommunications; Lecture Jin Pan from Guangzhou Institute of Technology, Xidian University; Lecture Jun Shen from Shanghai University; Dr. Yi Xie from Tsinghua University; Lecturer Yamin Li from Henan University of Science and Technology; and Dr. Xuan Jing for their help and support in my daily study and life in the **Ruiyun Data Security Lab** at Xidian University. I also wish to thank the other lab members, including Jiaojiao Wu, Chunyang Lv, Chenyang Chen, Xuexuan Hao, Sen Zhao, Anqi Jiang, Qiao'er Xu, Shen Lin, Yiwen Shi, Yundong Liu, Jiawei Li, Tong Zhao, Jiarui Chen, and others.

In addition, I am deeply indebted to the **Cyber Space Security Lab** members who worked with me during my visit to Purdue University. It is unforgettable to have struggled and thrived alongside Dr. Imtiaz Karim, Dr. Charalampos Katsis, Fabrizio Cicala, Adrian Shuai Li, Mirza Masfiqur Rahman, Kazi Samin Mubasshir, Yiwei Zhang, Zilin Shen, Laraib Sana, Yinjun Lin, Mir Imtiaz Mostafiz, Alireza Lotfi, Md Ajwad Akil, Md. Shamsul Kaonain, Yongye Su, Subangkar Karmaker Shanto, Enrico Pisanti, Chuanpu Fu, Chiara Luchini, Yani Liu, and Yu Zhang. Besides, I would like to express my gratitude to my other academic collaborators, Yuntao Du, Kaiyuan Zhang, Taisuke Mori, Guanyan Ou, Yuetian Chen, Zhizhen Yuan, in the **Trust Security Lab**.

I also want to extend my heartfelt thanks to my closest friends for their invaluable companionship during challenging times: Jiajia Zhan, Meipin Liu, Kun Zeng, Chenchen Sun, Xinru Wang, Minxue Wang, Yanxue Jia, Yue Cui, Lu Xian, Kailin Yao, etc.

Finally, I would like to express my heartfelt gratitude to my family for their unwavering love and support throughout my journey. They taught me kindness, resilience, and hope, and I carry that with me every day. No matter how much time passes, or how far I travel, they are my foundation. I am forever grateful for everything my parents have given me.

Author Biography

1. Basic Information

Mengdie Huang, female, from Nanjing, Jiangsu, born in October 1994, is a Ph.D. candidate in Cyber Security at School of Cyber Engineering, Xidian University, enrolled in 2019.

2. Educational Background

2019.09~2025.06, Xidian University, Ph.D. student, Major: Cyber Security

2022.09~2025.06, Purdue University, Visiting Ph.D., Major: Computer Science

2017.09~2019.07, Communication University of China, Master's degree, Major: Electronics and Communication Engineering

2013.09~2017.07, Huaiyin Institute of Technology, Bachelor's degree, Major: Communication Engineering

3. Research Achievements During Ph.D. Studies

3.1 Published Academic Papers

[1] **Mengdie Huang**, Yingjun Lin, Ninghui Li, Xiaofeng Chen*, Elisa Bertino. CARD: Robustness-Preserving Transfer Learning for Network Intrusion Detection via Contrastive Adversarial Representation Distillation[J]. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2025. DOI: 10.1109/TDSC.2025.3562600. (**CCF-A**)

[2] **Mengdie Huang**, Yingjun Lin, Xiaofeng Chen, Elisa Bertino*. Dimensional Robustness Certification for Deep Neural Networks in Network Intrusion Detection Systems[J]. *ACM Transactions on Privacy and Security (TOPS)*, 2025. DOI: 10.1145/371512. (**CCF-B**)

[3] **Mengdie Huang**, Yi Xie, Xiaofeng Chen*, Jin Li, Changyu Dong, Zheli Liu, Willy Susilo. Boost Off/On-Manifold Adversarial Robustness for Deep Learning with Latent Representation Mixup[C]. *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2023, 716-730. DOI: 10.1145/3579856.3595786. (**CACR-B**)

[4] **Mengdie Huang**, Yingjun Lin, Xiaofeng Chen*, Elisa Bertino. MARS: Robustness Certification for Deep Network Intrusion Detectors via Multi-Order Adaptive Randomized Smoothing. *IEEE International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom)*, 2024, 767-774. DOI: 10.1109/TrustCom63139.2024.00117. (**CCF-C, Best Paper**)

[5] **Mengdie Huang***, Hyunwoo Lee, Ashish Kundu, Xiaofeng Chen, Anand Mudgerikar, Ninghui Li, Elisa Bertino. ARIoTEDef: Adversarially Robust IoT Early Defense System based on Self-evolution against Multi-step Attacks[J]. *ACM Transactions on Internet of Things (TIOT)*, 2024, 5(3): 1-34. DOI: 10.1145/3660646.

3.2 Research Projects and Awards

[1] Key International Cooperation Project of the National Natural Science Foundation of China, Key Theories and Technologies for Encrypted Data Security Services in Open Fusion Environments, 2020.01-2024.12, Completed, Participant.

[2] Xidian University Doctoral Academic Scholarship, Second Prize, 2020-2022.

[3] Xidian University Outstanding Graduate Student Award, 2020-2022.

[4] Xidian University Excellent Paper Award of Annual Academic Conference, Second Prize, 2021.12.

[5] Xidian University Doctoral Academic Scholarship, First Prize, 2019.