Boosting Adversarial Robustness via Neural Architecture Search and Design

MINJING DONG

Doctor of Philosophy

Supervisor: Dr. Chang Xu

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

20 November 2023

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Student Name Signature

Minjing Dong

Authorship Attribution Statement

I designed the study and wrote the drafts of the papers that constitute some parts of the thesis. Chapter 3 was published as (Dong et al. 2020). Chapter 4 of the thesis was accepted as (Dong et al. 2021). Chapter 5 was published as (Dong and Xu 2023). Chapter 6 was accepted as (Dong et al. 2022). In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Student Name Signature
Minjing Dong

As the supervisor of the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Supervisor Name Signature
Chang Xu

List of Research Outcome

Research outcome covered in this thesis

(1) **Minjing Dong** and Chang Xu. Adversarial robustness via random projection filters. ***Conference on Computer Vision and Pattern Recognition 2023 (CVPR 2023)***, Vancouver, Canada, June 18-22 2023.

(2) **Minjing Dong**, Xinghao Chen, Yunhe Wang, and Chang Xu. Random normalization aggregation for adversarial defense. ***The thirty-sixth Conference on Neural Information Processing Systems (NeurIPS 2022)***, New Orleans, USA, November 28 - December 9 2022.

(3) **Minjing Dong**, Yunhe Wang, Xinghao Chen, and Chang Xu. Towards stable and robust addernets. ***The thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)***, December 6-14, 2021.

(4) **Minjing Dong**, Yanxi Li, Yunhe Wang, and Chang Xu. Adversarially robust neural architectures. arXiv preprint arXiv:2009.00902 (2020).

Other peer-refereed research outcome

(1) Yanxiang Ma, **Minjing Dong**, and Chang Xu. Adversarial robustness through random weight sampling. ***The thirty-seventh Conference on Neural Information Processing Systems (NeurIPS 2023)***.

(2) **Minjing Dong**, Xinghao Chen, Yunhe Wang, and Chang Xu. Improving lightweight adderNet via distillation from ℓ_2 to ℓ_1-Norm. ***IEEE Transactions on Image Processing (TIP)***, 2023.

(3) Linwei Tao*, **Minjing Dong***, and Chang Xu. Dual Focal Loss for Calibration. ***The fortieth International Conference on Machine Learning (ICML 2023)***, Hawaii, July 23-29, 2023.

(4) Linwei Tao, **Minjing Dong**, Daochang Liu, Changming Sun, and Chang Xu. Calibrating a deep neural network with its predecessors. ***The 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)***, Macao, China, August 19-25, 2023.

(5) Zhi Cheng, Yanxi Li, **Minjing Dong**, Xiu Su, Shan You, and Chang Xu. Neural architecture search for wide spectrum adversarial robustness. ***The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023)***, Washington DC, USA, February 7-14 2023. (**Distinguished Paper Award**)

(6) Haoyu Xie, Changqi Wang, Mingkai Zheng, **Minjing Dong**, Shan You, Chong Fu, and Chang Xu. Boosting semi-supervised semantic segmentation with probabilistic representations. ***The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023)***, Washington DC, USA, February 7-14 2023.

(7) **Minjing Dong** and Chang Xu. Skeleton-based human motion prediction with privileged supervision. ***IEEE Transactions on Neural Networks and Learning Systems (TNNLS)***, 2022, doi: 10.1109/TNNLS.2022.3166861.

(8) Yanxi Li, Xinghao Chen, **Minjing Dong**, Yehui Tang, Yunhe Wang, and Chang Xu. Spatial-channel token distillation for vision mlps. ***The 39th International Conference on Machine Learning (ICML 2022)***, Baltimore, USA, July 17-23 2022.

(9) Yanxi Li, **Minjing Dong**, Yixing Xu, Yunhe Wang, and Chang Xu. Neural architecture tuning with policy adaptation. ***Neurocomputing***, 2022, 485:196–204,2022.

(10) **Minjing Dong**, Yunhe Wang, Xinghao Chen, and Chang Xu. Handling long-tailed feature distribution in addernets. ***The thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)***, December 6-14, 2021.

(11) Xinghao Chen, Chang Xu, **Minjing Dong**, Chunjing Xu, and Yunhe Wang. An empirical study of adder neural networks for object detection. ***The thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)***, December 6-14, 2021.

(12) Yanxi Li, **Minjing Dong**, Yunhe Wang, and Chang Xu. Neural architecture search in a proxy validation loss landscape. *The 37th International Conference on Machine Learning (ICML 2020)*, July 12-18 2020.

(13) Shuo Yang, **Minjing Dong**, Yunhe Wang and Chang Xu. Adversarial recurrent time series imputation. ***IEEE Transactions on Neural Networks and Learning Systems (TNNLS)***, 2020, doi: 10.1109/TNNLS.2020. 3010524.

Other preprint research outcome

(1) Bo Han, Hao Peng, **Minjing Dong**, Chang Xu, Yi Ren, Yixuan Shen, and Yuheng Li. AMD: autoregressive motion diffusion model. arXiv preprint arXiv:2305.09381 (2023).

(2) Huihui Gong, **Minjing Dong**, Siqi Ma, Seyit Camtepe, Surya Nepal, Xiangyu Yue, and Chang Xu. Parameter-saving adversarial training: reinforcing multi-perturbation robustness with hypernetworks. arXiv preprint arXiv:2309.16207 (2023).

(3) Huihui Gong, **Minjing Dong**, Siqi Ma, Seyit Camtepe, Surya Nepal, and Chang Xu. Stealthy Physical Masked Face Recognition Attack via Adversarial Style Optimization. arXiv preprint arXiv:2309.09480 (2023).

(4) Xiaohuan Pei, Yanxi Li, **Minjing Dong**, and Chang Xu. Neural Architecture Retrieval. arXiv preprint arXiv:2307.07919 (2023).

(The star '*' indicates equal contribution.)

Abstract

Adversarial robustness in Deep Neural Networks (DNNs) is a critical and emerging field of research that addresses the vulnerability of DNNs to subtle, intentionally crafted perturbations in their input data. These perturbations, often imperceptible to the human eye, can lead to significant error increment in the network's predictions, while they can be easily derived via adversarial attacks in various data formats, such as image, text, and audio. This susceptibility poses serious security and trustworthy concerns in real-world applications such as autonomous driving, healthcare diagnostics, and cybersecurity. To enhance the trustworthiness of DNNs, lots of research efforts have been put into developing techniques that aim to improve DNNs ability to defend against such adversarial attacks, ensuring that trustworthy results can be provided in real-world scenarios. The main stream of adversarial robustness lies in the adversarial training strategies and regularizations. However, less attention has been paid to the DNN itself. Little is known about the influence of different neural network architectures or designs on adversarial robustness. To fulfill this knowledge gap, we propose to advance adversarial robustness via investigating neural architecture search and design in this thesis.

Firstly, we propose to connect adversarial robustness and neural architecture search. Through approximating Lipschitz constant of DNNs under NAS framework, we introduce an effective and efficient searching algorithm as well as corresponding sampling strategy to derive adversarially robust neural architecture. Secondly, we consider the basic operations in DNNs, such as the similarity measurement between weight parameters and feature maps. We show that the cross correlation in CNNs could be the important factor which amplifies the perturbations in feature maps, which decreases adversarial robustness. Based on the analysis, we introduce a robust inference strategy via the utilization of adder operation to eliminate the perturbation automatically. Thirdly, we explore the randomized defense scheme related to the CNN filters from a perspective of random projection. We extend the scope of Johnson-Lindenstrauss Lemma to the partial scenarios in convolutional filters, where a better

trade-off between natural and robust accuracy can be achieved. Lastly, we study the influence of normalization layers in DNNs on adversarial robustness. We establish the connection between adversarial transferability and normalization layers, from which we introduce a randomized defense scheme to fully utilize the potential of different normalization layers with low adversarial transferability to defend against adversarial attacks.

Acknowledgements

I would like to express my deepest gratitude to all those who have contributed to the completion of my doctoral studies. Due to their invaluable support and guidance, my journey has been filled with discoveries and growth.

First and foremost, I would like to thank my supervisor, Dr Chang Xu, for his unwavering support and guidance throughout my research journey. He not only provided expert knowledge in machine learning and deep learning but also encouraged me to think critically, set high standards, and persevere through challenges. I have received lots of insightful feedback and constructive criticism from Chang, which helped me to overcome various difficulties in both research and life during my pursuit of doctor of philosophy. His guidance, patience, and encouragement have been instrumental in shaping my academic development. I am honored to have had the opportunity to be supervised by Dr Chang Xu.

I am also greatly indebted to the strong support from Dr Yunhe Wang and Dr Xinghao Chen. I have learned a lot from their professional advice and close collaboration on the research of efficient neural networks.

I would also like to thank my research collaborators: Yanxi Li, Shuo Yang, Linwei Tao, Zhi Cheng, Hanting Chen, Huihui Gong, Xiaohuan Pei, Haoyu Xie, Yanxiang Ma, Yunke Wang, Bo Han, for their fruitful discussion and insights.

My sincere thanks go to my friends and colleagues, Jiajun Huang, Xiyu Wang, Xiu Su, shuyi Jiang, Anh-Dung Dinh, Daochang Liu, Chengbin Du, Yixiong Ding, Mingkai Zheng, Tao Huang, Chen Chen, Kai Han, Qiulin Zhang, Yixing Xu, Jianyuan Guo, Chuanjian Liu, Wenshuo Li, Zhenhua Liu, Ying Nie, Yehui Tang, Yiman Zhang, Min Zhong, An Xiao, for their strong support and encouragement.

I want to acknowledge the financial support provided by Faculty of Engineering Research Scholarship, The University of Sydney. Their support has enabled me to start my research and has been essential to the successful completion of doctoral studies.

Finally, I would like to thank my family and my wife Juewen Wang. Their support and understanding have been crucial in helping me navigate the challenges of doctoral studies.

Minjing Dong

Sydney, Australia, 2023

Contents

Statement of Originality ii

Authorship Attribution Statement iii

List of Research Outcome iv

Abstract vii

Acknowledgements ix

Contents xi

Notation xv

List of Figures xvi

List of Tables xviii

Chapter 1 Introduction 1

1.1 Thesis Contributions 4

1.1.1 Contribution of RACL 4

1.1.2 Contribution of AWN-R 5

1.1.3 Contribution of RPF 6

1.1.4 Contribution of RNA 6

1.2 Thesis Outline 7

Chapter 2 Literature review 9

2.1 Adversarial Attack 9

2.2 Adversarial Defense 10

2.3 Neural Architecture Search 11

2.4 Basic Operation Design 12

2.5 Normalization Layer 13
2.6 Random Projection 13

Chapter 3 Neural Architecture Search for Adversarial Robustness 15
3.1 Motivation 15
3.2 Methodology 17
3.2.1 Preliminary 17
3.2.2 Lipschitz Constraints in Neural Architecture 18
3.2.3 Confident Architecture Sampling 21
3.3 Experiments 27
3.3.1 Experimental Setup 28
3.3.2 Against White-box Attacks 29
3.3.3 Against Black-box Attacks 33
3.3.4 Robustness under Various Perturbation Size and Attack Iterations 35
3.3.5 Potential Pattern and Variance of RACL 37
3.3.6 Ablation Analysis 38
3.4 Conclusion 39

Chapter 4 Adder Filter Design for Adversarial Robustness 40
4.1 Motivation 40
4.2 Preliminary 42
4.3 Variance Study of AdderNet 43
4.3.1 Adaptive Weight Normalization for AdderNet 45
4.4 Activate Potential Adversarial Robustness 47
4.5 Experiments 50
4.5.1 Adversarial Robustness Evaluation 50
4.5.2 Robustness Comparison with Adversarial-trained CNN 52
4.5.3 Stability of AdderNets 53
4.5.4 Experiments on Object Detection 55
4.5.5 Ablation Studies 56
4.5.6 Adaptive Weight Normalization on CNN 58

4.5.7 Variants of Inference Strategy 58
4.5.8 Stability of Robustness 59
4.5.9 Classification Performance 60
4.6 Conclusion 60

Chapter 5 Random Filter Design for Adversarial Robustness 62
5.1 Motivation 62
5.2 Methodology 64
5.2.1 Preliminaries 64
5.2.2 Random Projection Filters 65
5.2.3 Adversarial Training with Random Projection 68
5.3 Experiments 70
5.3.1 Experimental Setup 70
5.3.2 Results on CIFAR 72
5.3.3 Evaluate with Stronger Attacks 74
5.3.4 State-of-the-art Comparison 75
5.3.5 Evaluation of Black-box Attacks 75
5.3.6 Comparisons with Noise Injection Techniques 76
5.3.7 Ablation Study 76
5.3.8 Multiple Runs 78
5.3.9 Evaluation on More Models, Norms, and Defense Techniques. 79
5.4 Conclusion 79

Chapter 6 Low-Transferability Normalization Search for Adversarial Robustness 80
6.1 Motivation 80
6.2 Adversarial Transferability with Different Normalization 82
6.2.1 Definition of Normalization Layers 83
6.2.2 Variation of Loss Function Smoothness 84
6.2.3 Normalization Layers and Adversarial Transferability 86
6.3 Random Normalization Aggregation 88
6.4 Experiments 90

6.4.1 Evaluation Setup 90
6.4.2 Results for Robustness 92
6.4.3 Comparison with RPI+RPT 95
6.4.4 Ablation Study 96
6.4.5 Stability of Robustness 98
6.5 Conclusions 98

Chapter 7 Conclusion and Future Work **99**

Appendix A Appendix for Chapter 5 **101**
A1 Proof of Theorem 1 101

Appendix B Appendix for Chapter 6 **105**
B1 Proof of Theorem 2 105

Appendix Bibliography **110**

Notation

Table of notations

Sign	Description
x	input image
y	annotated label vector
$\tilde{x}$	perturbed input image
δ	perturbation
ϵ	maximum perturbation size
$\|\cdot\|_p$	l_p-norm
o	operation
$\mathcal{O}$	pre-defined search space of operations
$\mathcal{A}$	architecture
W	model parameters
$\phi(\cdot)$	probability density function
$\Phi(\cdot)$	cumulative distribution function
$\mathbb{E}$	expectation
λ	Lipschitz constant
$\nabla_x \mathcal{L}$	gradient of loss function $\mathcal{L}$ with respect to input image x
$\mathcal{N}(\mu, \Sigma)$	normal distribution with mean μ and covariance matrix Σ
$\mathcal{LN}(\mu, \Sigma)$	log-normal distribution with mean μ and covariance matrix Σ

List of Figures

1.1 Algorithms for adversarial robustness via neural architectures. 7

3.1 An overview of proposed robust neural architecture search with confidence learning algorithm. Each node in the cell is computed with operations mixture under architecture parameters α for weighting operations and β for weighting inputs where α and β are sampled from multivariate log-normal distributions. Meanwhile, the Lipschitz constant of each edge and cell induce to univariate log-normal distributions. The Lipschitz constraint is formulated from cumulative distribution function. 19

3.2 An illustration of the difference between previous differentiable NAS and ours with confidence learning. 22

3.3 The visualization of normal and reduction cell searched by RACL are shown in (a) and (b). 28

3.4 Robustness evaluation under different perturbation sizes and attack iterations. 35

3.5 The visualization of cells searched by RACL through multiple runs. 37

4.1 Observations of AdderNet. Training and testing loss curves of ANN and CNN in (a). Histograms over the AdderNet features after batch normalization layer follow Gaussian distribution in (b). Histograms over the AdderNet weight follow Laplace distribution with a large variance while Conv weight follows Gaussian distribution with small variance in (c) and (d) respectively. 41

4.2 Histograms over the ANN and ANN-AWM features of an intermediate layer are shown in (a) and (b). 49

4.3 The evaluation of adversarial robustness under different PGD attack size is shown in (a) and different PGD attack steps in (b). The performance of intermediate weights sampled from ANN and ANN-AWN through linear interpolation in (c). Test loss curves of CNN, ANN, ANN-WS and ANN-AWN are shown in (d). 52

4.4 Distributions of features from different layers of ANN and ANN-AWN on different epochs. 54
5.1 An overview of proposed with Random Projection Filters with defense scheme. Part of the filters in convolutional layers are replaced by random projection, whose weight is randomly sampled from a Gaussian distribution. RP[A] and RP[I] denote the sampled Gaussian matrix of random projection filters during attack and infer phase respectively. 65
5.2 The evaluation of stronger PGD attacks with ResNet-18 on CIFAR-10 and CIFAR-100. 71
5.3 Ablation studies of Random Projection Filters on location, ratio, and weight decay. 75
6.1 (a) denotes the normalized dimensions of LN, GN, and IN, while (b) denotes BGN and BN. (c) and (d) denote the adversarial transferability among different types of normalizations. 81
6.2 (a) and (b) denote the smoothness of loss *w.r.t.* input with BN and IN respectively. (c) denotes the smoothness of BN, BGN, LN, GN, and IN with both 3D and 2D plots. 83
6.3 The histograms over the gradient cosine similarity of different normalization layers. 85
6.4 An illustration of Random Normalization Aggregation and Black-box Adversarial Training. 87
6.5 The evaluation of robustness under PGD attacks settings. (a) denotes larger attack iterations, and (b) denotes larger perturbation sizes. The adversarial transferability in random space is evaluated through sampling paths with different levels of diversity in (c). 94

List of Tables

3.1 Evaluation of RACL adversarial robustness on CIFAR-10, CIFAR-100, and Tiny-ImageNet compared with various NAS algorithms under white-box attacks. PGD^{20} denotes PGD attack with 20 iterations. Best results in bold. 27

3.2 Evaluation of RACL adversarial robustness on CIFAR-10 compared with other human-designed architectures and other robust NAS algorithms under white-box attacks. Best results in bold. 30

3.3 Comparison with existing defence techniques under PGD attack on different datasets. 32

3.4 Evaluation of RACL adversarial robustness on CIFAR-10 under transfer-based black-box attack setting. 33

3.5 Transferability test on CIFAR-10 among differnet models using PGD attack. The best results in each row are in bold. Underline denotes the white-box robustness. 34

3.6 Transferability Test on CIFAR-10 among different models under RFGSM Attack. The best results in each row are in bold. Underline denotes the white-box robustness. 34

3.7 Multiple runs of searched cells with adversarial training. The mean of clean or adversarial accuracy is reported with its error bar. 36

3.8 Ablation Analysis of RACL with respect to confidence learning, ρ, and η. 37

4.1 Adversarial robustness on CIFAR-10 under white-box attacks without adversarial training. -R denotes robust inference strategy which uses the running mean in batch normalization layer instead of tracked ones. BIM^7 denotes iterative attack with 7 steps. The best results in bold and the second best with underline. 49

4.2 Robustness Comparison with CNN defense techniques. AT denotes the usage of adversarial training. 53

4.3 Comparison of proposed approach on ANNs with other settings on PASCAL VOC 2012 benchmark. The $[\cdot]$ in backbone denotes the classification accuracy on ImageNet. 55

4.4 Adversarial robustness comparison of WS and AWN under different inference strategy with ResNet-20 on CIFAR-10 with natural training. -R denotes using running mean of current batch in BN layer. -r denotes using both running mean and variance. 57

4.5 Adversarial robustness evaluation of AWN on CNN and ANN with ResNet-32 on CIFAR-10. 57

4.6 Adversarial robustness evaluation of different inference strategy of ANN-AWN with ResNet-32 on CIFAR-10. 58

4.7 Robustness stability of proposed ANN-R-AWN with ResNet-20 on CIFAR-10 under different inference settings. 59

4.8 Classification performance evaluation. 60

5.1 The comparison with noise injection techniques with ResNet-18 on CIFAR-10 and CIFAR-100. 70

5.2 Adversarial robustness evaluation of randomized techniques with WideResNet on CIFAR-10. 72

5.3 Comparison with SOTA defense algorithms on CIFAR-10 and ImageNet. 73

5.4 Evaluation of black-box attacks. 76

5.5 Comparison with other noise injection techniques. 76

5.6 The evaluation results of 5 runs. 78

5.7 Results with ResNet-18 on CIFAR-10. 79

6.1 The adversarial robustness evaluation of adversarial-trained networks on CIFAR-10. 91

6.2 The adversarial robustness evaluation of adversarial-trained networks on CIFAR-100. 92

6.3 Comparison with defense algorithms. 94

6.4 Robustness evaluation of random space built from different normalization combinations under different attacks. 94

6.5 The adversarial robustness evaluation with normal attacks settings on CIFAR-10. 96

6.6 The adversarial robustness evaluation with normal attacks settings on CIFAR-100. 96

6.7 The adversarial robustness evaluation under adaptive attack setting on CIFAR-10/100. 96

6.8 Ablation studies of RNA with ResNet-18 on CIFAR-10, including random space designing, adversarial training, and layer-based constraint. 97

6.9 Stability evaluation of adversarial robustness with RNA. 98

CHAPTER 1

Introduction

Deep neural networks (DNNs) have shown remarkable performance in various applications, such as image classification (Krizhevsky et al. 2012; Simonyan and Zisserman 2014; He et al. 2016; Huang et al. 2017), object detection (Girshick 2015; Tian et al. 2019), and machine translation (Bahdanau et al. 2014; Chen et al. 2020c). Given the remarkable performance of DNNs on different data modalities, it becomes more critical to study the trustworthiness of DNNs since it reflects the reliability and safety in real-world applications. However, recent works have shown that DNNs are vulnerable to small perturbations on the input data, which has caused trustworthy issues in DNNs (Nguyen et al. 2015; Moosavi-Dezfooli et al. 2016; Moosavi-Dezfooli et al. 2017; Biggio et al. 2013; Xie et al. 2019).

Considering a model of image classification task, some perturbations which are imperceptible to human beings but make the model misclassify the input image can always be found (Goodfellow et al. 2014). The combination of original image and this kind of perturbation is denoted as an adversarial example, and the techniques to discover these adversarial examples are denoted as adversarial attacks. They could be generally categorized into two streams, the white-box and black-box attacks. In black-box setting, the attackers have no knowledge of victim models but can estimate the strong perturbation via surrogate models or huge number of queries (Guo et al. 2019b; Ilyas et al. 2018). In white-box setting, the attackers have full knowledge of victim model, including the model parameters, network architecture, and inference strategy (Szegedy et al. 2014; Madry et al. 2018). Since the gradients of victim models can be directly fetched, the crafted adversarial examples are more aggressive and the performance under white-box attacks is one of the key criteria of robustness evaluation. Given the increasing adversarial attacks under various settings (Madry et al. 2018; Andriushchenko

et al. 2020; Croce and Hein 2020), their threats can be rather challenging since it is difficult for existing popular backbone networks to show robustness when fed with adversarial examples.

Seeking adversarial robust networks becomes a key challenge to improve the trustworthiness of DNNs and has attracted lots of research interests. One of the most popular and effective techniques is adversarial training (Madry et al. 2018), which arguments the training data with adversarial examples within a fixed perturbation size. With the involvement of adversarial examples, DNNs are optimized to preserve their outputs for perturbed samples within the ℓ_p ball of all training input data. However, due to the increasingly advanced attack techniques, it is difficult for existing adversarially trained networks to achieve satisfactory robustness against all potential attacks. Furthermore, the training on stronger adversarial examples could hurt the natural generalization of models (Zhang et al. 2020), and there exists a trade-off between robustness and accuracy (Zhang et al. 2019a).

Besides the traditional adversarial training, the utilization of randomization in adversarial robustness has been proven effective. For example, Liu *et al*. (Liu et al. 2018b) propose to inject noise which is sampled from Gaussian distribution to the inputs of convolution layers. Some theoretical analyses have shown that randomized classifiers can easily outperform deterministic ones in defending against adversarial attacks (Pinot et al. 2019; Pinot et al. 2020). We mainly attribute the improvement of randomization in adversarial robustness evaluation to the fusion of features with noises, which prevents white-box attackers from obtaining the precise gradients of loss with respect to the inputs. Although the involvement of noise in the networks can be an effective defense mechanism, the design of noises, such as the way of injection, the magnitude of noise, etc., can also significantly influence the natural generalization of networks in practice. The trade-offs between the adversarial robustness and optimization difficulty are always ignored in the randomized techniques, which limits their superiority to deterministic models.

In addition to improving existing defense schemes, recent work has laid emphasis on the architecture search and DNN components design. For example, the strong connection between adversarial robustness and DNN components can be empirically found, such as normalization layers (Benz et al. 2021; Galloway et al. 2019), activation functions (Xie et al. 2020b), etc..

Although existing defense techniques show effectiveness against adversarial attacks, these defense strategies suffer from different issues which are remained to be tackled and there exist many unexplored potentials of DNNs in adversarial robustness. We mainly summarize them in the following four aspects:

- Firstly, the influence of neural architecture design on adversarial robustness. Although the adversarially trained networks show robustness on various attacks, the architectures of these networks are fixed during optimization, which limits the adversarial robustness potential of DNNs. The superior architectures designed by human experts, such as AlexNet and ResNet (Krizhevsky et al. 2012; He et al. 2016), suggest that the DNN performance is subject to the architecture of network. Boosting NAS studies also emphasize the influence of architecture. Hence, we ask a simple question: *Can the network be initialized with robust architecture to further obtain adversarial robustness?*
- Secondly, the vulnerability of convolution neural networks (CNNs) without adversarial training. The cross correlation is utilized in CNNs as the basic operation to measure the similarity between the input feature and weight parameters. This similarity measurement show superiority in various computer vision tasks, such as image classification, object detection, etc., however, it is difficult for CNNs to achieve adversarial robustness without adversarial training. We mainly attribute it to the fact that the perturbation in features can be amplified by the CNN weights. This observation motivates us to think about some more advanced similarity measurement in DNNs which can naturally perform strong adversarial robustness against attacks.
- Thirdly, the trade-offs between natural and robust accuracy in randomized defense. Different from conventional defense techniques, randomized defense aims at increasing the attacking difficulty of in searching adversarial examples via the involvement of randomness in the models, such as additive noises (Liu et al. 2018b; Li et al. 2019). The involvement of randomness in the models hinders adversaries from deriving precise input gradients. However, despite its powerful defense ability, the randomness leads to the increment of optimization difficulty and the decrease in

natural accuracy. Thus, a careful design of randomness is expected to achieve better trade-offs between natural and robust accuracy.

- Lastly, the potential of normalization layers in defense scheme. Existing works have discussed the influence of Batch Normalization (BN) and empirically shown that BN increases adversarial vulnerability and decreases adversarial transferability (Benz et al. 2021; Galloway et al. 2019). However, the theoretical analysis of this observation is insufficient, which can hardly provide insights to further robust network designs. Furthermore, there is no feasible solution to tackle this pitfall of BN layers in adversarial robustness given the wide usage of BN layers in CNNs. Exploring the defense scheme with BN layers could fulfill this research gap.

Following these four aspects of existing issues or unexplored potentials, this thesis introduces four algorithms to fulfill these gaps in the corresponding chapters, including Robust Neural Architecture Search with Confidence Learning (RACL), Adaptive Weight Normalization with Robust Inference (AWN-R), Random Projection Filters (RPF), and Random Normalization Aggregation (RNA).

1.1 Thesis Contributions

In Chapter 3 – 6, we introduce four different algorithms for adversarial robustness via neural architecture search and designs. An illustration of the covered algorithms in this thesis is shown in Figure 1.1. Each algorithm explores and tackles one of the aforementioned concern or potential. We mainly summarize the contributions of each algorithm in this section.

1.1.1 Contribution of RACL

To fulfill the research gap of superior adversarially robust neural architecture design, we aim to improve the adversarial robustness of the network from the architecture perspective. We explore the relationship among adversarial robustness, Lipschitz constant, and architecture parameters and show that an appropriate constraint on architecture parameters could reduce the Lipschitz constant to further improve the robustness. With optimized architecture parameters,

an adversarially robust neural architecture can be sampled. However, the importance of architecture parameters could vary from operation to operation or connection to connection, which highlights the importance of confidence learning of architecture parameters in the sampling strategy.

We approximate the Lipschitz constant of the entire network through a univariate log-normal distribution, whose mean and variance are related to architecture parameters. The confidence can be fulfilled through formulating a constraint on the distribution parameters based on the cumulative function. Compared with adversarially trained neural architectures searched by various NAS algorithms as well as efficient human-designed models, our algorithm empirically achieves the best performance among all the models under various attacks on different datasets.

1.1.2 Contribution of AWN-R

Based on the observation of the vulnerability of cross correlation in CNNs, we explore the potential of the replacement cross correlation with ℓ_1-norm in adder neural networks (ANNs) when it comes to adversarial robustness. ANN replaces the original convolutions with massive multiplications by cheap additions while achieving comparable performance thus yields a series of energy-efficient neural networks. Compared with convolutional neural networks (CNNs), the training of AdderNets is much more sophisticated including several techniques for adjusting gradient and batch normalization. In addition, variances of both weights and activations in resulting adder networks are very enormous which limits its performance and the potential for applying to other tasks. However, given the ℓ_1-norm similarity measurement. we find the potential of natural adversarial robustness in ANNs.

To enhance the stability and activate the robustness of AdderNets, we first thoroughly analyze the variance estimation of weight parameters and output features of an arbitrary adder layer. Then, we develop a weight normalization scheme for adaptively optimizing the weight distribution of AdderNets during the training procedure, which can reduce the perturbation on running mean and variance in batch normalization layers. Meanwhile, the proposed weight normalization can also be utilized to enhance the adversarial robustness of resulting networks.

Experiments conducted on several benchmarks demonstrate the superiority of the proposed approach for generating AdderNets with higher performance.

1.1.3 Contribution of RPF

Motivated by the delicate balance between robust and natural performance, we explore the random defense solution which achieves better trade-offs between the defense capability and representation preservation. Random projection is a classic dimensionality reduction technique with randomness, which we find can be a potential solution to the delicate trade-offs due to its distance preservation after the projection by the Johnson–Lindenstrauss lemma (Vershynin 2018) since it preserves representation preservation while involves randomness.

We take initiatives to explore randomized defense techniques in CNNs via random projection. To this end, we introduce *Random Projection Filters* (RPF), where part of filters in the CNN layer is replaced by the random projection. Together with the proposed defense scheme, RPF significantly increases the difficulties in discovering effective adversarial perturbations. To clarify the relationship between the desired trade-offs and the involved random projection in RPF, we first provide theoretical evidence that Johnson–Lindenstrauss lemma holds for the CNN layers with partial random projection. Furthermore, we establish the connections between the desired trade-offs and the Euclidean norm of trainable parameters in CNN layers with random projection. Based on the analysis, a simple training strategy is introduced to activate the superior adversarial robustness of RPF. Experimental results on CIFAR-10/100 and ImageNet show that RPF exhibits the best trade-offs between natural and robust accuracy compared with other baselines.

1.1.4 Contribution of RNA

To explore the potential of normalization layers in adversarial defense, we focus on the adversarial transferability instead of adversarial robustness. *Adversarial Transfersability*, i.e. the capability of adversarial examples to fool other models, is one of the intriguing properties in adversarial learning. Traditionally, this transferability is always regarded as a critical threat

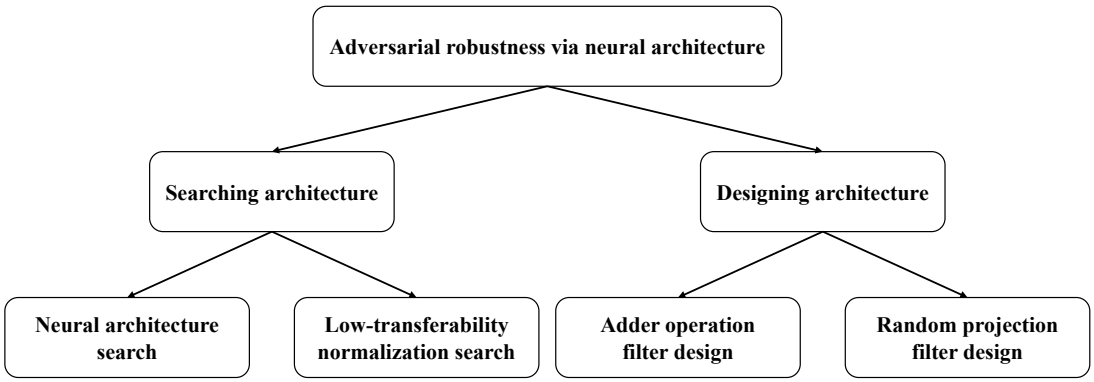


FIGURE 1.1. Algorithms for adversarial robustness via neural architectures.

to the defense against adversarial attacks, however, we argue that the network robustness can be significantly boosted by utilizing adversarial transferability from a new perspective.

We first discuss the influence of different popular normalization layers on the adversarial transferability, and then provide both empirical evidence and theoretical analysis to shed light on the relationship between normalization types and transferability. Based on our theoretical analysis, we propose a simple yet effective module named Random Normalization Aggregation (RNA) which replaces the batch normalization layers in the networks and aggregates different selected normalization types to form a huge random space.

Specifically, a random path is sampled during each inference procedure so that the network itself can be treated as an ensemble of a wide range of different models. With different normalization layers, we show that a desired random space can be built at a low cost. Since the entire random space is designed with low adversarial transferability, it is difficult to perform effective attacks even when the network parameters are accessible. We conduct extensive experiments on various models and datasets, and demonstrate the strong superiority of proposed algorithm.

1.2 Thesis Outline

As introduced above, this thesis delves into the solutions to existing issues and remaining potentials from a perspective of DNN search and design. This thesis is organized as follows:

In Chapter 1, we introduce the adversarial robustness and the existing concerns of different defense algorithms. We also highlight the unexplored potential of adversarial robustness in terms of DNN designs. Furthermore, we outline our main contributions of four algorithms which are introduced in this thesis.

In Chapter 2, we summarize the literature review of relevant topics, including adversarial attacks, adversarial defense, neural architecture search, basic operation design, normalization layers, and random projection.

In Chapter 3, we explore the connections between adversarial robustness and neural architectures. We introduce an efficient algorithm to discover the adversarially robust neural architectures, named Robust Neural Architecture Search with Confidence Learning (RACL).

In Chapter 4, we propose to replace the cross correlation in CNNs with the ℓ_1-norm distance in ANNs and investigate the potential of ANNs in natural adversarial robustness. Through the variance study of ANNs, we introduce a simple algorithm with designed inference strategy, named Adaptive Weight Normalization with Robust Inference (AWN-R).

In Chapter 5, we study the potential of random projection in randomized defense scheme for a better trade-offs between natural and robust performance. Utilizing the distancing preservation property, we propose to partially replace the CNN filters by random projection ones, named Random Projection Filters (RPF).

In Chapter 6, we investigate the role of normalization layers in defense scheme. Through empirical evidence and theoretical analysis, we introduce an effective defense algorithm via different normalization layers, named Random Normalization Aggregation (RNA).

In Chapter 7, we conclude the thesis and discuss the future directions.

CHAPTER 2

Literature review

In this chapter, we review the relevant literature of adversarial robustness as well as the related techniques introduced in the following chapters.

2.1 Adversarial Attack

The adversarial examples are first revealed by (Szegedy et al. 2014), in which Szegedy *et al*. demonstrated the vulnerability of DNNs to the perturbed inputs within a ℓ_p ball. To further explore the vulnerability of DNNs, various attacks have been developed under different settings (Goodfellow et al. 2014; Madry et al. 2018; Carlini and Wagner 2017; Croce and Hein 2020; Sun et al. 2022b). In general, adversarial attacks can be mainly categorized into white-box attacks and black-box attacks.

In white-box setting, the attackers have access to all the information of victim models, such as the model parameters and structure. Since the gradient information can be fetched, most white-box attacks utilize gradients to obtain the perturbations on the inputs which maximize the loss function. Goodfellow *et al*. introduce an efficient yet effective attack method via the sign of gradients, named Fast Gradient Sign Method (FGSM) (Goodfellow et al. 2014). Kurakin *et al*. proposed to adopt basic iterative method for FGSM, which achieves a higher attack success rate (Kurakin et al. 2018). Projection Gradient Descent (PGD) proposed to randomly initialize the adversarial examples within the ℓ_p ball (Madry et al. 2018). Carlini and Wagner (CW) attack proposed to treat adversarial attack as a constrained optimization problem (Carlini and Wagner 2017). Some feature-disruption-based attacks are introduced (Inkawhich et al. 2020; Yu et al. 2021). Dong *et al*. explored various momentum-based

iterative attack algorithm and proposed Momentum Iterative Fast Gradient Sign Method (MI-FGSM), which showed that the momentum term in the iterations can stabilize the updating direction and prevent local maxima (Dong et al. 2018).

In black-box setting, the attackers have no access to the information of victim models. One of the sub-categories of black-box attack is query-based methods where the perturbations can be approximated via huge number of queries (Cheng et al. 2018; Andriushchenko et al. 2020; Guo et al. 2019a). However, massive queries can be easily detected in real scenarios, which motivates some works to focus on efficiency (Sun et al. 2022a; Sun et al. 2022b). Another sub-category lies in transfer-based methods where the attacks have full access to a surrogate model and aims at generating adversarial examples with higher transferability (Papernot et al. 2016; Liu et al. 2016; Wang and He 2021). Although transfer-based methods dismiss the massive queries, they can hardly achieve satisfactory attack success rate on robust models.

Recently, an ensemble of multiple attacks is introduced (Croce and Hein 2020; Liu et al. 2022b). In Auto Attack (Croce and Hein 2020), four different diverse attacks including both white-box and black-box attacks are utilized in a specific order, which achieves state-of-the-art attacking performance. Due to its superior performance, Auto Attack is currently one of the most important criteria of network adversarial robustness evaluation.

2.2 Adversarial Defense

Defending adversarial attacks becomes a crucial problem which has attracted increasing attention (Madry et al. 2018; Cohen et al. 2019). The main stream of defence mechanisms lies in the adversarial training and it remains relatively resistant to most existing attacks. Vanilla adversarial training strategy simply takes adversarial examples as training data to form a min-max game during optimization (Madry et al. 2018). There exist a large number of variants of adversarial training algorithms, which improve the adversarial robustness performance (Rice et al. 2020; Shafahi et al. 2019; Xie et al. 2020b). Zhang *et al.* introduced friendly adversarial training which adopts early-stopped PGD attack to improve natural generalization of networks (Zhang et al. 2020). Rice *et al.* explored the importance of early-stopping

strategy in adversarial training (Rice et al. 2020). TRADES was introduced to achieve better trade-offs between adversarial robustness and natural accuracy (Zhang et al. 2019a).

Adversarial example detection is another stream which aims at discovering the adversarial examples and rejects them (Hendrycks and Gimpel 2016; Grosse et al. 2017). Feinman *et al*. proposed to randomize the classifier with Dropout to identify the adversarial examples based on the prediction variance (Feinman et al. 2017). Some regularization methods have been introduced to defend against attacks. (Cisse et al. 2017; Weng et al. 2018) proposed to constrain the Lipschitz constant of network to improve the adversarial robustness. Mustafa *et al*. introduced an effective constraint which forced the features for each class to lie inside a convex polytope and separated from those of other classes (Mustafa et al. 2020).

Besides traditional adversarial training, there exist randomized techniques for adversarial robustness (Pinot et al. 2020; He et al. 2019; Jeddi et al. 2020). Liu *et al*. proposed to inject random noises before the convolutional layers, which forms a noisy model to defend against adversarial examples (Liu et al. 2018b). Pinot *et al*. provided theoretical evidence that deterministic classifier can hardly ensure optimal robustness against all potential adversarial attacks and a mixture of classifiers can offer better robustness (Pinot et al. 2020). Fu *et al*. proposed to utilize random bits for adversarial defense (Fu et al. 2021). Although the methods of noise injection can be diverse, how the noise injection influence the natural generalization as well as convergence of networks has not been well explored, which could constrained the robustness.

2.3 Neural Architecture Search

Although vast approaches have been proposed to defend against adversarial samples, most of them focused on optimizing the weights based on different strategies, and the impact of architecture has been ignored. Recently, neural architecture search has received increasing attention due to its superior performance. Early NAS approaches heavily relied on macro searching which directly searches the entire network (Brock et al. 2017; Zoph and Le 2016). For efficiency, more NAS approaches have applied micro search space where the cell is

searched instead of the entire network, and the cells are stacked in series to compose the whole network (Pham et al. 2018; Zoph et al. 2018). Yang *et al*. proposed an efficient continuous evolutionary approach on the supernet where all the architectures share the parameters, which boosted the searching efficiency (Yang et al. 2020). Recently, the differentiable searching algorithm DARTS has been introduced to boost the searching speed through a relaxation on search space to form a supernet with operation mixture to achieve differentiable architecture searching (Liu et al. 2018a). Dong *et al*. introduced a differentiable sampler over the supernet with Gumbel-Softmax which improved the searching efficiency (Dong and Yang 2019). Xu *et al*. proposed to apply channel sampling for searching acceleration and add edge normalization to stabilize the searching phase (Xu et al. 2019). Recently, NAS has been applied to different areas, including adversarial robustness. Guo *et al*. empirically demonstrated that different architectures had different levels of robustness and proposed feature flow guided search to discover the robust neural architectures (Guo et al. 2020). Chen *et al*. proposed ABanditNAS with improved conventional bandit algorithm to search the architectures under enlarged search space to better defend against adversarial attacks (Chen et al. 2020a). One concern of NAS for adversarial robustness is the computational cost since both adversarial training and supernet optimization can be time-consuming. Kotyan *et al*. (Vargas et al. 2019) investigated the potential robust architecture in a broader search space including the concatenation and connections between dense and convolution layers, and demonstrated that there exist robust architectures which achieve inherent accuracy on adversarial examples. Different from (Vargas et al. 2019), our objective aims at discovering the robust architecture in the current popular search space benchmark (Liu et al. 2018a; Dong and Yang 2019) without expensive adversarial training via investigating the connection between Lipschitz constraint and adversarial robustness of architecture.

2.4 Basic Operation Design

In convolutional neural networks, the cross correlation is utilized for similarity measurement between filters and activation. However, this measurement requires massive multiplication, which has large energy consumption. For the actual deployment on resource-constrained

devices, more attention has been paid to other basic operations which can replace the one in CNNs to reduce energy consumption, such as quantization techniques which project full-precision weights and activations to fixed-point numbers with low bit-width (Zhou et al. 2016; Rastegari et al. 2016; Howard et al. 2019). Zhou *et al*. (Zhou et al. 2016) introduce DoReFa-Net which accelerates both training and inference phases via bit convolution kernels. Li *et al*. (Li et al. 2021b) shed light on the reconstruction granularity through exploring the second-order Taylor expansion and introduce block reconstruction to boost the performance of post-training quantization. Liu (Liu et al. 2022a) introduce a piecewise quantization method to tackle the gradient quantization noise. Besides quantization networks, the expensive multiplications in CNNs can be replaced by cheap additions. Chen *et al*. (Chen et al. 2020b) introduce ANNs which uses ℓ_1-norm for similarity measurement between filters and activations.

2.5 Normalization Layer

Normalization layer is a fundamental component in DNNs, which standardize features along different dimensions. It is widely used in different DNNs since it stabilize the gradient descent step, faster training, and provides better generalization. There exist various normalization layers, including Layer Normalization (LN), Group Normalization (GN), Instance Normalization (IN), Batch Normalization (BN), and Batch Group Normalization (BGN) (Ba et al. 2016; Wu and He 2018; Ulyanov et al. 2016; Ioffe and Szegedy 2015; Zhou et al. 2020). In terms of influence of BN on adversarial robustness, Xie *et al*. (Xie and Yuille 2019) explore the robustness at different network scales and introduce a mixture of two BN layers which take care of clean and adversarial examples separately to improve the trade-offs between clean and adversarial accuracy. The mixture of BN can also improve the generalization of network with adversarial training (Xie et al. 2020a). Benz *et al*. (Benz et al. 2021) provide empirical evidence that BN increase the adversarial vulnerability.

2.6 Random Projection

Random projection is a classic technique in dimensionality reduction (Bingham and Mannila 2001). Through controlling the distribution and dimensionality of random projection matrices,

the pairwise distances between any two data points can be preserved after the projection, which is stated in Johnson-Lindenstrauss lemma (Vershynin 2018). Furthermore, the projection is achieved by a simple linear transformation via random projection matrices, whose entities are sampled from a predefined distribution, random projection is both efficient and effective in practice. Due to its effectiveness, some work proposed to incorporate random projection into DNNs (Nachum et al. 2022; Zhang et al. 2019b).

CHAPTER 3

Neural Architecture Search for Adversarial Robustness

3.1 Motivation

Recent works (Nguyen et al. 2015; Moosavi-Dezfooli et al. 2016; Moosavi-Dezfooli et al. 2017; Biggio et al. 2013; Xie et al. 2019) have shown that DNNs are vulnerable to adversarial samples that can fool the networks to make wrong predictions with only perturbations of the input data, which has caused security issues. To deal with the threat of adversarial samples, the majority of existing works focus on robust training which optimizes the weights of robust DNNs through feeding adversarial samples generated by attack approaches (e.g. FGSM, PGD). However, less attention has been put into the architecture designs. A recent study has shown that different architectures tend to have different levels of adversarial robustness (Guo et al. 2020). Thus, the designing of robust neural architectures becomes essential for robustness improvement. However, the problem remains since designing a robust neural architecture can be rather expensive due to the substantial time cost and human effort, and the direct relationship between adversarial robustness and architectures is still unexplored.

To reduce the cost of discovering superior robust neural architecture, we made use of NAS algorithms which automatically discover the ideal architectures within a predefined search space. Recently, remarkable progress has achieved in NAS, including RL-based approaches (Zoph and Le 2016; Baker et al. 2016; Bello et al. 2017) and gradient-based approaches (Liu et al. 2018a; Dong and Yang 2019; Xu et al. 2019; Tang et al. 2020). In particular, DARTS (Liu et al. 2018a) introduced a differentiable method for architecture optimization through a continuous relaxation on discrete search space through forming a weighted sum of operations instead of discrete architecture selection, which significantly reduced the searching budget.

Although the NAS framework provided an efficient way to automatically discover the superior neural architectures with customized objective, the standard adversarial training required massive cost in generating the adversarial examples, which significantly decreased the search efficiency. Thus, we tried to dismiss the inner maximum of adversarial training to further accelerate the optimization of architecture through involving the Lipschitz constraint by exploring the influence of Lipschitz constant on adversarial robustness and how the architecture parameters impact the Lipschitz constant. In this thesis, we proposed to explore the relationship between adversarial robustness and the architecture of network through establishing their connections to Lipschitz constant under NAS framework.

Furthermore, the instability of differentiable NAS algorithm has been explored by previous work (Zela et al. 2019). Existing differentiable NAS algorithms used to utilize architecture parameters for sampling superior architectures, where all the elements of architecture parameters are “equally treated" for selection without exploring their discrepancies. For example, two nodes in the same cell may have different levels of freedom of selecting operation, however, they were only assigned with trainable parameters and applied with $argmax$ for selection after searching, which significantly reduced the reliability of sampled architecture and raises a demand for confidence learning of architecture parameters. Thus, we proposed to sample architecture parameters from trainable distributions instead of initializing them directly.

Our proposed algorithm Adversarially Robust Neural Architecture Search with Confidence Learning (RACL) starts from the approximation of Lipschitz constant of entire neural network under NAS framework, where we derive the relationship between Lipschitz constant and architecture parameters. We further propose to sample architecture parameters from log-normal distributions. With the usage of the properties of log-normal distribution, we show that the Lipschitz constant of entire network can be approximated with another log-normal distribution with mean and variance related to architecture parameters so that a constraint can be formulated in a form of cumulative function to achieve Lipschitz constraint on the architecture. Our algorithm achieves an efficient robust architecture search and RACL empirically achieves superior adversarial robustness compared with other NAS algorithms as well as state-of-the-art models through a series of experiments under different settings.

3.2 Methodology

In this section, we introduce the proposed robust architecture search with confidence learning (RACL) algorithm. Different from existing defence methods which only focus on weights optimization, we lay emphasis on the influence of architecture on adversarial robustness through exploring the relationship among robustness, architecture, and Lipschitz constant. Confidence learning is further involved to form a Lipschitz constraint on architecture parameters. With the proposed algorithm, the searched architectures can have stronger defensive power against adversarial examples.

3.2.1 Preliminary

Given the input $x \in \mathbb{R}^D$ and annotated label vector $y \in \mathbb{R}^M$ where M is the total number of classes, the neural network $\mathcal{H}$ maps perturbed input $\tilde{x} = x + \delta$ to a label vector $\hat{y} = \mathcal{H}(\tilde{x}; W, \mathcal{A})$. The network architecture is represented by $\mathcal{A}$, and its filter weight is denoted as W. The objective of adversarial attacks is to find the perturbed input $\tilde{x}$ which leads to wrong predictions through maximizing the classification loss as

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leqslant \epsilon}{\operatorname{argmax}} \mathcal{L}_{CE}(\mathcal{H}(\tilde{x}; W, \mathcal{A}), y), \tag{3.1}$$

where $\mathcal{L}_{CE}(\hat{y}, y) = -\sum_{i=1}^{M} y^{(i)} log(\hat{y}^{(i)})$, and the perturbation is constrained by its l_p-norm. Various powerful attacks have been proposed and shown high attack success rates, such as Fast Gradient Sign Method (FGSM) (Szegedy et al. 2014) and Projected Gradient Descent (PGD) (Madry et al. 2018). To defend against these attacks, regularizing the weight matrix of each layer to form a Lipschitz-constrained network has been proven to be beneficial for the adversarial robustness (Cisse et al. 2017; Weng et al. 2018).

Let $\mathcal{F} = \mathcal{L} \circ \mathcal{H}$ be the mapping from the input to the classification loss, and the difference of loss after an adversarial attack can be bounded as

$$\|\mathcal{F}(x + \delta, y; W, \mathcal{A}) - \mathcal{F}(x, y; W, \mathcal{A})\| \leqslant \lambda_{\mathcal{F}} \|\delta\|, \tag{3.2}$$

where $\lambda_{\mathcal{F}}$ is the Lipschitz constant of function $\mathcal{F}$ with respect to $\|.\|_p$. Together with $\|\delta\|_p \leqslant \epsilon$, the generalization error with perturbed input can be bounded as

$$\begin{aligned} \mathop{\mathbb{E}}_{x\backsim\mathcal{D}}[\mathcal{F}(\tilde{x})] &\leqslant \mathop{\mathbb{E}}_{x\backsim\mathcal{D}}[\mathcal{F}(x)] + \mathop{\mathbb{E}}_{x\backsim\mathcal{D}}[\max_{\|\tilde{x}-x\|\leqslant\epsilon} |\mathcal{F}(\tilde{x}) - \mathcal{F}(x)|] \\ &\leqslant \mathop{\mathbb{E}}_{x\backsim\mathcal{D}}[\mathcal{F}(x)] + \lambda_{\mathcal{F}} \cdot \epsilon, \end{aligned} \tag{3.3}$$

which suggests that neural networks can defend against adversarial examples with a smaller Lipschitz constant. Although it is difficult to derive the precise Lipschitz constant $\lambda_{\mathcal{F}}$ given a network, we can impose constraints on both the lower bound and upper bound of Lipschitz constant, which are denoted as $\underline{\lambda_{\mathcal{F}}}$ and $\overline{\lambda_{\mathcal{F}}}$ respectively. Thus, an adversarial robust formulation of neural architectures can be written as

$$\min_{\mathcal{A},W} \mathbb{E}[\mathcal{F}(x, y; W, \mathcal{A})] \; s.t. \; \underline{\lambda^*_{\mathcal{F}}} \leqslant \lambda_{\mathcal{F}} \leqslant \overline{\lambda^*_{\mathcal{F}}}, \tag{3.4}$$

where $\underline{\lambda^*_{\mathcal{F}}}$ and $\overline{\lambda^*_{\mathcal{F}}}$ are the optimal lower and upper bounds of Lipschitz constant. Existing works often consider a fixed network architecture $\mathcal{A}$ in Eq. (3.4), and focus on optimizing network weight for improved robustness, where the influence of architecture is ignored. Recent studies highlight the importance of architecture. Liu *et al.* conducts thorough experiments to empirically demonstrate that the better trade-offs of some pruning techniques mainly come from the architecture itself (Liu et al. 2018c). Boosting NAS algorithms involve optimization of architecture to obtain better performance with small model size (Liu et al. 2018a; Dong and Yang 2019). We are therefore motivated to investigate the influence of neural architecture on adversarial robustness.

3.2.2 Lipschitz Constraints in Neural Architecture

The discrete architecture $\mathcal{A}$ is determined by both connections and operations, which creates a huge search space. Differentiable Architecture Search algorithms provide an efficient solution through the continuous relaxation of the architecture representation (Liu et al. 2018a; Dong and Yang 2019; Xu et al. 2019). Within the differentiable NAS framework, we decompose the entire neural network into cells. Each cell I is a directed acyclic graph (DAG) consisting of an ordered sequence of n nodes, where each node denotes a latent representation that is

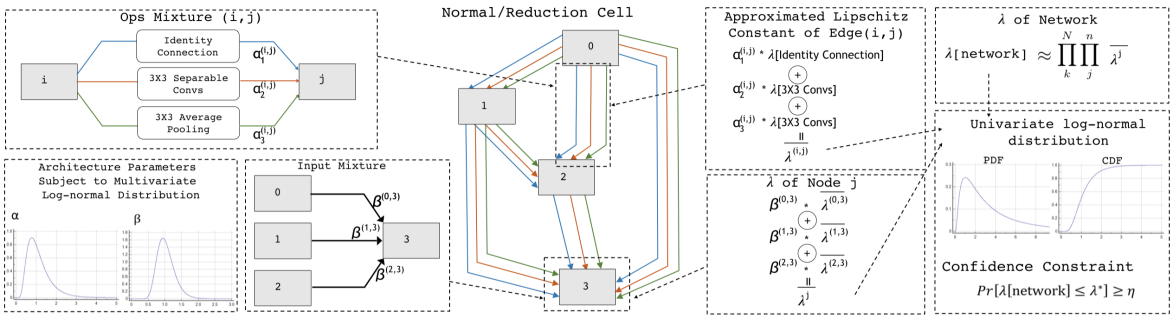


FIGURE 3.1. An overview of proposed robust neural architecture search with confidence learning algorithm. Each node in the cell is computed with operations mixture under architecture parameters α for weighting operations and β for weighting inputs where α and β are sampled from multivariate log-normal distributions. Meanwhile, the Lipschitz constant of each edge and cell induce to univariate log-normal distributions. The Lipschitz constraint is formulated from cumulative distribution function.

transformed from two previous latent representations and each edge (i, j) denotes an operation o from a pre-defined search space $\mathcal{O}$ which transforms $I^{(i)}$. Following (Xu et al. 2019), the architecture parameters α which weighs operations, and β which weighs input flows are introduced to form an operation mixture with weighted inputs. The intermediate node is computed as

$$I^{(j)} = \sum_{i<j} \beta^{(i,j)} \sum_{o \in \mathcal{O}} \alpha_o^{(i,j)} \cdot o(I^{(i)}), \tag{3.5}$$

where $I^{(0)}$ and $I^{(1)}$ are fixed as inputs nodes during the searching phase and the last node is formed by channel-wise concatenating of previous intermediate nodes $I = \cup_{i=2}^{n-1} I^{(i)}$ as the output of cell.

The entire neural network is constructed through two different types of cells including the normal cell, where all the operations have strides of 1, and the reduction cell, where the operations connected to the two inputs have strides of 2. With normal and reduction cells stacked in series, the entire neural network can be formed as $\mathcal{H} = I_1 \circ I_2 \circ \cdots \circ I_N \circ \mathcal{C}$, where N denotes the number of cells and $\mathcal{C}$ denotes the classifier. Following (Xu et al. 2019), after the searching phase, the operation o with the maximum $\beta^{(i,j)}\alpha_o^{(i,j)}$ for each edge (i, j)

is selected and the connection of each node j to its two precedents $i < j$ with maximum $\beta^{(i,j)}\alpha_o^{(i,j)}$ is selected so that a discrete superior architecture can be sampled from the supernet.

We now explore the relationship between architecture parameters α, β and Lipschitz constant of the network. Since the entire neural network is constructed by stacking cells in series as $[I_1, I_2, ..., I_N]$, Eq. 3.2 can be further decomposed as as

$$\begin{aligned} \|\mathcal{F}(\tilde{x}) - \mathcal{F}(x)\| &\leqslant \lambda_l \|\mathcal{H}(\tilde{x}) - \mathcal{H}(x)\| \\ &\leqslant \lambda_l \lambda_{\mathcal{C}} \|I_N(\tilde{x}) - I_N(x)\| \\ &\leqslant \lambda_l \lambda_{\mathcal{C}} \lambda(I_N) \|I_{N-1}(\tilde{x}) - I_{N-1}(x)\|, \end{aligned} \tag{3.6}$$

where λ_l, $\lambda_{\mathcal{C}}$ and $\lambda(I_N)$ denote the Lipschitz constants of the loss function, classifier, and cell I_N respectively. By rewriting $\|I_{N-1}(\tilde{x}) - I_{N-1}(x)\|$ in a format of its previous cells till the input of cell becomes the image for I_1 and considering $\|I_1(\tilde{x}) - I_1(x)\| \leqslant \lambda(I_1)\|\tilde{x} - x\| = \lambda(I_1)\|\delta\|$, Eq. 3.6 can be unfolded recursively and rewritten as

$$\|\mathcal{F}(\tilde{x}) - \mathcal{F}(x)\| \leqslant \lambda_{\mathcal{F}} \|\delta\| \leqslant \|\delta\| \lambda_l \lambda_{\mathcal{C}} \prod_{k}^{N} \lambda(I_k). \tag{3.7}$$

It is obvious that the adversarial robustness can be bounded by the Lipschitz constants of cells. Eq. 3.7 also suggests that the impact of perturbation grows exponentially with the number of cells, which further highlights the influence of cell designing.

As λ_l and $\lambda_{\mathcal{C}}$ in Eq. 3.7 are not related to the architecture, we next focus on the discussion on $\lambda(I_k)$. Based on the operation mixture defined in Eq. 3.5, the variation of node $I_k^{(j)}$ under perturbation can be written in a format of that in previous node $I_k^{(j)}$. For simplicity of notation, we omit the subscript k and for each node and we have

$$\begin{aligned} &\|I^{(j)}(\tilde{x}) - I^{(j)}(x)\| \leqslant \sum_{i<j} \beta^{(i,j)} \lambda^{(i,j)} \|I^{(i)}(\tilde{x}) - I^{(i)}(x)\|, \\ &s.t.\ \lambda^{(i,j)} \leqslant \sum_{o \in \mathcal{O}} \alpha_o^{(i,j)} \lambda_o, \end{aligned} \tag{3.8}$$

where $\lambda^{(i,j)}$ denotes the Lipschitz constant of transformation from node i to j and λ_o denotes the Lipschitz constant of operation o. Similarly, we can unfold Eq. 3.8 recursively for entire

cell by rewriting $\|I^{(i)}(\tilde{x}) - I^{(i)}(x)\|$ in a format of its previous node, and have

$$\lambda(I^{(j)}) \leqslant \sum_{i<j} \beta^{(i,j)} \sum_{o \in \mathcal{O}} \alpha_o^{(i,j)} \lambda_o. \tag{3.9}$$

Through substituting $\lambda(I_k)$ in Eq. 3.7 by the one in Eq. 3.9 and taking λ_l and $\lambda_{\mathcal{C}}$ as a unified constant C, the lipschitz constant $\lambda_{\mathcal{F}}$ is bounded by the product of the Lipschitz constant of intermediate nodes as

$$\begin{aligned} \lambda_{\mathcal{F}} &\leqslant C \prod_{k}^{N} \lambda(I_k) \leqslant C \prod_{k}^{N} \prod_{j}^{n} \lambda(I^{(j)}) \\ &\leqslant C \prod_{k}^{N} \prod_{j}^{n} \sum_{i<j} \beta^{(i,j)} \sum_{o \in \mathcal{O}} \alpha_o^{(i,j)} \lambda_o. \end{aligned} \tag{3.10}$$

According to the definition, the Lipschitz constant of operations without convolutional layers can be summarized as follows, (1). average pooling: $S^{-0.5}$ where S denotes the stride of pooling layer, (2). max pooling: 1, (3). identity connection: 1, (4). Zeroize: 0. For the rest operations including depth-wise separate conv and dilated depth-wise separate conv, we focus on the L_2 bounded perturbations and according to the definition of spectral norm, the Lipschitz constant of these operations is the spectral norm of its weight matrix where $\lambda_2^o = \|W^o\|_2$, which also is the maximum singular value of W, marked as Λ_1. However, directly computing Λ_1 is not practical through gradient descent. To achieve a differentiable optimization on Lipschitz constant, we make use of the power iteration method which can be applied for an efficient approximation of Λ_1 (Yoshida and Miyato 2017). Note that although the perturbation is L_2 bounded, the robustness against L_∞ can be also achieved, as stated by (Qian and Wegman 2018).

3.2.3 Confident Architecture Sampling

The architecture is determined by parameters α and β, which further influences the Lipschitz constants of the network, as shown in Eq. 3.10. Existing NAS algorithms used to initialize them as trainable parameters without in-depth analysis. However, these weightings on operations or connections naturally could have different levels of importance and freedom.

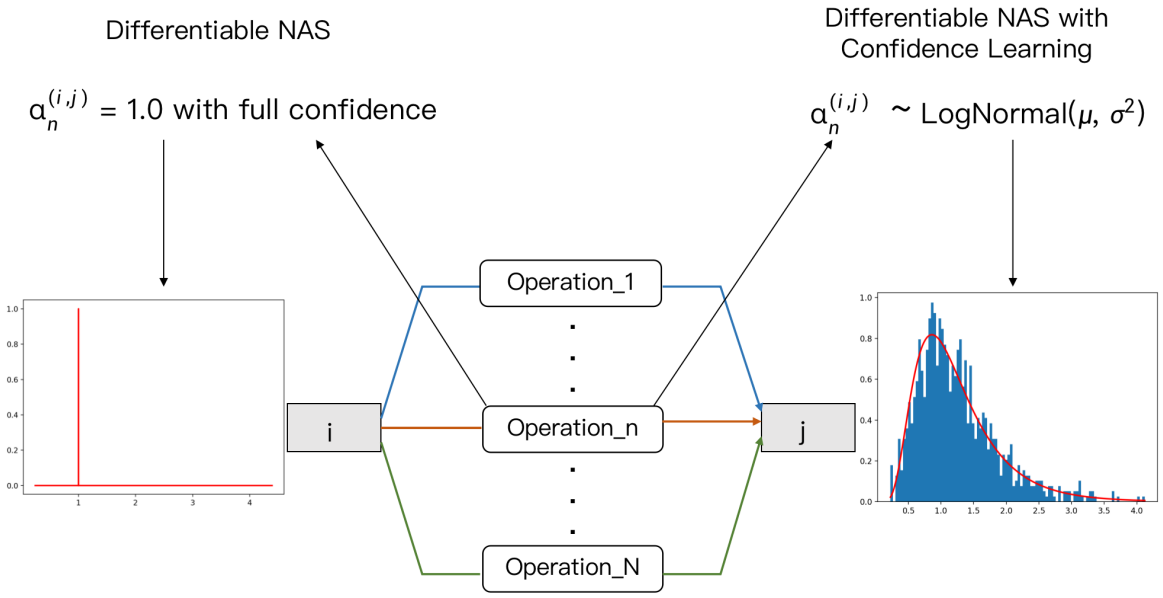


FIGURE 3.2. An illustration of the difference between previous differentiable NAS and ours with confidence learning.

E.g., the connection of the first node and one of the intermediate nodes may have different levels of freedom for the final selection, but they are simply assigned values with the same confidence for optimization and sampled with the maximum value in NAS framework. Thus, previous architecture parameters can hardly fulfill this requirement. Instead, we propose to explore the confidence on the architecture parameters by regarding them as variables sampled from distributions during architecture search. An illustration of the advantage of confidence learning is shown in Fig 3.2. For each architecture parameters, previous differentiable NAS algorithms made use of trained value with full confidence as shown in the left part, while our algorithm enables parameters to exploit their confidence as shown in the right part. Intuitively, the architecture optimization is highly uncertain due to the large search space. But existing NAS absolutely trusts all architecture parameters without discriminating the confidence on them. Taking α and β from the perspective of distributions, the variances will be optimized to indicate confidence in the values. RACL tends to exploit the operations of higher confidence and explore more potential good paths by investigating operations of lower confidence. The overall searching space can thus be well explored and exploited.

For distributions, a naive selection can be a multivariate normal distribution. However, according to the Lipschitz constant form in Eq. 3.10, the sampled values from this distribution need to be positive since $\lambda_{\mathcal{F}}$ is always positive and negative values from distribution will make the constraint cease to be effective. Thus, we turn to log-normal distribution $\mathcal{LN}$ since it guarantees positive sampled values. Note that a random variable is log-normally distributed $\mathcal{LN}(\mu, \Sigma)$ if the logarithm of it is normally distributed $\mathcal{N}(\mu, \Sigma)$. For simplicity, the following mean and variance denote those of the logarithm. Most importantly, there are several nice properties, including the weighted sum of multiple independent $\mathcal{LN}_{1,...,n}$ can be approximated with another $\mathcal{LN}$ and the product of multiple independent $\mathcal{LN}_{1,...,n}$ induces to $\mathcal{LN}$ with parameters μ and Σ of the sum of those in $\mathcal{LN}_{1,...,n}$. Thus we propose to sample α from multivariate log-normal distributions, denoted as $\mathcal{LN}(\mu^\alpha, \Sigma^\alpha)$, with mean $\mu^\alpha \in \mathbb{R}^d$ and covariance matrix $\Sigma^\alpha \in \mathbb{R}^{d\times d}$ with diagonal standard deviation $\sigma^\alpha \in \mathbb{R}^d$ where d denotes the dimension of α. Similarly, we sample β from $\mathcal{LN}(\mu^\beta, \Sigma^\beta)$.

Back to the Lipschitz constant, the multivariate log-normal distribution over α induces a univariate log-normal distribution over the upper boundary of Lipschitz constant of edge based on the operation mixture $\sum_{o\in\mathcal{O}} \alpha_o^{(i,j)} \lambda_o$ since it can be treated as the weighted sum of multiple log-normal distributed variables. Note that λ_o is treated as constant here since the weights are fixed when optimizing architecture parameters. The is proposed Lipschitz confidence constraint is shown in Fig. 3.1. Although there is no closed-form expression of its probability density function, the distribution can be approximated using the properties of log-normal distribution as:

Property 1. If a log-normal variable $X \backsim \mathcal{LN}(\mu, \sigma^2)$ is multiplied by a constant a, $aX \backsim \mathcal{LN}(\mu + ln(a), \sigma^2)$.

Property 2. If multiple independent log-normal variables, denoted as $X_1, X_2, ..., X_n$, are multiplied, $X_1 \cdot X_2 \cdots X_n \backsim \mathcal{LN}(\sum_{i=1}^n \mu_i, \sum_{i=1}^n \sigma_i^2)$

Following (Marlow 1967), the sum of log-normal distributions can be approximated by another log-normal distribution as below:

PROPOSITION 1. *If multiple independent log-normal variables, denoted as $X_1, X_2, ..., X_n$, are added, the sum $Z = \sum_{i=1}^{n} X_i$ can be approximated by another log-normal distribution $\mathcal{LN}(\mu_Z, \sigma_Z^2)$ with variance $\sigma_Z^2 = ln[\frac{\sum e^{(2(\mu_i)+\sigma_i^2)}(e^{(\sigma_i)^2}-1)}{(\sum e^{(\mu_+\sigma_i^2/2)})^2} + 1])$ and mean $\mu_Z = ln[\sum e^{(\mu_i+\sigma_i^2/2)}] - \frac{\sigma_Z}{2}$.*

Thus, in Eq. 3.8, the distribution of $\sum_{o\in\mathcal{O}} \alpha_o^{(i,j)}\lambda_o$ can be treated as a weighted sum of multiple independent log-normal distributions and can be approximated with these properties and Proposition 1. Similarly, in Eq. 3.9, $\sum_{i<j} \beta^{(i,j)} \sum_{o\in\mathcal{O}} \alpha_o^{(i,j)}\lambda_o$ can be treated as the sum of multiple products of two independent log-normal distributions which can also be approximated. Based on the properties of log-normal distribution, the upper boundary of Lipschitz constant of entire network can be approximated accordingly,

$$\begin{aligned}
\overline{\lambda^{(i,j)}} &= \sum_{o\in\mathcal{O}} \alpha_o^{(i,j)}\lambda_o \backsim \mathcal{LN}(ln[\sum_o e^{(\mu_o^{\alpha\prime}+(\sigma_o^{\alpha})^2/2)}] - \frac{\sigma_{I^{(i,j)}}^2}{2}, \sigma_{I^{(i,j)}}^2), \\
\sigma_{I^{(i,j)}}^2 &= ln[\frac{\sum_o e^{(2(\mu_o^{\alpha\prime})+(\sigma_o^{\alpha})^2)}(e^{(\sigma_o^{\alpha})^2}-1)}{(\sum_o e^{(\mu_o^{\alpha\prime})+(\sigma_o^{\alpha})^2/2)})^2} + 1]), \\
\mu_o^{\alpha\prime} &= \mu_o^{\alpha} + ln(\lambda_o),
\end{aligned} \tag{3.11}$$

where $\overline{\lambda^{(i,j)}}$ denotes the upper boundary of $\lambda^{(i,j)}$. For simplicity, we denote the mean for $\overline{\lambda^{(i,j)}}$ as $\mu_{I^{(i,j)}}$ and variance as $\sigma_{I^{(i,j)}}^2$. Similarly, we sample β from a multivariate log-normal distribution $\mathcal{N}(\mu^{\beta}, \Sigma^{\beta})$. For variable $\beta^{(i,j)}\overline{\lambda^{(i,j)}}$, it can be treated as the product of two log-normal distributions, which also follows a log-normal distribution whose mean is the sum of means of two distributions and variance as well. Thus, to generalize the distribution over edge $\overline{\lambda^{(i,j)}}$ to the one over intermediate node $\overline{\lambda^{(j)}}$, we replace o with j, $\mu_o^{\alpha} + ln(\lambda_o)$ with $\mu_{(i,j)}^{\beta} + \mu_{I^{(i,j)}}$, and $(\sigma_o^{\alpha})^2$ with $(\sigma_{(i,j)}^{\beta})^2 + \sigma_{I^{(i,j)}}^2$ in Eq 3.11 and obtain the log-normal distribution of $\overline{\lambda^{(j)}} = \sum_{i<j} \beta^{(i,j)}\overline{\lambda^{(i,j)}}$ as

$$\begin{aligned}
\overline{\lambda^{(j)}} &= \sum_{i<j} \beta^{(i,j)}\overline{\lambda^{(i,j)}} \backsim \mathcal{LN}(ln[\sum_o e^{(\mu_{(i,j)}^{\beta}{}'+[\sigma_{(i,j)}^{\beta}{}']/2)}] - \frac{\sigma_{I^{(j)}}^2}{2}, \sigma_{I^{(j)}}^2), \\
\sigma_{I^{(j)}}^2 &= ln[\frac{\sum_j e^{(2(\mu_{(i,j)}^{\beta}{}')+[\sigma_{(i,j)}^{\beta}{}'])}(e^{[\sigma_{(i,j)}^{\beta}{}']}-1)}{(\sum_o e^{(\mu_{(i,j)}^{\beta}{}'+[(\sigma_{(i,j)}^{\beta}{}']/2)})^2} + 1]), \\
\mu_{(i,j)}^{\beta}{}' &= \mu_{(i,j)}^{\beta} + \mu_{I^{(i,j)}}, \sigma_{(i,j)}^{\beta}{}' = (\sigma_{(i,j)}^{\beta})^2 + \sigma_{I^{(i,j)}}^2,
\end{aligned} \tag{3.12}$$

with mean and variance which are denoted as $\mu_{I^{(j)}}$ and $\sigma^2_{I^{(j)}}$. According to Eq. 3.10, $\lambda_{\mathcal{F}}$ is bounded by the product of $\overline{\lambda^{(j)}}$. Thus, $\overline{\lambda_{\mathcal{F}}}$ follows the log-normal distribution with mean $\mu = In(C) + \sum_k^N \sum_j^n \mu_{I^{(j)}}$ and variance $\sigma^2 = \sum_k^N \sum_j^n \sigma^2_{I^{(j)}}$. We introduce a confidence hyperparameter $\eta \in [0, 1]$ to enable confidence learning with such an constraint as

$$\begin{aligned} &Pr_{\alpha,\beta}[\overline{\lambda_{\mathcal{F}}} \leqslant \overline{\lambda^*_{\mathcal{F}}}] \\ &= Pr_{\alpha,\beta}[C \prod_k^N \prod_j^n \sum_{i<j} \beta^{(i,j)} \sum_{o \in \mathcal{O}} \alpha_o^{(i,j)} \lambda_o \leqslant \overline{\lambda^*_{\mathcal{F}}}] \geqslant \eta, \end{aligned} \tag{3.13}$$

where $\overline{\lambda^*_{\mathcal{F}}}$ is the desired Lipschitz constant upper bound of $\mathcal{F}$, Note that in Eq. 3.13, the variance of $\overline{\lambda_{\mathcal{F}}}$ is reduced to satisfy the inequality, which strengths the confidence on the approximation of Lipschitz constant of $\lambda_{\mathcal{F}}$, compared with the one in Eq. 3.10 without confidence learning. To obtain a convex constraint in μ and Σ, we reformulate Eq. 3.13 through the format of cumulative function as

$$\begin{aligned} Pr[\overline{\lambda_{\mathcal{F}}} \leqslant \overline{\lambda^*_{\mathcal{F}}}] &= Pr[\frac{ln(\overline{\lambda_{\mathcal{F}}}) - \mu}{\sigma} \leqslant \frac{ln(\overline{\lambda^*_{\mathcal{F}}}) - \mu}{\sigma}] \\ &= \Phi(\frac{ln(\overline{\lambda^*_{\mathcal{F}}}) - \mu}{\sigma}), \end{aligned} \tag{3.14}$$

where Φ denotes the cumulative function of the normal distribution since $\frac{ln(\overline{\lambda_{\mathcal{F}}}) - \mu}{\sigma}$ is a random variable following the normal distribution. Thus, we establish direct relationship among μ, σ and η as

$$\frac{ln(\overline{\lambda^*_{\mathcal{F}}}) - \mu}{\sigma} \geqslant \Phi^{-1}(\eta). \tag{3.15}$$

Through omitting the square root on σ, we achieve a convex constraint. Besides the upper bound of Lipschitz constant, we propose to minimize the lower bound $\underline{\lambda_{\mathcal{F}}}$ together to better control $\lambda_{\mathcal{F}}$. Taking the advantage of the fact that $\|\nabla \mathcal{F}(x, y; W, \mathcal{A})\| \leq \lambda_{\mathcal{F}}$, we simply take $\underline{\lambda_{\mathcal{F}}} = \|\nabla \mathcal{F}(x, y; W, \mathcal{A})\|$. Together with the constraint in Eq. 3.15, we reformulate the optimization objective in Eq. 3.4 as

$$\begin{aligned} &\min_{\mu^\alpha, \Sigma^\alpha, \mu^\beta, \Sigma^\beta, W} \mathcal{L}_{CE}(\mathcal{F}(x; W, \mathcal{A}), y) + \|\nabla \mathcal{F}(x, y; W, \mathcal{A})\|, \\ &s.t.\ ln(\overline{\lambda^*_{\mathcal{F}}}) - \mu \geqslant \Phi^{-1}(\eta)\sigma^2, \mathcal{A} \backsim \mathcal{LN}(\mu^\alpha, \Sigma^\alpha), \mathcal{LN}(\mu^\beta, \Sigma^\beta). \end{aligned} \tag{3.16}$$

Algorithm 1 Robust Neural Architecture Search with High Confidence Algorithm

Input: The training set is split into $\mathcal{D}_T$ and $\mathcal{D}_V$; Batch size n; Hyperparameter λ^*, ρ, η; Initialize multivariate log-normal distributions $\mathcal{LN}(\mu^\alpha, \Sigma^\alpha)$ and $\mathcal{LN}(\mu^\beta, \Sigma^\beta)$; Initialize $\mathcal{H}$ with W;
while not converge **do**
Sample α and β from $\mathcal{LN}(\mu^\alpha, \Sigma^\alpha)$ and $\mathcal{LN}(\mu^\beta, \Sigma^\beta)$ based on reparameterization trick
Sample batch of data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{D}_T$
Optimize W with $\sum_{i=1}^n \mathcal{L}_{CE}(x_i, y_i; W, \alpha, \beta) + \underline{\lambda_{\mathcal{F}}}$
Sample batch of data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{D}_V$
Optimize $\mu^\alpha, \Sigma^\alpha, \mu^\beta, \Sigma^\beta$ with ADMM framework
$\mu_{t+1} \leftarrow \mu_t - \gamma \nabla_\mu [\mathcal{L}_{CE} + \underline{\lambda_{\mathcal{F}}} + \theta(c(\mu, \Sigma_t)) + \frac{\rho}{2}\|c(\mu, \Sigma_t)\|_F^2]$
$\sigma_{t+1} \leftarrow \sigma_t - \gamma \nabla_\sigma [\mathcal{L}_{CE} + \underline{\lambda_{\mathcal{F}}} + \theta(c(\mu_t, \Sigma)) + \frac{\rho}{2}\|c(\mu_t, \Sigma)\|_F^2]$
Optimize θ with $\theta_{t+1} \leftarrow \theta_t + \rho \cdot c(\mu_t, \Sigma_t)$
end while
Sample the normal and reduction cell based on sampled α and β
Retrain the searched architecture from scratch on training set

Intuitively, the constraint in Eq. 3.16 reveals the influence of σ on sampling architecture parameters. As σ increases, the value of μ decreases to satisfy the inequality where the corresponding μ^α and μ^β become 0 for relatively large σ, which implies that the operations or connections are unlikely to be sampled when its corresponding confidence is low. Thus, the architecture can be sampled based on its confidence in the Lipschitz constraint. We apply the ADMM optimization framework to solve this constrained optimization through incorporating the constraint to form a minimax problem so that Eq. 3.16 can be rewritten as

$$\begin{aligned} &\min_{\mu^\alpha, \Sigma^\alpha, \mu^\beta, \Sigma^\beta} \max_\theta \mathcal{L}_{CE} + \underline{\lambda_{\mathcal{F}}} + \theta(c(\mu, \Sigma)) + \frac{\rho}{2}\|c(\mu, \Sigma)\|_F^2, \\ &c(\mu, \Sigma) = \mu + \Phi^{-1}(\eta)\sigma^2 - ln(\overline{\lambda_{\mathcal{F}}^*}), \end{aligned} \tag{3.17}$$

where θ is the dual variable and ρ is positive number predefined in ADMM. The first step is to update μ while fixing other variables and the second step is to update σ while fixing other variables as

$$\begin{aligned} \mu_{t+1} &\leftarrow \mu_t - \gamma \nabla_\mu [\mathcal{L}_{CE} + \underline{\lambda_{\mathcal{F}}} + \theta(c(\mu, \Sigma_t)) + \frac{\rho}{2}\|c(\mu, \Sigma_t)\|_F^2], \\ \sigma_{t+1} &\leftarrow \sigma_t - \gamma \nabla_\sigma [\mathcal{L}_{CE} + \underline{\lambda_{\mathcal{F}}} + \theta(c(\mu_t, \Sigma)) + \frac{\rho}{2}\|c(\mu_t, \Sigma)\|_F^2], \end{aligned} \tag{3.18}$$

TABLE 3.1. Evaluation of RACL adversarial robustness on CIFAR-10, CIFAR-100, and Tiny-ImageNet compared with various NAS algorithms under white-box attacks. PGD^{20} denotes PGD attack with 20 iterations. Best results in bold.

Dataset	Model	Params	Natural	FGSM	MIM	PGD^{20}	PGD^{100}	CW	AA
CIFAR-10	AmoebaNet	3.2M	82.28%	59.12%	57.26%	53.69%	53.34%	78.63%	47.88%
	NASNet	3.8M	**84.37%**	61.38%	58.72%	53.35%	52.84%	80.69%	48.19%
	DARTS	3.3M	80.65%	59.63%	57.55%	54.04%	53.73%	77.01%	48.13%
	PC-DARTS	3.6M	84.32%	61.08%	58.10%	53.01%	52.36%	80.54%	47.95%
	RACL(ours)	3.3M	84.04%	**62.55%**	**60.00%**	**55.68%**	**55.32%**	**80.90%**	**50.07%**
CIFAR-100	AmoebaNet	3.2M	56.51%	32.67%	31.44%	29.70%	29.66%	49.03%	25.26%
	NASNet	3.8M	57.97%	31.54%	30.14%	28.58%	28.44%	49.64%	24.42%
	DARTS	3.3M	**58.67%**	32.71%	31.14%	29.21%	29.11%	50.32%	24.30%
	PC-DARTS	3.6M	57.20%	31.85%	30.46%	28.62%	28.50%	49.40%	24.10%
	RACL(ours)	3.3M	57.83%	**33.89%**	**32.41%**	**30.41%**	**30.15%**	**52.56%**	**25.55%**
Tiny-ImageNet	AmoebaNet	3.2M	47.84%	31.44%	30.57%	30.12%	30.09%	42.56%	-
	NASNet	3.8M	47.85%	30.76%	29.80%	29.47%	29.44%	41.93%	-
	DARTS	3.3M	48.20%	31.38%	30.71%	30.30%	30.25%	42.23%	-
	PC-DARTS	3.6M	47.24%	30.04%	29.18%	28.55%	28.53%	40.91%	-
	RACL(ours)	3.3M	**48.86%**	**31.98%**	**31.12%**	**30.63%**	**30.63%**	**42.99%**	-

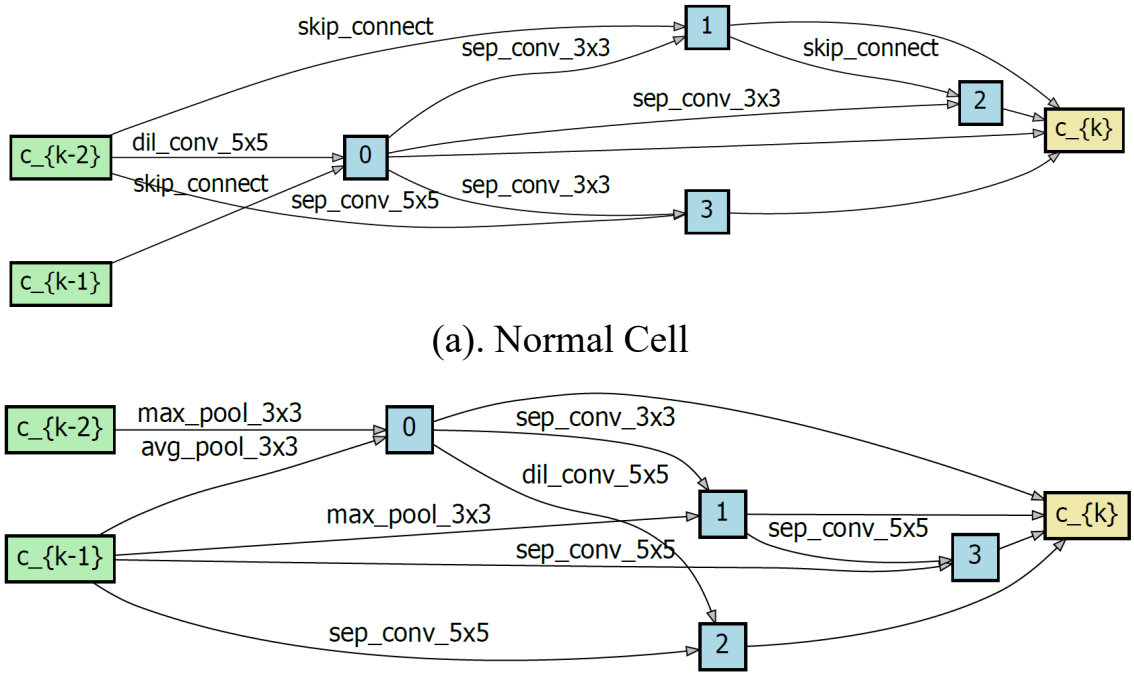
where $\mu^\alpha, \Sigma^\alpha, \mu^\beta, \Sigma^\beta$ are updated through back-propagation. The dual variable θ is updated with learning rate of ρ as

$$\theta_{t+1} \leftarrow \theta_t + \rho \cdot c(\mu_t, \Sigma_t) \tag{3.19}$$

The entire robust neural architecture search with confidence learning algorithm, denoted as RACL, is shown in Alg. 1. With proposed algorithm, we impose confidence learning on the values of architecture parameters α and β, which strengthens the confidence of robust architecture sampling.

3.3 Experiments

In this section, we conduct a series of experiments to empirically demonstrate the effectiveness of proposed RACL algorithm. We retrain the searched neural architecture and compare it with various neural architectures searched by NAS algorithms as well as state-of-the-art network architectures. We show that under various adversarial attack settings, the robust



(a). Normal Cell

(b). Reduction Cell

FIGURE 3.3. The visualization of normal and reduction cell searched by RACL are shown in (a) and (b).

neural architectures searched by RACL always achieve better robustness than other baselines.

3.3.1 Experimental Setup

Neural Architecture Search Setup Following previous works (Liu et al. 2018a; Xu et al. 2019), we search the robust neural architectures on CIFAR-10 dataset which contains 50K training images and 10K validation images over 10 classes. During the searching phase, the training set is divided into two parts with equal sizes for architecture and weight optimization respectively. The search space includes 8 candidates: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, skip connection, and zero, as suggested by previous works (Xu et al. 2019; Dong and Yang 2019). The supernet is constructed by stacking 8 cells including 6 normal cells and 2 reduction cells, each of which contains 6 nodes. For the training settings, we follow the setups of PC-DARTS

(Xu et al. 2019). The searching phase takes 50 epochs with a batch size of 128. We use SGD with momentum. The initial learning rate is 0.1 with a momentum of 0.9 and a weight decay is 3×10^{-4} to update the supernet weights. Architecture parameters were updated with Adam with a learning rate of 6×10^{-4} and a weight decay of 1×10^{-3}. The searching time of RACL takes 0.5 GPU days.

Datasets and Retrain Details We extensively evaluate the proposed algorithm on three datasets including CIFAR-10, CIFAR-100, and Tiny-ImageNet, which are widely compared by other previous work. The searched superior neural architecture is sampled based on the proposed sampling strategy. Following the setting in NAS (Xu et al. 2019; Liu et al. 2018a), we stack searched cells to form a 20-layer network and retrained it with the entire training set. For the evaluation stage, we adopt the popular adversarial training framework to retrain all the baselines. We train the network from scratch for 100 epochs with a batch size of 128 on the entire training set. We use SGD optimizer with an initial learning rate of 0.1, momentum of 0.9, and a weight decay of 2×10^{-4}. The norm gradient clipping is set to 5. Following (Wang et al. 2019), the hyperparameter which balances the adversarial loss and KL divergence is set to 6. Since we focus on the impact of architecture on adversarial robustness, we train the searched architectures as well as state-of-the-art network architectures with the same adversarial training setting. Through training these architectures in the same adversarial manner, we conduct a fair comparison among different architectures and demonstrate how they improve or constrain the adversarial robustness. For CIFAR-10 and CIFAR-100, we use adversarial training with the total perturbation size $\epsilon = 8/255$. The maximum number of attack iterations is set to 10 with a step size of $2/255$. For Tiny-ImageNet, we set $\epsilon = 4/255$. The maximum number of attack iterations is set to 6 with a step size of $2/255$. An illustration of the searched normal cell and reduction cell is shown in Figure 3.3, more analysis on searched robust neural architectures will be covered in Sec. 3.3.5

3.3.2 Against White-box Attacks

To evaluate the superiority of proposed RACL, we compare the searched cells with SOTA NAS algorithms, including DARTS (Liu et al. 2018a), PC-DARTS (Xu et al. 2019), NASNet

TABLE 3.2. Evaluation of RACL adversarial robustness on CIFAR-10 compared with other human-designed architectures and other robust NAS algorithms under white-box attacks. Best results in bold.

Model	Params	Natural	FGSM	PGD^{20}	PGD^{100}	MIM
ResNet-18	11.17M	78.38%	49.81%	45.60%	45.10%	45.23%
ResNet-50	23.52M	79.15%	51.46%	45.84%	45.35%	45.53%
WRN-28-10	36.48M	86.43%	53.57%	47.10%	46.90%	47.04%
DenseNet-121	6.95M	82.72%	54.14%	47.93%	47.46%	48.19%
ABanditNAS	5.19M	**90.64%**	-	50.51%	-	54.19%
RobNet-S	4.41M	78.05%	53.93%	48.32%	48.07%	48.98%
RobNet-M	5.56M	78.33%	54.55%	49.13%	48.96%	49.34%
RobNet-L	6.89M	78.57%	54.98%	49.44%	49.24%	49.92%
RobNet-free	5.49M	82.79%	58.38%	52.74%	52.57%	52.95%
RACL(ours)	3.34M	84.04%	**62.55%**	**55.68%**	**55.32%**	**60.00%**

(Zoph et al. 2018), AmoebaNet (Real et al. 2019). We also compare RACL with SOTA human-designed network architectures, such as ResNet and DenseNet (He et al. 2016; Huang et al. 2017). Furthermore, some NAS algorithms targeting the adversarial robustness are also included for comparison, including RobNet (Guo et al. 2020) and ABanditNAS (Chen et al. 2020a). Moreover, we also compare our results with other defence mechanisms, including Stochastic Weight Averaging (SWA) (Chen et al. 2021a) and Instance Adaptive Adversarial training (IAAT) (Balaji et al. 2019). For robustness evaluation, we choose various popular powerful attacks including Fast Gradient Sign Method (FGSM) (Szegedy et al. 2014), Momentum Iterative Method (MIM) (Dong et al. 2017), Projected Gradient Descent (PGD) (Madry et al. 2018), CW attack (Carlini and Wagner 2017), and Auto Attack (Croce and Hein 2020). Consistent with previous adversarial literature (Madry et al. 2018; Zhang et al. 2019a), the perturbation is considered under l_∞ norm with the total perturbation size of $8/255$ on CIFAR-10/100 and $4/255$ on Tiny-ImageNet. For CW attack, the steps are set to 1000 with a learning rate of 0.01.

Evaluation on CIFAR-10, CIFAR-100, and Tiny-ImageNet Although adversarial training is a strong defence method, the impact of architecture is always ignored. In this experiment, we demonstrate that constructing networks via the neural architectures searched by RACL can further improve the robustness after adversarial training. For a fair comparison, we retrain the searched cells using PGD adversarial training for all the models to evaluate the robustness of

RACL on the main benchmark of defence mechanisms. The number of PGD attack iterations is set to 20 and 100 with a step size of $2/255$, as suggested by (Guo et al. 2020). The detailed evaluation results are shown in Table 3.1. The best result for each column is highlighted in bold. As shown in Table 3.1, RACL achieves better adversarial accuracy than other state-of-the-art neural architectures on all the datasets. For example, compared with our baseline PC-DARTS on CIFAR-10, though both RACL and PC-DARTS achieve similar clean accuracy and model size, their performance with adversarial training varies differently. RACL achieves an accuracy of 62.55% under FGSM attack, with 1.47% improvement ($61.08\% \rightarrow 62.55\%$) over that of PC-DARTS, and 2.96% improvement ($52.36\% \rightarrow 55.32\%$) over that of PC-DARTS under PGD^{100} attack. Furthermore, RACL achieves the best robust accuracy than other baselines under different attacks on CIFAR-100 and Tiny-ImageNet. For example, RACL achieves an accuracy of 52.26% under Auto Attack, with 1.25% improvement ($24.30\% \rightarrow 25.55\%$) over that of DARTS on CIFAR-100. Similarly, RACL achieves an accuracy of 42.99% under CW attack, with 1.06% improvement ($41.93\% \rightarrow 42.99\%$) over that of NASNet on Tiny-ImageNet. We empirically show that RACL consistently achieves the best robust performance compared with other NAS algorithms with the same search space under various attacks, which indicates that the adversarial robustness can be further improved through imposing Lipschitz constraint on architecture parameters.

Comparison with Human-designed Architectures and Robust NAS Algorithms on CIFAR-10 Besides the standard NAS algorithms, there exist some NAS algorithm targeting adversarial robustness as well as some popular human-designed architectures which are widely compared in adversarial robustness benchmarks. We include ResNet-18, ResNet-50, WideResNet-28-10, and DenseNet-121 for comparison. The results are shown in Table 3.2. Compared with these human-designed architectures, RACL shows obvious superiority of robust accuracy over all the baselines under various attacks with fewer parameters. In terms of robust NAS algorithms, RobNet applies robust architecture search algorithm to explore a RobNet family under different budgets (Guo et al. 2020). Compared with RobNet-S, RobNet-M and RobNet-L, RACL consistently achieves the best performance with a large gap. Compared with RobNet-free which relaxes the cell-based constraint, RACL still achieves better results with fewer parameters. For example, RACL achieves an accuracy of 55.32%

TABLE 3.3. Comparison with existing defence techniques under PGD attack on different datasets.

Attack	Defence	CIFAR-10	CIFAR-100	Tiny-ImageNet
PGD^{20}	FAT (Zhang et al. 2020)	45.31%	27.38%	17.50%
	SWA (Chen et al. 2021a)	52.14%	28.28%	21.84%
	NADAR (Li et al. 2021a)	53.43%	28.40%	21.14%
	RACL(ours)	**55.68%**	**30.41%**	**30.63%**
PGD^{100}	IAAT (Balaji et al. 2019)	46.50%	24.22%	-
	RobNet-L (Guo et al. 2020)	49.24%	23.19%	19.90%
	RobNet-free (Guo et al. 2020)	52.57%	23.87%	20.87%
	RACL(ours)	**55.32%**	**30.15%**	**21.48%**

under PGD^{100} attack, with 2.75% improvement ($52.57\% \rightarrow 55.32\%$) over that of RobNet-free. Compared with ABanditNAS which includes denoise operations in its search space, RACL outperforms it in adversarial accuracy. For example, RACL achieves an accuracy of 55.32% under PGD^{100} attack, with 5.81% improvement ($54.19\% \rightarrow 60.00\%$) over that of ABanditNAS. Overall, RACL achieves superior trade-offs among parameters, clean accuracy, and adversarial accuracy, which highlights the effectiveness and efficiency of proposed RACL algorithm.

Comparison with existing defence mechanisms We argue that initializing a network with robust neural architecture can be regarded as an efficient defence method against adversarial samples. To illustrate how robust architecture improves the performance of adversarial training, we compare RACL with previously proposed defence mechanisms on different datasets, including CIFAR-10, CIFAR-100, and Tiny-ImageNet. The perturbation budget ϵ is set to $8/255$ for CIFAR-10 and CIFAR-100. For Tiny-ImageNet, we consider two perturbation budgets. Following (Li et al. 2021a; Chen et al. 2021a), the perturbation budget of PGD attack ϵ is set to $4/255$ with 20 iterations as PGD^{20}. We also consider another stronger attacking setting in Tiny-ImageNet, which follows (Guo et al. 2020). The perturbation budget of PGD attack ϵ is set to $8/255$ with 100 iterations as PGD^{100}. We include various defence mechanisms for comparison. RACL was also compared with FAT (Zhang et al. 2020) which aims at better trade-offs between natural accuracy and robustness, SWA (Chen et al. 2021a) which introduces weight smoothing to tackle the overfitting issue in adversarial training, IAAT (Balaji et al. 2019) which enforces sample-specific perturbation margins for a better

TABLE 3.4. Evaluation of RACL adversarial robustness on CIFAR-10 under transfer-based black-box attack setting.

Model	FGSM	MIM	PGD^{20}
AmoebaNet	81.92%	82.04%	82.61%
NasNet	82.32%	82.60%	83.21%
DARTS	78.76%	78.83%	79.29%
PC-DARTS	82.47%	82.65%	83.06%
RACL(ours)	**82.78%**	**82.92%**	**83.45%**

generalization, and NADAR (Li et al. 2021a) which proposes to search the dilation network for adversarial robustness. The detailed results are shown in Table 3.3. Compared with all the SOTA defence techniques, RACL consistently achieves the best performance in all the scenarios, which demonstrates the superiority of RACL as defence mechanism. Note that RACL can collaborate with other adversarial training algorithms to achieve potentially better performance.

3.3.3 Against Black-box Attacks

Transfer-based Black-box Attack Evaluation We next evaluate the robustness of RACL under black-box attacks. Following previous literature (Madry et al. 2018; Papernot et al. 2016), we apply transfer-based black-box attacks which generate adversarial samples using a victim model and feed them to the target models. In this work, we take a ResNet-110 network as the victim model. The transferred adversarial samples are generated through FGSM, MIM and PGD attacks. The adversarial accuracy of different architectures is compared after they are fed with these transferred adversarial samples, as shown in Table 3.4. Compared with other standard NAS algorithms, RACL achieves the highest robust accuracy in all the scenarios under these transfer-based attacks, which highlights the adversarial robustness of the proposed algorithm against transfer-based black-box attacks.

Transferability Test on CIFAR-10 under PGD Attack Following (Guo et al. 2020), we further conduct the transferability test on CIFAR-10. We use different NAS algorithms as source models to generate adversarial samples through 10-iteration PGD attack and feed them to other target models as cross black-box attacks. The results are shown in Table 3.5. Each

TABLE 3.5. Transferability test on CIFAR-10 among differnet models using PGD attack. The best results in each row are in bold. Underline denotes the white-box robustness.

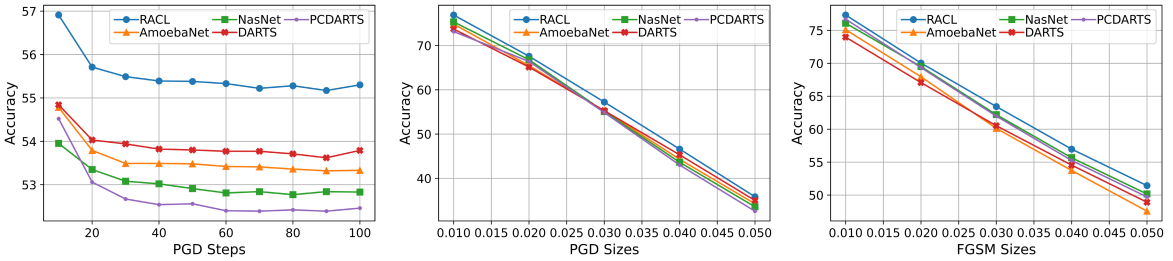
Source \ Target	AmoebaNet	NasNet	DARTS	PC-DARTS	ours
AmoebaNet	53.69	66.79	64.39	66.50	**67.21**
NasNet	64.77	53.35	64.31	65.62	**66.22**
DARTS	65.03	66.91	54.04	66.74	**67.04**
PC-DARTS	64.37	65.45	63.91	53.01	**66.23**
RACL(ours)	64.49	**66.24**	64.06	65.90	55.68

row denotes the robust accuracy of different target models under the black-box attack from the same source model. Correspondingly, each column denotes the robustness of a target model under attack from different source models. Comparing each row, RACL achieves the best accuracy under the attacks from different source models, which indicates that although these architectures are searched within the same search space, they show different robustness under attacks. The large gap between RACL and other baselines also highlights the superiority of RACL under black-box settings. Furthermore, through comparing the transferability between each model pair, RACL tends to generate stronger adversarial samples. *E.g.*, RACL $\rightarrow$ AmoebaNet achieves the successful attack success rate of 35.52% and AmoebaNet $\rightarrow$ ours achieves the successful attack success rate of 29.59%. Taking NASNet, DARTS and PC-DARTS as target models, RACL generates the adversarial samples which achieve the highest attack success rate except for the white-box attack.

Transferability Test on CIFAR-10 under RFGSM Attack Furthermore, we provide

TABLE 3.6. Transferability Test on CIFAR-10 among different models under RFGSM Attack. The best results in each row are in bold. Underline denotes the white-box robustness.

Source \ Target	AmoebaNet	NasNet	DARTS	PC-DARTS	ours
AmoebaNet	76.68	80.97	77.13	80.93	**81.22**
NasNet	78.93	78.52	77.18	80.85	**81.09**
DARTS	78.81	80.92	75.26	80.82	**81.16**
PC-DARTS	78.70	80.68	77.02	78.43	**81.16**
RACL(ours)	78.78	80.73	77.18	**80.83**	78.83



(a). Adversarial accuracy on CIFAR-10 under different PGD iterations

(b). Adversarial accuracy on CIFAR-10 under different PGD total perturbation size

(c). Adversarial accuracy on CIFAR-10 under different FGSM total perturbation size

FIGURE 3.4. Robustness evaluation under different perturbation sizes and attack iterations.

additional robustness evaluation of RACL through transferability test on CIFAR-10 under RFGSM attack (Tramèr et al. 2017a). The detailed results are shown in Table 3.6. The underline denotes the adversarial accuracy under white-box RFGSM attack where the total perturbation is set to $8/255$. Comparing the accuracy on diagonal, RACL achieves the best white-box performance under RFGSM attack. Each row denotes the robustness of different target models under the black-box attack from the same source model. Comparing each column, RACL shows strong adversarial transferability as the source model. Comparing each row, RACL achieves better black-box adversarial accuracy in all the scenarios as shown in Table 3.6 with bold. *E.g.*, as shown in the fourth row, PC-DARTS $\rightarrow$ RACL achieves the successful attack success rate of 18.84%, PC-DARTS $\rightarrow$ AmoebaNet of 21.30%, PC-DARTS $\rightarrow$ NASNet of 19.32% and PC-DARTS $\rightarrow$ DARTS of 22.98%. Similarly, RACL achieves strong adversarial transferability with RFGSM attack. Thus, RACL shows superior adversarial robustness against different transferred-based attacks, which demonstrates the effectiveness of our algorithm.

3.3.4 Robustness under Various Perturbation Size and Attack Iterations

Robustness under Increasing Attack Iterations We further conduct experiments with different white-box attack parameters, including the size of perturbation and the number of iterations. Following (Guo et al. 2020), we strengthen the adversarial attack through boosting the attack iterations to 100 for PGD attack with a step size of $2/255$. The comparison with other baselines is shown in Figure 3.4 (a) where RACL consistently achieves the best accuracy

TABLE 3.7. Multiple runs of searched cells with adversarial training. The mean of clean or adversarial accuracy is reported with its error bar.

Models	Clean	FGSM	PGD^{20}	MIM	CW	AA
AmoebaNet	81.86±0.22	59.19±0.32	53.43±0.45	57.07±0.49	78.33±0.63	47.64±0.37
NasNet	**84.05±0.64**	59.49±0.53	53.37±0.70	58.16±0.55	79.57±0.74	48.34±0.46
DARTS	80.84±0.88	59.49±0.53	53.82±0.59	57.32±0.45	77.34±0.99	48.31±0.38
PC-DARTS	84.01±0.68	60.85±0.26	53.30±0.51	58.17±0.46	79.93±0.56	47.56±0.50
RACL	83.98±0.22	**62.48±0.10**	**55.50±0.38**	**60.01±0.39**	**80.11±0.39**	**50.14±0.33**

for different PGD iterations. Furthermore, RACL shows relatively stronger defence capability against PGD attacks with more iterations. For example, NasNet achieves 53.35% under PGD^{20} and 52.83% under PGD^{100} with a gap of 0.52% while RACL achieves 55.68% under PGD^{20} and 55.32% under PGD^{100} with a gap of 0.36%, which demonstrates that RACL can better remain the robustness after more attack iterations. Compared to RobNet family with the same search space on PGD^{100} (Guo et al. 2020), RACL achieved better performance with fewer parameters than RobNet-small, RobNet-medium, RobNet-large, and RobNet-free but 7.25%, 6.36%, 6.08%, and 2.75% accuracy improvement respectively, which also shows the efficiency of RACL.

Robustness under Increasing Perturbation Size Besides attack iterations, we evaluate the adversarial robustness under different perturbation budgets. As shown in Figure 3.4 (b, c), the total perturbation size ranges from 0.01 to 0.05 for both PGD and FGSM attacks. Our proposed RACL algorithm always performs better than other baselines under different perturbation budgets, which illustrates that RACL can provide a stronger defence against various adversarial attacks. Similarly, our advantage becomes more obvious when the attack was allowed with a larger total perturbation size. *E.g.*, comparing NasNet with RACL on the $PGD_{0.01}$ and $PGD_{0.05}$, the gap increases 0.71% when the attack size grew ($75.33\% \rightarrow 76.89\%$ on $PGD_{0.01}$ and $33.57\% \rightarrow 35.84\%$ on $PGD_{0.05}$); comparing AmoebaNet with RACL on the $FGSM_{0.01}$ and $FGSM_{0.05}$, the gap increases 1.65% ($75.10\% \rightarrow 77.33\%$ on $FGSM_{0.01}$ and $47.55\% \rightarrow 51.43\%$ on $FGSM_{0.05}$), which highlights the adversarial robustness of RACL within a wider perturbation space for various attacks.

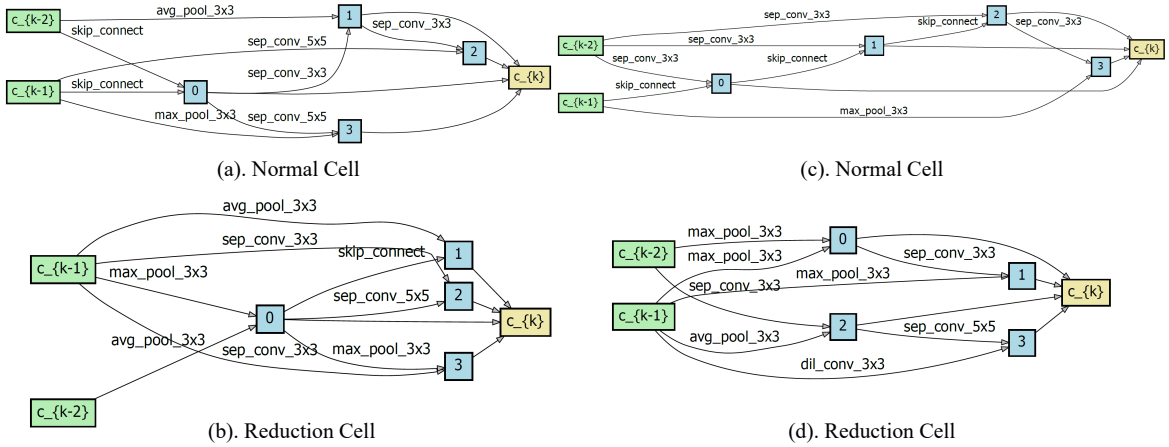


FIGURE 3.5. The visualization of cells searched by RACL through multiple runs.

TABLE 3.8. Ablation Analysis of RACL with respect to confidence learning, ρ, and η.

Setting	Clean	FGSM	PGD^{20}	MIM	CW	AA
Random Search	81.55	58.73	51.36	55.72	76.63	46.82
w/o Gradient Norm	82.54	60.33	53.87	58.04	78.94	48.25
w/o CL	**84.35**	61.64	54.58	58.68	78.72	48.89
$\eta = 0.9$, $\rho = 0.01$	83.06	61.60	55.66	59.34	79.51	48.98
$\eta = 0.7$, $\rho = 0.001$	84.30	61.58	55.00	59.14	80.36	49.03
$\eta = 0.9$, $\rho = 0.001$	84.04	**62.07**	**55.68**	**60.00**	**80.90**	**50.07**

3.3.5 Potential Pattern and Variance of RACL

Visualization of Searched Cells Besides the cells visualized in Fig. 3.3, we run RACL several times to explore the potential patterns RACL tended to discover and provided more insights into the searched robust neural architectures. More searched architectures are given in Fig. 3.5. Together with the searched cells in Fig. 3.3, we showed that there exist some potential patterns which RACL prefers. There always exists a ResNet-like pattern in the searched normal cells. For example, the input of each node tends to be a combination of skip connection and another operation with trainable parameters, such as the Node 0, 1, 2 in Fig. 3.3 (a) and Fig . 3.5 (c). Besides the ResNet-like pattern in the normal cells, RACL tends to select pooling layers such as 3×3 max pooling instead of skip connection in the reduction

cells, as shown in Figure 3.3 (b) and Figure 3.5 (b), (d). Overall, the searched cells of RACL look like a tuned version of ResNet.

Robustness Stability of Searched Cells To further demonstrate the effectiveness of RACL, we report the error bar of RACL as well as other baselines through multiple runs to evaluate the robustness stability of searched cells. Following previous NAS work (Liu et al. 2018a), we retrain all the baselines for multiple times and report the performance of neural architectures searched by different NAS algorithms and RACL (out of 5 runs). The detailed results are shown in Table 3.7. For each algorithm, we report the average clean and robust accuracy with the standard error. Comparing each column, RACL consistently achieves the best average robust accuracy under various attacks after multiple runs. For example, RACL achieves an average accuracy of 50.14% under Auto Attack and 55.50% under PGD^{20}, which is around 2% higher than other baselines. For the error bar, RACL has smaller fluctuation in almost all the scenarios among different NAS algorithms, which demonstrates that RACL can discover robust neural architectures with better robustness and stability.

3.3.6 Ablation Analysis

In this section, we conduct ablation studies on the hyperparameters of RACL algorithm as well as confidence learning. The ablation study results are shown in Table 3.8. We first apply the random search algorithm within the pre-defined search space to rule out the possibility that the major improvement comes from the search space. We randomly sampled 10 models and selected the best one for comparison. We then remove the confidence learning and apply the constraint in Eq. 3.10 to evaluate the effectiveness of confidence learning. Similarly, we remove the gradient norm constraint in Eq. 3.16 to evaluate the effectiveness of lower bound constraint. Comparing the first and other rows, the random search algorithm cannot achieve competitive results within a pre-defined search space, which demonstrates the necessity of discovering robust neural architectures. Comparing the second and last row, the searched architecture without confidence learning tends to have a relatively higher natural accuracy. On the contrary, our proposed RACL achieves a relatively large increment in adversarial accuracy with confidence learning, which highlighted the importance of proposed confident

architecture sampling. We then investigated the influence of hyperparameter ρ and reported the performance of searched robust cell on CIFAR-10 under different values of ρ. Through comparison, ρ with a large value could hurt the classification performance on clean images. On the contrary, ρ with a small value reduces the influence of Lipschitz constraint and results in inferior adversarial accuracy. The influence of confidence hyperparameter η is also investigated. From Eq. 3.16, η controls the balance between the mean and variance of Lipschitz constant $\overline{\lambda_{\mathcal{F}}}$. Through cross-validation, η is set to 0.9 to obtain the best adversarial accuracy.

3.4 Conclusion

In this chapter, we propose to tackle the vulnerability of neural networks by incorporating NAS frameworks. Through sampling architecture parameters from trainable log-normal distributions, we show that the approximated Lipschitz constant of the entire network can be formulated as a univariate log-normal distribution, which enables the proposed algorithm, Robust Architecture with Confidence Learning to form confidence learning of architecture parameters on the robustness through a Lipschitz constraint. Thorough experiments demonstrate the influence of architecture on adversarial robustness and the effectiveness of RACL under various attacks on different datasets.

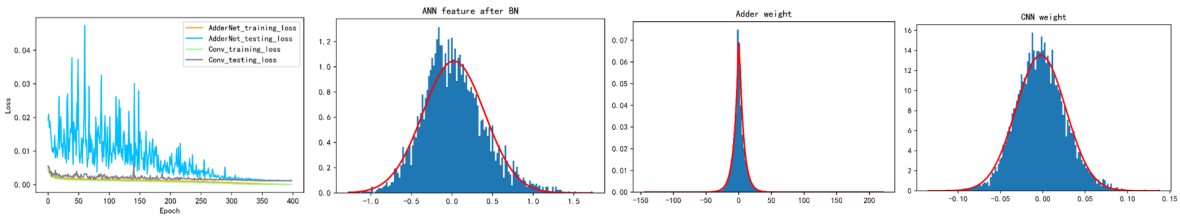
CHAPTER 4

Adder Filter Design for Adversarial Robustness

4.1 Motivation

Convolutional Neural Networks (CNNs) utilizes cross-correlation as the basic operation to measure the similarity between weight parameters and activations, which has been widely adopted in various computer vision tasks. However, this similarity measurement could limits the performance of DNNs in adversarial robustness, which motivated us to look for some robust basic operations which can naturally defend against adversarial perturbations. Recently, Chen *et al.*(Chen et al. 2020b) advocate the use of ℓ_1-distance for similarity measure instead of cross-correlation in CNNs to replace multiplications with additions. Compared with multiplications, addition operations are much cheap, which benefits the power-efficiency (Wang et al. 2020a; Sze et al. 2017; You et al. 2020). Adder neural network (ANN) has demonstrated extraordinary performance on several computer vision tasks with huge energy reduction, which can be seen as a good complement to the classical CNNs. Different from existing work, we delve into the theoretically understanding of ANNs and explore their potential in adversarial robustness.

To investigate this potential, we first study the difference caused by the replacement from cross correlation to ℓ_1-distance. The performance achieved by ANNs on classification is impressive, but some observations raise our concerns. E.g., the optimization of ANNs is not stable as CNNs where the test accuracy curve has dramatic fluctuations during the optimization, as shown in Figure 4.1 (a). The test accuracy of ANNs has a large variation until the end of training while that of convolution networks is much more stable. Furthermore, weights of ANNs have demonstrated a significantly different statistical property from those of CNN



(a). Loss Curve (b). Adder feature after BN (c). Adder weight (d). Conv weight

FIGURE 4.1. Observations of AdderNet. Training and testing loss curves of ANN and CNN in (a). Histograms over the AdderNet features after batch normalization layer follow Gaussian distribution in (b). Histograms over the AdderNet weight follow Laplace distribution with a large variance while Conv weight follows Gaussian distribution with small variance in (c) and (d) respectively.

weights. The histograms over ANN and CNN weights are shown in Figure 4.1 (c) and (d) respectively. The red curve in (c) denotes a Laplace distribution with a mean of 0.32 and a variance of 88.47 and the one in (d) denotes a normal distribution with a mean of -0.002 and a variance of 0.001. Although the means of these two distributions are similar, there exists a serious discrepancy between their variances. Given this statistical property difference, those sophisticated training strategies previously developed for CNN might not fulfill their potential when straightforwardly applied to ANNs.

In this chapter, we provide a complete workflow of adversarial robustness via adder filters. First, we provide an exhaustive analysis of the variance in ANNs. Taking a one-layer adder forward as an example, we demonstrate the large variance of ANN weights can be the major cause of the instability of running mean and variance in batch normalization layer which vibrates the test accuracy. Weight normalization is therefore a natural idea to constrain the variance of weights. To preserve the representation capability of the normalized weights, trainable scaling and shift coefficients are further introduced to achieve an Adaptive Weight Normalization (AWN). By doing so, the variance of ANN weights can be normalized to acceptable levels which significantly stabilizes the batch normalization layers and makes pretrained ANN easily applied to other tasks. Notably, we identify a natural advantage of ANNs to be robust to adversarial perturbations, stemming from ℓ_1-distance and the running mean of batch normalization layer in ANNs. The reduction of weight variation across channels further boosts the defense ability against attacks. Without adversarial training, AWN with

ANN on ResNet-32 improves adversarial accuracy by 59.73% compared with CNN under PGD attacks. AWN presents superior stability under a series of evaluations and outperforms vanilla ANN in detection task on VOC benchmark with 2.7 mAP improvement.

4.2 Preliminary

Adder Neural Networks. To significantly reduce energy costs, Chen *et al.*(Chen et al. 2020b) proposed an Adder Neural Network to release the burden of multiplications in traditional convolution networks by replacing them with additions. Consider an intermediate layer in deep convolution neural network with weight $W \in \mathbb{R}^{d\times d\times c_{in}\times c_{out}}$ where d denotes the kernel size and c_{in} and c_{out} are the number of input and output channel respectively. Given the input feature map $X \in \mathbb{R}^{H\times W\times c_{in}}$ where H and W are the height and width of input feature, the adder operation is defined as

$$Y(m,n,c) = -\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{c_{in}} |X(m+i,n+j,k) - W(i,j,k,c)|. \tag{4.1}$$

Although the adder operation enables ANNs to achieve similar performance in classification tasks with energy efficiency, the observations in Figure 4.1 shadow ANNs. The unstable test accuracy curve and large variance almost everywhere in ANNs arouse our interest in the analysis of ANN variance.

Adversarial Attack. Given the input $x \in \mathbb{R}^D$ and the annotated label $y \in \mathbb{R}^C$ where D is the dimension of input and C is the number of classes, the network $\mathcal{N}$ maps perturbed input $\tilde{x}$ to $\tilde{y} = \mathcal{N}(\tilde{x}; W)$ where $\tilde{x} = x + \delta$. The objective of adversarial attacks is to find the perturbed input which maximizes the classification loss as

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p\leqslant\epsilon}{\operatorname{argmax}} \ell(\mathcal{N}(\tilde{x}, W), y), \tag{4.2}$$

where the perturbation is constrained by its l_p-norm. Through variance study of ANNs and ℓ_1-distance analysis, we demonstrate the potential adversarial robustness of ANNs can be activated through proposed inference strategy with AWN.

4.3 Variance Study of AdderNet

To analyze the variance of ANNs, we first consider a one-layer adder forward with batch normalization (Ioffe and Szegedy 2015) and ReLU as activation function. Given output feature y_{l-1} from previous layer, the forwarding of layer l is formulated as

$$x_l = max(0, y_{l-1}),\ y'_l = -\sum^{n} |x_l - W_l|,\ y_l = \gamma \frac{y'_l - \mu_l}{\sigma_l} + \beta, \tag{4.3}$$

where μ_l and σ_l denote the mean and standard deviation, respectively, W_l is the weight, n represents the size of weight of one output channel $n = d \times d \times c_{in}$, and γ and β stand for the rescale and shift parameters.

Before we proceed to the weight variance analysis, we first make several assumptions that are empirically plausible. After the batch normalization, y_{l-1} is supposed to follow a normal distribution $\mathcal{N}(0, \sigma^2)$. From empirically observation of a ResNet-18 with ANN on ImageNet, as shown in Figure 4.1 (b) and (c), adder feature after BN layer follows a distribution with mean of 0.02 and variance of 0.38 while the corresponding weight with mean of 0.32 and variance of 88.47, the adder weights follow Laplace distributions with a large variance but their mean is close to 0 after training. Thus, we assume W_l follows $\mathcal{L}(\mu, b)$ with the mean as μ and the variance as $2b^2$ where $2b^2 \gg \sigma^2$.

Now we compute the mean and variance of each variable. The activation x_l follows the Rectified Gaussian distribution. With the law of total expectation, the mean of x_l forms the one-side truncation of lower tail $E[x_l | x_l > 0]$ which can be computed based on the property of Truncated normal distribution as

$$\begin{aligned} E[x_l] &= E[x_l | x_l > 0] \cdot P(x_l > 0) + E[x_l | x_l \leq 0] \cdot P(x_l \leq 0) \\ &= E[x_l | x_l > 0] \cdot P(x_l > 0) + 0 = [0 + \sigma \frac{\phi(0)}{1 - \Phi(0)}] \cdot \frac{1}{2} = \frac{\sigma}{\sqrt{2\pi}}, \end{aligned} \tag{4.4}$$

where $\phi(\cdot)$ denotes the probability of standard normal distribution and $\Phi(\cdot)$ denotes the cumulative distribution function. Similarly, with the law of total expectation, the variance of x_l is broken down into variants of expectation and the variance of one-side truncation of

lower tail $Var(x_l|x_l > 0)$, which can be computed through Truncated normal distribution as

$$\begin{aligned} Var[x_l] &= E[x_l^2] - (E[x_l])^2 = -(E[x_l])^2 + E[x_l^2|x_l > 0] \cdot P(x_l > 0) \\ &= P(x_l > 0)[Var(x_l|x_l > 0) + (E[x_l|x_l > 0])^2] - (E[x_l])^2 \\ &= [\sigma^2(1 - (\frac{\phi(0)}{1 - \Phi(0)})^2) + \frac{4\sigma^2}{2\pi}] \cdot \frac{1}{2} - \frac{\sigma^2}{2\pi} = \sigma^2(\frac{1}{2} - \frac{1}{2\pi}). \end{aligned} \tag{4.5}$$

Since we assume $2b^2 \gg \sigma^2$ and the variance is further reduced after activation, the distribution of $x_l - W_l$ is overwhelmed by W_l to form a Laplace distribution as $\mathcal{L}(\frac{\sigma}{\sqrt{2\pi}} - \mu, \sqrt{b^2 + \frac{(\pi-1)\sigma^2}{4\pi}})$. For simplicity, let $\tau_\mu = \frac{\sigma}{\sqrt{2\pi}} - \mu$ and $b + \tau_\sigma = \sqrt{b^2 + \frac{(\pi-1)\sigma^2}{4\pi}}$. Note that the standard deviation of output last layer σ and the mean of weight μ are both close to zero, which makes τ_μ and τ_σ small values. Although it is difficult to directly derive the closed-form distribution expression of y_l', it can be approximated based on variable $x_l - W_l - \tau_\mu$. According to the properties of Laplace distribution, the absolute function of $\mathcal{L}(0, b)$ follows an Exponential distribution, with which y_l' can be approximated as

$$\begin{aligned} &y_l' = -\sum^n |x_l - W_l| \geq -\sum^n [|x_l - W_l - \tau_\mu| + |\tau_\mu|], \\ &\text{where } |x_l - W_l - \tau_\mu| \sim \text{Exp}([b + \tau_\sigma]^{-1}), \end{aligned} \tag{4.6}$$

where Exp denotes Exponential distribution. Based on the property of Exponential distribution, $E[|x_l - W_l|] = b + \tau_\sigma$ and $Var[|x_l - W_l|] = (b + \tau_\sigma)^2$. With Eq. 4.6, the lower boundary of $E[y_l']$ can be derived as

$$E|x_l - W_l| \leq E|x_l - W_l - \tau_\mu| + |\tau_\mu|, \quad E[y_l'] \geq -n(b + \tau_\sigma + |\tau_\mu|). \tag{4.7}$$

With inequality in Eq. 4.7, the lower boundary of $Var[y_l']$ can be derived by the law of the unconscious statistician as

$$\begin{aligned} Var[y_l'] &= \sum^n [E[|x_l - W_l|^2] - (E[|x_l - W_l|])^2] \\ &= \sum^n [Var(x_l - W_l) + (E(x_l - W_l))^2 - (E(|x_l - W_l|))^2] \\ &\geq n[Var(x_l - W_l) + (E(x_l - W_l))^2 - (E|x_l - W_l - \tau_\mu| + |\tau_\mu|)^2] \\ &= n[2(b + \tau_\sigma)^2 + \tau_\mu^2 - (b + \tau_\sigma + |\tau_\mu|)^2] = n[(b + \tau_\sigma)^2 - 2|\tau_\mu|(b + \tau_\sigma)]. \end{aligned} \tag{4.8}$$

Taking the lower boundary of mean and variance of y_l' in Eq. 4.7, 4.8, the batch normalization layer can be computed as

$$y_l = \gamma \frac{y_l' + n(b + \tau_\sigma + |\tau_\mu|)}{\sqrt{n[(b+\tau_\sigma)^2 - 2|\tau_\mu|(b+\tau_\sigma)]}} + \beta. \tag{4.9}$$

Note that the batch normalization layers use running mean and variance of current batch in training phase while moving averages in testing phase. Although precise $E[y_l']$ and $Var[y_l']$ can be computed in training phase, the moving averages vary dramatically since b in Eq. 4.9 grows from a small value to a large one during optimization, as shown in Figure 4.1 (c) where the distribution of W_l with initial variance of 1.0 is optimized to the one with variance of 88.47. Thus, the large variation of batch statistics results in the instability of testing loss curve while the training loss curve is similar to the one of CNNs, as shown in Figure 4.1 (a).

4.3.1 Adaptive Weight Normalization for AdderNet

Based on these observations and analysis, we propose to make some efficient modifications to current ANN optimization. From Eq. 4.9, the running mean and variance mainly depend on the standard deviation of adder weights so that large magnitude of weights could destabilize the statistics in batch normalization, which indicates that adder weights need normalization to prevent them from being updated to a distribution with large variance. A naive approach is Weight Standardization proposed by Qiao *et al.*(Qiao et al. 2019) as

$$\begin{aligned} W'_{i,j} &= \frac{W_{i,j} - \mu_{W_i}}{\sigma_{W_i}}, \text{ where } \mu_{W_i} = \frac{1}{n}\sum_{j=1}^{n} W_{i,j}, \\ \sigma_{W_i} &= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(W_{i,j} - \mu_{W_i})^2 + \epsilon}, \end{aligned} \tag{4.10}$$

where $W \in \mathbb{R}^{c_{out} \times n}$ denotes the permuted adder weight, $||$ denotes concatenation operation and ϵ is added for numerical stability. In Eq. 4.10, each output channel of adder weight is normalized to a distribution with a mean of 0 and variance of 1, which stabilizes the running mean and variance of batch normalization layer according to Eq. 4.9. However, with weight standardization directly applied to ANNs, there exists a dramatic accuracy drop. For example,

ANN with weight standardization achieves 90.62% in ResNet-20 on CIFAR-10 while original ANN achieves 91.84%, which causes 1.22% accuracy drop. Although adder weights are normalized to guarantee a stable test phase, rigorous mean and variance are strictly assigned to adder weights, which constrains the representation power of ANNs. Since the similarity between filter and input feature is measured by ℓ_1-distance in ANNs, the magnitude of weight values can be rather sensitive for network expression capability. Thus, directly applying weight standardization to ANNs can be easily stunk in the local optimum without exploring wider space of adder weights. Instead, we propose to normalize adder weights with trainable variables for each output channel, which preserves the representation power. Eq. 4.10 can be rewritten as

$$W'_{i,j} = \nu_i \frac{W_{i,j} - \mu_{W_i}}{\sigma_{W_i}} + \upsilon_i, \tag{4.11}$$

where ν_i and υ_i are trainable variables similar to β and γ in batch normalization layer. Thus, the magnitude of weight values can be automatically adjusted to fit the potential levels of freedom of adder weights. Furthermore, previous analysis demonstrates that the magnitude of gradient w.r.t the input X and the filter W in ANNs is much smaller than that in CNNs (Chen et al. 2020b). With the incorporation of these two parameters, the gradient w.r.t the filter W can be automatically adjusted. The gradient of loss ℓ w.r.t the weight $W_{i,j}$ is computed as

$$\frac{\partial \ell}{\partial W_{i,j}} = \sum_{c=1}^{n} \frac{\nu_i}{n^2 \sigma_{W_i}} \Big\{ \frac{\partial \ell}{\partial W'_{i,j}} - \frac{\partial \ell}{\partial W'_{i,c}} [1 + \frac{(W_{i,j} - W_{i,c})(W_{i,c} - \mu_{W_i})}{\sigma_{W_i}}] \Big\}. \tag{4.12}$$

In Eq. 4.12, the gradient of W is amplified by ν_i, which relieves the gradient reduction in ANNs.

4.4 Activate Potential Adversarial Robustness

Consider the intermediate layer forwarding of ANNs and CNNs with perturbation δ_l, the disturbance of an element on the output feature map before BN layer can be computed as

$$\begin{aligned}\tilde{y'}_{lCNN} - y'_{lCNN} &= \sum_{i=1}^{n}[(x_{l,i} - \delta_{l,i}) \times W_{l,i} - x_{l,i} \times W_{l,i}] = \sum_{i=1}^{n}(-\delta_{l,i}) \times W_{l,i}, \\ \tilde{y'}_{lANN} - y'_{lANN} &= \sum_{i=1}^{n}[|x_{l,i} - W_{l,i}| - |x_{l,i} - \delta_{l,i} - W_{l,i}|] \approx \sum_{i=1}^{n} \pm|\delta_{l,i}|,\end{aligned} \tag{4.13}$$

where $\pm$ denotes the choices of two possible signs including addition and subtraction. Note that $|x_{l,i} - W_{l,i}|$ follows a distribution with large mean and variance in ANNs. We assume that δ_l follows a Gaussian distribution with zero mean and smaller variance $\mathcal{N}(0, \sigma_\delta^2)$ where $\sigma_\delta^2 < Var[W_l]$, with which the subtraction $|x_{l,i} - W_{l,i}| - |x_{l,i} - \delta_{l,i} - W_{l,i}|$ has a high probability to be either δ_l or $-\delta_l$. We show that it is difficult for $\pm|\delta_{l,i}|$ to have large variance in ANNs under adversarial attacks. In Eq. 4.13, if $Var[\pm|\delta_{l,i}|]$ grows in ANNs, all the elements in $\pm|\delta_{l,i}|$ tend to select different signs, which automatically eliminate each other to make $\tilde{y'}_{lANN} - y'_{lANN} \approx 0$. Thus, if attacks succeed, $Var[\pm|\delta_{l,i}|]$ needs reduction to guarantee larger perturbation on feature map. To ensure maximum disturbance, all the elements in $\pm|\delta_{l,i}|$ have the same signs, which forms a sum of n half-normal distributions. On the contrary, each $\delta_{l,i}$ in CNNs is transformed by different $W_{l,i}$ in Eq. 4.13. The variance of disturbance before BN layer can be computed as

$$\begin{aligned}Var[\tilde{y'_l} - y'_l]_{CNN} &= Var[\sum_{i=1}^{n}(-\delta_{l,i}) \times W_{l,i}] = n\sigma_\delta^2(Var[W_l] + (E[W_l])^2), \\ Var[\tilde{y'_l} - y'_l]_{ANN} &\approx Var[\sum_{i=1}^{n}|\delta_{l,i}|] = n\sigma_\delta^2(1 - \frac{2}{\pi}).\end{aligned} \tag{4.14}$$

Eq. 4.14 indicates the output disturbance variation in CNNs depends on both the statistics of W and δ, and could vary for different output channels while the one in ANNs only depends on the statistics of δ for all the output channels. The major difference lies in BN layer where both ANN and CNN disturbances are re-scaled and the variance of perturbation on next layer can be computed as

$$
\begin{aligned}
Var[\delta_{l+1}]_{CNN} &= n\sigma_\delta^2(Var[W_l] + (E[W_l])^2)/[Std(\tilde{y'}_{lCNN})]^2, \\
Var[\delta_{l+1}]_{ANN} &\approx n\sigma_\delta^2(1 - \frac{2}{\pi})/[Std(\tilde{y'}_{lANN})]^2.
\end{aligned}
\tag{4.15}
$$

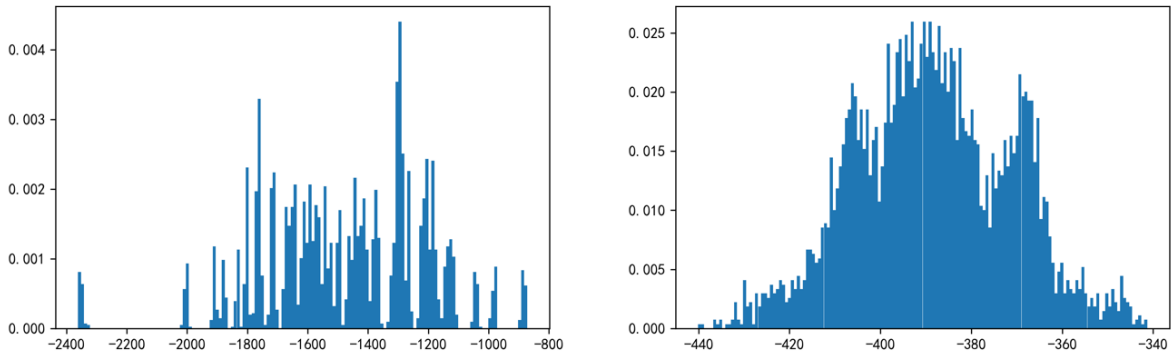
Note that the variance of $\tilde{y'}_{lANN}$ is much larger than $\tilde{y'}_{lCNN}$. Thus, ANNs have a much smaller disturbance variance on $l+1$ layer, which suggests that the perturbations of all the elements on $\tilde{y'_l}$ are similar and can be eliminated through subtraction of a scalar value while CNNs cannot copy that success. To activate the adversarial robustness through utilizing this property of ANNs, a simple yet effective method is utilizing the running mean in batch normalization layer for automatic disturbance elimination. In Eq. 4.9, τ_μ becomes $\tau_\mu - E[\delta]$ while $b + \tau_\sigma$ nearly remains the same under attack settings since the disturbance has small variance. The forwarding of BN layer can be computed as

$$
\begin{aligned}
\tilde{y} &= \gamma \frac{\tilde{y'} + n(b + \tau_\sigma + |\tau_\mu - E[\delta]|)}{\sqrt{n[(b+\tau_\sigma)^2 - 2|\tau_\mu - E[\delta]|(b+\tau_\sigma)]}} + \beta \\
&\approx \gamma \frac{y' + \sum_i^n \pm|\delta_i| + n(b + \tau_\sigma + |\tau_\mu|) - n|E[\delta]|}{\sqrt{n[(b+\tau_\sigma)^2 - 2|\tau_\mu - E[\delta]|(b+\tau_\sigma)]}} + \beta \\
&\approx \gamma \frac{y' + n(b + \tau_\sigma + |\tau_\mu|)}{\sqrt{n[(b+\tau_\sigma)^2 - 2|\tau_\mu - E[\delta]|(b+\tau_\sigma)]}} + \beta.
\end{aligned}
\tag{4.16}
$$

The difference between Eq. 4.9 and Eq. 4.16 lies in the running variance, which demonstrates that the disturbance can be significantly relieved through computing the running mean in BN layer while the variance is expected to be constant to eliminate the difference brought by $E[\delta]$. Thus, we propose ANN robust inference strategy as

$$
y^* = \gamma \frac{\tilde{y} - \text{running mean}}{\text{tracked variance}} + \beta. \tag{4.17}
$$

However, in some cases where $|x_{l,i} - W_{l,i}|$ and $|x_{l,i} - \delta_{l,i} - W_{l,i}|$ have different signs, the perturbations in feature space can be amplified by W in ANNs. It indicates the trade-offs between the magnitude of W and variance of ANN features, where our proposed ANN-AWN can successfully achieves better trade-offs. In addition, considering the actual case where the signs of $\pm|\delta_{l,i}|$ are not the same everywhere, the proposed ANN robust inference strategy cannot work appropriately and results in residual perturbations which will be re-scaled by tracked variance after BN layer. In Eq. 4.13, the perturbations of different output channels



(a). Histograms over ANN feature before BN

(b). Histograms over ANN-AWN feature before BN

FIGURE 4.2. Histograms over the ANN and ANN-AWM features of an intermediate layer are shown in (a) and (b).

TABLE 4.1. Adversarial robustness on CIFAR-10 under white-box attacks without adversarial training. -R denotes robust inference strategy which uses the running mean in batch normalization layer instead of tracked ones. BIM^7 denotes iterative attack with 7 steps. The best results in bold and the second best with underline.

Model	Method	#Mul.	#Add.	Clean	FGSM	BIM^7	PGD^7	MIM^5	$RFGSM^5$
ResNet-20	CNN	41.17M	41.17M	**92.68**	16.33	0.00	0.00	0.01	0.00
	ANN	0.45M	81.89M	91.72	18.42	0.00	0.00	0.04	0.00
	CNN-R	41.17M	41.17M	90.62	17.23	3.46	3.67	4.23	0.06
	ANN-R	0.45M	81.89M	90.95	29.93	29.30	29.72	32.25	3.38
	ANN-R-AWN	0.45M	81.89M	90.55	**45.93**	**42.62**	**43.39**	**46.52**	**18.36**
ResNet-32	CNN	69.12M	69.12M	**92.78**	23.55	0.00	0.01	0.10	0.00
	ANN	0.45M	137.79M	92.48	35.85	0.03	0.11	1.04	0.02
	CNN-R	69.12M	69.12M	91.32	20.41	5.15	5.27	6.09	0.07
	ANN-R	0.45M	137.79M	91.68	19.74	15.96	16.08	17.48	0.07
	ANN-R-AWN	0.45M	137.79M	91.25	**61.30**	**59.41**	**59.74**	**61.54**	**39.79**

are similar in ANNs, which indicates that the noise of feature map can hardly cross channels. However, this property will be broken if tracked variances have enormous differences among channels. Thus, our proposed ANN-AWN can successfully relieve the variation across channels to further improve adversarial robustness. As shown in Figure 4.2, each peak in ANN feature histogram denotes the features of one channel, which illustrates that different channels of ANN feature map have tremendous variations while ANN-AWN has a much smooth histogram.

4.5 Experiments

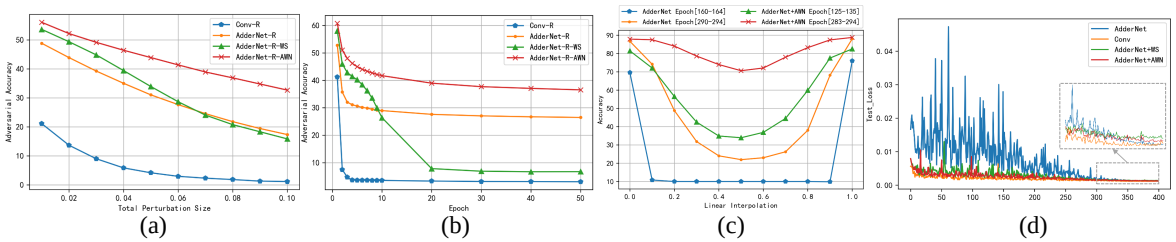
In this section, we empirically evaluate the superiority of the proposed approach on different tasks and datasets, including the adversarial robustness, object detection and optimization stability.

4.5.1 Adversarial Robustness Evaluation

White-box Attacks Setup. To demonstrate how the proposed adaptive weight normalization activates the potential adversarial robustness of AdderNet, we conduct a series of experiments. Following (Chen et al. 2020b), we first train both CNNs and ANNs with ResNet20 and ResNet32 on CIFAR-10 under the same settings. CIFAR-10 dataset contains $50K$ training images and $10K$ validation images with size of 32×32 over 10 classes. We use SGD optimizer with an initial learning rate of 0.1, momentum of 0.9 and a weight decay of 5×10^{-4}. The model is trained on single V100, which takes 400 epochs with a batch size of 256 and a cosine learning rate schedule. The learning rate of trainable parameter ν and υ in AWN is rescaled by a hyper-parameter which we set to be 1×10^{-5}. For adversarial robustness evaluation, we conduct white-box attacks on these models including Fast Gradient Sign Method (FGSM) (Szegedy et al. 2014), Basic Iterative Method (BIM) (Kurakin et al. 2018), Projected Gradient Descent (PGD) (Madry et al. 2018), Momentum Iterative Method (MIM) (Dong et al. 2018), and RFGSM (Tramèr et al. 2017a) to generate adversarial examples. Following previous adversarial literature (Madry et al. 2018; Zhang et al. 2019a), the adversarial perturbation is considered under l_∞ norm with the total perturbation size of $8/255$. In iterative attack settings, the step size is set to $2/255$. The number of iterations is set to 7 for PGD and BIM attacks, and 5 for MIM and RFGSM attacks. Note that different from traditional defense algorithms which generate adversarial samples for adversarial training (Shafahi et al. 2019) or search robust architectures (Dong et al. 2020; Li et al. 2021a), our proposed algorithm utilizes the properties of ANNs to achieve adversarial robustness without any training tricks or modifications of architecture. The experimental results are shown in Table 4.1. Note that all the models are trained with clean images, which dismisses the expensive adversarial training.

Against White-box Attacks. As shown in the first two rows in Table 4.1, these adversarial attacks successfully mislead both CNN and ANN to provide wrong predictions. Although ANN has slightly better adversarial accuracy compared to CNN, such as $0.01\% \rightarrow 0.04\%$ under MIM^5 attack, the potential adversarial robustness shown in Eq. 4.13 cannot be activated with normal settings. Thus, we replace the tracked mean in batch normalization layer with the running mean of current batch based on Eq. 4.17 and compare ANN with CNN in the third and fourth rows, which are denoted as -R. Comparing CNN-R and ANN-R, there exists an enormous adversarial robustness improvement. For example, ANN-R achieves $\mathbf{26.05}\%$ accuracy increment from $3.67\% \rightarrow 29.72\%$ under PGD^7 compared with CNN-R, and improves the accuracy by $\mathbf{32.21}\%$ from $0.04\% \rightarrow 32.25\%$ under MIM^5 compared with ANN. The comparison between ANN and ANN-R empirically demonstrates that appropriate utilization of the running mean in batch normalization layer of ANNs can significantly activate the adversarial robustness of ANNs. Furthermore, the comparison between CNN-R and ANN-R shows strong evidence of the natural robustness difference between CNNs and ANNs and indicates that the ℓ_1 distance and the independence between perturbation and adder weight provides much better defense than CNNs, which is consistent with the aforementioned variance analysis. The evaluation of proposed AWN is shown in the fifth row. For all the attacks, ANN-R-AWN achieves the best results, which demonstrates the effectiveness of proposed AWN. Comparing ANN-R and ANN-R-AWN, AWN shows obvious superior adversarial robustness. For example, AWN improves adversarial accuracy by $\mathbf{13.67}\%$ from $29.72\% \rightarrow 43.39\%$ under PGD^7 and $\mathbf{14.98}\%$ from $3.38\% \rightarrow 18.36\%$ under RFGSM^5, which illustrates that adversarial robustness can be further boosted through narrowing the difference among channels to relieve the perturbation transformation. On ResNet-32, ANN-R achieves worse performance than the one with ResNet-18, which demonstrates that proposed robust inference strategy is not sufficient for superior adversarial robustness. However, ANN-R-AWN consistently achieves better performance, which outperforms other baselines in all the adversarial scenarios, which indicates that the robustness of proposed AWN can generalize to deeper models.

Against Gradually Enhanced Attacks. We highlight the superiority of proposed ANN-R-AWN through enhancing the attacks from different aspects to evaluate the adversarial



(a) (b) (c) (d)

FIGURE 4.3. The evaluation of adversarial robustness under different PGD attack size is shown in (a) and different PGD attack steps in (b). The performance of intermediate weights sampled from ANN and ANN-AWN through linear interpolation in (c). Test loss curves of CNN, ANN, ANN-WS and ANN-AWN are shown in (d).

robustness under more powerful attacks. We use the naturally trained models with ResNet-20 on CIFAR-10 for evaluation. In the first scenario, the total perturbation size ϵ of PGD attack increases from 0.01 to 0.1 with a step size of $\epsilon/7$. In the second scenario, the iterations of PGD attacks are enhanced from 1 to 50. The evaluation results are shown in Figure 4.3 (a) and (b). ANN-R-AWN obtains much better defense ability than ANN-R, CNN-R and ANN-R-WS baselines when the attack size grows. For example, the adversarial accuracy of CNN-R quickly dive to 2.96%, ANN-R to 27.67% and ANN-R-WS to 28.66 when the PGD attack size reaches 0.06 while ANN-R-AWN maintains 32.67% when the PGD attack size reaches 0.1. Similarly, ANN-R-AWN also achieves superior robustness over other baselines with increasing steps of PGD attack. For example, the adversarial accuracy of ANN-R under PGD^{50} becomes 26.48% while ANN-R-AWN maintains 36.52% with $\mathbf{10.04}\%$ increment, which demonstrates the effectiveness of AWN. The enormous accuracy drop of ANN-R-WS along with attack epoch and coherent robustness of our proposed AWN illustrate the necessity of introduced adaptive rescale and shift parameters in weight normalization. Thus, the advantage of AWN becomes more obvious under more powerful attacks, which highlights the superiority of proposed AWN.

4.5.2 Robustness Comparison with Adversarial-trained CNN

We further provide the comparison with other advanced defense techniques on CNN, as shown in Table 4.2. Adversarial training is one of the most effective approaches for defending

TABLE 4.2. Robustness Comparison with CNN defense techniques. AT denotes the usage of adversarial training.

Method	AT	Clean	FGSM	BIM^7	PGD^7	MIM^{40}	CW^{30}
PGD-AT	✓	87.14	55.63	48.29	49.79	45.16	46.97
ALP	✓	89.79	60.29	50.62	51.89	45.97	47.69
TLA	✓	86.21	58.88	52.60	53.87	50.09	**50.69**
ANN-AWN-R	✗	**91.25**	**61.30**	**59.41**	**59.74**	**66.43**	50.60

adversarial examples and different variants have been proposed, such as PGD-AT (Madry et al. 2018), ALP (Kurakin et al. 2018) and TLA (Mao et al. 2019). We evaluate these algorithms under various attacks with the same settings, such as FGSM, BIM, PGD, MIM as well as CW. Our proposed ANN-AWN-R achieves the best performance since the perturbation brought by adversarial examples can be automatically eliminated by BN layers and attacking space across channels is reduced by AWN. Note that our algorithm only needs a natural training without feeding any adversarial examples, however, the adversarial robustness of ANN-AWN-R still outperforms CNN with defense techniques, which demonstrates the effectiveness of proposed algorithm on ANNs.

4.5.3 Stability of AdderNets

Stability Evaluation through Linear Interpolation. To conduct a quantitative evaluation of AdderNet stability, we propose to evaluate the smoothness of optimization landscape through tracking the performance of AdderNet parameters which are sampled between different training epochs. To highlight the stability of our proposed AWN, we select adjacent epochs for ANN and ANN-AWN trained with ResNet-20 on CIFAR-10 and sample 9 intermediate weights between two epochs through linear interpolation. The comparison is shown in Figure 4.3 (c). To evaluate the stability of early training stage, we select epoch $160, 164$ for ANN as the blue curve and select epoch $125, 135$ for ANN-AWN as the green curve. Although we set earlier epoch with larger intervals for ANN-AWN, our proposed algorithm forms a much more smooth curve with a smaller variance 336.00 compared to the one of ANN 587.28. In the later training stage, we select epoch $290, 294$ for ANN as the orange curve and select epoch $283 - 294$ for ANN-AWN as the red curve. Comparing with ANN,

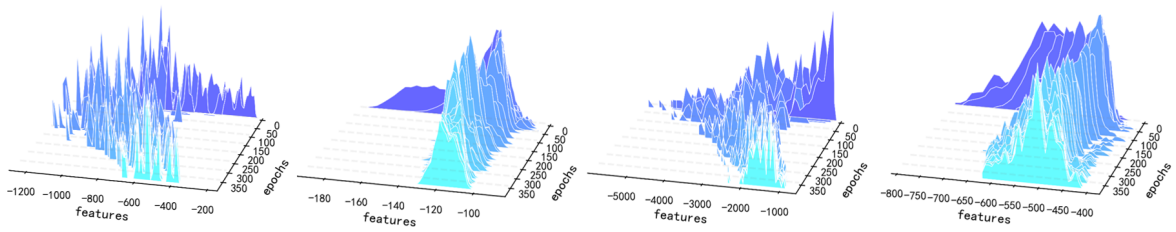


FIGURE 4.4. Distributions of features from different layers of ANN and ANN-AWN on different epochs.

our proposed algorithm achieves variance reduction of 585.97 from 627.54 to 41.67. The performance of parameters sampled from linear interpolation reflects the stability of models and the smoothness of the optimization landscape. With the proposed AWN, the optimization of AdderNet can be significantly stabilized, which could benefit the situations where adder layers are difficult to optimize.

Trade-offs Between Stability and Accuracy. To further illustrate how AWN takes effect, we keep track of the test loss during AdderNet optimization and the results of both CNN and ANN with ResNet-20 on CIFAR-10 are shown in Figure 4.3 (d). The test loss curve of AdderNet is quite unstable before 300 epochs while CNNs, ANN with weight standardization and ANN with adaptive weight normalization all achieve relatively smoothed loss curve, which indicates that the instability of AdderNet mainly comes from the large variance of adder weights and the proposed AWN eliminate it successfully. The dotted area covers from 300 to 400 epochs. Although AdderNet cannot achieve stability due to the large variance of weights, ANNs can still achieve similar performance as CNNs since the variation of adder weights is reduced during the end of training. However, WS fails to achieve a similar classification performance and is stuck at local optimum, which demonstrates that the normalization on adder weights could hurt the expressive power. On the contrary, our proposed AWN achieves relatively better performance through incorporating the adaptive trainable parameters for adder weights, which enables them to shift and re-scale back to restore the original performance.

Feature Distribution Analysis. We visualize and track the feature distributions of AdderNet with or without AWN during the training, as shown in Figure 4.4. We randomly sample features before batch normalization layer at different epochs and compute histograms over

TABLE 4.3. Comparison of proposed approach on ANNs with other settings on PASCAL VOC 2012 benchmark. The $[\cdot]$ in backbone denotes the classification accuracy on ImageNet.

Model	Backbone	Neck	mAP
CNN-FPN	Res18-CNN	CNN	69.3
ANN-FPN	Res18-ANN [67.0]	ANN	68.6
ANN-WS-FPN	Res18-ANN-WS [64.1]	ANN-WS	67.0
ANN-AWN-FPN	Res18-ANN-AWN [67.1]	ANN-AWN	69.4

them. Both ANN and ANN-AWN feature distributions in the 2nd and 19th layers are shown in Figure 4.4 (a),(b),(c) and (d) respectively. ANN feature distributions vary dramatically during the optimization, which enormously disturbs the tracking of mean and variance in batch normalization layer. For example, the sampled feature on 21-th epoch in (c) has mean of -2738.58 and standard deviation of 1042.44 while the one on 386-th epoch has mean of -1476.52 and standard deviation of 266.97 with 1262.06 increment on mean and 775.47 reduction on standard deviation. However, those in ANN-AWN become much milder, which stabilizes ANNs. For example, the sampled feature on 21-th epoch in (d) has mean of -940.89 and standard deviation of 269.39 and the one on 385-th epoch has mean of -564.59 and standard deviation of 60.65 with 376.3 increment on mean and 208.74 reduction on standard deviation. Furthermore, the difference among channels are effectively reduced by AWN to constrain the perturbation space of adversarial examples, E.g., the standard deviation of ANN features on the second layer after training becomes 87.41 while that of ANN-AWN becomes 31.13 with a massive reduction, which potentially provides better defense ability against more powerful adversarial attacks.

4.5.4 Experiments on Object Detection

To further illustrate the advantage of imposing stability on AdderNets, we conduct experiments on object detection with ANNs on PASCAL VOC (VOC) dataset. VOC contains 20 object classes, the training set includes 10K images which are the union of VOC 2007 and VOC 2012, and the VOC 2007 test set with 4.9K images is used for evaluation. The mAP scores using Iou at 0.5 are reported. All the models are trained with the same setting. Based on the

variance study of ANNs, we unfreeze BatchNorm during the training. Following (Chen et al. 2021b), we insert more shortcuts in the neck part. We use an initial learning rate of 0.008 with a linear warmup for 500 iterations, momentum of 0.9, weight decay of 1×10^{-4} and a cosine learning rate strategy. All the models are trained on 4 V100 GPUs with SGD optimizer for 12 epochs with a batch size of 4. For the detector baseline, we include both CNN and vanilla ANN for comparison. ANN-FPN replaces the convolution layers with adder layers in the pretrained ResNet-18 backbone and neck of Faster R-CNN (Ren et al. 2015). Through applying different types of ANNs to detection, we conduct comparison among the CNN, vanilla ANN (Chen et al. 2020b), ANN-WS and ANN-AWN. The detailed evaluation is shown in Table 4.3. In Backbone column, the number in brackets denotes the classification accuracy pretrained on ImageNet. Comparing ANN with our proposed AWN, although they achieve similar classification performance, AWN improves the mAP score by $\mathbf{0.8}$ from $68.6 \rightarrow 69.4$. Even comparing with CNN-FPN which has superior classification performance, our proposed AWN still outperforms it, which demonstrates the necessity of stability. With a much more smooth loss landscape, the optimization of AdderNet on other tasks can be easier and more stable. However, ANN-WS is not competitive with other baselines, with $\mathbf{2.4}$ mAP reduction compared with AWN, which empirically verifies that directly normalizing adder weights could limit the ability of feature extraction and performance for other tasks. Note that there exists an enormous accuracy drop of pretrained ANN-WS, which significantly constrains its detection performance. Thus, besides stability, the representation power of ANNs can be rather important in terms of applying ANNs to other tasks. On the contrary, our proposed AWN achieves better trade-offs between classification performance and stability through an adaptive scheme, which together achieves the superior mAP score in detection task.

4.5.5 Ablation Studies

We conduct ablation studies on proposed ANN-R-AWN to illustrate the effectiveness of adaptive training parameters and proposed robust inference strategy. We have already shown that weight standardization can easily be stuck at a local optimum in Figure 4.3 (d). Although the analysis in Sec 4.4 works for both WS and AWN, we empirically verifies that the gaps

TABLE 4.4. Adversarial robustness comparison of WS and AWN under different inference strategy with ResNet-20 on CIFAR-10 with natural training. -R denotes using running mean of current batch in BN layer. -r denotes using both running mean and variance.

Model	Clean	FGSM	BIM^7	PGD^7	MIM^5	RFGSM^5
ANN-r-WS	88.10	49.24	21.46	22.96	32.26	7.31
ANN-r-AWN	89.81	**49.85**	22.54	24.27	32.54	8.26
ANN-R-WS	89.38	40.65	31.35	36.08	42.3	7.78
ANN-R-AWN	**90.55**	45.93	**42.62**	**43.39**	**46.52**	**18.36**

TABLE 4.5. Adversarial robustness evaluation of AWN on CNN and ANN with ResNet-32 on CIFAR-10.

Model	Clean	FGSM	BIM^7	PGD^7	MIM^5	RFGSM^5
CNN-R	91.32	20.41	5.15	5.27	6.09	0.07
ANN-R	91.68	19.74	15.98	16.08	17.48	0.07
CNN-R-AWN	92.33	21.71	5.74	5.94	7.16	0.05
ANN-R-AWN	91.25	**61.30**	**59.41**	**59.74**	**61.54**	**39.79**

of clean accuracy between WS and AWN still exist in adversarial accuracy. The results are shown in Table 4.4 where WS and AWN are further attacked and evaluated to demonstrate the influence of performance drop on clean and adversarial accuracy. Comparing WS and AWN, our proposed AWN consistently outperforms WS in both adversarial and clean accuracy, which indicates that directly normalizing adder weights could hurt the representation power of ANNs and restrict the adversarial accuracy of ANNs. With proposed parameters ν and υ in Eq. 4.11, AWN successfully relieves this problem through exploring the balance between expressive power and weight magnitude reduction, which achieves better classification performance and adversarial robustness. We further evaluate the effectiveness of proposed ANN robust inference strategy as in Eq. 4.17. We denote the strategy which replaces both tracked mean and variance with running ones as -r and our proposed one as -R. Comparing two inference methods, our proposed strategy consistently outperforms -r, which verifies that the activated robustness mainly comes from the running mean which automatically eliminates the perturbations.

TABLE 4.6. Adversarial robustness evaluation of different inference strategy of ANN-AWN with ResNet-32 on CIFAR-10.

Running Mean	Running Variance	Clean	FGSM	BIM^7	PGD^7	MIM^5	RFGSM^5
✗	✗	92.26	29.70	0.05	0.08	0.84	0.01
✗	✓	88.95	17.23	10.49	11.69	16.74	8.63
✓	✗	91.25	**61.30**	**59.41**	**59.74**	**61.54**	**39.79**
✓	✓	90.38	54.69	25.65	26.55	38.56	14.03

4.5.6 Adaptive Weight Normalization on CNN

Our proposed Adaptive Weight Normalization is based on the analysis of the variance of ANN features and specifically designed for ANNs. We further evaluate the performance of AWN on CNNs. As shown in Table 4.5, with the involvement of AWN, CNN obtains slight better adversarial robustness. However, comparing with ANN, the improvement robustness of CNN with AWN is marginal, which demonstrates that the superior collaboration of proposed AWN with ANNs and shows strong evidence of potential robustness of ANNs.

4.5.7 Variants of Inference Strategy

Our proposed ANN robust inference strategy is derived from the analysis of variance in ANNs. To illustrate the effectiveness of proposed inference strategy, we evaluate several variants of them with ResNet-32 on CIFAR-10. Our proposed robust inference strategy makes use of running mean of current batch and the tracked variance in batch normalization layer, which denotes the third row in Table 4.6. Comparing with other variants of inference strategy, our proposed one achieves the best robustness with a large margin. The evaluation shows strong evidence that the perturbations can be eliminated by the subtraction of a single scalar value on feature map, which is consistent with our analysis.

TABLE 4.7. Robustness stability of proposed ANN-R-AWN with ResNet-20 on CIFAR-10 under different inference settings.

Batch size	Shuffle	Clean	FGSM	BIM^7	PGD^7	MIM^5	RFGSM^5
96	✗	90.55	45.93	42.62	43.39	46.52	18.36
96	✓	90.59	46.81	42.26	43.42	46.65	18.19
64	✗	90.56	45.58	41.53	42.42	46.03	18.15
256	✗	90.78	46.93	42.62	43.28	46.51	20.31

4.5.8 Stability of Robustness

Different from traditional defense methods which are mainly based on adversarial training, our proposed robust inference strategy works during testing phase. Since our proposed ANN-R-AWN makes use of the running mean of current batch, the performance could vary due to different inference settings. Here we provide stability test of robustness under different batch size and shuffle settings to eliminate the concerns that the superior adversarial robustness comes from other aspects.

As shown in Table 4.7, the stability of adversarial robustness is evaluated under different inference settings. We first show whether the shuffle of test set influences the performance. Comparing the first two rows, ANN-R-AWN achieves similar performance with or without shuffle. Furthermore, we forward test set with different batch size as shown in the last two rows. It is obvious that both clean and adversarial accuracy has slight increment with increasing batch size, which matches the well-known observation that networks with batch normalization layer gets better performance with larger batch size. For adversarial robustness, ANN-R-AWN still outperforms CNN and ANN with a large margin when a smaller batch size of 64 is selected, which illustrates that the superior adversarial robustness mainly comes from proposed ANN robust inference strategy and adaptive weight normalization. For a fair comparison, all the empirical experiments of adversarial robustness are conducted with the same inference setting of a batch size of 96 without shuffle.

4.5.9 Classification Performance

To illustrate the effectiveness of adaptive weight normalization, we evaluate the classification performance of ANN, ANN-WS and ANN-AWN on CIFAR-10 and ImageNet. Note that our main contribution lies on the stability and robustness of AdderNets. The evaluation of classification are included here for the completeness.

TABLE 4.8. Classification performance evaluation.

Method	Model	Dataset	Accuracy
ANN	ResNet-20	CIFAR-10	91.84
ANN-WS	ResNet-20	CIFAR-10	90.62
ANN-AWN	ResNet-20	CIFAR-10	91.42
ANN	ResNet-18	ImageNet	67.00
ANN-WS	ResNet-18	ImageNet	64.17
ANN-AWN	ResNet-18	ImageNet	67.11

The comparison is shown in table 4.8. The exists a trade-offs between ANN classification performance and stability, as discussed in Section 4.2. As shown in the first three rows, both ANN-WS and ANN-AWN have accuracy drop, however, ANN-WS has relatively larger drop from $91.84\% \rightarrow 90.62\%$ and ANN-AWN has acceptable drop from $91.84\% \rightarrow 91.42\%$. The gap becomes more obvious when methods are evaluated on ImageNet. Our proposed ANN-AWN achieves slightly better performance than ANN while ANN-WS has a dramatic accuracy drop, which empirically verified our analysis in Sec 3.2 that the performance could be largely constrained without incorporating the shift and scale parameters.

4.6 Conclusion

In this chapter, we investigate the major concerns of AdderNets through approximating the mean and variance of output features of an arbitrary adder layer. With a derived lower boundary, we show that the instability of AdderNets mainly comes from drastic fluctuations of running mean and variance in batch normalization layer whose computation is dominated by the variance of weights. Our proposed adaptive weight normalization (AWN) works with AdderNets to optimize adder weight distributions adaptively, which significantly improves

the stability and leads to smooth landscape. Our analysis of the adder layer forwarding reveals the potential superior defense ability of AdderNets against perturbations and proposed robust inference strategy together with AWN successfully activate the adversarial robustness. Experiments conducted on stability and robustness demonstrate the superior performance of the proposed ANN-AWN.

CHAPTER 5

Random Filter Design for Adversarial Robustness

5.1 Motivation

Although Deep Neural Networks (DNNs) have become a popular technique in various scientific fields (He et al. 2016; Zhang et al. 2019b; Yang et al. 2022), the vulnerability of DNNs reveals the high risk of deployment in real scenarios, especially under the attack of adversarial examples (Goodfellow et al. 2014; Madry et al. 2018). Some tiny and imperceptible perturbations to network inputs could result in the major changes of outputs, which can be easily crafted through various adversarial attack strategies (Andriushchenko et al. 2020; Croce and Hein 2020; Goodfellow et al. 2014). Adversarial attacks could be generally categorized into two streams, the white-box and black-box attacks. In black-box setting, the attackers have no knowledge of victim models but can estimate the strong perturbation via surrogate models or huge number of queries (Guo et al. 2019b; Ilyas et al. 2018). In white-box setting, the attackers have full knowledge of victim model, including the model parameters, network architecture, and inference strategy (Szegedy et al. 2014; Madry et al. 2018). Since the gradients of victim models can be directly fetched, the crafted adversarial examples are more aggressive and the performance under white-box attacks is one of the key criteria of robustness evaluation.

Seeking adversarial robust networks becomes a key challenge when it comes to the deployment of DNNs. One of the most popular and effective techniques is adversarial training (Madry et al. 2018), which arguments the training data with adversarial examples within a fixed perturbation size. With the involvement of adversarial examples, DNNs are optimized to preserve their outputs for perturbed samples within the ℓ_p ball of all training input data. However, due to

the increasingly advanced attack techniques, it is difficult for existing adversarially trained networks to achieve satisfactory robustness against all potential attacks. Furthermore, the training on stronger adversarial examples could hurt the natural generalization of models (Zhang et al. 2020), and there exists a trade-off between robustness and accuracy (Zhang et al. 2019a).

Besides the traditional adversarial training, the utilization of randomization in adversarial robustness has been proven effective. For example, Liu *et al*. (Liu et al. 2018b) propose to inject noise which is sampled from Gaussian distribution to the inputs of convolution layers. Some theoretical analyses have shown that randomized classifiers can easily outperform deterministic ones in defending against adversarial attacks (Pinot et al. 2019; Pinot et al. 2020). We mainly attribute the improvement of randomization in adversarial robustness evaluation to the fusion of features with noises, which prevents white-box attackers from obtaining the precise gradients of loss with respect to the inputs. Although the involvement of noise in the networks can be an effective defense mechanism, the design of noises, such as the way of injection, the magnitude of noise, etc., can also significantly influence the natural generalization of networks in practice. The trade-offs between the adversarial robustness and optimization difficulty are always ignored in the randomized techniques, which limits their superiority to deterministic models.

In this work, we introduce randomness into deep neural networks with the help of random projection filters. Random projection is a simple yet effective technique for dimension reduction, which can approximately preserve the pairwise distance between any two data points from a higher-dimensional space in the projected lower-dimensional space under certain conditions. The theoretical and empirical advantages offered by random projection thus inspire a new way to explore the potential of noise injection with better trade-offs in Convolutional Neural Networks (CNNs). We propose to *partially replace the convolutional filters with the random projection filters*. Theoretically, we extend the scope of Johnson-Lindenstrauss Lemma (Vershynin 2018) to cover the convolutions, where partial convolutional filters are randomly sampled from a zero-mean Gaussian distribution. Pairwise example distance can also be approximately preserved under the new convolutions defined by random projection

filters, if the number of random projection filters is lower bounded in terms of the weight norm of the remaining convolutional filters. Motivated by these observations, we introduce a simple and efficient defense scheme via the proposed Random Projection Filters (RPF). As parameters of random projection filters are randomly sampled during forwarding, the attackers have no knowledge of upcoming sampled parameters even if in white-box attack settings. The effectiveness of proposed RPF is verified via extensive empirical evaluations in our experiments.

5.2 Methodology

5.2.1 Preliminaries

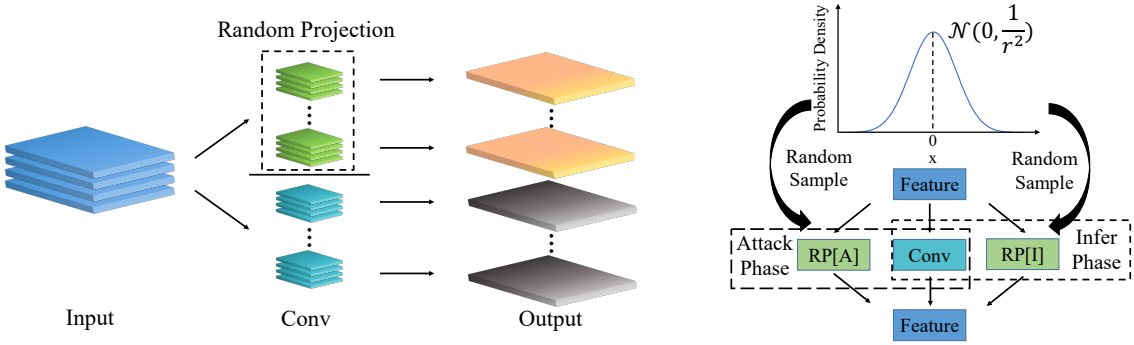
Adversarial Training Given a classifier f with parameters θ which maps the input image $\mathcal{X} \in \mathbb{R}^D$ to the logits $f_\theta(\mathcal{X}) \in \mathbb{R}^C$ where D and C denote the dimension of original image and number of classes respectively, the adversarial example $\mathcal{X}^{adv} = \mathcal{X} + \delta$ is defined as

$$\max_{\mathcal{X}^{adv}} \mathcal{L}(f_\theta(\mathcal{X}^{adv}), y), \text{ s.t. } \|\mathcal{X}^{adv} - \mathcal{X}\|_p \leq \epsilon, \tag{5.1}$$

where $\mathcal{L}(,)$ denotes the loss function (e.g.cross-entropy loss), ϵ denotes the maximum perturbation size and y denotes the ground truth label. In adversarial training strategy, the adversarial examples are generated and fed to the classifier to form a min-max optimization as

$$\min_{\theta} \max_{\mathcal{X}^{adv}} \mathcal{L}(f_\theta(\mathcal{X}^{adv}), y), \text{ s.t. } \|\mathcal{X}^{adv} - \mathcal{X}\|_p \leq \epsilon. \tag{5.2}$$

Random Projection The random projection is a linear transformation from D dimensions to D' dimensions via a random matrix $R \in \mathbb{R}^{D \times D'}$ where each entry is drawn from an independent identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, 1)$ and the columns are normalized to have unit lengths. Given data point $x \in \mathbb{R}^D$, the random projected data point $x' \in \mathbb{R}^{D'}$ can be derived as $x' = xR$. In CNNs, we can simply replace the filter parameters with the i.i.d. zero-mean Gaussian weights.



(a). Random Projection Filters (b). Defense with Random Gaussian Matrix

FIGURE 5.1. An overview of proposed with Random Projection Filters with defense scheme. Part of the filters in convolutional layers are replaced by random projection, whose weight is randomly sampled from a Gaussian distribution. RP[A] and RP[I] denote the sampled Gaussian matrix of random projection filters during attack and infer phase respectively.

5.2.2 Random Projection Filters

White-box attacks have full access to network including the parameters and architecture and it is difficult for networks to defend against various white-box attacks since adversarial perturbations can be easily found via gradient ascent. Thus, to prevent attackers from deriving precise gradient on input image, we propose to involve some noises during the network inference. However, the magnitude of noise and the manner of the noise involvement can significantly influence the optimization, which implies that a careful design of noise is necessary to achieve a better trade-offs between adversarial robustness and optimization difficulty.

Motivated by the distance preservation of random projection, we propose to incorporate random projection into CNNs to achieve a better trade-offs. The core idea of random projection mainly lies in the Johnson-Lindenstrauss lemma which states that a projection of data points of high dimension to an appropriate lower dimensional space can preserve the distances among the data points. By definition, given a linear mapping $F : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ and a set of data points $\mathbb{X}$ with size of m, for $D' > 8(ln\ m)/\epsilon^2$

$$(1-\epsilon)\|x_i - x_j\|^2 \leq \|F(x_i) - F(x_j)\|^2 \leq (1+\epsilon)\|x_i - x_j\|^2, \tag{5.3}$$

for all x_i, x_j in $\mathbb{X}$. Since convolution is a linear mapping, Johnson-Lindenstrauss lemma holds for CNNs. According to Eq. 5.3, the dimension of projected space plays an important role in random projection. Intuitively, a higher ratio of random projection in CNNs could make it difficult for white-box attackers to obtain adversarial perturbations, however, it also brings huge noise to network optimization. Thus, to balance these trade-offs, we propose to replace a bunch of the convolutional filters in CNN layers with random projection, as shown in Figure 5.1 (a). Formally, given the input feature $x \in \mathbb{R}^{n\times n\times d}$ where n and d denote the size and dimension of feature respectively, and a single filter of CNN $F \in \mathbb{R}^{r\times r\times d}$ where r denotes the kernel size, the output z is given by

$$z(p,q) = F * [x]^r_{p,q} = \sum_{i=0}^{r}\sum_{j=0}^{r}\sum_{k=0}^{d} F(i,j,k)\cdot x(p+i, q+j, k), \tag{5.4}$$

where $[x]^r_{p,q}$ denotes the subarea of x for convolutional operation with row from p to $p+r-1$ and column from q to $q+r-1$. For a convolutional layer which contains N filters $F_1, \ldots, F_N$, we divide these filters into two parts. We denote $F_1, \ldots, F_{N_r}$ as the random projection filters with parameters randomly sampled from a zero-mean Gaussian distribution, and denote $F_{N_r+1}, \ldots, F_N$ as the traditional convolutional filters with trainable parameters. The output z can be formulated as

$$\begin{aligned} &z(p,q) = \left[F_{1,\ldots,N_r} * [x]^r_{p,q}, F_{N_r+1,\ldots,N} * [x]^r_{p,q}\right], \\ &\text{where } F_1, \ldots, F_{N_r} \sim \mathcal{N}(0, \sigma^2), \end{aligned} \tag{5.5}$$

where $\left[,\right]$ denotes the concatenation and σ^2 denotes the variance of random projection filters. With Eq. 5.5, the trade-offs between adversarial defense capability and network optimization difficulty can be explored via adjusting N_r. The Johnson-Lindenstrauss lemma in CNN layers has been studied in (Nachum et al. 2022). In this work, under mild assumptions, we further generalize it to the random projection scenario where only N_r output features are derived via random projection while the others via optimized convolutional filters. Since batch normalization layers with affine transformation are widely adopted in CNNs, we assume that the inputs to the filters follows Gaussian distribution with mean of β and variance of γ^2 where β and γ are the affine parameters of batch normalization layers. We also assume that

the variance of trainable filters $F_{N_{r+1}}, \ldots, F_N$ is same as the one of random projection filters $F_1, \ldots, F_{N_r}$.

THEOREM 1. *Let $x, y \in \mathbb{R}^{n \times n \times d}$ be the input to the filters, which follow Gaussian distribution $x, y \sim \mathcal{N}(\beta, \gamma^2)$. Consider we have N filters $F_1, \ldots, F_N \in \mathbb{R}^{r \times r \times d}$, in which $F_1, \ldots, F_{N_r}$ denote the random projection matrices where all the entries are drawn from i.i.d. $\mathcal{N}(0, \frac{1}{r^2})$ while $F_{N_{r+1}}, \ldots, F_N$ denote the trainable parameters of convolutional layer with mean of μ and variance of $\frac{1}{r^2}$ where r denotes the kernel size. We assume that*

$$\max_{i,j} \|[x]_{ij}^r\| \leq R, \quad \max_{i,j} \|[y]_{ij}^r\| \leq R, \quad \max_i \|F_i\| \leq W, \tag{5.6}$$

and we denote $K = n^2 max\{\frac{C_0^2 R^2}{r^2}, (r^2 d\beta\mu + C_0 W\gamma)^2\}$ and $D = \mu^2\beta^2 n^2 r^4 d^2$. Then the probability that the distance between x, y cannot be preserved after convolutional operation F can be upper bounded as

$$\begin{aligned} &\mathbb{P}\left(\left|\frac{1}{N}\sum_{l=1}^{N}\langle F_l * x, F_l * y\rangle - \langle x, y\rangle\right| \geq \epsilon\right) \leq \delta, \ \textit{for } \delta > 0 \textit{ and} \\ &N_r > \begin{cases} \frac{(D-\epsilon)N + K log\frac{2Cn^2}{\delta}}{D}, & \textit{if } \frac{\epsilon - \frac{N-N_r}{N}D}{K} \leq \frac{(\epsilon - \frac{N-N_r}{N}D)^2}{K^2} \\ \frac{(D-\epsilon)N + NK\sqrt{log\frac{2CNn^2}{\delta}}}{D}, & \textit{otherwise} \end{cases} \end{aligned} \tag{5.7}$$

where C and C_0 are absolute constants.

In Theorem 1, we define the distance between two data points x, y as $\langle x, y\rangle$ and the geometric representation preservation as the scenario where the sum of absolute differences between $\langle F_l * x, F_l * y\rangle$ and $\langle x, y\rangle$ can be bounded by ϵ. The proof mainly utilizes the Bernstein's inequality (Vershynin 2018) and the detailed proof is provided in the supplementary material. In Eq. 5.7, we can see that the probability of breaking geometric representation preservation can be upper bounded by δ if an appropriate N_r is selected for this convolutional layer. It indicates a lower bound of the number of random projection filters. Since our objective is the better trade-offs between network optimization difficulty and defense capability, we propose to reduce the number of random projection filters N_r while meeting the constraint in Eq. 5.7 so that geometric representation preservation holds and the noises introduced by random projection filters do not damage the convergence and performance of networks. Besides the

Algorithm 2 Adversarial Training with Random Projection

Input: Network with random projection filters f_θ; Number of random projection filters N_r; Weight decay of random projection filters α; Perturbation size ϵ; Attack step size η; Attack iterations t; Training set $\{\mathcal{X}, \mathcal{Y}\}$;
while not converge **do**
  Sample a batch of data $\{x, y\}_{i=1}^{n}$ from $\{\mathcal{X}, \mathcal{Y}\}$;
  for F with random projection filters **do**
    $F_1, \ldots, F_{N_r} \sim \mathcal{N}(0, \frac{1}{r^2})$
  end for
  Random initialize adversarial perturbation δ;
  for $i \leftarrow 1$ to t **do**
    $\delta = \delta + \eta \cdot \text{sign}(\nabla_x \mathcal{L}(f_\theta(x^{adv}), y)$;
    Clip x^{adv} = $\text{Clip}_x^\epsilon\{x + \delta\}$;
  end for
  for F with random projection filters **do**
    $F_1, \ldots, F_{N_r} \sim \mathcal{N}(0, \frac{1}{r^2})$;
  end for
  $\theta = \theta - \nabla_\theta \Big(\mathcal{L}(f_\theta(x^{adv}), y) + \alpha \| F_{N_r+1}, \ldots, F_N \| \Big)$;
end while

constants, the lower bound of N_r is dominated by $K = n^2 max\{\frac{C_0^2 R^2}{r^2}, (r^2 d\beta\mu + C_0 W\gamma)^2\}$. In practice, the maximum Euclidean norm of input subareas R can be well-controlled due to batch normalization layers while the weight norm of trainable parameters $F_{N_{r+1}}, \ldots, F_N$ cannot. Thus, to reduce the burden of N_r, we propose to impose a larger weight decay to the $F_{N_{r+1}}, \ldots, F_N$, which minimizes W to relieve the constraint in Eq. 5.7. Thus, the objective in Eq. 5.2 can be reformulated as

$$\begin{aligned} &\min_\theta \max_{\mathcal{X}^{adv}} \mathcal{L}(f_\theta(\mathcal{X}^{adv}), y) + \alpha \| F_{N_r+1}, \ldots, F_N \|, \\ &\text{s.t. } \| \mathcal{X}^{adv} - \mathcal{X} \|_p \leq \epsilon, \end{aligned} \tag{5.8}$$

where α denotes the hyperparameter of weight decay.

5.2.3 Adversarial Training with Random Projection

Existing white-box attacks can easily discover an aggressive perturbation δ for a fixed network f via gradient ascent, however, it is difficult for the generated adversarial example

$x' = x + \delta$ to attack another network f' successfully (Wang and He 2021; Tramèr et al. 2017b). Since the parameters of random projection filters $F_1, \ldots, F_{N_r}$ are randomly sampled from $\mathcal{N}(0, \frac{1}{r^2})$, we individually sample parameters for random projection filters during attacking and inference phase, which is denoted as $F_{1:N_r}[A]$ and $F_{1:N_r}[I]$ respectively. Considering the partial derivative of output feature z with respect to the input adversarial feature x^{adv}

$$\begin{aligned} \frac{\partial z(p, q, 1:N_r)}{\partial x^{adv}(p+i, q+j, k)} &= F_{1:N_r}[A](i, j, k) \\ &\neq F_{1:N_r}[I](i, j, k), \end{aligned} \tag{5.9}$$

which indicates that the difference between $F_{1:N_r}[A]$ and $F_{1:N_r}[I]$ can significantly influence the gradient of adversarial examples. $f_{\theta[A]}$ denotes the parameters to be optimized in a network with the random projection filters $F_{1:N_r}[A]$, while $f_{\theta[I]}$ corresponds to the parameters to be optimized in one network with $F_{1:N_r}[I]$. The mix-max optimization in Eq. 5.2 can be reformulated as

$$\begin{aligned} &\min_{\theta[I]} \max_{\mathcal{X}^{adv}} \mathcal{L}(f_{\theta[A]}(\mathcal{X}^{adv}), y) + \alpha \|F_{N_r+1}, \ldots, F_N\|, \\ &\text{s.t. } \|\mathcal{X}^{adv} - \mathcal{X}\|_p \leq \epsilon, \end{aligned} \tag{5.10}$$

where adversarial examples have been produced from a network with random projection filters $F_{1:N_r}[A]$ and then the adversarial examples are used to train a network with random projection filters $F_{1:N_r}[I]$. The details of adversarial training with random projection is shown in Algorithm 2.

With the involvement of random projection in convolutional filters and corresponding adversarial training strategy in Algorithm 2, CNNs can perform a strong defense during inference phase, which is illustrated in Figure 5.1 (b). Considering a white-box attack which has access to the current sampled random projection parameters $F_{1:N_r}[A]$ and generates adversarial example $\mathcal{X}^{adv}$ of $f_{\theta[A]}$ successfully, $F_{1:N_r}[A]$ is re-sampled and becomes $F_{1:N_r}[I]$ during evaluation so that $\mathcal{X}^{adv}$ can hardly be generalized to $f_{\theta[I]}$. Together with the fact that Theorem 1 holds for random Gaussian matrix sampling strategy, RPF achieves better trade-offs between clean accuracy and adversarial robustness.

TABLE 5.1. The comparison with noise injection techniques with ResNet-18 on CIFAR-10 and CIFAR-100.

Dataset	Method	Clean	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AA
CIFAR-10	AT	81.84	56.70	52.16	78.46	54.96	0.35	47.69
	Additive	81.24	59.19	57.61	80.84	57.83	73.44	62.25
	Multiplicative	83.16	61.92	59.49	82.80	59.48	78.28	63.78
	RPF(Ours)	**83.79**	**62.71**	**61.27**	**83.60**	**60.72**	**79.43**	**64.38**
CIFAR-100	AT	55.81	31.33	28.71	50.94	30.26	0.79	24.48
	Additive	53.34	31.72	31.13	52.50	31.16	46.76	36.37
	Multiplicative	54.52	34.09	32.58	54.61	32.45	50.90	38.13
	RPF (Ours)	**56.88**	**37.67**	**37.37**	**56.59**	**35.31**	**54.39**	**42.88**

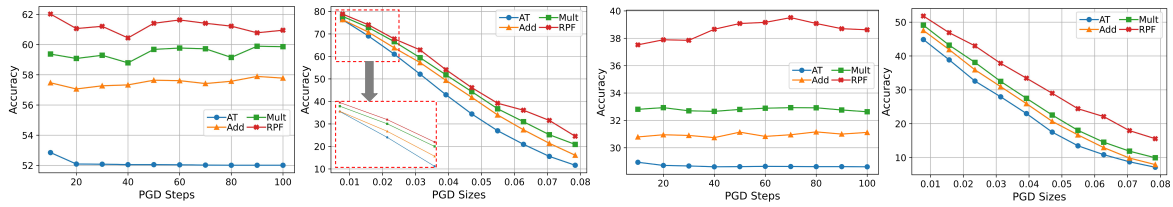
5.3 Experiments

5.3.1 Experimental Setup

Datasets Following previous work (Rice et al. 2020; Madry et al. 2018), we include multiple datasets in our evaluation, including CIFAR-10/100 and ImageNet. CIFAR-10 and CIFAR-100 datasets (Krizhevsky, Hinton et al. 2009) have 10 and 100 categories respectively. Each of them contains $60K$ color images with size of 32×32, including $50K$ training images and $10K$ validation images. ImageNet dataset (He et al. 2016) contains $1.2M$ training images and $50k$ testing images with size of 224×224 from 1000 categories.

Models Note that our proposed RPF can be easily applied to any CNN-based models via partially replacing CNN filters with random projection ones. Thus, we evaluate the performance of RPF on several widely compared models in the field of adversarial robustness, including ResNet-18 (Krizhevsky, Hinton et al. 2009) and WideResnet-34-10 (Zagoruyko and Komodakis 2016) on CIFAR-10/100 as well as ResNet-50 (Krizhevsky, Hinton et al. 2009) on ImageNet.

Training Strategy We follow the protocol of state-of-the-art adversarial training strategy (Rice et al. 2020) to setup our experiments on CIFAR-10/100. We train the network for 200 epochs with a batch size of 128 via SGD with momentum of 0.9. The learning rate is set to 0.1 and the weight decay is set to 5×10^{-4}. We use a piecewise decay learning rate scheduler



(a) CIFAR-10 PGD Steps (b) CIFAR-10 PGD Sizes (c) CIFAR-100 PGD Steps (d) CIFAR-100 PGD Sizes

FIGURE 5.2. The evaluation of stronger PGD attacks with ResNet-18 on CIFAR-10 and CIFAR-100.

with a decay factor of 0.1 at 100 and 150 epoch. For adversarial example generation, PGD-10 is used with the a maximum perturbation size $\epsilon = 8/255$. The step size of PGD is set to $2/255$. On ImageNet, we train the network for 90 epochs with a batch size of 1024 via SGD with momentum of 0.9. The learning rate is set to 0.02 and the weight decay is set to 1×10^{-4}. We use a cosine learning rate scheduler. For adversarial example generation, PGD-2 is used with the a maximum perturbation size $\epsilon = 4/255$.

Attacks For the adversarial robustness evaluation of proposed RPF, we conduct extensive experiments on various attacks, including Fast Gradient Sign Method (FGSM) (Szegedy et al. 2014), Projected Gradient Descent (PGD) (Madry et al. 2018), CW attack (Carlini and Wagner 2017), Momentum-based Iterative Fast Gradient Sign Method (MIFGSM) (Dong et al. 2018), DeepFool (Moosavi-Dezfooli et al. 2016) and Auto Attack (Croce and Hein 2020). We follow the standard protocol (Kim 2020) to setup the attacks. The maximum perturbation size ϵ is set to $8/255$ for FGSM, PGD, MIGFSM, and Auto Attack. The step size is set to $2/255$ for PGD and MIGFSM, and the steps are 20 and 5 for PGD and MIGFSM respectively. For CW attack, the learning rate is set to 0.01 with 1000 steps. For DeepFool, the steps are set to 50 with an overshoot of 0.02. On ImageNet, ϵ is set to $4/255$ with steps of 10 and 50.

Baselines We include extensive baselines for comparison. We compare RPF with some randomize techniques, such as additive noise (Liu et al. 2018b) and random bits (Fu et al. 2021). We also include another strong baseline for comparison which replaces the additive noise with the multiplicative noise. In addition, some other defense techniques are also involved in our comparison, including RobustWRN (Huang et al. 2021), AWP (Wu et al. 2020), SAT (Xie et al. 2020b), LLR (Qin et al. 2019), and RobNet (Guo et al. 2020).

TABLE 5.2. Adversarial robustness evaluation of randomized techniques with WideResNet on CIFAR-10.

Method	FGSM	PGD^{20}	MIFGSM	AA
AT (Rice et al. 2020)	60.65	55.06	58.47	52.24
Random Bit (Fu et al. 2021)	57.95	53.96	56.32	53.30
Additive (Liu et al. 2018b)	62.36	58.47	60.58	60.55
Multiplicative	62.01	57.48	59.79	57.99
RPF (Ours)	**63.95**	**63.71**	**60.77**	**68.71**

5.3.2 Results on CIFAR

To demonstrate the effectiveness of proposed RPF, we first perform six different attacks. Besides the deterministic classifier with adversarial training denoted as AT, we include the additive noise injection. We follow the setting of (Liu et al. 2018b) to conduct noise injection where some sampled noises are added to the input of convolution layers. Furthermore, we also construct another stronger baseline, namely multiplicative noise injection, which simply fuses the feature maps via multiplying noises. Additive noises are sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$ while multiplicative noises are sampled from $\mathcal{N}(1, 1)$. Although these baselines are simple, they can achieve satisfactory adversarial robustness in our defense scheme. The comparison results are shown in Table 5.1. All the baselines are adversarially trained with the same setting. On CIFAR-10, the additive noise injection can achieve 57.61% robust accuracy under PGD^{20} attack and 59.19% under FGSM attack. Compared with deterministic AT baseline, additive noise injection improves the baseline by 5.45% under PGD^{20} attack and 2.49% under FGSM attack, which demonstrates that the randomized techniques can improve the adversarial robustness against current popular white-box attacks. Besides additive noise injection, the multiplicative noise injection baseline also shows superiority to AT baseline. Comparing additive and multiplicative noise injections, the multiplicative one has better performance. For example, multiplicative noise achieves 82.80% robust accuracy under CW attack with a gap of 1.96% an 63.78% under Auto Attack with a gap of 1.53%, which indicates that the noise injected to CNNs as well as the method of injection play important roles in the adversarial robustness. The natural accuracy decrement of additive noise injection also implies that there exists a trade-offs between natural generalization of network and adversarial robustness if randomized techniques are utilized. Thus, RPF is introduced

TABLE 5.3. Comparison with SOTA defense algorithms on CIFAR-10 and ImageNet.

Method	CIFAR-10		ImageNet	
	PGD^{20}	AA	PGD^{10}	PGD^{50}
Overfit (Rice et al. 2020)	55.06	52.24	39.85	39.19
RobustWRN (Huang et al. 2021)	59.13	52.48	31.14	-
AWP (Wu et al. 2020)	58.14	54.04	-	-
RobNet (Guo et al. 2020)	52.74	-	37.16	37.15
Random Bit (Fu et al. 2021)	53.96	53.30	42.88	42.72
SAT (Xie et al. 2020b)	56.01	51.83	-	42.30
LLR (Qin et al. 2019)	54.24	-	-	47.00
RPF (Ours)	**63.71**	**68.71**	**56.56**	**55.41**

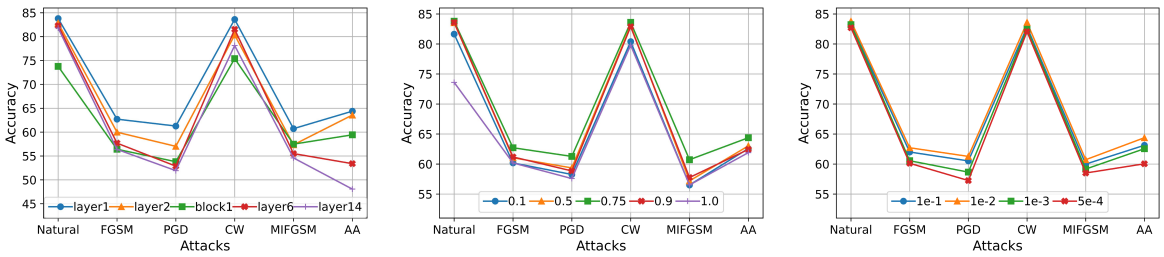
to tackle this problem via the involvement of random projection. With the assistance of the geometric representation preservation property, our algorithm can achieve better trade-offs than these baselines. To illustrate, on ResNet-18, our proposed RPF achieves 83.79% natural accuracy, 61.27% robust accuracy under PGD^{20} attack, 79.43% under DeepFool attack, and 64.38% under Auto Attack, with a obvious gap between RPF and all the baselines. Under 6 different attacks, our proposed RPF achieves the best performance in all the scenarios as well as the highest clean accuracy. This superiority can also be generalized to CIFAR-100. RPF improves the robust accuracy of AT baseline by 18.40% under Auto Attack, 8.66% under PGD^{20} attack, and 5.05% under MIFGSM attack. Furthermore, RPF achieves a clean accuracy of 56.88%, which improves all the noise injection techniques by a considerable gap. On the contrary, the superiority of both additive and multiplicative noises to AT baseline becomes much slighter, which highlights the necessity of proposed RPF as a more advanced noise injection techniques for adversarial robustness.

We also provide adversarial robustness evaluation with WideResNet-34-10 on CIFAR-10, and compare the results with other randomized baselines. The comparison is presented in Table 5.2. Similar to the results on ResNet-18, our proposed RPF achieves strong adversarial robustness. Under powerful Auto Attack, RPF remains a accuracy of 68.71% with a gap of 16.47 to deterministic AT classifier and a gap of 8.16% to the additive noise injection. Compared to other randomized techniques, such as random bit, RPF also show clear advantage, with a robust accuracy of 63.71% under PGD^{20} compared to 53.93% in random bits. The extensive

experimental results under various attacks with multiple models on different datasets show strong empirical evidence that our proposed RPF can achieve superior adversarial robustness.

5.3.3 Evaluate with Stronger Attacks

Besides the standard evaluation of adversarial robustness under various attacks in Table 5.1, we also provide the performance of AT baseline, noise injection baselines, and our algorithm under stronger attacks. The evaluation is conducted on CIFAR-10 and CIFAR-100, as shown in Figure 5.2. We mainly consider two scenarios, including the PGD attacks with more steps and the PGD attacks with larger maximum perturbation size ϵ. Specifically, we consider the attacking scenario of PGD$^{10}, \ldots,$ PGD^{100}, and $\epsilon \in [2/255, 20/255]$. The randomized techniques are insensitive to the increasing PGD steps, as illustrated in Figure 5.2 (a) and (c). Different from the deterministic AT classifier whose robust accuracy is inversely proportional to the number of steps, all the noise injection based method still have chance to achieve a relatively higher robust accuracy even under PGD^{100}. Among all the techniques, RPF achieves the best performance under different PGD steps. Taking PGD size into consideration, all the robust methods have a large drop with the increment of perturbation size since the search space of adversarial perturbations becomes much larger. The results are illustrated in Figure 5.2 (b) and (d). Compared with baselines, RPF shows more robust performance under larger perturbation sizes. On CIFAR-10, RPF achieves 78.73% accuracy under PGD with $\epsilon = 2/255$ and 24.53% under PGD with $\epsilon = 20/255$, where the drop is 54.31%. Additive noise injection has a drop of $60.29\%(76.37\% \rightarrow 16.08\%)$ and multiplicative noise injection has a drop of $56.92\%(77.80\% \rightarrow 20.88\%)$. Similarly, on CIFAR-100, the drop of RPF is 36.22%, 39.63% for additive noise injection, and 39.16% for multiplicative noise injection, where RPF has a much smaller accuracy drop. Thus, among all the noise injection techniques, RPF performs better resistance against stronger PGD attacks. In all the scenarios including PGD steps and sizes, RPF consistently achieves the best robust accuracy, which highlights the superior defense capability of RPF.



(a) Location of RPF (b) Ratio of RPF (c) Weight Decay of Conv

FIGURE 5.3. Ablation studies of Random Projection Filters on location, ratio, and weight decay.

5.3.4 State-of-the-art Comparison

We further provide the comparison with state-of-the-art defense techniques to demonstrate the effectiveness of proposed RPF. We consider two popular benchmarks which are widely compared, including WideResNet-34-10 (WRN-34-10) on CIFAR-10 and ResNet-50 on ImageNet. For evaluation, we select PGD^{20} and Auto Attack on CIFAR. On ImageNet, we report the robust accuracy under PGD^{10} and PGD^{50} attacks. For baselines Overfit (Rice et al. 2020) and Random bit(Fu et al. 2021), we reproduce the results with the official implementation. For the rest results, we cite them from the original paper. Compared with these baselines, RPF performs much stronger defense. RPF achieves 5.57% higher than AWP with PGD^{20} and 15.41% higher than Random Bits with Auto Attack on CIFAR-10. Similarly, RPF achieves 19.4% higher than RobNet with PGD^{10} and 8.41% higher than LLR with PGD^{50} on ImageNet. We mainly attribute the success of RPF to the theoretical-guided design of randomized techniques. Note that RPF can be easily integrated into other state-of-the-art defense techniques to further improve the performance since RPF is orthogonal to other baselines and there is no extra parameters involved.

5.3.5 Evaluation of Black-box Attacks

We evaluate RPF under black-box attacks Square (Andriushchenko et al. 2020) and Pixle (Pomponi et al. 2022) with ResNet-18 on CIFAR-10 and CIFAR-100. Query number is set to 5000 in Square and the maximum patch size is 10×10 in Pixle. The advantage of RPF over AT can be found in Table 5.4 where RPF achieves better robust accuracy in all the scenarios.

TABLE 5.4. Evaluation of black-box attacks.

	C-10		C-100			C-10		C-100	
Attack	AT	RPF	AT	RPF	Attack	AT	RPF	AT	RPF
Square	53.64	**76.56**	29.57	**48.21**	Pixle	8.21	**44.84**	1.16	**23.39**

TABLE 5.5. Comparison with other noise injection techniques.

Method	R20		R32		R44		R56	
	FGSM	PGD	FGSM	PGD	FGSM	PGD	FGSM	PGD
PNI	54.40	45.90	51.50	43.50	55.80	48.50	53.90	46.30
Learn2Perturb	58.41	51.13	59.94	54.62	61.32	54.62	61.53	54.62
RPF	**63.27**	**60.94**	**62.52**	**60.78**	**63.39**	**62.47**	**62.30**	**60.97**

5.3.6 Comparisons with Noise Injection Techniques

Different from (He et al. 2019; Jeddi et al. 2020) which utilize additive noises, RPF replaces partial filters with random projection to form concatenate noise. Following the same setting in (Jeddi et al. 2020), we apply RPF on ResNet-20/32/44/56. RPF performs better than PNI (He et al. 2019) and Learn2Perturb (Jeddi et al. 2020) with relatively large margins, as shown in Table 5.5.

5.3.7 Ablation Study

During the setup of random projection filters, multiple components could influence the performance, including the location of random projection filters in the network, the ratio of random projection, and the weight decay of the other convolution filters. We further provide more empirical evidence to verify the observations in Theorem 1.

Random Projection Filters Location We first replace a specific convolution layer with the one with random projection filters in different locations of network. Taking ResNet-18 as an example, we propose to apply random projection in different layers or entire block and the number of random projection filters is set to 48. The natural accuracy and adversarial robustness under different scenarios are shown in Figure 5.3 (a). Considering the replacement of an entire block, the injected noise could be redundant and overwhelm the natural

generalization, which makes both natural and robust accuracy drop to some extent. RPF on the first layer achieves a natural accuracy of 83.79% and robust accuracy of 61.27% under PGD attack while the natural accuracy drops to 73.76% and robust accuracy to 53.80% with RPF on the first block. Thus, we propose to apply random projection to a specific layer in the network. There exist a clear tendency that the robust accuracy decreases if we deploy random projection filters in the deeper layers. For example, the robust accuracy is 63.57% under Auto Attack with RPF on the 2^{nd} layer while 53.39% on the 6^{th} layer and 48.06% on the 14^{th} layer. According to Eq. 5.7, the geometric representation preservation holds if the number of random projection filters N_r is large than the term which is proportional to the total number of filters N. When it goes deeper in ResNet-18, N keeps increasing, which requires larger N_r to guarantee the bound in the deeper layers where the redundant random projection filters could hurt the trade-offs between robustness and natural generalization. Thus, we apply random projection filters to the first layer in our experiments to achieve better trade-offs.

Ratio of Random Projection Filters We then explore how the ratio of random projection filters in the first layer of ResNet-18 influence the adversarial robustness. The results are shown in Figure 5.3 (b). According to Eq. 5.7, the number of random projection filters N_r is required to be sufficient, however, directly setting $N_r \approx N$ could involves redundant noise to the network. For illustration, applying RPF with a ratio of 0.75 can achieve the natural accuracy of 83.79% and robust accuracy of 61.27% under PGD attack. On the contrary, the natural accuracy becomes 73.60 with a RPF ratio of 1.0 due to the redundant random projection filters, and the robust accuracy becomes 58.26% under PGD attack with a RPF ratio of 0.1 due to the insufficient random projection filters. Thus, the empirical observations of the RPF ratio is consistent with the analysis of Theorem 1.

Weight Norm Study According to Eq. 5.7, the requirement of N_r can be further relieved via the reduction of weight norm of convolution filters of that layer besides the random projection, which motivates us to adjust the weight decay of these convolution filters via α in Eq. 5.8. We further provide empirical evidence that the weight norm could play an important role in the our defense scheme through setting different weight decays for the trainable parameters in the first layer of ResNet-18, as shown in Figure 5.3 (c). The traditional

TABLE 5.6. The evaluation results of 5 runs.

Method	Clean	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
Add	81.09	59.51	57.46	80.83	57.64	73.56	62.23
	81.02	59.24	57.84	80.77	57.49	73.61	62.02
	81.49	59.88	57.25	80.90	57.83	73.65	62.10
	80.94	59.23	57.83	81.36	57.86	73.57	62.11
	81.24	59.19	57.61	80.84	57.83	73.44	62.25
Add Avg	81.16	59.41	57.60	80.94	57.73	73.57	62.14
Multi	82.91	61.89	59.77	82.70	59.96	78.49	63.96
	82.76	61.89	59.43	82.77	59.54	78.98	64.03
	83.08	61.77	59.05	82.73	59.36	78.52	63.94
	82.74	61.98	59.34	82.27	59.37	78.70	64.04
	83.16	61.92	59.49	82.80	59.48	78.28	63.78
Multi Avg	82.93	61.89	59.42	82.65	59.54	78.59	63.95
RPF	83.75	62.87	60.75	83.62	60.59	78.96	64.71
	83.48	63.19	60.88	83.63	60.39	79.74	64.72
	83.73	62.87	60.89	83.62	61.94	79.71	64.29
	83.80	61.95	62.12	83.34	61.57	79.31	65.06
	83.79	62.71	61.27	83.60	60.72	79.43	64.38
RPF(Ours) Avg	**83.72**	**62.72**	**61.19**	**83.56**	**61.04**	**79.43**	**64.63**

weight decay is set to 5×10^{-4} for ResNet-18 on CIFAR-10, however, it cannot achieve satisfactory performance, with 60.04% robust accuracy under Auto Attack. On the contrary, the variants with larger weight decays, such as 1×10^{-2} or 1×10^{-1}, can achieve 64.38% and 63.12% respectively, which empirically verify the effectiveness of Eq. 5.8 and the correctness of Theorem 1.

5.3.8 Multiple Runs

Since, the proposed RPF is a randomized defense techniques, we further provide the results of multiple runs of proposed random projection filters as well as additive and multiplicative noise injection with ResNet-18 on CIFAR-10. The results are shown in Table 5.6. For each scenario, we conduct evaluation with 5 times under different seeds. Our proposed RPF consistently achieves the best performance.

TABLE 5.7. Results with ResNet-18 on CIFAR-10.

Setting	Method	Clean	FGSM	PGD	MIFGSM	AA
DenseNet	AT	82.94	59.36	55.32	57.67	51.83
	RPF	**85.19**	**60.90**	**57.00**	**58.92**	**59.91**
SqueezeNet	AT	76.71	51.95	47.29	49.91	42.06
	RPF	**82.66**	**64.59**	**62.99**	**60.83**	**69.06**
Vgg16 BN	AT	79.30	53.87	48.40	51.62	44.17
	RPF	**82.41**	**61.92**	**61.09**	**61.40**	**64.41**
IN	AT	81.05	52.13	42.96	48.50	39.82
	RPF	**84.00**	**56.67**	**49.46**	**52.36**	**52.46**
LN	AT	78.07	52.91	45.57	50.30	41.35
	RPF	**82.38**	**57.42**	**50.73**	**53.80**	**54.12**
Defense	MART	77.35	56.04	52.22	54.65	45.55
	RPF	**82.11**	**62.65**	**60.40**	**60.97**	**64.46**

5.3.9 Evaluation on More Models, Norms, and Defense Techniques.

We apply RPF on different models including densenet121, squeezenet, and vgg. We also include evaluation on different normalizations including instance norm and layer norm (Ba et al. 2016; Ulyanov et al. 2016). Furthermore, we include MART+RPF in our evaluation (Wang et al. 2020b). Our proposed RPF shows consistent improvements in all the scenarios, as shown in Table 5.7.

5.4 Conclusion

In this chapter, we propose to utilize random projection as the noise injection to perform a randomized defense technique against adversarial examples. Through the generalization of Johnson-Lindenstrauss lemma to the scenario where partial convolution filters can replaced by random projection, we theoretically show the correlations between weight norm, the number of random projection filters, and the total number of filters. Based on these observations, we introduce Random Projection Filters as a strong defense scheme. Through sufficient evaluation with various models and datasets, we present the superiority of proposed algorithm to other baselines.

CHAPTER 6

Low-Transferability Normalization Search for Adversarial Robustness

---

6.1 Motivation

Deep Neural Networks (DNNs) have achieved impressive performance in various tasks (Krizhevsky et al. 2012; Ren et al. 2015; Ronneberger et al. 2015). However, it is well known that DNNs are susceptible to maliciously generated adversarial examples (Szegedy et al. 2014; Goodfellow et al. 2014). Through imperceptible perturbations on the model inputs during inference stage, the model is misled to wrong predictions at a high rate. Since then, a wide range of attack techniques have been proposed under different settings and show strong attack capability. For example, attackers have full access to the model architecture and parameters, which forms white-box attacks (Goodfellow et al. 2014; Madry et al. 2018), and attackers have limited query access to the model, which forms black-box attacks (Ilyas et al. 2018; Brendel et al. 2017). Since the high attack success rates of these techniques reveal the high risk of DNNs, defenses against adversarial examples have received increasing attention and adversarial robustness becomes one of the key criteria.

To mitigate this risk, adversarial training is proposed to yield robust models through training on generated adversarial examples (Goodfellow et al. 2014; Madry et al. 2018). Besides training procedure, some regularization and image preprocessing techniques are introduced to improve adversarial robustness (Zhang et al. 2019a; Xie et al. 2017). Recent work note that the architecture and module designs could play important roles in the robustness (Guo et al. 2020; Xie et al. 2019). Hence, we pay more attention to the basic modules in the network which are seldom considered for improving robustness, such as normalization layers. Existing works have discussed the influence of Batch Normalization (BN) and empirically shown

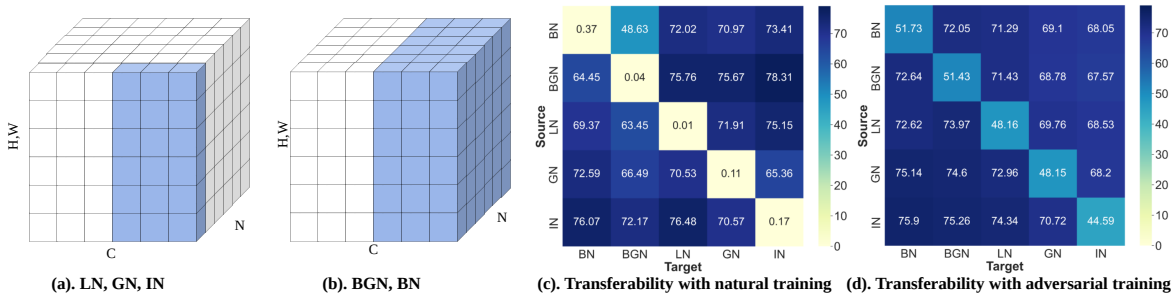


FIGURE 6.1. (a) denotes the normalized dimensions of LN, GN, and IN, while (b) denotes BGN and BN. (c) and (d) denote the adversarial transferability among different types of normalizations.

that BN increases adversarial vulnerability and decreases adversarial transferability (Benz et al. 2021; Galloway et al. 2019). However, the theoretical analysis of this observation is insufficient and how to tackle this pitfall or even utilize this property to defend attacks is unexplored.

In this chapter, we provide the workflow to search normalization layers for adversarial robustness. We first take numerous normalizations into consideration, including Layer Normalization (LN), Group Normalization (GN), Instance Normalization (IN), Batch Normalization (BN), and Batch Group Normalization (BGN) (Ba et al. 2016; Wu and He 2018; Ulyanov et al. 2016; Ioffe and Szegedy 2015; Zhou et al. 2020), as shown in Figure 6.1 (a) and (b). To evaluate the influence of different normalizations on the robustness, we conduct PGD-7 attack (Madry et al. 2018) to both natural and adversarial trained networks with different normalizations on CIFAR-10, as shown in the diagonals of Figure 6.1 (c) and (d). Not surprisingly, BN obtains the best robustness compared to other variants. However, we have an intriguing observation after the transferability evaluations among different normalizations. As illustrated in the heatmaps, the adversarial accuracies in most scenarios are around 70% when fed with transferred adversarial examples, while those under white-box attack are around 50%. This huge gap mainly comes from the adversarial transferability among normalizations. Motivated by this observation, we first explore the relationship between adversarial transferability and normalizations, and show that the gradient similarity and loss smoothness are the key factors of the discrepancy in transferability among different normalizations. Based on the theoretical evidence, we propose to aggregate different types of normalizations to form random space in

the inference phase where the adversarial transferability can be significantly reduced. With designed random space, the inference phase naturally forms the black-box setting for those attackers who have access to model parameters due to the colossal random space. Together with the proposed black-box adversarial training framework, the adversarial robustness is substantially improved with less reduction of natural accuracy. For example, with the same adversarial training setting, the proposed algorithm improves the natural accuracy by 2.45% and the adversarial accuracy by 8.53% with ResNet-18 on CIFAR-10 under PGD^{20} attack. Our contributions can be summarized as:

1) We provide both empirical and theoretical evidence that the upper bound of adversarial transferability is influenced by the types and parameters of normalization layers.

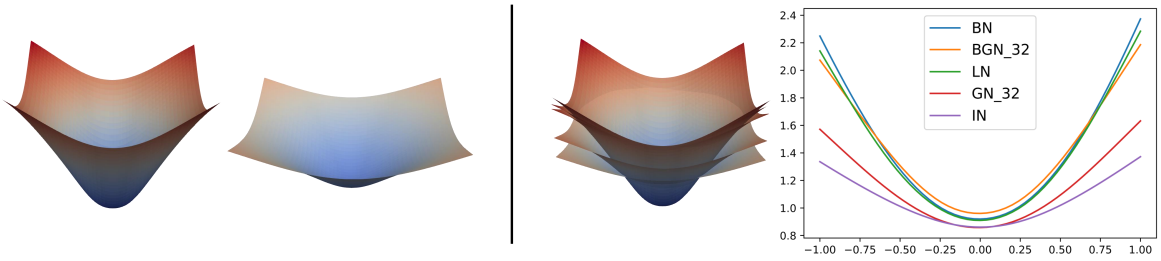
2) We propose a novel Random Normalization Aggregation (RNA) module which replaces the normalization layers to create huge random space with weak adversarial transferability. Together with a natural black-box adversarial training, RNA boosts the defense ability.

3) We conduct extensive experiments to demonstrate the superiority of RNA on different benchmark datasets and networks. Different variants and components are also studied.

6.2 Adversarial Transferability with Different Normalization

In this section, we reveal the connections between adversarial transferability and normalization layers. We first consider a network which is identified with an hypothesis h from a space $\mathcal{H}$. The network h is optimized with the loss function $\mathcal{L}$ on input $\mathcal{X}$ and labels $\mathcal{Y}$. The objective is formulated as

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \underset{x,y \sim \mathcal{X},\mathcal{Y}}{\mathbb{E}} [\mathcal{L}(h(x), y)]. \tag{6.1}$$



(a). BN smoothness (b). IN smoothness (c). Smoothness of different normalization on 3D and 2D

FIGURE 6.2. (a) and (b) denote the smoothness of loss *w.r.t.* input with BN and IN respectively. (c) denotes the smoothness of BN, BGN, LN, GN, and IN with both 3D and 2D plots.

Given a target network h and inputs $\{x, y\}$, the adversarial examples are defined as perturbed input $\tilde{x} = x + \delta$, which makes the network h misclassify through maximizing the classification loss as

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leqslant \epsilon}{\operatorname{argmax}} \mathcal{L}(h(\tilde{x}), y), \tag{6.2}$$

where the perturbation δ is constrained by its l_p-norm. Adversarial transferability denotes an inherent property of $\tilde{\mathcal{X}}$ that these adversarial examples can also boost the classification loss $\mathcal{L}(h'(\tilde{x}), y)$ of other networks as well, where $h' \in \mathcal{H}$. We empirically demonstrate that transferability is influenced by the normalization layers in the network h, as shown in Figure 6.1 (d). For example, taking BN and IN as source models to generate adversarial examples, the adversarial accuracies of LN are 71.29% and 74.34% respectively. We further provide more theoretical analysis of their relationships.

6.2.1 Definition of Normalization Layers

Batch Normalization is known as important basic module in DNNs, which improves the network performance, and a wide variety of variants are introduced where the activations are normalized with different dimensions as well as sizes. To cover most types of normalization, we divide them into two categories. An illustration is shown in Figure 6.1 (a) and (b), where LN, GN, and IN compute the mean and variance for each example with different group sizes during inference, while BN and BGN adopt the pre-calculated mini batch statistics which are

computed by moving average in the training phase. Note that LN and IN are special cases of GN, which takes the minimum or maximum group number. Likewise, BN is a special case of BGN. For simplicity, we use GN and BGN to cover all these normalizations. Considering the activations $y \in \mathbb{R}^{d\times N}$ where N denotes the batch size and d denotes the number of features, the normalized outputs after BGN with group number of $s_{(BGN)}$ and those of GN with group number of $s_{(GN)}$ during inference stage are formulated as

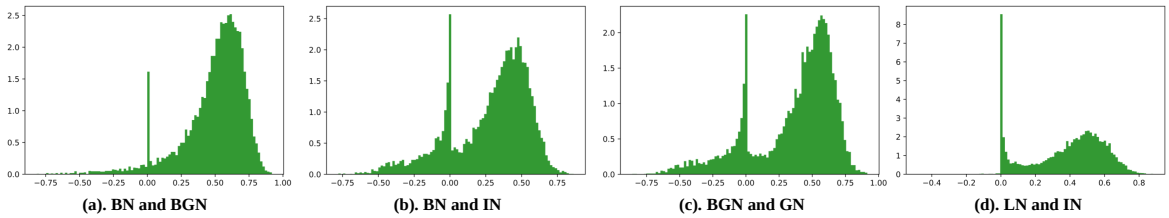
$$
\begin{aligned}
&(\hat{y}^{(k)}_{(BGN)})_i = \frac{(y^{(k)})_i - (\mu_{(BGN)})_i}{(\sigma_{(BGN)})_i}, \;\; (z^{(k)}_{(BGN)})_i = \gamma_{(BGN)} * (\hat{y}^{(k)}_{(BGN)})_i + \beta_{(BGN)}, \text{ for } 1 \le k \le N, \\
&(\hat{y}^{(k)}_{(GN)})_i = \gamma_{(GN)} \frac{(y^{(k)})_i - (\mu^{(k)}_{(GN)})_i}{(\sigma^{(k)}_{(GN)})_i} + \beta_{(GN)}, \;\; (z^{(k)}_{(GN)})_i = \gamma_{(GN)} * (\hat{y}^{(k)}_{(GN)})_i + \beta_{(GN)}, \text{ for } 1 \le k \le N, \\
&\text{where } \;\; (\mu^{(k)}_{(GN)})_i = \frac{1}{G}\sum_{j=1}^{G} (y^{(k)})_{G\lfloor \frac{i}{G} \rfloor + j}, \;\; (\sigma^{(k)}_{(GN)})_i = \sqrt{\frac{1}{G}\sum_{j=1}^{G} ((y^{(k)})_{G\lfloor \frac{i}{G} \rfloor + j} - (\mu^{(k)}_{(GN)})_i)^2},
\end{aligned}
\tag{6.3}
$$

where $G = \lfloor \frac{d}{s_{(GN)}} \rfloor$ denotes the group size of GN, $(\mu_{(BGN)})_i$ and $(\sigma_{(BGN)})_i$ denote the tracked mean and standard deviation of group $\lfloor \frac{i \cdot s_{(BGN)}}{d} \rfloor$.

6.2.2 Variation of Loss Function Smoothness

Existing work on adversarial transferability reveals that the adversarial transferability is mainly influenced by the dimensionality of the space of adversarial examples, since the adversarial subspaces of two networks are more likely to intersect with the growth of this dimensionality. (Goodfellow et al. 2014; Tramèr et al. 2017b). The size of space of adversarial examples can be estimated by the maximum number of orthogonal vectors r_i which are aligned with the gradient $g = \nabla_{\mathcal{X}} \mathcal{L}(h(\mathcal{X}), \mathcal{Y})$. In (Tramèr et al. 2017b), a tight bound is derived as $g^\top r_i \ge \frac{\epsilon \|g\|_2}{\sqrt{k}}$ where k denotes the maximum number of r_i, which implies that the smoothness of loss function is inversely proportional to the adversarial transferability. Thus, we analyze the influence of different normalization layers on the smoothness of loss function, including GN and BGN.

For simplicity, we dismiss the usage of k in the following equations since the computation of both GN and BGN during inference is independent of batch size. We denote the loss with GN



(a). BN and BGN (b). BN and IN (c). BGN and GN (d). LN and IN

FIGURE 6.3. The histograms over the gradient cosine similarity of different normalization layers.

as $\hat{\mathcal{L}}_{gn}$ and the loss with BGN as $\hat{\mathcal{L}}_{bgn}$. Since the mean and variance are computed based on current group for both GN and BGN, we compute the partial derivative of loss *w.r.t.* a group Y_j instead of y_i where $Y_j = y_{[G\lfloor \frac{i}{G} \rfloor : G\lfloor \frac{i}{G} \rfloor + G]}$. Similarly, Z_j denotes the activations of a group after normalization layers. Based on Eq. 6.3, the partial derivative of $\hat{\mathcal{L}}_{gn}$ and $\hat{\mathcal{L}}_{bgn}$ *w.r.t.* Y_j can be given as

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}_{gn}}{\partial Y_j} &= \frac{\gamma_{gn}}{G \cdot \sigma_j^{gn}} (G \cdot \frac{\partial \hat{\mathcal{L}}_{gn}}{\partial Z_j} - 1\langle 1, \frac{\partial \hat{\mathcal{L}}_{gn}}{\partial Z_j} \rangle - \hat{Y}_j \langle \frac{\partial \hat{\mathcal{L}}_{gn}}{\partial Z_j}, \hat{Y}_j \rangle), \\ \frac{\partial \hat{\mathcal{L}}_{bgn}}{\partial Y_j} &= \frac{\gamma_{bgn}}{\sigma_j^{bgn}} \frac{\partial \hat{\mathcal{L}}_{bgn}}{\partial Z_j}, \end{aligned} \tag{6.4}$$

where $\langle , \rangle$ denotes the inner product, σ_j^{gn} denotes the standard deviation of Y_j, and σ_j^{bgn} denotes the tracked standard deviation of Y_j. For simplicity, we denote $\hat{g} = \frac{\partial \hat{\mathcal{L}}}{\partial Y_j}$ and $g = \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}$. Taking the advantage of the fact that the mean of Y_j is zero and its norm is $\sqrt{G}$, the squared norm of the partial derivative of GN and BGN can be derived as

$$\|\hat{g}_{gn}\|^2 = \frac{\gamma_{gn}^2}{(\sigma_j^{gn})^2} (\|g_{gn}\|^2 - \frac{1}{G}\langle 1, g_{gn} \rangle^2 - \frac{1}{G}\langle g_{gn}, \hat{Y}_j \rangle^2), \quad \|\hat{g}_{bgn}\|^2 = \frac{\gamma_{bgn}^2}{(\sigma_j^{bgn})^2} \|g_{bgn}\|^2. \tag{6.5}$$

Besides the smoothness of the loss, we further consider the smoothness of the gradients of the loss for GN and BGN. Following (Santurkar et al. 2018), we compute the "effective" β-smoothness through the quadratic form of Hessian of the loss *w.r.t.* the group activations in the normalized gradient direction, which measures the change of gradients with perturbations in the gradient direction. For simplicity, we denote the hessian *w.r.t.* the layer output as $\hat{H} = \frac{\partial \hat{\mathcal{L}}}{\partial Y_j \partial Y_j}$, the hessian *w.r.t.* the normalization output as $H = \frac{\partial \hat{\mathcal{L}}}{\partial Z_j \partial Z_j}$, the normalized

gradient as $\hat{g}' = \frac{\hat{g}}{\|\hat{g}\|}$ and $g' = \frac{g}{\|g\|}$. For GN and BGN, we have

$$\hat{g}'^{\top}_{gn}\hat{H}_{gn}\hat{g}'_{gn} \leq \frac{\gamma^2_{gn}}{(\sigma^{gn}_j)^2}\left[g'^{\top}_{gn}H_{gn}g'_{gn} - \frac{1}{G\cdot\gamma_{gn}}\langle g_{gn}, \hat{Y}_j\rangle\right], \;\; \hat{g}'^{\top}_{bgn}\hat{H}_{bgn}\hat{g}'_{bgn} \leq \frac{\gamma^2_{bgn}}{(\sigma^{bgn}_j)^2}\left[g'^{\top}_{bgn}H_{bgn}g'_{bgn}\right]. \tag{6.6}$$

6.2.3 Normalization Layers and Adversarial Transferability

The sufficient conditions and the bounds of adversarial transferability between two networks have been discussed in (Yang et al. 2021). We extend this result to the networks with different normalization layers. Since we focus on the influence of different normalization layers, we assume that these networks share the same loss function and weight parameters W, which makes $\frac{\partial\hat{\mathcal{L}}_{gn}}{\partial Z_j} = \frac{\partial\hat{\mathcal{L}}_{bgn}}{\partial Z_j}$ and $\frac{\partial\hat{\mathcal{L}}_{gn}}{\partial Z_j\partial Z_j} = \frac{\partial\hat{\mathcal{L}}_{bgn}}{\partial Z_j\partial Z_j}$. Meanwhile, Eq. 6.5 and Eq. 6.6 can be easily generalized to the input x since $\frac{\partial\hat{\mathcal{L}}}{\partial x} = \frac{\partial\hat{\mathcal{L}}}{\partial\hat{Y}}W$. With this assumption, the connections between normalization layers and adversarial transferability can be established via bounded gradient norm and β-smoothness in Eq. 6.5 and Eq. 6.6 as

THEOREM 2. *Given two networks h_a and h_b with different normalization layers, the adversarial perturbation under white-box attack is δ on x with attack target label $y_{\mathcal{A}}$ and true label $y_{\mathcal{T}}$. Assume h_a and h_b are "effective" β_a and β_b-smooth respectively, the level of adversarial transferability T between networks h_a and h_b within the perturbation ball $\|\delta\|_2 \leq \epsilon$ can be upper bounded by*

$$T \leq \frac{\mathcal{R}_a + \mathcal{R}_b}{\min(\mathcal{L}(x, y_{\mathcal{A}})) - \max(\|\nabla_x\mathcal{L}\|)\epsilon(\sqrt{\frac{1+\bar{S}}{2}} + 1) - \max(\beta_a, \beta_b)\epsilon^2}, \tag{6.7}$$

where T denotes the attack successful rate, $\min(\mathcal{L}(x, y_{\mathcal{A}})) = \min_{x\sim\mathcal{X}}(\mathcal{L}_a(x, y_{\mathcal{A}}), \mathcal{L}_b(x, y'))$, $\mathcal{R}_a$ and $\mathcal{R}_b$ denotes the empirical risks of network h_a and h_b, $\bar{S}$ denotes the upper loss gradient similarity, and $\max(\|\nabla_x\mathcal{L}\|) = \max_{x\sim\mathcal{X}, y\sim\{y_{\mathcal{T}}, y_{\mathcal{A}}\}}(\|\nabla_x\mathcal{L}_a(x, y)\|, \|\nabla_x\mathcal{L}_b(x, y)\|)$. Since the networks share the same loss function and weight parameters, we denote the influence of weight parameters as some constant C_g on gradient norm and C_H on gradient smoothness. The partial derivative and Hessian of loss w.r.t. the normalization output are the same for different normalization, denoted as g and H respectively. The gradient norm, β_a, and β_b in

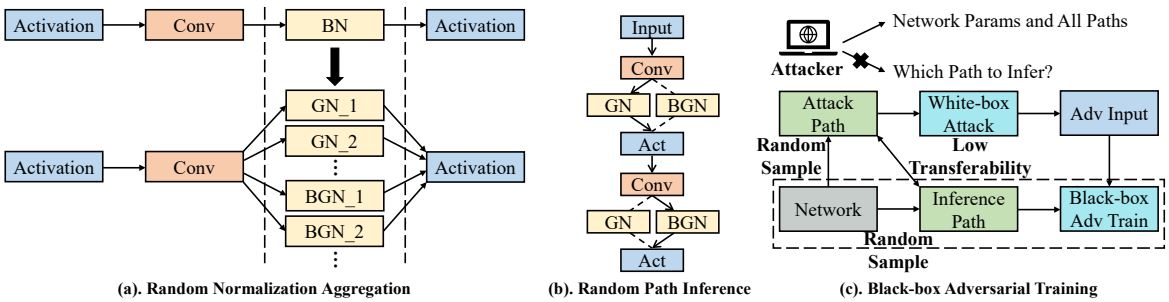


FIGURE 6.4. An illustration of Random Normalization Aggregation and Black-box Adversarial Training.

Eq. 6.7 can be bounded as

$$\begin{aligned}
\|\nabla_x \mathcal{L}\| &\leq C_g \cdot \max\left(\frac{|\gamma_{gn}|}{\sigma_j^{gn}}\sqrt{\|g\|^2 - \frac{1}{G}\langle 1, g\rangle^2 - \frac{1}{G}\langle g, \hat{Y}_j\rangle^2}, \frac{|\gamma_{bgn}|}{\sigma_j^{bgn}}\|g\|\right), \\
\beta_{a,b} &\leq C_H \cdot \max\left(\frac{\gamma_{gn}^2}{(\sigma_j^{gn})^2}\left[g'^\top H g' - \frac{1}{G\cdot\gamma_{gn}}\langle g, \hat{Y}_j\rangle\right], \frac{\gamma_{bgn}^2}{(\sigma_j^{bgn})^2}\left[g'^\top H g'\right]\right).
\end{aligned} \tag{6.8}$$

Combining Eq. 6.7 and 6.8, we observe that the upper bound of adversarial transferability is controlled by the gradient magnitude and gradient smoothness, which is further bounded according to the type and parameters of normalization layers. Specifically, given the same γ and σ for GN and BGN, GN achieves a smaller gradient norm and better gradient smoothness than BGN, which decreases the upper bound of adversarial transferability. Furthermore, the group size G in GN plays an important role in smoothness. With smaller G, the smoothness of GN increases, and thus the upper bound of adversarial transferability decreases. Similar observations can be found in empirical evidence. As shown in Figure 6.2, the loss landscapes of different normalization layers *w.r.t.* input are visualized, which demonstrates that different normalization layers have different smoothness. Furthermore, IN achieves the best performance in smoothness, which corresponds to the observation in Eq. 6.8, since IN has the minimum group size. The attack success rate is relatively low when the source model is IN, as shown in Figure 6.1 (c) and (d), which corresponds to the observation in Theorem 2 that the adversarial transferability decreases when the network is smoother.

6.3 Random Normalization Aggregation

Since the adversarial transferability is strongly correlated with the type of normalization layers, we ask a simple question: *Can we utilize the bounded adversarial transferability among normalization layers to defense against white-box attacks?* In this work, we propose a Random Normalization Aggregation (RNA) module, which replaces the BN layer in the network. As shown in Figure 6.4 (a), the normalization layers becomes a combination of different normalization sampled from GNs and BGNs, where the underline denotes the group number. Specifically, the network maintains different normalization layers while only one normalization is randomly selected for each layer during forwarding, as shown in Figure 6.4 (b). Through incorporating randomization in normalization layers, the network with RNA module can be treated as a "supernet" with multiple paths. Back to white-box defense setting, we assume that the attackers have access to the network parameters. The adversarial examples are generated through backward on a randomly sampled path, and then fed to another randomly sampled path due to RNA module, which makes the entire white-box attack become a "black-box" attack, as illustrated in Figure 6.4 (c). Thus, together with the adversarial transferability study in Section 6.2, it is natural to create a network with random space in normalization layers where the adversarial transferability is significantly constrained. To achieve a strong defense against adversarial attacks, some concerns still remain: (1). The number of paths are required to be extremely large to reduce the probability of sampling the same path with random sampling strategy; (2). The collaboration with traditional adversarial training; (3). The normalization types need to be carefully selected to enforce low adversarial transferability. We discussion these concerns as follows.

Path Increment in Random Space The adversarial transferability among different normalization has been discussed in Figure 6.1 (c) and (d). However, the size of random space also matters for effective defense against attacks. If the attackers can sample the same path during attack and inference phases with a high probability, the adversarial accuracy will decrease tremendously. To tackle this issue, we introduce layer-wise randomization of RNA module, which randomly samples the normalization for each layer in the network. As shown in Figure 6.4 (b), different normalization types are sampled for different layers, which exponentially

Algorithm 3 Random Normalization Aggregation with Black-box Adversarial Training

Input: The training tuple $\{\mathcal{X}, \mathcal{Y}\}$; Path set $\mathcal{P}$; Attack step size η; Attack iterations t; Perturbation size ϵ; Network h with parameters W;
Replace BN layers with RNA modules, and initialize the network.
while not converge **do**
  Sample a batch of data {x, y} from $\{\mathcal{X}, \mathcal{Y}\}$;
  Randomly sample a path p_a from $\mathcal{P}$;
  Initialize adversarial perturbation δ;
  for $i \leftarrow 1$ to t **do**
    $\delta = clip_{\epsilon}[\delta + \eta \cdot \text{sign}(\nabla_x \mathcal{L}(h(x; p_a), y)]$;
  end for
  Randomly sample a path p from $\mathcal{P}$;
  $W = W - \nabla_W \mathcal{L}(h(x + \delta; p), y)$;
end while

increases the number of paths. Given the n normalization types in RNA and L layers in the network, the size of random space becomes n^L, which reduces the probability of sampling the same path during attack and inference phase to $\frac{1}{n^L} << \frac{1}{n}$.

Black-box Adversarial Training It is natural to incorporate RNA module into adversarial training. Consistent with inference phase, we randomly sample a path p_a and conduct white-box attack to generate adversarial examples $\tilde{\mathcal{X}}_{p_a}$. Different from traditional adversarial training which optimizes p_a through feeding $\tilde{\mathcal{X}}_{p_a}$, we feed $\tilde{\mathcal{X}}_{p_a}$ to another randomly sampled path p, which forms a "black-box" adversarial training, as illustrated in Figure 6.4 (c). Eq. 6.1 and Eq. 6.2 can be reformulated as

$$h^* = \underset{h \in \mathcal{H}}{\text{argmin}} \underset{x,y \sim \mathcal{X},\mathcal{Y}; p \sim \mathcal{P}}{\mathbb{E}} [\mathcal{L}(h(\tilde{x}_{p_a}; p), y)], \text{ where } \tilde{x}_{p_a} = \underset{\tilde{x}_{p_a}: \|\tilde{x}_{p_a} - x\|_p \leqslant \epsilon}{\text{argmax}} \mathcal{L}(h(\tilde{x}_{p_a}; p_a), y), \quad (6.9)$$

where $\mathcal{P}$ denotes the space of paths. The training procedure is shown in Algorithm 3.

Normalization Types Selection In RNA module, multiple normalization types are maintained to form random space. According to Theorem 2, the adversarial transferability is bounded by different components, including gradient similarity, empirical risks, gradient magnitude, and gradient smoothness. We first provide empirical evidence that normalization layers from the same category defined in Section 6.2.1 tend to have higher gradient similarity. As shown in Figure 6.3, we visualize the histograms over the cosine similarity of two networks

with different normalization layers. For example, BN and BGN belong to the same category, and their gradient similarity is much higher than that between BN and IN, comparing Figure 6.3 (a) and (b). Since the gradient similarity is proportional to the upper bounds of adversarial transferability in Eq. 6.7, we propose to select normalization from different categories. Thus, RNA module samples the normalization types from both GN and BGN with small group sizes in our experiments, while the evaluation of other combinations is also included.

6.4 Experiments

In this section, we provide sufficient evaluation of RNA module on various models and datasets compared with different normalization layers as well as defense algorithms.

6.4.1 Evaluation Setup

CIFAR-10/100 We first conduct experiments on CIFAR-10/100 (Krizhevsky, Hinton et al. 2009) datasets, which contain 50K training images and 10K testing images with size of 32×32 from 10/100 categories. The networks we use are ResNet-18 (Krizhevsky, Hinton et al. 2009) and WideResNet-32 (WRN) (Zagoruyko and Komodakis 2016). The SGD optimizer with a momentum of 0.9 is used. The weight decay is set to 5×10^{-4}. The initial learning rate is set to 0.1 with a piecewise decay learning rate scheduler. All the baselines are trained with 200 epochs with a batch size of 128. The PGD-10 with $\epsilon = 8/255$ and step size of $2/255$ is adopted in the adversarial training setting. For the RNA module, we utilize BN and IN to form the random space in normalization layers. The experiments are performed on one V100 GPU using Pytorch (Paszke et al. 2019) and Mindspore (Huawei 2020).

ImageNet The effectiveness of proposed RNA is also evaluated on ImageNet (He et al. 2016), which contains 1.2M training images and 50K testing images with size of 224×224 from 1000 categories. The networks we use are ResNet-50 (Krizhevsky, Hinton et al. 2009). The SGD optimizer with a momentum of 0.9 is used. The weight decay is set to 1×10^{-4}. The initial learning rate is set to 0.02 with a cosine learning rate scheduler. We load a pretrained ResNet-50 and then adversarailly train the network for 60 epochs with a batch

TABLE 6.1. The adversarial robustness evaluation of adversarial-trained networks on CIFAR-10.

Model	Method	Natural	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
ResNet-18	BN	81.84	56.70	52.16	78.46	54.96	0.35	47.69
	BGN32	77.28	54.60	50.71	73.57	53.05	0.39	46.12
	IN	77.07	50.07	42.76	72.51	47.63	2.27	38.30
	GN32	76.69	52.60	45.66	72.92	50.26	0.72	41.83
	LN	79.81	53.88	45.44	75.51	50.72	1.13	41.48
	RNA(Ours)	**84.29**	**63.10**	**60.69**	**84.45**	**60.70**	**76.73**	**65.61**
WideResNet	BN	85.27	60.65	55.06	82.24	58.47	0.40	52.24
	BGN32	83.70	59.66	54.96	80.25	57.85	0.37	51.38
	IN	84.11	58.14	50.37	80.13	55.37	1.62	46.81
	GN32	83.45	58.95	51.94	79.55	56.70	1.37	47.99
	LN	83.24	57.80	49.74	79.39	54.68	0.95	46.44
	RNA(Ours)	**86.46**	**65.73**	**63.34**	**85.68**	**62.84**	**78.18**	**67.88**

size of 512. The PGD-2 with $\epsilon = 4/255$ is adopted in the adversarial training setting. For the RNA module, we utilize BGNs and GNs with group size of 1 and 2 to form the random space. The experiments are performed on eight V100 GPUs.

Baselines and Attacks Our proposed RNA modules replace the normalization layers in the network. Thus, various normalization layers are involved for comparison, including BN, IN, LN, GN, and BGN (Ba et al. 2016; Wu and He 2018; Ulyanov et al. 2016; Ioffe and Szegedy 2015; Zhou et al. 2020). On CIFAR-10/100, we evaluate the robustness of all the baselines under different strong attacks from TorchAttacks (Kim 2020). For Fast Gradient Sign Method (FGSM) (Szegedy et al. 2014), the perturbation size ϵ is set to $8/255$. For Projected Gradient Descent (PGD) (Madry et al. 2018), ϵ is set to $8/255$ with step size of $2/255$, and the steps are set to 20. For CW attack (Carlini and Wagner 2017), the steps are set to 1000 with learning rate of 0.01. For Momentum variant of Iterative Fast Gradient Sign Method (MIFGSM) (Dong et al. 2017), ϵ is set to $8/255$ with a step size of $2/255$, and the steps are set to 5 with decay of 1.0. For DeepFool (Moosavi-Dezfooli et al. 2016), the steps are set to 50 with overshoot of 0.02. For Auto Attack (Croce and Hein 2020), ϵ is set to $8/255$. On ImageNet, we evaluate the robustness of under PGD attacks with ϵ of $4/255$ and steps of 50.

TABLE 6.2. The adversarial robustness evaluation of adversarial-trained networks on CIFAR-100.

Model	Method	Natural	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
ResNet-18	BN	55.81	31.33	28.71	50.94	30.26	0.79	24.48
	BGN32	53.16	30.11	27.75	48.74	29.11	0.54	23.29
	IN	52.92	27.56	23.16	47.70	25.91	2.86	19.33
	GN32	51.00	28.32	25.10	45.82	27.10	0.79	21.00
	LN	48.82	27.05	23.83	44.07	26.93	0.80	19.97
	RNA(Ours)	**56.79**	**36.76**	**35.55**	**56.86**	**34.00**	**51.77**	**42.12**
WideResNet	BN	60.11	35.40	31.69	57.11	34.14	0.23	28.36
	BGN32	58.54	34.13	30.97	54.08	33.01	0.46	26.92
	IN	56.71	31.96	28.09	51.98	30.33	1.83	24.25
	GN32	59.08	33.56	29.94	53.89	32.30	1.12	25.78
	LN	57.09	32.92	29.75	52.19	31.73	0.79	25.71
	RNA(Ours)	**60.57**	**37.87**	**36.04**	**60.21**	**35.58**	**55.16**	**42.43**

6.4.2 Results for Robustness

Main Results We first evaluate the performance of RNA on CIFAR-10 and CIFAR-100 under different types of attacks. The detailed results are shown in Table 6.1 and 6.2. Popular normalization layers show similar performance on robustness. However, with a random space of different normalization layers, the robustness is significantly improved. Comparing RNA with other baselines, RNA consistently achieves the best performance under all attacks, and show strong superiority. For example, RNA with ResNet-18 achieves 65.61% under Auto Attack on CIFAR-10, which is 17.92% higher than BN. Similarly, RNA with WRN achieves 55.16% under DeepFool attacks on CIFAR-100, which is 53.33% higher than IN. The boosted adversarial accuracy shows strong empirical evidence that the constrained adversarial transferability in random space can provide satisfactory defense capability. Furthermore, with proposed black-box adversarial training, RNA achieves better natural accuracy. For example, RNA with WRN achieves 86.46% on CIFAR-10, which improves BN by 1.19%. We mainly attribute this improvement to the fact that the generated adversarial examples can be treated as a "weaker" attack example to other paths during optimization, which naturally achieves better trade-offs between natural and adversarial accuracy.

Stronger PGD Attacks We further evaluate the defense capability of RNA under stronger PGD attacks, which enhance the number of iterations and enlarge the perturbation size. The comparison with other baselines are shown in Figure 6.5 (a) and (b). Comparing RNA with other baselines, RNA achieves stable robustness under different attack iterations, such as 60.70% on PGD^{10} and 59.94% on PGD^{100}. Meanwhile, the PGD accuracy of RNA is much higher than all other baselines. For example, RNA improves BN baselines by a margin of 7.93%. In terms of larger perturbation size, RNA achieves the best robustness in all the scenarios, and the lowest decrement among all the methods. Specifically, RNA achieves 80.06% with ϵ of $2/255$ and 24.90% with ϵ of $20/255$, whose gap is 55.16%. For comparison, the gap of BN is 64.68% and GN is 58.40%.

Comparison with SOTA Defense Methods To demonstrate the superiority of RNA, we include several SOTA defense algorithms for comparison. RobustWRN (Huang et al. 2021) explores the importance of network width and depth on robustness. AWP (Wu et al. 2020) proposes to regularize the flatness of weight loss landscape to achieve robustness. RobNet (Guo et al. 2020) introduce a NAS framework for robustness. RPI+RPT (Fu et al. 2021) utilizes randomized precision for adversarial defense. SAT (Xie et al. 2020b) proposes to replace ReLU with its smooth approximations, which exhibits robustness. The results are shown in Table 6.3. We use WRN on CIFAR-10 and ResNet-50 on ImageNet. All the baselines are evaluated under the attack of PGD^{20} and AutoAttack (AA) on CIFAR-10 and PGD^{50} on ImageNet. Our proposed RNA module achieves the best performance in all the scenarios. On CIFAR-10, RNA achieves 67.88% under AutoAttack, with 13.84% improvement compared with AWP. On ImageNet, RNA achieves 54.61% under PGD^{50}, with 12.31% improvement compared with SAT. Note that RNA replaces the normalization layer, which is orthogonal to other defense techniques. Similarly, the adversarial training in our setting can be replaced by other advanced training strategies to achieve potentially better performance.

Adversarial Transferability in Random Space For a better illustration of the adversarial transferability in the random space built from RNA, we conduct the adversarial transferability study of ResNet-18 applied with RNA on CIFAR-10. We first define the path difference as the

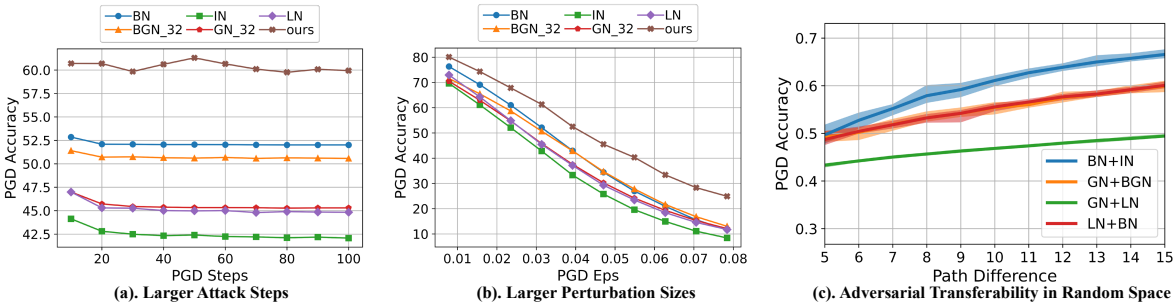


FIGURE 6.5. The evaluation of robustness under PGD attacks settings. (a) denotes larger attack iterations, and (b) denotes larger perturbation sizes. The adversarial transferability in random space is evaluated through sampling paths with different levels of diversity in (c).

TABLE 6.3. Comparison with defense algorithms.

Method	CIFAR-10		ImageNet
	PGD^{20}	AA	PGD^{50}
RobustWRN (Huang et al. 2021)	59.13	52.48	31.14
AWP (Wu et al. 2020)	58.14	54.04	-
RobNet (Guo et al. 2020)	52.74	-	37.15
RPI+RPT (Fu et al. 2021)	53.96	53.30	42.72
SAT (Xie et al. 2020b)	56.01	51.83	42.30
RNA(Ours)	**63.34**	**67.88**	**54.61**

TABLE 6.4. Robustness evaluation of random space built from different normalization combinations under different attacks.

Normalization	PGD^{20}	DeepFool	AutoAttack
BN	52.16	0.35	47.69
GN+BGN	55.40	70.26	58.90
GN+LN	46.67	62.85	47.96
LN+BN	55.67	68.14	58.73
IN+BN	**60.69**	**76.73**	**65.61**

number of different normalization layers between attack and inference paths. For example, a path difference of 7 denotes two paths selecting different normalization layers in 7 layers during forwarding. For each path difference, we then randomly sample 10 path pairs for transferability evaluation. We also include different normalization combinations in RNA for comparison. The results are shown in Figure 6.5 (c), which illustrates the PGD accuracy under transferred attacks between path pairs along increasing path differences. The filled

areas denote the maximum and minimum PGD accuracy. It is obvious that the combination of BN and IN achieves the best performance, which also corresponds to the analysis in Theorem 2. With random sampling strategy, the path difference is always around 10 since the number of normalization layers is 20 in this network. For example, the combination of BN and IN achieves an average PGD accuracy of 61.09% when the path difference becomes 10. Compared with the baseline BN which achieves 52.16% PGD accuracy, this lower adversarial transferability in our random space brings a strong defense capability against adversarial attacks.

6.4.3 Comparison with RPI+RPT

To demonstrate the necessity of reducing the adversarial transferability in random space and the superiority of proposed black-box adversarial training, we provide more detailed comparison with RPI+RPT (Fu et al. 2021). Note that we strictly follow the same training recipe of RPI+RPT (Fu et al. 2021) in this section. We train all the models for 160 epochs with PGD-7 adversarial training, which is different from the one introduced in Section 5.1. For a fair comparison, we consider two different attack settings.

We first consider the normal attack setting where the number of attack iterations are fixed for different examples. Specifically, PGD^{20} denotes that the number of steps for PGD is 20 for all the examples. We take ResNet-18 and WideResNet32 as the models for evaluation. For the results of RPI+RPT, we use the official implementation to train the ResNet-18, and we load the pretrained models from official implementation for WideResNet32. Similar to main results in the work, we consider a wide range of attacks to evaluate the robustness, including FGSM, PGD^{20}, CW, MIFGSM, DeepFool and AutoAttack. The detailed results are shown in Table 6.5 and 6.6. Our proposed RNA achieves better performance under different attacks with large margins compared to RPI+RPT.

Besides the normal attack setting, we also consider a relatively weak attack setting evaluated in (Fu et al. 2021) where the attackers stop the iteration of attacks when the network misclassifies the perturbed example. Thus, the number of attack iterations can be different for

TABLE 6.5. The adversarial robustness evaluation with normal attacks settings on CIFAR-10.

Model	Method	Natural	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
ResNet-18	RPI+RPT	82.84	58.31	52.18	80.02	56.09	27.81	49.70
	RNA(Ours)	**85.33**	**63.88**	**59.79**	**84.42**	**61.05**	**76.83**	**66.35**
WideResNet	RPI+RPT	81.59	57.95	53.96	79.05	56.32	27.61	53.30
	RNA(Ours)	**86.22**	**64.85**	**60.49**	**85.87**	**61.21**	**57.21**	**62.91**

TABLE 6.6. The adversarial robustness evaluation with normal attacks settings on CIFAR-100.

Model	Method	Natural	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
ResNet-18	RPI+RPT	56.72	33.04	29.98	52.87	31.76	20.07	28.14
	RNA(Ours)	**57.62**	**36.62**	**35.51**	**57.14**	**35.84**	**53.12**	**42.41**
WideResNet	RPI+RPT	60.04	35.78	32.46	56.74	34.29	21.74	31.17
	RNA(Ours)	**61.36**	**37.76**	**35.98**	**60.80**	**35.63**	**54.48**	**41.23**

TABLE 6.7. The adversarial robustness evaluation under adaptive attack setting on CIFAR-10/100.

Model	Method	CIFAR-10		CIFAR-100	
		Natural	PGD^{20}	Natural	PGD^{20}
ResNet-18	RPI+RPT	82.64	65.77	56.97	41.75
	RNA(Ours)	**85.33**	**78.58**	**57.62**	**51.62**
WideResNet	RPI+RPT	81.52	66.75	58.41	40.45
	RNA(Ours)	**86.22**	**78.33**	**61.36**	**53.55**

different examples. Specifically, PGD^{20} here denotes the maximum attack iterations for all the examples while the actual attack iterations are always smaller than 20. Under this attack, the adversarial accuracy is much higher due to the adversarial transferability. The comparison under this adaptive attack setting is provided in Table 6.7. Similarly, we include ResNet-18 and WideResNet32 for comparison on CIFAR-10/100. Our proposed RNA achieves better performance with large margins under a relatively weak adaptive PGD^{20} attack.

6.4.4 Ablation Study

Different Normalization Combinations We first provide more quantitative results of different combinations of normalization layers in RNA module. We conduct comparison with ResNet-18 on CIFAR-10. As shown in Table 6.4, we evaluate the performance under PGD^{20},

TABLE 6.8. Ablation studies of RNA with ResNet-18 on CIFAR-10, including random space designing, adversarial training, and layer-based constraint.

Random Space	ADV Train	Layer-free	Natural	FGSM	PGD^{20}	CW	MIFGSM	DeepFool	AutoAttack
BN+IN+GN+LN	WhiteBox	✗	10.45	-	-	-	-	-	-
GN[1-64]	WhiteBox	✗	75.50	53.62	49.57	74.63	51.81	47.20	50.57
GN[1-64]	BlackBox	✗	77.54	55.06	51.32	76.55	52.44	58.72	52.59
BGN+GN[1-64]	BlackBox	✗	73.81	53.98	52.98	73.19	53.07	63.78	56.86
BGN+GN[1-64]	BlackBox	✓	61.72	46.73	46.42	61.89	45.69	56.82	51.26
BGN+GN[1-16]	BlackBox	✓	70.99	51.66	49.74	70.48	51.58	63.63	55.98
BGN+GN[1-4]	BlackBox	✓	78.26	58.91	56.29	77.74	56.31	70.83	61.08
BGN+GN[1-1]	BlackBox	✓	**84.29**	**63.10**	**60.69**	**84.45**	**60.70**	**76.73**	**65.61**

DeepFool and Auto Attack, and the combination of IN and BN achieves the best robustness in all the scenarios. Consistent with the observation in Theorem 2, the combination of LN and BN has slightly worse performance than that of IN and BN, since IN is a smoother normalization than LN. Similarly, the combination of GN and LN achieves the worst performance, since LN is a special case of GN so that they have high gradient similarity in our empirical observation, as discussed in Figure 6.3. Thus, utilizing different normalization layers with smaller group size from GN and BGN, RNA module can form random space with low adversarial transferability to better improve the defense ability.

Effectiveness of Different Components We next demonstrate the effectiveness of each component introduced in RNA module. The detailed results are shown in Table 6.8 where [.] denotes the range of group size. Besides the normalization combinations, we include more discussion of the random space designing. To form a random space in normalization layers, it is natural to consider a combination of BN, IN, GN, and LN, however, it is difficult to optimize, as shown in the first row of Table 6.8. With the normalization layers selected from GNs, the optimization becomes stable, however, the robustness is not competitive, as shown in the second row. The involvement of black-box adversarial training significantly improves the robustness, as shown in the third row. Through expanding the random space with BGNs, the defense capability is slightly improved due to the doubled number of paths. However, the size of random space is still limited. After removing the layer-based constraint, the number of paths exponentially increases, the size of random space for each layer can be reduced for better trade-offs, as shown in the last 4 rows. Comparing BGN+GN[1-1] with GN[1-64], a better random space designing with an appropriate adversarial training strategy can achieve

TABLE 6.9. Stability evaluation of adversarial robustness with RNA.

Method	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
RNA(Ours)	59.79	60.10	59.98	59.97	60.46	59.15	59.86	59.66	59.66	60.72

an improvement of 15.04% under Auto Attack, which demonstrates the necessity of these components.

6.4.5 Stability of Robustness

During the inference phase, we randomly sample paths from the random space we build so that the adversarial robustness could vary according to the sampled paths. To evaluate the stability of robustness with RNA module, we rerun the evaluation for 10 times with ResNet-18 on CIFAR-10. The results are shown in Table 6.9. The maximum accuracy is 60.72% and the minimum accuracy is 59.15%. The average accuracy of 10 runs is 59.94% and the variance is 0.19. Thus, although the performance depends on the randomly sampled paths, the stability of robustness is verified in empirical evaluations since the entire random space is built with weak adversarial transferability.

6.5 Conclusions

In this chapter, we explore the importance of normalization layers in adversarial robustness where the transferability among different normalization layers can be utilized to boost the defense capability. A Random Normalization Aggregation (RNA) module is proposed to form random space with low adversarial transferability for defense against adversarial attacks. We provide sufficient empirical evidence and theoretical analysis to reveal the connections between adversarial transferability and normalization types, which guides the random space designing. With the involvement of black-box adversarial training strategy and the relaxation of layer-based constraint, the robustness provided by RNA module is significantly strengthened. We demonstrate the superiority of RNA module via comprehensive experiments across different network architectures, attack settings, and benchmark datasets. Our work can provide valuable insights into the network module design for robustness.

CHAPTER 7

Conclusion and Future Work

Adversarial robustness is one of the essential components to guarantee the trustworthiness of DNNs in real-world applications. To tackle adversarial robustness, the majority of existing work focus on the adversarial training strategies, which involve adversarial examples into training phase to improve the adversarial robustness under white-box attacks. Although adversarial training is regarded as one of the most effective defense techniques against attacks, it suffers from different concerns and the potentials of DNN defense capability have not been fully explored in terms of various DNN designs. These research gaps are summarized as follows: (1) influence of neural architecture on adversarial robustness; (2) vulnerability of CNNs without adversarial training; (3) trade-offs between natural and robust accuracy in randomized defense; (4) potential of normalization layers in defense scheme. In this thesis, we introduce four different algorithms to tackle the aforementioned issues respectively from the perspective of DNN designs. For the first potential, we propose to establish the connections between adversarial robustness and neural architecture. Through formulating the Lipschitz constant in NAS framework, we introduce a novel searching algorithm called Robust Neural Architecture Search with Confidence Learning (RACL) to discover optimal architectures against attacks. For the second issue, we investigate the vulnerability from the perspective of basic operations in DNNs. Instead of utilizing cross correlation as similarity measurement in CNNs, we propose to utilize ℓ_1-norm distance to constrain the perturbation magnitude in feature space. We introduce a novel strategy called Adaptive Weight Normalization with Robust Inference (AWN-R) to automatically eliminate perturbation without adversarial training. For the third issue, we explore the trade-offs between natural and robust performance via random projection. Motivated by the distance preservation property of random projection, we further extend the correctness of distance preservation to the scenario where partial CNN

filters are replaced by random projection filters. From which, we introduce a novel algorithm called Random Projection Filters (RPF), which achieves better trade-offs between natural and robust accuracy. For the last potential, we provide theoretical evidence that different normalization layers influence the adversarial transferability. Based on these observations, we introduce a simple yet efficient defense scheme, called Random Normalization Aggregation (RNA), which aggregates different types of normalization layers to construct random space with low adversarial transferability to perform strong defense capability. For all the introduced algorithm, we provide sufficient empirical or theoretical analysis, and conduct extensive experiments on popular benchmarks.

Despite the promising results in this thesis, the level of adversarial robustness remains to be far away from the requirements for actual deployment and the potential of DNN search and design in adversarial robustness has not been fully explored yet. Firstly, it is difficult for the studied adversarial robustness to be generalized to all potential attacks, such as different attack types (ℓ_1-norm, ℓ_2-norm, etc.), different perturbation sizes, and different settings (white-box, data poisoning, etc.). Thus, the universal adversarial robustness against all potential attacks can be valuable and challenging. Secondly, there exist strong connections among different components in trustworthiness, such as adversarial robustness, calibration, uncertainty, etc., however, there is no clear definition or benchmark to unify and evaluate the trustworthiness of DNNs. Thus, a one-for-all benchmark of trustworthiness could be essential in this field. Lastly, the trade-offs among basic requirements of real-world applications could be important in exploring adversarial robustness, such as natural performance, network efficiency, etc.. I believe that all the algorithms and results introduced in this thesis can provide insights to motivate further research of adversarial robustness via DNN search and design to fulfill aforementioned research gaps.

APPENDIX A

Appendix for Chapter 5

A1 Proof of Theorem 1

In Section 5.2.2, we explore the trade-offs between natural and robust performance via investigating the distance preservation property in the proposed RPF. We show that the distance preservation property can be achieved with a control on the weight norm. We provide the proof of Theorem 1 as follows:

PROOF. We consider a single filter in convolution layer $F \in \mathbb{R}^{r\times r\times d}$ with mean of μ and variance of $\sigma^2 = \frac{1}{r^2}$ and the input $x, y \sim \mathcal{N}(\beta, \gamma^2)$. For simplicity. We denote $k = r \times r \times d$ and $\mathbb{Z}_n$ as the set of $\{0, \ldots, n-1\}$. We first prove the following simple results: Let $u, v \in \mathbb{R}^{r\times r\times d}$ and $Z_1 = u^T F, Z_2 = v^T F$, then we have

$$\begin{aligned} &\mathbb{E}[FF^T] = cov(F) + E[F]E[F]^T = \sigma^2 I + \mu^2, \\ &\mathbb{E}[Z_1 \cdot Z_2] = u^T \mathbb{E}[FF^T]v = \mu^2 \cdot \sum u \cdot \sum v + \sigma^2 \langle u, v\rangle, \end{aligned} \tag{A.1}$$

where I denotes identity matrix. Now we replace u and v with $[x]^r_{i,j}$ and $[y]^r_{i,j}$ respectively. Given the fact that $\langle x, y\rangle = \frac{1}{r^2}\sum_{i,j\in\mathbb{Z}_n}\langle [x]^r_{i,j}, [y]^r_{i,j}\rangle$, the expectation of the dot product of two filter output can be written as

$$\begin{aligned} \mathbb{E}[\langle F * x, F * y\rangle] &= \sum_{i,j\in\mathbb{Z}_n} \mathbb{E}[\langle F, [x]^r_{i,j}\rangle \cdot \langle F, [y]^r_{i,j}\rangle] \\ &= \sum_{i,j\in\mathbb{Z}_n} \mu^2 \sum^{k} [x]^r_{i,j} \cdot \sum^{k} [y]^r_{i,j} + \sigma^2 \langle [x]^r_{i,j}, [y]^r_{i,j}\rangle \\ &= \langle x, y\rangle + \sum_{i,j\in\mathbb{Z}_n} \mu^2 \cdot k^2 \cdot \beta^2 \end{aligned} \tag{A.2}$$

Similarly, we consider a single random projection filter $F \in \mathbb{R}^{r\times r\times d}$ with zero mean and variance of $\frac{1}{r^2}$.

$$\mathbb{E}[\langle F * x, F * y\rangle] = \sum_{i,j\in\mathbb{Z}_n} \mathbb{E}[\langle F, [x]^r_{i,j}\rangle \cdot \langle F, [y]^r_{i,j}\rangle] = \sum_{i,j\in\mathbb{Z}_n} \frac{1}{r^2}\langle [x]^r_{i,j}, [y]^r_{i,j}\rangle = \langle x, y\rangle \quad \text{(A.3)}$$

For simplicity, we denote $X_{ijl} = \langle F_l, [x]^r_{ij}\rangle$ and $Y_{ijl} = \langle F_l, [y]^r_{ij}\rangle$. Now we consider all the filters including random projection filters $F_1, \ldots, F_{N_r}$ and convolutional filters $F_{N_{r+1}}, \ldots, F_N$. The probability that the absolute difference between the inputs and outputs is large than ϵ can be derived as

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{1}{N}\sum_{l=1}^{N}\langle F_l * x, F_l * y\rangle - \langle x, y\rangle\right| \geq \epsilon\right)\\
=&\mathbb{P}\left(\left|\frac{1}{N}\left(\sum_{l=1}^{N}(\langle F_l * x, F_l * y\rangle - \mathbb{E}[\langle F_l * x, F_l * y\rangle]) + (\sum_{l=1}^{N-N_r}\sum_{i,j\in\mathbb{Z}_n}\mu_l^2 k^2\beta^2)\right)\right| \geq \epsilon\right)\\
\leq&\mathbb{P}\left(\left|\frac{1}{N}\sum_{l=1}^{N}(\langle F_l * x, F_l * y\rangle - \mathbb{E}[\langle F_l * x, F_l * y\rangle])\right| + \left|\frac{1}{N}\sum_{l=1}^{N-N_r}\sum_{i,j\in\mathbb{Z}_n}\mu_l^2 k^2\beta^2\right| \geq \epsilon\right)\\
=&\mathbb{P}\left(\left|\frac{1}{N}\sum_{l\in[N];i,j\in\mathbb{Z}_n}\langle X_{ijl}, Y_{ijl}\rangle - \mathbb{E}[\langle X_{ijl}, Y_{ijl}\rangle]\right| \geq \epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 k^2\right)\\
\leq&\mathbb{P}\left(\frac{1}{N}\sum_{i,j\in\mathbb{Z}_n}\left|\sum_{l\in[N]}\langle X_{ijl}, Y_{ijl}\rangle - \mathbb{E}[\langle X_{ijl}, Y_{ijl}\rangle]\right| \geq \epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 k^2\right)\\
\leq&\mathbb{P}\left(\frac{n^2}{N}\max_{i,j\in\mathbb{Z}_n}\left|\sum_{l\in[N]}\langle X_{ijl}, Y_{ijl}\rangle - \mathbb{E}[\langle X_{ijl}, Y_{ijl}\rangle]\right| \geq \epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 k^2\right)\\
\leq&\sum_{i,j\in\mathbb{Z}_n}\mathbb{P}\left(\frac{n^2}{N}\left|\sum_{l\in[N]}\langle X_{ijl}, Y_{ijl}\rangle - \mathbb{E}[\langle X_{ijl}, Y_{ijl}\rangle]\right| \geq \epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 k^2\right)
\end{aligned} \quad \text{(A.4)}$$

For the convolutional filters, $X_{ijl} = \langle F_l, [x]^r_{ij}\rangle$ and $Y_{ijl} = \langle F_l, [y]^r_{ij}\rangle$ are linear combination of i.i.d. Gaussian RVs since $x, y \sim \mathcal{N}(\beta, \gamma^2)$. Thus, X_{ijl} and Y_{ijl} are sub-Gaussian RVs with mean of $\beta k\mu$ and variance of $\gamma^2\|F_l\|^2$. The sub-gaussian norm of $\langle F_l, [x]^r_{ij}\rangle$ can be computed

as

$$\Big\|\langle F_l, [x]_{ij}^r\rangle\Big\|_{\psi_2} = \Big\|X_{ijl}\Big\|_{\psi_2} = \Big\|\beta k\mu + \gamma^2\|F_l\|^2 z\Big\|_{\psi_2} \leq \Big\|\beta k\mu\Big\|_{\psi_2} + \Big\|\gamma\|F_l\|z\Big\|_{\psi_2} \leq k\beta\mu + C_0 W\gamma \tag{A.5}$$

and $\|\langle F_l, [y]_{ij}^r\rangle\|_{\psi_2} = \|Y_{ijl}\|_{\psi_2} \leq k\beta\mu + C_0 W\gamma$ where C_0 denotes an absolute constant. According to the product of sub-Gaussians property and centering property (Vershynin 2018), we have $\|XY\|_{\psi_1} \leq \|X\|_{\psi_2}\|Y\|_{\psi_2}$ and $\|X - \mathbb{E}[X]\|_{\psi_1} \leq C\|X\|_{\psi_1}$. Thus, we have

$$\|X_{ijl}Y_{ijl} - \mathbb{E}[X_{ijl}Y_{ijl}]\|_{\psi_1} \leq (k\beta\mu + C_0 W\gamma)^2. \tag{A.6}$$

Similarly, for the random projection filters, we have $\|\langle F_l, [x]_{ij}^r\rangle\|_{\psi_2} = \|X_{ijl}\|_{\psi_2} \leq \frac{C_0 R}{r}$ and $\|\langle F_l, [y]_{ij}^r\rangle\|_{\psi_2} = \|Y_{ijl}\|_{\psi_2} \leq \frac{C_0 R}{r}$. According to the product of sub-Gaussians property and centering property, we have

$$\|X_{ijl}Y_{ijl} - \mathbb{E}[X_{ijl}Y_{ijl}]\|_{\psi_1} \leq C_0^2\nu^2 R^2, \tag{A.7}$$

According to Bernstein's inequality for sub-exponentials, let $X_1, \ldots, X_N$ be independent zero-mean sub-exponential RVs. Then, for all $t \geq 0$

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} X_i\right| \geq t\right) \leq 2exp\Big\{-min\Big\{\frac{t^2}{K^2}, \frac{t}{K}\Big\} \cdot c \cdot N\Big\}, \tag{A.8}$$

where $K = \max_i \|X_i\|_{\psi_1}$ and $c > 0$ is an absolute constant.

Together with results in Eq. A.6 and Eq. A.7, the probability in Eq. A.4 can be bounded as

$$\begin{aligned}
&\sum_{i,j\in\mathbb{Z}_n} \mathbb{P}\left(\frac{n^2}{N}\left|\sum_{l\in[N]}\langle X_{ijl}, Y_{ijl}\rangle - \mathbb{E}[\langle X_{ijl}, Y_{ijl}\rangle]\right| \geq \epsilon - \frac{N - N_r}{N}\mu^2\beta^2 n^2 r^4 d^2\right) \\
&\leq 2n^2 exp\Big\{-min\Big\{\frac{(\epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 r^4 d^2)^2}{n^4 max\{C_0^4\nu^4 R^4, (k\beta\mu + C_0 W\gamma)^4\}}, \frac{\epsilon - \frac{N-N_r}{N}\mu^2\beta^2 n^2 r^4 d^2}{n^2 max\{C_0^2\nu^2 R^2, (k\beta\mu + C_0 W\gamma)^2\}}\Big\} \cdot c \cdot N\Big\}
\end{aligned} \tag{A.9}$$

We denote $K = n^2 max\{C_0^2\nu^2R^2, (k\beta\mu + C_0W\gamma)^2\}$ and $D = \mu^2\beta^2n^2r^4d^2$. If $\frac{\epsilon - \frac{N-N_r}{N}D}{K} \leq \frac{(\epsilon - \frac{N-N_r}{N}D)^2}{K^2}$, we have

$$\begin{aligned} \delta &> 2cn^2 exp\Big\{ - \frac{\epsilon N - D(N-N_r)}{K} \Big\} \\ log\frac{\delta}{2cn^2} &> -\frac{(\epsilon - D)N + DN_r}{K} \\ Klog\frac{2cn^2}{\delta} &< (\epsilon - D)N + DN_r \\ N_r &> \frac{(D-\epsilon)N + Klog\frac{2cn^2}{\delta}}{D} \end{aligned} \tag{A.10}$$

Similarly, if $\frac{\epsilon - \frac{N-N_r}{N}D}{K} > \frac{(\epsilon - \frac{N-N_r}{N}D)^2}{K^2}$, we have

$$N_r > \frac{(D-\epsilon)N + NK\sqrt{log\frac{2cNn^2}{\delta}}}{D} \tag{A.11}$$

□

APPENDIX B

Appendix for Chapter 6

B1 Proof of Theorem 2

In section 6.2.3, we provide theoretical analysis of the connections between adversarial transferability and normalization layers. We show that the adversarial transferability is controlled by the smoothness and the group size of normalization layers controls the smoothness. In this section, we provide the proof of Theorem 2 as follows:

PROOF. Following (Yang et al. 2021), we further extend the upper bound of adversarial transferability to the network with different normalization layers. To establish the connections between normalization types and adversarial transferability, we first provide some useful simple but useful facts of different normalization layers, including Group Norm (GN) and Batch Group Norm (BGN).

During the inference stage, the computation of both GN and BGN are independent of batch size. Thus, we dismiss the discussion of batch. We first consider the network with GN. Given the activations $y \in \mathbb{R}^d$ where d denotes the number of features, the normalized outputs after GN with group number of g and during inference stage are formulated as

$$\begin{aligned} &\hat{y}_i = \gamma \frac{y_i - \mu(i)}{\sigma(i)} + \beta, \quad z_i = \gamma * \hat{y}_i + \beta, \\ &\text{where } \ \mu(i) = \frac{1}{\lfloor \frac{d}{g} \rfloor} \sum_{i=0}^{\lfloor \frac{d}{g} \rfloor - 1} y_{\lfloor \frac{i \cdot g}{d} \rfloor + i}, \quad \sigma(i) = \sqrt{\frac{1}{\lfloor \frac{d}{g} \rfloor} \sum_{i=0}^{\lfloor \frac{d}{g} \rfloor - 1} (y_{\lfloor \frac{i \cdot g}{d} \rfloor + i} - \mu(i))^2}, \end{aligned} \tag{B.1}$$

For simplicity, we denote the group size $G = \lfloor \frac{d}{g} \rfloor$ and a group of activations $Y_j = y_{[\lfloor \frac{i \cdot g}{d} \rfloor : \lfloor \frac{i \cdot g}{d} \rfloor + G]}$. The partial derivative of the loss $\hat{\mathcal{L}}$ w.r.t. y for GN is given as

$$
\begin{aligned}
\frac{\partial \hat{\mathcal{L}}}{\partial Y_j} &= \frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j}\frac{\partial \hat{Y}_j}{\partial Y_j} + \frac{\partial \hat{\mathcal{L}}}{\partial \mu_j}\frac{\partial \mu_j}{\partial Y_j} + \frac{\partial \hat{\mathcal{L}}}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial Y_j} \\
&= (\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j}\frac{\partial \hat{Y}_j}{\partial Y_j}) + (\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j}\frac{\partial \hat{Y}_j}{\partial \mu_j} + \frac{\partial \hat{\mathcal{L}}}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial \mu_j})\frac{\partial \mu_i}{\partial Y_j} + (\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j}\frac{\partial \hat{Y}_j}{\partial \sigma_j})\frac{\partial \sigma_j}{\partial Y_j} \\
&= \frac{1}{\sigma_j}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j} + \frac{1}{G}((\sum_{i=1}^{G}\frac{-1}{\sigma_j}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i}) + \frac{\partial \hat{\mathcal{L}}}{\partial \sigma_j}(\frac{(\sigma_j)^{-1}}{2G}\sum_{i=1}^{G}-2(y_i-\mu_j))) + (-1\sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i}(\sigma_j)^{-2}(y_i-\mu_j))\frac{\partial \sigma_j}{\partial Y_i} \\
&= \frac{1}{\sigma_j}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j} + \frac{1}{G}(\sum_{i=1}^{G}\frac{-1}{\sigma_j}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i}) + \frac{-(\sigma_j)^{-1}}{2}\frac{2(Y_j-\mu_j)}{G}(\sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i}(\sigma_j)^{-2}(y_i-\mu_j)) \\
&= \frac{1}{G\sigma_j}(G\frac{\partial \hat{\mathcal{L}}}{\partial \hat{Y}_j} - \sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i} - \frac{(Y_j-\mu_j)}{\sigma_j}\sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial \hat{y}_i}\frac{(y_i-\mu_j)}{\sigma_j}) \\
&= \frac{\gamma_j}{G\sigma_j}(G\frac{\partial \hat{\mathcal{L}}}{\partial Z_j} - \sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial z_i} - \hat{Y}_j\sum_{i=1}^{G}\frac{\partial \hat{\mathcal{L}}}{\partial z_i}\frac{(y_i-\mu_j)}{\sigma_j}).
\end{aligned}
\tag{B.2}
$$

Eq. B.2 can be vectorized as

$$
\frac{\partial \hat{\mathcal{L}}}{\partial Y_j} = \frac{\gamma_j}{G\sigma_j}(G\frac{\partial \hat{\mathcal{L}}}{\partial Z_j} - 1\langle 1, \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}\rangle - \hat{Y}_j\langle \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}, \hat{Y}_j\rangle). \tag{B.3}
$$

Let $\mu_g = \frac{1}{G}\langle 1, \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}\rangle$. Note that $\|\hat{Y}_j\| = \sqrt{G}$. Eq. B.3 can be written as

$$
\frac{\partial \hat{\mathcal{L}}}{\partial Y_j} = \frac{\gamma_j}{\sigma_j}((\frac{\partial \hat{\mathcal{L}}}{\partial Z_j} - 1\mu_g) - \frac{\hat{Y}_j}{\|\hat{Y}_j\|}\langle \frac{\partial \hat{\mathcal{L}}}{\partial Z_j} - 1\mu_g, \frac{\hat{Y}_j}{\|\hat{Y}_j\|}\rangle) \tag{B.4}
$$

The squared norm of the partial derivative can be derived as

$$
\begin{aligned}
\|\frac{\partial \hat{\mathcal{L}}}{\partial Y_j}\|^2 &= \frac{\gamma_j^2}{(\sigma_j)^2}(\|(\frac{\partial \hat{\mathcal{L}}}{\partial z_j} - 1\mu_g)\|^2 - \langle \frac{\partial \hat{\mathcal{L}}}{\partial z_j} - 1\mu_g, \frac{\hat{y}_j}{\|\hat{y}_j\|}\rangle^2) \\
&= \frac{\gamma_j^2}{(\sigma_i)^2}(\|\frac{\partial \hat{\mathcal{L}}}{\partial Z_j}\|^2 - \frac{1}{G}\langle 1, \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}\rangle^2 - \frac{1}{G}\langle \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}, \hat{Y}_j\rangle^2)
\end{aligned}
\tag{B.5}
$$

Similarly, we consider the network with BGN. Given the activations $y \in \mathbb{R}^d$ where d denotes the number of features, the normalized outputs after BGN with group number of g and during

inference stage are formulated as

$$\hat{y}_i = \gamma \frac{y_i - \mu(i)}{\sigma(i)} + \beta, \quad z_i = \gamma * \hat{y}_i + \beta, \tag{B.6}$$

where $\mu(i)$ and $\sigma(i)$ here denote the tracked mean and standard deviation which are fixed during inference stage. Similarly, we denote the group size $G = \lfloor \frac{d}{g} \rfloor$ and a group of activations $Y_j = y_{[\lfloor \frac{i \cdot g}{d} \rfloor : \lfloor \frac{i \cdot g}{d} \rfloor + G]}$. The partial derivative of the loss $\hat{\mathcal{L}}$ *w.r.t.* y for BGN is given as

$$\frac{\partial \hat{\mathcal{L}}}{\partial Y_j} = \frac{\partial \hat{\mathcal{L}}}{\partial Z_j} \frac{\partial Z_j}{\partial \hat{Y}_j} \frac{\partial \hat{Y}_j}{\partial Y_j} = \frac{\gamma_j}{\sigma_j} \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}, \tag{B.7}$$

The squared norm of the partial derivative can be derived as

$$\| \frac{\partial \hat{\mathcal{L}}}{\partial Y_j} \|^2 = \frac{\gamma_j^2}{\sigma_j^2} \| \frac{\partial \hat{\mathcal{L}}}{\partial Z_j} \|^2, \tag{B.8}$$

Note that Eq. 6.5 in the chapter 6 is a combination of Eq. B.5 and Eq. B.8. We denote the influence of weight parameters as some constant C_g on gradient norm since we assume the networks are identical except for the normalization layers. And the partial derivative of the loss *w.r.t.* the output of normalization $g = \frac{\partial \hat{\mathcal{L}}}{\partial Z_j}$ is identical for networks due to the same loss function and weight parameters. Combining the assumptions with Eq. B.5 and Eq. B.8, we can derive the upper bound of the gradient norm through Y_j of networks with GN and GBN as

$$\begin{aligned} \| \nabla_x \hat{\mathcal{L}}_{gn} \| &\leq C_g \cdot \frac{|\gamma_{gn}|}{\sigma_j^{gn}} \sqrt{\|g\|^2 - \frac{1}{G} \langle 1, g \rangle^2 - \frac{1}{G} \langle g, \hat{Y}_j \rangle^2}, \\ \| \nabla_x \hat{\mathcal{L}}_{bgn} \| &\leq C_g \cdot \frac{|\gamma_{bgn}|}{\sigma_j^{bgn}} \|g\|, \end{aligned} \tag{B.9}$$

Following (Santurkar et al. 2018), we can generalize the results of loss smoothness to the gradient smoothness via Hessian. Note that the partial derivative of GN during inference in Eq. B.5 has the same format of the partial derivative of BN during training which is studied in (Santurkar et al. 2018) as

$$\| \frac{\partial \hat{\mathcal{L}}}{\partial y_j} \|^2 = \frac{\gamma^2}{\sigma^2} (\| \frac{\partial \hat{\mathcal{L}}}{\partial z_j} \|^2 - \frac{1}{m} \langle 1, \frac{\partial \hat{\mathcal{L}}}{\partial z_j} \rangle^2 - \frac{1}{m} \langle \frac{\partial \hat{\mathcal{L}}}{\partial z_j}, \hat{y}_j \rangle^2) \tag{B.10}$$

The differences are that Y_j denotes a group of activations in GN while y_j denotes a batch of activations in BN, and G denotes the size of group while m denotes the batch size. Thus,

we can easily generalize the smoothness of gradient in BN during training phase which is discussed in (Santurkar et al. 2018) to the one in GN during inference phase. In (Santurkar et al. 2018), the "effective" β-smoothness is defined as the changes of gradients as we move in the direction of gradients, which corresponds to $\hat{g}^T\hat{H}\hat{g}$ where $\hat{H}$ denotes the hessian *w.r.t.* the output of activations. We slightly modify $\hat{g}$ to $\hat{g}' = \frac{\hat{g}}{\|\hat{g}\|}$ so that $\hat{g}'^T\hat{H}\hat{g}'$ corresponds to the β smoothness in the direction of gradients. Through replacing a batch of activation y_j to a group of activation Y_j, batch size m to group size G, and g to normalized one g', the upper bound of gradient smoothness of BN in (Santurkar et al. 2018) can be reformulated for GN as

$$\hat{g}'^T\hat{H}\hat{g}' \leq \frac{\gamma^2}{\sigma^2}\left[g'^\top Hg' - \frac{1}{G\gamma}\langle g, \hat{Y_j}\rangle\right] \tag{B.11}$$

Since BGN uses fixed mean and variance during inference, Tthe upper bound of gradient smoothness of BGN during inference can be derived as

$$\hat{g}'^T\hat{H}\hat{g}' \leq \frac{\gamma^2}{\sigma^2}[g'^\top Hg'] \tag{B.12}$$

Similarly, we denote the influence of weight parameters as some constant C_H on gradient smoothness since we assume the networks are identical except for the normalization layers. And the partial derivative of the loss *w.r.t.* the output of normalization $H = \frac{\partial\hat{\mathcal{L}}}{\partial Z_j \partial Z_j}$ is identical for networks due to the same loss function and weight parameters. The β-smoothness of GN and BGN in the direction of gradients can be upper bounded as

$$\begin{aligned}
\beta_{gn} &\leq C_H \cdot \frac{\gamma_{gn}^2}{(\sigma_j^{gn})^2}\left[g'^\top Hg' - \frac{1}{G\gamma_{gn}}\langle g, \hat{Y_j}\rangle\right], \\
\beta_{bgn} &\leq C_H \cdot \frac{\gamma_{bgn}^2}{(\sigma_j^{bgn})^2}\left[g'^\top Hg'\right],
\end{aligned} \tag{B.13}$$

With the results in Eq. B.9 and Eq. B.13, we can easily extend the upper bound of adversarial transferability in (Yang et al. 2021) to the networks with GN and BGN through replacing the assumption of $\|\nabla_x\mathcal{L}\| \leq B$ with the upper bound in Eq. B.9 and the assumption of β-smoothness to those in Eq.B.13 since the usage of β-smoothness in the upper bound of adversarial transferability in (Yang et al. 2021) lies in

$$\mathcal{L}(x, y) + \delta \cdot \nabla_x\mathcal{L}(x, y) + \frac{\beta}{2}\|\delta\|^2 \geq \mathcal{L}(x + \delta, y), \tag{B.14}$$

where the perturbation δ is generated by adversarial attack via gradient ascent. Thus, the assumption of β-smoothness in (Yang et al. 2021) is equivalent to the β-smoothness in the direction of gradients in Eq. B.13. Finally, combining the upper bound of transferability in (Yang et al. 2021) with Eq. B.9 and Eq. B.13 via max function gives the desired results. $\square$

Bibliography

Andriushchenko, Maksym et al. (2020). 'Square attack: a query-efficient black-box adversarial attack via random search'. In: *European conference on computer vision*. Springer, pp. 484–501.

Ba, Jimmy Lei, Jamie Ryan Kiros and Geoffrey E. Hinton (2016). *Layer Normalization*. DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2014). 'Neural machine translation by jointly learning to align and translate'. In: *arXiv preprint arXiv:1409.0473*.

Baker, Bowen et al. (2016). 'Designing neural network architectures using reinforcement learning'. In: *arXiv preprint arXiv:1611.02167*.

Balaji, Yogesh, Tom Goldstein and Judy Hoffman (2019). 'Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets'. In: *arXiv preprint arXiv:1910.08051*.

Bello, Irwan et al. (2017). 'Neural optimizer search with reinforcement learning'. In: *International Conference on Machine Learning*. PMLR, pp. 459–468.

Benz, Philipp, Chaoning Zhang and In So Kweon (2021). 'Batch normalization increases adversarial vulnerability and decreases adversarial transferability: A non-robust feature perspective'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7818–7827.

Biggio, Battista et al. (2013). 'Evasion attacks against machine learning at test time'. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer, pp. 387–402.

Bingham, Ella and Heikki Mannila (2001). 'Random projection in dimensionality reduction: applications to image and text data'. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250.

Brendel, Wieland, Jonas Rauber and Matthias Bethge (2017). 'Decision-based adversarial attacks: Reliable attacks against black-box machine learning models'. In: *arXiv preprint arXiv:1712.04248*.

Brock, Andrew et al. (2017). 'Smash: one-shot model architecture search through hypernetworks'. In: *arXiv preprint arXiv:1708.05344*.

Carlini, Nicholas and David Wagner (2017). 'Towards evaluating the robustness of neural networks'. In: *2017 ieee symposium on security and privacy (sp)*. Ieee, pp. 39–57.

Chen, Hanlin et al. (2020a). 'Anti-bandit neural architecture search for model defense'. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*. Springer, pp. 70–85.

Chen, Hanting et al. (2020b). 'AdderNet: Do We Really Need Multiplications in Deep Learning?' In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, Hanting et al. (2020c). 'Distilling portable generative adversarial networks for image translation'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 3585–3592.

Chen, Tianlong et al. (2021a). 'Robust Overfitting may be mitigated by properly learned smoothening'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=qZzy5urZw9.

Chen, Xinghao et al. (2021b). 'An empirical study of adder neural networks for object detection'. In: *Advances in Neural Information Processing Systems* 34, pp. 6894–6905.

Cheng, Minhao et al. (2018). 'Query-efficient hard-label black-box attack: An optimization-based approach'. In: *arXiv preprint arXiv:1807.04457*.

Cisse, Moustapha et al. (2017). 'Parseval networks: Improving robustness to adversarial examples'. In: *International conference on machine learning*. PMLR, pp. 854–863.

Cohen, Jeremy, Elan Rosenfeld and Zico Kolter (2019). 'Certified adversarial robustness via randomized smoothing'. In: *international conference on machine learning*. PMLR, pp. 1310–1320.

Croce, Francesco and Matthias Hein (2020). 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'. In: *International conference on machine learning*. PMLR, pp. 2206–2216.

Dong, Minjing and Chang Xu (2023). 'Adversarial Robustness via Random Projection Filters'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4077–4086.

Dong, Minjing et al. (2020). 'Adversarially robust neural architectures'. In: *arXiv preprint arXiv:2009.00902*.

Dong, Minjing et al. (2021). 'Towards stable and robust addernets'. In: *Advances in Neural Information Processing Systems* 34, pp. 13255–13265.

Dong, Minjing et al. (2022). 'Random normalization aggregation for adversarial defense'. In: *Advances in Neural Information Processing Systems* 35, pp. 33676–33688.

Dong, Xuanyi and Yi Yang (2019). 'Searching for a robust neural architecture in four gpu hours'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770.

Dong, Yinpeng et al. (2017). 'Discovering adversarial examples with momentum'. In: *arXiv preprint arXiv:1710.06081* 5.

Dong, Yinpeng et al. (2018). 'Boosting adversarial attacks with momentum'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193.

Feinman, Reuben et al. (2017). 'Detecting adversarial samples from artifacts'. In: *arXiv preprint arXiv:1703.00410*.

Fu, Yonggan et al. (2021). 'Double-win quant: Aggressively winning robustness of quantized deep neural networks via random precision training and inference'. In: *International Conference on Machine Learning*. PMLR, pp. 3492–3504.

Galloway, Angus et al. (2019). 'Batch normalization is a cause of adversarial vulnerability'. In: *arXiv preprint arXiv:1905.02161*.

Girshick, Ross (2015). 'Fast r-cnn'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Goodfellow, Ian J, Jonathon Shlens and Christian Szegedy (2014). 'Explaining and harnessing adversarial examples'. In: *arXiv preprint arXiv:1412.6572*.

Grosse, Kathrin et al. (2017). 'On the (statistical) detection of adversarial examples'. In: *arXiv preprint arXiv:1702.06280*.

Guo, Chuan et al. (2019a). 'Simple black-box adversarial attacks'. In: *International Conference on Machine Learning*. PMLR, pp. 2484–2493.

Guo, Minghao et al. (2020). 'When nas meets robustness: In search of robust architectures against adversarial attacks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 631–640.

Guo, Yiwen, Ziang Yan and Changshui Zhang (2019b). 'Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks'. In: *Advances in Neural Information Processing Systems* 32.

He, Kaiming et al. (2016). 'Deep residual learning for image recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Zhezhi, Adnan Siraj Rakin and Deliang Fan (2019). 'Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 588–597.

Hendrycks, Dan and Kevin Gimpel (2016). 'Early methods for detecting adversarial images'. In: *arXiv preprint arXiv:1608.00530*.

Howard, Andrew et al. (2019). 'Searching for mobilenetv3'. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324.

Huang, Gao et al. (2017). 'Densely connected convolutional networks'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Huang, Hanxun et al. (2021). 'Exploring architectural ingredients of adversarially robust deep neural networks'. In: *Advances in Neural Information Processing Systems* 34, pp. 5545–5559.

Huawei (2020). *MindSpore*. https://www.mindspore.cn/.

Ilyas, Andrew et al. (2018). 'Black-box adversarial attacks with limited queries and information'. In: *International conference on machine learning*. PMLR, pp. 2137–2146.

Inkawhich, Nathan et al. (2020). 'Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability'. In: *Advances in Neural Information Processing Systems* 33, pp. 20791–20801.

Ioffe, Sergey and Christian Szegedy (2015). 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. In: *CoRR* abs/1502.03167. arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

Jeddi, Ahmadreza et al. (2020). 'Learn2perturb: an end-to-end feature perturbation learning to improve adversarial robustness'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1241–1250.

Kim, Hoki (2020). 'Torchattacks: A pytorch repository for adversarial attacks'. In: *arXiv preprint arXiv:2010.01950*.

Krizhevsky, Alex, Geoffrey Hinton et al. (2009). 'Learning multiple layers of features from tiny images'. In.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton (2012). 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., pp. 1097–1105.

Kurakin, Alexey, Ian J Goodfellow and Samy Bengio (2018). 'Adversarial examples in the physical world'. In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, pp. 99–112.

Li, Bai et al. (2019). 'Certified adversarial robustness with additive noise'. In: *Advances in neural information processing systems* 32.

Li, Yanxi et al. (2021a). 'Neural architecture dilation for adversarial robustness'. In: *Advances in Neural Information Processing Systems* 34, pp. 29578–29589.

Li, Yuhang et al. (2021b). 'Brecq: Pushing the limit of post-training quantization by block reconstruction'. In: *arXiv preprint arXiv:2102.05426*.

Liu, Chang et al. (2022a). 'Rethinking the Importance of Quantization Bias, toward Full Low-bit Training'. In: *IEEE Transactions on Image Processing*.

Liu, Hanxiao, Karen Simonyan and Yiming Yang (2018a). 'Darts: Differentiable architecture search'. In: *arXiv preprint arXiv:1806.09055*.

Liu, Xuanqing et al. (2018b). 'Towards robust neural networks via random self-ensemble'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 369–385.

Liu, Yanpei et al. (2016). 'Delving into transferable adversarial examples and black-box attacks'. In: *arXiv preprint arXiv:1611.02770*.

Liu, Ye et al. (2022b). 'Practical evaluation of adversarial robustness via adaptive auto attack'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15105–15114.

Liu, Zhuang et al. (2018c). 'Rethinking the value of network pruning'. In: *arXiv preprint arXiv:1810.05270*.

Madry, Aleksander et al. (2018). 'Towards Deep Learning Models Resistant to Adversarial Attacks'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJzIBfZAb.

Mao, Chengzhi et al. (2019). 'Metric learning for adversarial robustness'. In: *NeurIPS*, pp. 480–491.

Marlow, N. A. (1967). 'A normal limit theorem for power sums of independent random variables'. In: *The Bell System Technical Journal* 46.9, pp. 2081–2089.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi and Pascal Frossard (2016). 'Deepfool: a simple and accurate method to fool deep neural networks'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582.

Moosavi-Dezfooli, Seyed-Mohsen et al. (2017). 'Universal adversarial perturbations'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773.

Mustafa, Aamir et al. (2020). 'Deeply supervised discriminative learning for adversarial defense'. In: *IEEE transactions on pattern analysis and machine intelligence* 43.9, pp. 3154–3166.

Nachum, Ido et al. (2022). 'A Johnson-Lindenstrauss Framework for Randomly Initialized CNNs'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=YX0lrvdPQc.

Nguyen, Anh, Jason Yosinski and Jeff Clune (2015). 'Deep neural networks are easily fooled: High confidence predictions for unrecognizable images'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436.

Papernot, Nicolas, Patrick McDaniel and Ian Goodfellow (2016). 'Transferability in machine learning: from phenomena to black-box attacks using adversarial samples'. In: *arXiv preprint arXiv:1605.07277*.

Paszke, Adam et al. (2019). 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *NeurIPS*.

Pham, Hieu et al. (2018). 'Efficient neural architecture search via parameters sharing'. In: *International conference on machine learning*. PMLR, pp. 4095–4104.

Pinot, Rafael et al. (2019). 'Theoretical evidence for adversarial robustness through randomization'. In: *Advances in Neural Information Processing Systems* 32.

Pinot, Rafael et al. (2020). 'Randomization matters how to defend against strong adversarial attacks'. In: *International Conference on Machine Learning*. PMLR, pp. 7717–7727.

Pomponi, Jary, Simone Scardapane and Aurelio Uncini (2022). 'Pixle: a fast and effective black-box attack based on rearranging pixels'. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–7.

Qian, Haifeng and Mark N Wegman (2018). 'L2-nonexpansive neural networks'. In: *arXiv preprint arXiv:1802.07896*.

Qiao, Siyuan et al. (2019). 'Weight standardization'. In: *arXiv preprint arXiv:1903.10520* 2.3, p. 8.

Qin, Chongli et al. (2019). 'Adversarial robustness through local linearization'. In: *Advances in Neural Information Processing Systems* 32.

Rastegari, Mohammad et al. (2016). 'Xnor-net: Imagenet classification using binary convolutional neural networks'. In: *European conference on computer vision*. Springer, pp. 525–542.

Real, Esteban et al. (2019). 'Regularized evolution for image classifier architecture search'. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01, pp. 4780–4789.

Ren, Shaoqing et al. (2015). 'Faster r-cnn: Towards real-time object detection with region proposal networks'. In: *Advances in neural information processing systems* 28.

Rice, Leslie, Eric Wong and Zico Kolter (2020). 'Overfitting in adversarially robust deep learning'. In: *International Conference on Machine Learning*. PMLR, pp. 8093–8104.

Ronneberger, Olaf, Philipp Fischer and Thomas Brox (2015). 'U-net: Convolutional networks for biomedical image segmentation'. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, pp. 234–241.

Santurkar, Shibani et al. (2018). 'How does batch normalization help optimization?' In: *Advances in neural information processing systems* 31.

Shafahi, Ali et al. (2019). 'Adversarial training for free!' In: *Advances in Neural Information Processing Systems* 32.

Simonyan, Karen and Andrew Zisserman (2014). 'Very deep convolutional networks for large-scale image recognition'. In: *arXiv preprint arXiv:1409.1556*.

Sun, Xuxiang et al. (2022a). 'Exploring effective data for surrogate training towards black-box attack'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15355–15364.

Sun, Xuxiang et al. (2022b). 'Query-efficient decision-based attack via sampling distribution reshaping'. In: *Pattern Recognition* 129, p. 108728.

Sze, Vivienne et al. (2017). 'Efficient processing of deep neural networks: A tutorial and survey'. In: *Proceedings of the IEEE* 105.12, pp. 2295–2329.

Szegedy, Christian et al. (2014). 'Intriguing properties of neural networks'. In: *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1312.6199.

Tang, Yehui et al. (2020). 'A semi-supervised assessor of neural architectures'. In: *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1810–1819.

Tian, Zhi et al. (2019). 'Fcos: Fully convolutional one-stage object detection'. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636.

Tramèr, Florian et al. (2017a). 'Ensemble adversarial training: Attacks and defenses'. In: *arXiv preprint arXiv:1705.07204*.

Tramèr, Florian et al. (2017b). 'The space of transferable adversarial examples'. In: *arXiv preprint arXiv:1704.03453*.

Ulyanov, Dmitry, Andrea Vedaldi and Victor S. Lempitsky (2016). 'Instance Normalization: The Missing Ingredient for Fast Stylization'. In: *CoRR* abs/1607.08022. arXiv: 1607.08022. URL: http://arxiv.org/abs/1607.08022.

Vargas, Danilo Vasconcellos, Shashank Kotyan and SPM IIIT-NR (2019). 'Evolving robust neural architectures to defend from adversarial attacks'. In: *arXiv preprint arXiv:1906.11667* 3.

Vershynin, Roman (2018). *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press.

Wang, Meng et al. (2020a). 'A survey on large-scale machine learning'. In: *IEEE Transactions on Knowledge and Data Engineering* 34.6, pp. 2574–2594.

Wang, Xiaosen and Kun He (2021). 'Enhancing the transferability of adversarial attacks through variance tuning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1924–1933.

Wang, Yisen et al. (2019). 'Improving adversarial robustness requires revisiting misclassified examples'. In: *International conference on learning representations*.

— (2020b). 'Improving adversarial robustness requires revisiting misclassified examples'. In: *International Conference on Learning Representations*.

Weng, Tsui-Wei et al. (2018). 'Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BkUHlMZ0b.

Wu, Dongxian, Shu-Tao Xia and Yisen Wang (2020). 'Adversarial weight perturbation helps robust generalization'. In: *Advances in Neural Information Processing Systems* 33, pp. 2958–2969.

Wu, Yuxin and Kaiming He (2018). 'Group Normalization'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*.

Xie, Cihang and Alan Yuille (2019). 'Intriguing properties of adversarial training at scale'. In: *arXiv preprint arXiv:1906.03787*.

Xie, Cihang et al. (2017). 'Mitigating adversarial effects through randomization'. In: *arXiv preprint arXiv:1711.01991*.

Xie, Cihang et al. (2019). 'Feature denoising for improving adversarial robustness'. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 501–509.

Xie, Cihang et al. (2020a). 'Adversarial examples improve image recognition'. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 819–828.

Xie, Cihang et al. (2020b). 'Smooth adversarial training'. In: *arXiv preprint arXiv:2006.14536*.

Xu, Yuhui et al. (2019). 'Pc-darts: Partial channel connections for memory-efficient architecture search'. In: *arXiv preprint arXiv:1907.05737*.

Yang, Xingyi et al. (2022). 'Deep model reassembly'. In: *Advances in neural information processing systems* 35, pp. 25739–25753.

Yang, Zhaohui et al. (2020). 'CARS: Continuous Evolution for Efficient Neural Architecture Search'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Zhuolin et al. (2021). 'Trs: Transferability reduced ensemble via promoting gradient diversity and model smoothness'. In: *Advances in Neural Information Processing Systems* 34, pp. 17642–17655.

Yoshida, Yuichi and Takeru Miyato (2017). 'Spectral norm regularization for improving the generalizability of deep learning'. In: *arXiv preprint arXiv:1705.10941*.

You, Haoran et al. (2020). 'Shiftaddnet: A hardware-inspired deep network'. In: *Advances in Neural Information Processing Systems* 33, pp. 2771–2783.

Yu, Yunrui, Xitong Gao and Cheng-Zhong Xu (2021). 'Lafeat: Piercing through adversarial defenses with latent features'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5735–5745.

Zagoruyko, Sergey and Nikos Komodakis (2016). 'Wide residual networks'. In: *arXiv preprint arXiv:1605.07146*.

Zela, Arber et al. (2019). 'Understanding and robustifying differentiable architecture search'. In: *arXiv preprint arXiv:1909.09656*.

Zhang, Hongyang et al. (2019a). 'Theoretically principled trade-off between robustness and accuracy'. In: *International conference on machine learning*. PMLR, pp. 7472–7482.

Zhang, Hongyi, Yann N Dauphin and Tengyu Ma (2019b). 'Fixup initialization: Residual learning without normalization'. In: *arXiv preprint arXiv:1901.09321*.

Zhang, Jingfeng et al. (2020). 'Attacks which do not kill training make adversarial learning stronger'. In: *International conference on machine learning*. PMLR, pp. 11278–11287.

Zhou, Shuchang et al. (2016). 'Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients'. In: *arXiv preprint arXiv:1606.06160*.

Zhou, Xiao-Yun et al. (2020). 'Batch Group Normalization'. In: *CoRR* abs/2012.02782. arXiv: 2012.02782. URL: https://arxiv.org/abs/2012.02782.

Zoph, Barret and Quoc V Le (2016). 'Neural architecture search with reinforcement learning'. In: *arXiv preprint arXiv:1611.01578*.

Zoph, Barret et al. (2018). 'Learning transferable architectures for scalable image recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710.