ROBUST LEARNING OF DEEP NEURAL NETWORKS UNDER DATA CORRUPTION

By

Boyang Liu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2022

ABSTRACT

ROBUST LEARNING OF DEEP NEURAL NETWORKS UNDER DATA CORRUPTION

By

Boyang Liu

Training deep neural networks in the presence of corrupted data is challenging as the corrupted data points may significantly impact generalization performance of the models. Unfortunately, the data corruption issue widely exists in many application domains, including but not limited to, healthcare, environmental sciences, autonomous driving, and social media analytics. Although there have been some previous studies that aim to enhance the robustness of machine learning models against data corruption, most of them either lack theoretical robustness guarantees or unable to scale to the millions of model parameters governing deep neural networks. The goal of this thesis is to design robust machine learning algorithms that **1)** effectively deal with different types of data corruption, **2)** have sound theoretical guarantees on robustness, and **3)** scalable to large number of parameters in deep neural networks. There are two general approaches to enhance model robustness against data corruption. The first approach is to detect and remove the corrupted data while the second approach is to design robust learning algorithms that can tolerate some fraction of corrupted data. In this thesis, I had developed two robust unsupervised anomaly detection algorithms and two robust supervised learning algorithm for corrupted supervision and backdoor attack. Specifically, in Chapter 2, I proposed the Robust Collaborative Autoencoder (RCA) approach to enhance the robustness of vanilla autoencoder methods against natural corruption. In Chapter 3, I developed Robust RealNVP, a robust density estimation technique for unsupervised anomaly detection tasks given concentrated anomalies. Chapter 4 presents the Provable Robust Learning (PRL) approach, which is a robust algorithm against agnostic corrupted supervision. In Chapter 5, a meta-algorithm to defend against backdoor attacks is proposed by exploring the connection between label corruption and backdoor data poisoning attack. Extensive experiments on multiple benchmark datasets have demonstrated the robustness of the proposed algorithms under different types of corruption.

This thesis is dedicated to my parents Songmei Liu, Song Liu, my wife Yunshi Liang, as well as my coming daughter, Shuman Liu.

ACKNOWLEDGEMENTS

Time flies, it's hard to believe I've spent five years at Michigan State University, and as my doctoral career draws to a close, so many emotions come to my mind.

First and foremost, I would like to thank my advisors Prof. Pang-Ning Tan and Prof. Jiayu Zhou. I thank Professor Tan for teaching me the correct and rigorous scientific research attitude. Your enthusiasm and persistence in scientific research have influenced me all the time in the past five years. I will always remember your patience and valuable advice when we discussing every detail, every formula. I am grateful to Prof. Zhou for his encouragement over the past five years and giving me full research freedom. You always push me one step further in research, which benefits me a lot. Without Prof. Tan and Prof. Zhou, I couldn't have completed my Ph.D., nor could I have grown from a machine learning beginner to a machine learning researcher with some knowledge, although this knowledge is still very limited considering the vastness of machine learning field.

Secondly, I would like to thank my defense committee members Prof. Jiliang Tang and Prof. Kendra Spence Cheruvelil for their support and valuable advice during my PhD career. I thank Professor Tang for his valuable advice about my research. I thank Professor Kendra for his insights on limnology, which have been very helpful for me to build better/explainable machine learning models.

Also, I would like to thank my colleagues at Michigan State University. In particular, I would like to thank Ding Wang for his brilliant mind, which gave me endless inspiration when I was stuck in a mathematical proof. I thank Kaixiang Lin for spending his time discussing some interesting machine learning problem with me, which greatly broadens my knowledge. I am grateful to Mengying Sun and Wei Wang for their encouragement and the delicious food they made during the pandemic. I am grateful to Qiaozi Gao, Xi Liu, Qi Wang and Shaohua Yang for those countless happy times. I would also like to thank all my lab colleagues and friends including Liyang xie, Inci M. Baytas, Ikechukwu Uchendu, Junyuan Hong, Andy Tang, Xitong Zhang, Jingwen Shi, Nan Du, Farzan Masrour, Tyler Wilson, Zhuangdi Zhu, and Courtland VanDam.

Finally, I would like to thank my parents Song Liu and Songmei Liu for their unconditional support and love. I would not be where I am today without you. I would like to thank my kind and beautiful wife Yunshi Liang, who always stand and company with me and make me a better person. I would also like to thank my soon-to-be daughter Shuman Liu, who has brought my wife and I endless joy during this pandemic.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xii

LIST OF ALGORITHMS xiv

CHAPTER 1 INTRODUCTION 1
- 1.1 Natural Corruption 1
- 1.2 Adversarial Corruption 4
- 1.3 Scope of the Thesis 7
- 1.4 Thesis Contributions 7
- 1.5 Publications 7
- 1.6 Thesis Outline 8

CHAPTER 2 ROBUST COLLABORATIVE AUTOENCODER (RCA) 9
- 2.1 Introduction 9
- 2.2 Related Work 11
- 2.3 Methodology 12
  - 2.3.0.1 Sample Selection 12
  - 2.3.0.2 Ensemble Evaluation 17
- 2.4 Experiment 18
  - 2.4.0.1 Results on Synthetic Data 18
  - 2.4.0.2 Results on Real-World Data 19
- 2.5 Conclusion 23
- 2.6 Proofs of Theorem 24
  - 2.6.1 Proof of Theorem 1 24
  - 2.6.2 Proof of Theorem 2 27
  - 2.6.3 Proof of Theorem 3 29

CHAPTER 3 ROBUST DENSITY ESTIMATION FOR UNSUPERVISED ANOMALY DETECTION 32
- 3.1 Introduction 32
- 3.2 Related Work 35
  - 3.2.1 Density Estimation and Anomaly Detection 35
  - 3.2.2 Lipschitz Regularization 36
  - 3.2.3 Robust Optimization 37
- 3.3 Preliminaries 38
- 3.4 Methodology 39
  - 3.4.1 Robustness Analysis from Optimization Perspective 39
  - 3.4.2 Robust Gradient Estimation Method for Density Estimation 41
  - 3.4.3 Spectral Normalization 43
  - 3.4.4 Lipschitz Regularization for Flow-based Model 44

3.4.5 Anomaly Detection from Trained Network . . . 46
3.5 Experiment . . . 46
3.5.1 Experiments on Synthetic Data . . . 46
3.5.2 Experiments on Real-World Data . . . 47
3.5.3 Results for ODDS data . . . 50
3.5.4 Results for CIFAR10 . . . 50
3.5.5 Ablation Study for Spectral Normalization . . . 52
3.5.6 Sensitivity Analysis for ϵ . . . 52
3.6 Conclusion . . . 53

CHAPTER 4 ROBUST LEARNING DEEP NEURAL NETWORKS UNDER AGNOSTIC CORRUPTED SUPERVISION . . . 55
4.1 Introduction . . . 55
4.2 Related Work . . . 57
4.3 Methodology . . . 58
4.3.1 Stochastic Gradient Descent with Biased Gradient . . . 59
4.3.2 Robust Gradient Estimation for General Data Corruption . . . 61
4.3.3 Robust Gradient Estimation for One Dimensional Corrupted Supervision . . . 62
4.3.4 Extension to Multi-Dimensional Corrupted Supervision . . . 64
4.3.5 Comparison against Other Robust Optimization Methods . . . 66
4.3.6 Relationship to Self-Paced Learning (SPL) . . . 67
4.3.7 Combining with Co-teaching Style Training . . . 70
4.4 Experimental Results . . . 70
4.4.1 Regression Results . . . 73
4.4.2 Classification Results . . . 74
4.4.3 Case Study on Limnology Dataset . . . 75
4.4.4 Sensitivity Analysis . . . 77
4.5 Conclusion . . . 78
4.6 Supplementary Empirical Results on Running Time . . . 78
4.7 Proofs of Theorem . . . 78
4.7.1 Proof of Theorem 4 . . . 78
4.7.2 Proof of Corollary 1 . . . 79
4.7.3 Proof of Lemma 4 . . . 81
4.7.4 Proof of Theorem 5 . . . 83
4.7.5 Proof of Lemma 2 . . . 83

CHAPTER 5 DEFENDING BACKDOOR ATTACKS VIA ROBUSTNESS AGAINST NOISY LABEL . . . 85
5.1 Introduction . . . 85
5.2 Related Work . . . 87
5.2.1 Robust Deep Learning Against Adversarial Attack . . . 87
5.2.2 Robust Deep Learning Against Noisy Labels . . . 87
5.2.3 Data Poisoning Backdoor Attack and its Defense . . . 88
5.3 Preliminaries . . . 89
5.3.1 Learning with Noisy Labels . . . 89

5.3.2 Problem Setting of Backdoor Attacks . . . 89
5.4 Methodology . . . 90
5.4.1 A black-box robust algorithm against noisy labels . . . 91
5.4.2 Theoretical Justification . . . 93
5.4.3 A semi-supervised algorithm . . . 95
5.4.4 How to choose the noisy label algorithm . . . 96
5.5 Experiment . . . 97
5.6 Proof of Theorems . . . 101
5.6.1 Proof of Inequality in Eq. 5.3 . . . 101
5.6.2 Proof of theorem 1 and corollary 2 . . . 102
5.6.3 Proof of Proposition 3 . . . 104
5.7 More Discussions about the Proposed Framework . . . 104
5.7.1 Noisy Label Algorithm . . . 104
5.7.2 Ablation Study of Inner Maximization and Outer Minimization . . . 105
5.7.3 Discussion about Lipschitz regularization and adversarial training . . . 105
5.7.4 Supplementary Experiment Results . . . 106
5.7.4.1 Experiment Hyperparameters . . . 106
5.7.4.2 Experiment on MNIST . . . 107

CHAPTER 6 CONCLUSION & FUTURE WORK . . . 110
6.0.1 Conclusion . . . 110
6.0.2 Future Work . . . 110

BIBLIOGRAPHY . . . 112

LIST OF TABLES

Table 1.1: Correlation between changes in land use/land cover variables between 2001 and 2016 for raw lake data obtained from the LAGOS-NE database. 2

Table 1.2: Correlation between changes in land use/land cover variables between 2001 and 2016 for the remaining lakes after removing outliers using isolation forest. . 2

Table 1.3: R^2 for multiple linear regression (MLR) and multiple linear regression with local outlier factor (LOF-MLR) for predicting total phosphorous in lakes using other lake water quality variables (including total nitrogen, NO_2NO_3, secchi, etc.). 3

Table 2.1: Performance comparison of RCA against baseline methods in terms of their average and standard deviation of AUC scores across 10 random initializations. . 19

Table 2.2: Comparison of RCA against baseline methods in terms of (#win-#draw-#loss) on 19 benchmark datasets with different proportion of imputed missing values in the data. Results for RCA-E (no ensemble), RCA-SS (no sample selection), AE (no ensemble and no sample selection) are for ablation study. 21

Table 2.3: Average and standard deviation of AUC scores (for 10 random initialization) as anomaly ratio parameter is varied from ϵ (i.e., true anomaly ratio of the data) to $\epsilon + \Delta\epsilon$. If ϵ is less than 0.05 or 0.1, then $\Delta\epsilon = 0.05$ and $\Delta\epsilon = 0.1$ will be truncated to 0. 22

Table 3.1: Average and standard deviation of AUC scores for the ODDS datasets (across 5 different random seeds). The last row corresponds to average rank of each method. 51

Table 3.2: Average and standard deviation of AUC scores for the CIFAR10 dataset (across 5 different random seeds). The first column shows the category chosen as normal class. 'class'-c means the training anomalies are clustered whereas 'class'-u means the anomalies are uniformly sampled from other classes. 51

Table 4.1: R-square on CelebA clean testing data, and the standard deviation is from last ten epochs and 5 random seeds. 71

Table 4.2: Classification accuracy for clean testing data on CIFAR10 and CIFAR100 with training on ***symmetric*** and ***pairflip*** label corruption. The standard deviation is from last ten epochs and 3 random seeds. 71

Table 4.3: Main Hyperparmeters 73

Table 4.4: Sensitivity analysis for over-estimated/under-estimated ϵ. 73

Table 4.5: Prediction Results for Limonology Dataset. The numbers are averaged R-square across 3 random seeds. As we can see, standard training deep neural network cannot defend against both type of corruption while PRL showes robustness for such attack. 77

Table 4.6: Execution Time of Single Epoch in CIFAR-10 Data 78

Table 5.1: Performance on CIFAR10. ϵ is the corruption rate. 97

Table 5.2: Performance on CIFAR100. ϵ is the corruption rate. 98

Table 5.3: Accuracy on CIFAR10 in semi-supervised setting. ϵ is the corruption rate. . . . 100

Table 5.4: Sensitivity analysis of ϵ. Average top-1 accuracy across three random seeds. The first number is the clean accuracy while the second number is the poisoned accuracy. The hyperparameter ϵ is fixed to be 0.5 while the ground truth ϵ is varied. 101

Table 5.5: Overview of noisy-label defending algorithms, which achieve robustness against up to 45% of pairwise flipping label noises. 104

Table 5.6: Ablation study on CIFAR10. ϵ is the corruption rate. 105

Table 5.7: Ablation study on CIFAR100. ϵ is the corruption rate. 106

Table 5.8: Performance on MNIST. ϵ is the corruption rate. 108

LIST OF FIGURES

Figure 1.1: An illustration of the backdoor attack. 4

Figure 2.1: An illustration of the training phase for the proposed RCA framework. 10

Figure 2.2: The first two columns are results for 10% and 40% anomaly ratio, respectively. The last column shows the fraction of points with highest reconstruction loss that are true anomalies. The top diagram is for 10% anomalies while the bottom is for 40% anomalies. 18

Figure 2.3: Comparison of RCVA and VAE in terms of AUC score. RCVA outperforms VAE on most datasets, suggesting that the RCA framework can improve the performance of other DNNs such as VAE. 23

Figure 2.4: Effect of varying number of DNNs in RCA on AUC scores. RCA corresponds to a twin network while K-RCA has K DNNs. 24

Figure 3.1: Effect of anomalies on density estimation for flow-based models. The upper left figure represents the clean data while the bottom left figure represents data contaminated with a small fraction of clustered anomalies (shown as orange points). The upper right figure shows the results of Real NVP results on the clean data while the bottom right figure shows the results for contaminated data. Observe that Real NVP assigns large density to the region of clustered anomalies, which leads to its suboptimal performance. 34

Figure 3.2: Effect of uniform anomalies on density estimation. Normal data are generated from a two-moon distribution while anomalies are from a uniform distribution (see left-most figure). Density estimation results of Real NVP, TrimRealNVP, and RobustRealNVP are shown from left to right. 48

Figure 3.3: Effect of clustered anomalies on density estimation. Normal data are generated from a two-moon distribution while anomalies are from a concentrated Gaussian (see left-most figure). Density estimation results of Real NVP, TrimRealNVP, and RobustRealNVP are shown from left to right. 49

Figure 3.4: Ablation study for the spectral normalization module. The TrimRealNVP is the RobustRealNVP without the spectral normalization module. 52

Figure 3.5: Sensitivity analysis for estimated ϵ. The ground truth ϵ is 0.1. We give the results of the estimated ϵ equals to 0.05, 0.075, 0.1, 0.125, 0.15, 0.2. The y-value is the averaged AUC across 5 random seeds. 53

Figure 3.6: Sensitivity Analysis of ground-truth Training anomaly ratio for uniform anomalies from 0.1 to 0.4. Red is RobustRealNVP, blue is Real NVP. 54

Figure 4.1: Geometric illustration of the difference between loss filtering and gradient norm filtering. 68

Figure 4.2: CelebA Testing Curve During Training. The corruption ratios are [0.1, 0.2, 0.3, 0.4] from left to right. The corruption types are [linadv, signflip, uninoise, mixture, pairflip] from up to bottom. X axis represents the epoch number, Y axis represents the testing r-square. In some experiment, there is no curve for Standard and NormClip since they gave negative r-square, which will effect the plotting scale. The shadow represents the confidence interval, which is calculated across 5 random seed. As we see, PRL(G), PRL(L), and Co-PRL(L) are robust against different types of corruptions. 74

Figure 4.3: CIFAR10 and CIFAR100 Testing Curve During Training. X axis represents the epoch number, Y axis represents the testing accuracy. The shadow represents the confidence interval, which is calculated across 3 random seed. The first two rows are results for CIFAR10 while the last two rows are results for CIFAR100. For each dataset, the first row represents the symmetric noise while the second row represents the pairflip noise. The corruption ratios for symmetric noise are [0.3, 0.5, 0.7] from left to right while the corruption ratios for pairflip noise are [0.25, 0.35, 0.45] from left to right. As we see, PRL(L), and Co-PRL(L) are robust against different types of corruptions. 76

Figure 5.1: Illustration of our meta algorithm. By combining the minimax objective and noisy label algorithm, we could reduce a backdoor attack problem to a label flipping attack. The left most is the clean original data. The second shows corrupted samples. The third figure shows the inner maximization step while the last figure shows the outer minimization step. 107

Figure 5.2: Example of clean and various poisoned samples.badnet patch attack: trigger is a 3×3 black-white checkerboard and it is added to the right bottom corner of the image. blending attack: trigger is a fixed Gaussian noise which has the same dimension as the image. The corrupted image generated by $\mathbf{x}_i^\epsilon = (1-\alpha)\mathbf{x}_i + \alpha\mathbf{t}$. In our experiment, we set the α as 0.1. 107

Figure 5.3: Example of label flipping attacks on both binary feature values and continuous feature values. 108

LIST OF ALGORITHMS

Algorithm 1: Robust Collaborative Autoencoders 13

Algorithm 2: Robust Gradient Estimation for Density Estimation 42

Algorithm 3: (***PRL(G)***) Provable Robust Learning for General Corrupted Data 61

Algorithm 4: (***PRL(L)***) Efficient Provable Robust Learning for Corrupted Supervision . . 64

Algorithm 5: Co-PRL(L), Collaborative Provable Robust Learning 70

Algorithm 6: Meta Algorithm for Robust Learning Against Backdoor Attacks 93

Algorithm 7: Semi-Supervised Algorithm for Robust Learning Against Backdoor Attacks 96

CHAPTER 1

INTRODUCTION

Deep neural network (DNN) has achieved huge success in recent years. With the large amount of data available as we move into the era of big data, DNN has shown extraordinary generalization performance in many applications including image classification, language translation, climate modeling, medical diagnosis, etc. One huge advantage of using DNN is its expressive power, which makes it possible to fit any complex, continuous functions. However, this advantage has also become a major weakness of DNN when the training data is contaminated. A series of studies have shown that DNN can easily memorize corrupted examples in the data (Zhang et al., 2016), which in turn, would severely degrade its generalization performance. Thus, it is imperative to develop a robust learning algorithm for DNN that will defend against such data corruptions.

Data corruption, which refers to perturbation of the original values of the data, widely exists in many real-world applications. The corrupted data points, which are also commonly known as noise, may manifest themselves as outliers or anomalies[1] in the data. For predictive modeling applications, the corrupted values can be present in the predictor variables, target/response variables, or both. Corruption in data can be generally divided into two categories depending on whether the corruption is introduced intentionally. If the corruption is introduced unintentionally, then it is referred to as natural corruption. If the corruption is introduced deliberately, with a certain purpose (e.g., to degrade or manipulate the model performance), then it is often referred to as adversarial corruption. The distinction between the two types of data corruptions will be discussed in the next subsections.

1.1 Natural Corruption

Natural corruption may occur in data due to a variety of factors. Common examples include low-quality photos caused by camera issues, incorrectly labeled data points introduced during data

[1] Note that outliers/anomalies are unusual values in the data. Not all corrupted values are outliers/anomalies nor all outliers/anomalies are corrupted values.

	agriculture	urban	wetlands	forest	grass	shrub
agriculture	1.000000	-0.455317	0.024345	-0.121643	-0.257747	-0.025230
urban	-0.455317	1.000000	-0.302412	-0.170223	-0.087192	-0.067282
wetlands	0.024345	-0.302412	1.000000	0.025391	0.009387	0.013533
forest	-0.121643	-0.170223	0.025391	1.000000	-0.431688	-0.486350
grass	-0.257747	-0.087192	0.009387	-0.431688	1.000000	-0.130568
shrub	-0.025230	-0.067282	0.013533	-0.486350	-0.130568	1.000000

Table 1.1: Correlation between changes in land use/land cover variables between 2001 and 2016 for raw lake data obtained from the LAGOS-NE database.

	agriculture	urban	wetlands	forest	grass	shrub
agriculture	1.000000	-0.104697	-0.013885	-0.146293	-0.386797	-0.061773
urban	-0.104697	1.000000	-0.034925	-0.077534	-0.032570	-0.019336
wetlands	-0.013885	-0.034925	1.000000	-0.003893	0.014305	0.000290
forest	-0.146293	-0.077534	-0.003893	1.000000	-0.445040	-0.534568
grass	-0.386797	-0.032570	0.014305	-0.445040	1.000000	-0.042667
shrub	-0.061773	-0.019336	0.000290	-0.534568	-0.042667	1.000000

Table 1.2: Correlation between changes in land use/land cover variables between 2001 and 2016 for the remaining lakes after removing outliers using isolation forest.

annotation, faulty sensor measurements, etc. Since the corruption is introduced unintentionally, rather than being carefully designed, such corruption would generally degrade/harm the model performance only if the corrupted values are significantly different from their original (unperturbed or clean) values. Such natural corruption is usually detectable based on the abnormal values (e.g., sensor measurements that lie outside their normal range of values) or unusual combination of values by using techniques such as unsupervised anomaly detection (AD). After removing the detected anomalies, the downstream tasks will be able to give better generalization performance (Chandola et al., 2009).

While natural corruption may be unavoidable, the presence of the corrupted data may lead to misleading conclusions about the relationships present in the data or degradation of the overall model performance. To illustrate the potential risk of natural corruption in data, consider the following two examples from the ecology domain. In the first example, we compute the correlation between changes in the land use/land cover variables associated with different lakes in the north-east region of the United States. The original data was obtained from the LAGOS-NE database (Soranno et al., 2017).

	MLR	LOF-MLR
R^2	0.78	0.82

Table 1.3: R^2 for multiple linear regression (MLR) and multiple linear regression with local outlier factor (LOF-MLR) for predicting total phosphorous in lakes using other lake water quality variables (including total nitrogen, NO_2NO_3, secchi, etc.).

Changes in 6 land use/land cover variables (including agriculture, urban, wetlands, forest, grass, and shrub) between the years 2001 to 2016 are initially computed for each lake. We then calculate a correlation matrix of the raw data and compare them against the correlation matrix computed after applying isolation forest (Liu et al., 2008), a popular unsupervised anomaly detection approach, to remove the outliers. The results shown in Tables 1.1 and 1.2, respectively, suggest that the correlation values can alter significantly when the outliers are removed. For example, the correlation between urban and agriculture changes from -0.46 to -0.10 whereas the correlation between agriculture and wetlands changes from positive correlation to negative correlation. Assuming the outliers removed are true corruption of the data, this analysis suggests that such corruption can potentially alter the observed correlations, which leads to conflicting conclusions.

In the second example, the possibility of corrupted data to degrade the performance of regression models is demonstrated. In this example, a regression is trained to predict the total phosphorus (TP) in lakes using data from the LAGOS-NE database (Soranno et al., 2017). The predictor variables used include total nitrogen, chlorophyll, secchi, NO_2NO_3, etc. There are 10,470 observations in the dataset. We compared the results of applying multiple linear regression on the original data against the results after removing 5% of the anomalies in the training data using an unsupervised anomaly detection method known as local outlier factor (Breunig et al., 2000). Table 1.3 showed the average R-square of 10-fold cross validation on testing data for both methods. The results in Table 1.3 suggest that applying LOF to remove outliers prior to performing multiple linear regression can produce higher R-square values, which is important if the outliers are natural corruption of the data.

Although anomaly detection can detect unusual values due to natural corruption, adopting the anomaly detection method to the deep neural networks is a challenge. One key difficulty of using deep neural networks for unsupervised AD is the overparameterization of DNN. The overparameterization

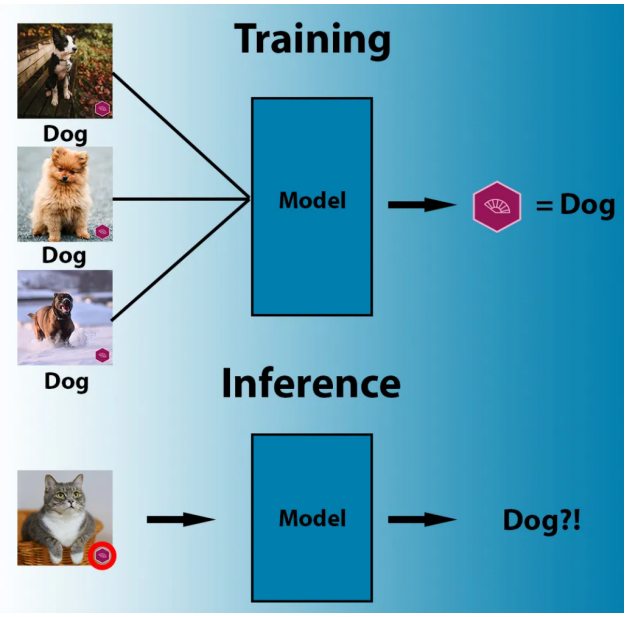


Figure 1.1: An illustration of the backdoor attack.

makes the DNN to easily fit even the anomalies in the data (Zhang et al., 2016), thus degrading the performance of the AD algorithm. For example, a classical AD method is the auto-encoder, which reconstructs the data by using a pair of encoder and decoder. Data points with large reconstruction loss would be considered as corrupted. The underlying prerequisite for auto-encoder to work well in AD is that the anomalies should be harder to reconstruct compared to normal data points. However, for DNN, this assumption does not hold generally since the autoencoder (AE) could easily memorize all the training examples. As a consequence, it is possible for both the clean and corrupted data to have zero anomaly scores, making it difficult to distinguish them. There is no approach that guarantees we can filter out the true anomalies. Thus, it is important to make the training process of AE more robust to alleviate the ill-effect brought upon by the corrupted data. More details on this would be discussed in the next few chapters.

1.2 Adversarial Corruption

Another category of data corruption is adversarial corruption. Unlike natural corruption, adversarial corruption is often carefully designed by an adversary, who aims to attack the model in order to degrade its performance by manipulating its input data. To successfully compromise the model,

the data must be corrupted in such a way that makes it difficult to distinguish the corrupted data points from the non-corrupted ones. Thus, by design, anomaly detection algorithms may not be able to detect such adversarial corruptions since the perturbed data are often non-outliers. Instead of using anomaly detection, a more effective approach is to develop a robust learning algorithm that guarantees the model trained on corrupted data will not be too different from one trained on the clean (uncorrupted) data.

Adversarial corruption can be found in many application domains. One example application is in image classification (see Figure 1.1). Suppose the adversary would like to manipulate the model such that the model would classifier his images as the dog even though it is not. The adversary can design a trigger by simply perturbing a few pixels in the training image and labeled them as the dog. The model would be learned based on the corrupted training data. During the inference phase, given a non-dog image, the adversary would add the trigger to it before passing it to the model. Finally, the model would classifier it as a dog since the model detected the trigger, and all training data with the trigger belongs to the class dog. The whole process is shown in figure 1.1.
In other words, an adversary could simply manipulate the output by adding triggers to the input data. It is therefore important to develop robust algorithms that will defend against such adversarial data corruption for trustworthy AI.

One natural question is whether we can use unsupervised anomaly detection to defend against adversarial corruption. Unfortunately, it is very difficult to use anomaly detection to detect adversarial corruption since the adversary can manipulate the data in such a way that the corrupted data resembles normal data despite being “harmful" to the model. To achieve robustness against adversarial corruption, a typical approach is to design a robust learning algorithm such that changing a small fraction of the data will not significantly alter the model output. We gave a simple Gaussian mean estimation example below to illustrate the robust learning approach. We first define ε-corrupted set, which is also known as Huber’s contamination model (Huber et al., 1973).

Let $\mathcal{F}$ be a family of probabilistic models. We say that a set of N samples is ε-corrupted from $\mathcal{F}$ if it is generated as follows:

- N samples are drawn from an unknown $F \in \mathcal{F}$
- An omniscient adversary inspects these samples and changes arbitrarily an ε-fraction of them.

In particular, we assume the adversary knows everything, which includes, but not limited to, the clean data distribution and the learning algorithm. The robust mean estimation problem for a Gaussian distribution is defined as follows:

Definition 1 (Robust Estimation for Gaussian Mean) *Let $\mu \in \mathbb{R}^d$, $\sigma > 0$, and let $\varepsilon \in [0, 1/2)$. Let $S = \{X_1, ..., X_n\}$ be an ε-corrupted set of samples from $\mathcal{N}(\mu, \sigma^2)$. Given S, ε, σ, output $\hat{\mu}$ minimizing $\|\hat{\mu} - \mu\|_2$*

Here, by simply using the empirical mean will not give a robust estimation since an outlier can ruin the mean estimation task. One solution is to use median as the estimator, as it has been shown that median is the optimal estimator for one-dimensional Gaussian mean estimation. It is easy to see that as long as the corruption ratio, ϵ, is less than 0.5, no matter how the adversary injects the corruption, the estimator would not be too far from the ground truth mean. However, using anomaly detection methods would not guarantee that the corruption data would be correctly filtered out since the adversary could hide the corrupted data in the normal data, and it is impossible to filter out the corrupted data perfectly. Nevertheless, although the median is the optimal robust mean estimator in one dimensional Gaussian, it is not a good estimator for high dimensional data. More details would be discussed in later chapters.

Many robust learning algorithms can tolerate a small fraction of adversarial corruption in the training data. However, to the best of our knowledge, previous robust algorithms are designed for shallow models such as linear regression or support vector machine, and are not specifically designed for DNN. As a consequence, some of the algorithms either lose their robustness guarantees or are computationally infeasible due to their high space-time complexity. Thus, it is imperative to develop a robust learning algorithm that can deal with adversarial corruption yet scalable to DNN. To achieve robustness with theoretical guarantees, we may need stronger assumptions (e.g., corruptions only happen to the labels), and we will discuss these details in the next few chapters.

1.3 Scope of the Thesis

There are many existing robust algorithms that achieve training robustness against different types of data corruption. However, most of them are not designed for deep neural networks. The scope of this thesis focuses on the problem of training robust DNN. Note that there is another branch of research that focuses on robustness in testing. Testing robustness assumes the training data is clean, but the test data has been adversarially corrupted. Even though this is also a huge research topic, this thesis mainly focuses on solving the robustness issue in training. We leave the testing robustness as a subject for future research.

1.4 Thesis Contributions

The goal of this thesis is to develop robust learning algorithms for DNN against both natural and adversarial corruptions in the training data. A good robust learning algorithm for DNN must have the following characteristics. First, the algorithm should be able to tolerate/detect corrupted data both empirically and theoretically. Second, the underlying assumptions of the algorithm should not be too strong. Finally, the algorithm itself should be scalable to DNNs, with large number of learning parameters. In the following sections, I will introduce several robust learning algorithms that satisfy these characteristics. First, I have developed RCA (Robust Collaborative Autoencoder), a DNN-based unsupervised anomaly detection algorithm to detect natural corruption. Second, I introduce a robust algorithm called PRL (Provable Robust Learning) to train DNN to defend against adversarial label corruption. Third, I propose a robust algorithm called RobustRealNVP (Robust real-valued non-volume preserving transformations) for learning the density of data given adversarial corruption. At last, I proposed a framework that connects the backdoor attack and noisy label attack. The framework enables a robust meta-algorithm that can defend against strong backdoor attacks.

1.5 Publications

Some content in this dissertation was adapted from the following works:

- Liu, B., Wang, D., Lin, K., Tan, P. N., & Zhou, J. (2021, August). RCA: A Deep Collaborative

Autoencoder Approach for Anomaly Detection. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21).

- Liu, B., Tan, P. N., & Zhou, J. (2022, April). Unsupervised Anomaly Detection by Robust Density Estimation. In Proceedings of the AAAI conference on artificial intelligence (AAAI-22).

- Liu, B., Sun, M., Wang, D., Tan, P. N., & Zhou, J. (2021, July). Learning Deep Neural Networks under Agnostic Corrupted Supervision. In International Conference on Machine Learning (pp. 6957-6967). PMLR.

- Liu, B., Zhu, Z., Tan, P. N., & Zhou, J. (2021). Defending Backdoor Data Poisoning Attacks by Using Noisy Label Defense Algorithm (in preparation).

1.6 Thesis Outline

In chapter 2 and chapter 3, we will introduce two anomaly detection algorithm: robust collaborative autoencoder (RCA) and Robust RealNVP, to defend against natural corruption. In chapter 4, we introduce provable robust learning (PRL), a robust algorithm to defend against agnostic supervision corruption. In chapter 5, we introduce a meta framework, which draws a principle connection between noisy label attack and backdoor attack. The connection enables a robust algorithm, which can effectively defend against backdoor data poisoning attack. Finally, conclusion and future study are discussed in chapter 6.

CHAPTER 2

ROBUST COLLABORATIVE AUTOENCODER (RCA)

In this chapter, we introduce the Robust Collaborative Autoencoder, a robust unsupervised anomaly detection algorithm which could alleviate the training on corrupted data issue.

2.1 Introduction

Anomaly detection (AD) is the task of identifying unusual or abnormal observations in the data. It has been widely used in many applications, including credit fraud detection, malware detection, and medical diagnosis. Current approaches can be divided into supervised or unsupervised learning methods. Supervised AD requires labeled examples to train the AD models whereas unsupervised AD, which is the focus of our paper, does not require label information. Instead, unsupervised AD implicitly assumes there are more normal than anomalous instances in the data Chandola et al. (2009). Deep autoencoders are perhaps one of the most widely used unsupervised AD methods Chandola et al. (2009); Sakurada and Yairi (2014); Vincent et al. (2010). An autoencoder compresses the original data by learning its hidden representation in a way that minimizes the reconstruction loss. It is based on the working assumption that normal observations are easier to compress than anomalies. Unfortunately, such an assumption does not generally hold for DNNs, which are often over-parameterized and have the capability to fit well even to the anomalies Zhang et al. (2016). Thus, the DNN-based unsupervised AD methods must consider the trade-off between model capacity and overfitting to the anomalies to achieve good performance.

Our work is motivated by recent progress on robustness of DNNs for noisy labeled data by learning the weights of the samples during training Jiang et al. (2017); Han et al. (2018). For unsupervised AD, our goal is to learn the weights in such a way that normal observations are assigned higher weights than anomalies when calculating reconstruction error. The weights can be used to reduce the influence of anomalies when updating the model for learning a feature representation of the data. However, existing approaches for weight learning are inapplicable to

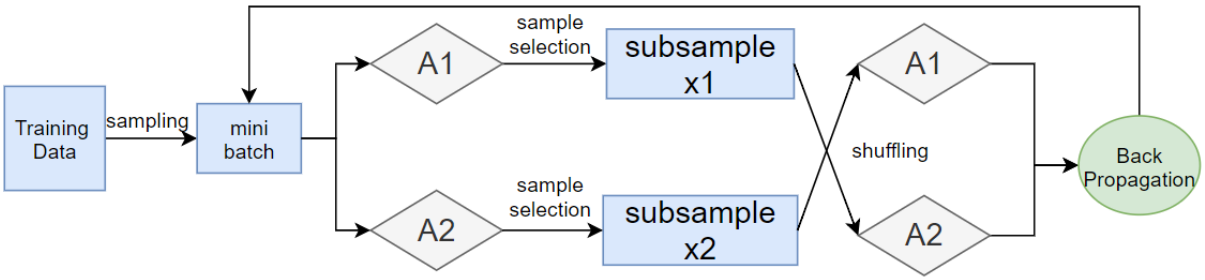


Figure 2.1: An illustration of the training phase for the proposed RCA framework.

unsupervised AD as they require label information. To address this challenge, we propose a robust collaborative autoencoders (RCA) method that trains a set of autoencoders in a collaborative fashion and jointly learns their model parameters and sample weights. Specifically, given a mini-batch, each autoencoder would learn a feature representation and selects a subset of the samples with lowest reconstruction errors. By discarding samples with high reconstruction errors, the learning algorithm will be more focused on fitting the clean data, thereby reducing its risk of memorizing anomalies. However, by selecting only easy-to-fit samples, this may lead to premature convergence of the algorithm without sufficient exploration of the loss surface. To address this issue, each autoencoder will shuffle its selected samples to another autoencoder, who will use the exchanged samples to update its model weights. The sample selection and shuffling procedure is illustrated in Figure 2.1. During the testing phase, we apply a dropout mechanism to produce multiple output predictions for each test point by repeating the forward pass multiple times. These ensemble of outputs are then aggregated to obtain a final robust estimate of the anomaly score.

The main contributions of this paper are as follows. First, we present a framework for unsupervised deep AD using robust collaborative autoencoders (RCA) to prevent model overfitting due to anomalies. Second, we provide theoretical analysis to understand the mechanism behind RCA. Our analysis shows that the worst-case scenario for RCA is better than conventional autoencoders and provides the conditions under which RCA will detect the anomalies. Third, we show that RCA outperforms state-of-the-art unsupervised AD methods for the majority of the datasets used in this study, even if there are missing values present in the data. In addition, RCA also enhances the

performance of more advanced autoencoders such as variational autoencoders in unsupervised AD tasks.

2.2 Related Work

Many methods have been developed over the years for unsupervised AD Chandola et al. (2009). Reconstruction-based methods, such as principal component analysis (PCA) and autoencoders, project the input data to a lower-dimensional manifold before transforming it back to the original feature space. The distances between the input and reconstructed data is used as anomaly scores of the data points. More recently, Zhou and Paffenroth (2017) combined robust PCA with an autoencoder to decompose the data into a mixture of normal and anomaly parts. Zong et al. (2018) jointly learned a low dimensional embedding and density of the data, using the density of each point as its anomaly score while Ruff et al. (2018) extended the traditional one-class SVM approach to a deep learning setting. Huang et al. (2019) uses self-supervised learning to perform semi-supervised anomaly detection for images. Wang et al. (2019b) applied end-to-end self-supervised learning to unsupervised AD. However, their approach is only applicable to image data as it requires augmentation operations such as rotation and patch reshuffling.

Current deep AD methods will not prevent the network from incorporating anomalies into their learned representation. One way to address the issue is by assigning a weight to each data point. For example, in self-paced learning Kumar et al. (2010), the algorithm assigns higher weights to easier-to-classify examples and lower weights to harder ones. This strategy was also adopted by other supervised methods for learning from noisy labeled data, including mentornet Jiang et al. (2017) and co-teaching Han et al. (2018). Extending the weight learning methods to unsupervised AD is a key novelty of our work. Theoretical studies on the benefits of choosing samples with smaller loss to drive the optimization algorithm can be found in Shen and Sanghavi (2018); Shah et al. (2020).

2.3 Methodology

This section introduces the RCA framework and analyzes its theoretical properties. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the input data, where n is the number of observations and d is the number of features. Our goal is to classify each $x_i \in \mathbf{X}$ as an anomaly or a normal observation. Let $O \subset \mathbf{X}$ be the set of true anomalies in the data and $\epsilon = |O|/n$ be the anomaly ratio, which is determined based on the amount of suspected anomalies in the data or the proportion the user is willing to inspect and verify.

In the RCA framework, we train a set of k autoencoders with different initializations. For brevity, we assume $k = 2$ even though RCA is applicable to more than 2 autoencoders. In each iteration during training, the autoencoders will each apply a forward pass on a mini-batch randomly sampled from the training data and compute the reconstruction error of each data point in the mini-batch. Each autoencoder will then sort the data points according to their reconstruction errors and selects the points with lowest reconstruction error to be exchanged with another autoencoder. Each autoencoder subsequently performs a back-propagation step to update its model parameters using the samples it receives from another autoencoder. Upon convergence, the averaged reconstruction error of each data point is treated as its overall anomaly score. A pseudocode for RCA with $k = 2$ autoencoders is shown in Algorithm 1.

RCA differs from conventional autoencoders in several ways. First, its autoencoders are trained with a subset of data points having low reconstruction errors. The selected points are then exchanged among the autoencoders to avoid premature convergence. During the testing phase, each autoencoder applies a dropout mechanism to generate multiple outputs. The averaged ensemble output is used as the predicted anomaly score. Details of these steps are given next.

2.3.0.1 Sample Selection

We present theoretical results to motivate our sample selection approach. Specifically, we demonstrate the robustness of RCA against contamination (anomalies) in the training data by showing that RCA converges to a similar solution as if it had been trained on clean (normal) data without anomalies.

Algorithm 1: Robust Collaborative Autoencoders

input: training data $\mathbf{X}_{\text{trn}}$, test data $\mathbf{X}_{\text{tst}}$, anomaly ratio ϵ, dropout rate r, decay rate α, and max_epoch for training
initialize autoencoders $\mathcal{A}_1$ and $\mathcal{A}_2$; sample selection $\beta = 1$;
Training Phase

while epoch $\leq$ max_epoch **do**
  for minibatch $\mathbf{S}$ in $\mathbf{X}_{\text{trn}}$ **do**
    $\hat{\mathbf{S}}_1 \leftarrow \text{forward}(\mathcal{A}_1, \mathbf{S}, dropout = 0)$, $\hat{\mathbf{S}}_2 \leftarrow \text{forward}(\mathcal{A}_2, \mathbf{S}, dropout = 0)$
    $\mathbf{c}_1 \leftarrow \text{sample\_selection}(\hat{\mathbf{S}}_1, \mathbf{S}, \beta)$, $\mathbf{c}_2 \leftarrow \text{sample\_selection}(\hat{\mathbf{S}}_2, \mathbf{S}, \beta)$
    $\hat{\mathbf{S}}_1 \leftarrow \text{forward}(\mathcal{A}_1, \mathbf{S}[\mathbf{c}_2], dropout = r)$, $\hat{\mathbf{S}}_2 \leftarrow \text{forward}(\mathcal{A}_2, \mathbf{S}[\mathbf{c}_1], dropout = r)$
    $\mathcal{A}_1 \leftarrow \text{backprop}(\hat{\mathbf{S}}_1, \mathbf{S}[\mathbf{c}_2], dropout = r)$, $\mathcal{A}_2 \leftarrow \text{backprop}(\hat{\mathbf{S}}_2, \mathbf{S}[\mathbf{c}_1], dropout = r)$
  end for
  $\beta = \max(\beta - \dfrac{\epsilon}{\alpha \times \text{max\_epoch}}, 1 - \epsilon)$
end while
$\mathcal{A}_1^* = \mathcal{A}_1$ and $\mathcal{A}_2^* = \mathcal{A}_2$
Testing Phase
$\xi = []$
for i = 1 **to** v **do**
  $\xi_1 = \text{forward}(\mathcal{A}_1^*, \mathbf{X}_{\text{tst}}, dropout = r)$
  $\xi_2 = \text{forward}(\mathcal{A}_2^*, \mathbf{X}_{\text{tst}}, dropout = r)$
  $\xi.\text{append}((\xi_1 + \xi_2)/2)$
end for
return $anomaly\_score = average(\xi)$

Next, we show that RCA is better than vanilla SGD when the anomaly ratio is large or when the anomalies are very different from normal data. Finally, we show that RCA will correctly select all the normal points under some convexity assumption.

Given a mini-batch, $\mathbf{X}_m \subset \mathbf{X}$, our sample selection procedure chooses a subset of points with lowest reconstruction error as "clean" samples to update the parameters of the autoencoder. The selected points may vary from one iteration to another depending on which subset of points are in the mini-batch and which points have lower reconstruction error within the mini-batch. To avoid discarding the data points prematurely, we use a linear decay function from $\beta = 1$ (all points within the mini-batch are chosen) until $\beta = 1 - \epsilon$ to gradually reduce the proportion of selected samples (see last line of training phase in Algorithm 1). The rationale for this approach is that we observe the autoencoders to overfit the anomalies only when the number of training epochs is large.

Let k be the mini-batch size and $\mathbf{w}$ be the current parameter of an autoencoder. Our algorithm selects $(\beta \times 100)\%$ of the data points with lowest reconstruction errors in the mini-batch to update the autoencoder. Let $p_i(\mathbf{w})$ be the probability that a data point with i^{th} smallest reconstruction error (among all n points in the entire dataset) is chosen to update the parameters of the autoencoder.

Assuming sampling without replacement, we consider two cases: $i \leq \beta k$ and $i > \beta k$. In the first case, the data point with i^{th} smallest error will be selected as long as it is in the mini-batch. In the second case, the point is chosen only if it is part of the mini-batch and has among the (βk)-th lowest errors in the mini-batch:

$$p_i(\mathbf{w}) = \begin{cases} \frac{\binom{n-1}{k-1}}{\binom{n}{k}} = \frac{k}{n} & \text{if } i \leq \beta k, \\ \frac{\sum_{j=0}^{\beta k-1} \binom{i-1}{j}\binom{n-i}{k-j-1}}{\binom{n}{k}} & \text{otherwise.} \end{cases} \tag{2.1}$$

The objective function for the autoencoder with sample selection can thus be expressed as follows:

$$\min_{\mathbf{w}} \hat{F}(\mathbf{w}) = \sum_{i=1}^{n} p_i(\mathbf{w}) f(\mathbf{x}_i, \mathbf{w}) \equiv \sum_{i=1}^{n} p_i(\mathbf{w}) f_i(\mathbf{w})$$

where $f(\mathbf{x}_i, \mathbf{w}) = f_i(\mathbf{w})$ is the individual reconstruction loss for $\mathbf{x}_i$. Suppose $\Omega(\mathbf{w}^*_{sr})$ is the set of stationary points for $\hat{F}(\mathbf{w})$ and $\Omega_i(\mathbf{w}^*)$ is the corresponding set of stationary points for each individual loss, $f_i(\mathbf{w})$. Let $F(\mathbf{w}) = \sum_{i \notin O} f_i(\mathbf{w})$ be the "clean" objective function, where anomalies have been excluded from the training data and $\Omega(\mathbf{w}^*)$ be its set of stationary points. Furthermore, let $\tilde{F}(w) = \sum_{i=1}^{n} f_i(\mathbf{w})$ be the objective function if no sample selection is performed and $\Omega(\mathbf{w}^*_{ns})$ is its corresponding set of stationary points.

Our analysis on properties of RCA is based on the following assumptions:

Assumption 1 (Gradient Regularity) $\max_{i,\mathbf{w}} \|\nabla f_i(\mathbf{w})\| \leq G$.

Assumption 2 (Individual L-smooth) *For every individual loss f_i, $\forall p, q : \|\nabla f_i(\mathbf{w}_p) - \nabla f_i(\mathbf{w}_q)\| \leq L_i \|\mathbf{w}_p - \mathbf{w}_q\|$.*

Assumption 3 (Equal Minima) *Same minimum value for every individual loss: $\forall i, j : \min_{\mathbf{w}} f_i(\mathbf{w}) = \min_{\mathbf{w}} f_j(\mathbf{w})$.*

Assumption 4 (Individual Strong Convexity) *For every individual loss f_i, $\forall p, q : \|\nabla f_i(\mathbf{w}_p) - \nabla f_i(\mathbf{w}_q)\| \geq \mu_i \|\mathbf{w}_p - \mathbf{w}_q\|$.*

We denote $L_{\max} = \max_i(L_i)$, $L_{\min} = \min_i(L_i)$, $\mu_{\max} = \max_i(\mu_i)$, and $\mu_{\min} = \min_i(\mu_i)$. Since $F(\mathbf{w})$ is the sum over the loss for clean data, it is easy to see that Assumption 2 implies $F(\mathbf{w})$ is

$n(1-\epsilon)L_{\max}$ smoothness, while Assumption 4 implies that $F(\mathbf{w})$ is $n(1-\epsilon)\mu_{\min}$ convex. We thus define $M = n(1-\epsilon)L_{\max}$, and $m = n(1-\epsilon)\mu_{\min}$.

Remark 1 *Assumptions 1 and 2 are commonly used in non-convex optimization. Assumption 3 is not a strong assumption in an over-parameterized DNN setting Zhang et al. (2016). While Assumption 4 is perhaps the strongest assumption, it is only needed to demonstrate correctness of our algorithm (Theorem 3). A similar convex assumption was used in Shah et al. (2020) to prove the correctness of their algorithm.*

We define the constants $\delta > 0$ and $\phi \geq 1$ as follows:

$$\delta \geq \max_{\mathbf{x}\in\Omega_i(\mathbf{w}^*),\mathbf{y}\in\Omega(\mathbf{w}^*)} \|\mathbf{x}-\mathbf{y}\|, \forall i \notin O \tag{2.2}$$

$$\delta \leq \min_{\mathbf{z}\in\Omega_j(\mathbf{w}^*),\mathbf{y}\in\Omega(\mathbf{w}^*)} \|\mathbf{z}-\mathbf{y}\|, \forall j \in O,$$

$$\max_{\mathbf{z}\in\Omega_j(\mathbf{w}^*),\mathbf{y}\in\Omega(\mathbf{w}^*)} \|\mathbf{z}-\mathbf{y}\| \leq \phi\delta, \quad \forall j \in O. \tag{2.3}$$

Note that under the convex assumption, the above equations reduce to: $\|\mathbf{w}_i^* - \mathbf{w}^*\| \leq \delta \leq \|\mathbf{w}_j^* - \mathbf{w}^*\| \leq \phi\delta, \quad \forall i \notin O, \ \forall j \in O$. These inequalities provide bounds on the distance between $\mathbf{w}_j^*$ of anomalies and $\mathbf{w}^*$ for clean data.

First, based on Assumptions 1 and 2, the following theorem shows that optimizing $\hat{F}(\mathbf{w})$ will give a C-approximate solution to $\Omega(\mathbf{w}^*)$.

Theorem 1 *Let $F(\mathbf{w}) = \sum_{i\notin O} f_i(\mathbf{w})$ be a twice differentiable function. Consider the sequence $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \cdots, \mathbf{w}^{(t)}$ generated by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)}\nabla_{\mathbf{w}^{(t)}}\hat{F}(\mathbf{w}^{(t)})$ and let $\max_{\mathbf{w}^{(t)}} \|\nabla_{\mathbf{w}^{(t)}}F(\mathbf{w}^{(t)}) - \nabla_{\mathbf{w}^{(t)}}\hat{F}(\mathbf{w}^{(t)})\|^2 = C$. Based on Assumptions 1 and 2, if $\sum_{t=1}^{\infty}\eta^{(t)} = \infty$, $\sum_{t=1}^{\infty}\eta^{(t)^2} \leq \infty$, then $\min_{t=0,1,\cdots,T} \|\nabla F(\mathbf{w}^{(t)})\|^2 \to C$ as $T \to \infty$.*

Remark 2 *The theorem shows how the presence of anomalies in training data affects the gradient norm of the clean objective function. If the training data has no anomalies and $p_i = \frac{1}{N}$, then $C = 0$. When data is noisy, there is no guarantee that $C = 0$. Instead, C is controlled by the choice of p_i.*

Since $\|\nabla F(\mathbf{w}^*)\| = 0$, Theorem 1 shows our sample selection method enables convergence to a C-approximate solution of the objective function for clean data. The theorem below compares our solution against the solution found when trained on the entire data (with no sample selection).

Theorem 2 *Let $F(\mathbf{w}) = \sum_{i \notin O} f_i(\mathbf{w})$ be a twice differentiable function with a bound C defined in Theorem 1. Consider the sequence $\{\mathbf{w}_{RCA}\}$ generated by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \nabla_{\mathbf{w}^{(t)}} \hat{F}(\mathbf{w}^{(t)})$. Based on Assumptions 1 and 2 and assume $C \leq (\min(n\epsilon G, M\delta))^2$, if $\sum_{t=1}^{\infty} \eta^{(t)} = \infty$, $\sum_{t=1}^{\infty} \eta^{(t)^2} \leq \infty$, then there exists a large enough T and $\tilde{\mathbf{w}}_{ns} \in \Omega(\mathbf{w}^*_{ns})$ such that $\min_{t=0,1,\ldots,T} \|\nabla F(\mathbf{w}^{(t)}_{RCA})\| \leq \|\nabla F(\tilde{\mathbf{w}}_{ns})\|$.*

Remark 3 *The above theorem is for worst case bound. A similar result is given in Shah et al. (2020) but with a stronger convex assumption. Although it is for worst case scenario, our experiments show that our sample selection method generally outperforms DNN methods that use all the data. The condition $C \leq (\min(n\epsilon G, M\delta))^2$ will more likely hold when the anomaly ratio ϵ is large or when δ, distance between normal data and anomalies, is large, consistent with our expectation.*

Below we give a sufficient condition for guaranteeing correctness when Assumption 4 holds. Suppose $\forall i \notin O : f_i(\mathbf{w}^*) = 0$ and $\forall j \in O : f_j(\mathbf{w}^*) > 0$. Assuming $f(\mathbf{w})$ is convex and its gradient is upper bounded, let $\mathcal{B}_r(\boldsymbol{w}^*) = \left\{\boldsymbol{w} \mid f_i(\boldsymbol{w}) < f_j(\boldsymbol{w}), \forall i \notin \mathbb{O}, j \in \mathbb{O}, \|\boldsymbol{w} - \boldsymbol{w}^*\| \leq r\right\}$. $\mathcal{B}_r(\boldsymbol{w}^*)$ describes a ball of radius $r > 0$ around the optimal point for which normal observations have a smaller loss than anomalies. The following theorem describes a sufficient condition for our algorithm to converge within the ball.

Theorem 3 *Let $F(\mathbf{w}) = \sum_{i \notin O} f_i(\mathbf{w})$ be a twice differentiable function and $\kappa = \sqrt{\frac{L^c_{\max}}{\mu^o_{\min}}}$, where $L^c_{max} = \max_{i \notin O}(L_i)$ is the maximum Lipschitz smoothness for clean data and $\mu^o_{min} = \min_{j \in O}(\mu_j)$ is the minimum convexity for anomalies. Consider the sequence $\{\mathbf{w}_{RCA}\}$ generated by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \nabla_{\mathbf{w}^{(t)}} \hat{F}(\mathbf{w}^{(t)})$ and assume $\max_{\mathbf{w}^{(t)}} \|\nabla_{\mathbf{w}^{(t)}} F(\mathbf{w}^{(t)}) - \nabla_{\mathbf{w}^{(t)}} \hat{F}(\mathbf{w}^{(t)})\|^2 = C$. Based on Assumptions 1-4, if $\sum_{t=1}^{\infty} \eta^{(t)} = \infty$, $\sum_{t=1}^{\infty} \eta^{(t)^2} \leq \infty$, and $C \leq \left(\frac{\delta}{(1+\kappa)m}\right)^2 = \mathbb{O}\left(\frac{\delta}{\kappa}\right)^2$, then there exists $r > 0$ such that $\mathbf{w}^*_{sr} \in \mathcal{B}_r(\boldsymbol{w}^*)$.*

The proof is given in the supplementary materials. This guarantee depends on having a sufficiently small C, which is related to δ, the nearest distance between anomalies and the normal points, as well as the landscape of the loss surface κ. A small κ suggests that the loss surface will be very sharp for anomalies (large μ^o_{min}) but flat for normal data (small L^c_{max}). In this case, most regions in the loss surface will have smaller loss on the normal data and larger loss on the anomalies (under assumption of equal minima). As a result, the anomalies have smaller probability to be selected than normal points by our proposed algorithm since they have larger loss.

The above analysis shows that sample selection helps our method to have better convergence to the stationary points for clean data. Nevertheless, our ultimate goal is to improve test performance, not just convergence to stationary points of clean data. When sample selection is applied to just one autoencoder, the algorithm may converge too quickly as we use only samples with low reconstruction loss to compute the gradient, making it susceptible to overfitting Zhang et al. (2016). Thus, instead of using only the self-selected samples for model update, we train the autoencoders collaboratively and shuffle the selected samples between them to avoid overfitting. A similar strategy has proven to be effective in supervised learning for data with noisy labels Han et al. (2018).

2.3.0.2 Ensemble Evaluation

Unsupervised AD using an ensemble of model outputs has been shown to be highly effective in previous studies Liu et al. (2008); Zhao et al. (2019); Emmott et al. (2015); Aggarwal and Sathe (2017). However, incorporating ensemble method into deep learning is challenging as it is expensive to train a large number of DNNs. In RCA, we use dropout Srivastava et al. (2014) to emulate the ensemble process. Dropouts are typically used during training to avoid model overfitting. In RCA, we employ the dropout mechanism during testing as well. Specifically, we use many networks of perturbed structures to perform multiple forward passes over the data in order to obtain a set of reconstruction errors for each test point. The final anomaly score is computed by averaging the reconstruction errors. Although dropout may increase reconstruction error, we expect a more robust estimation of the ranking information for anomaly score using this procedure.

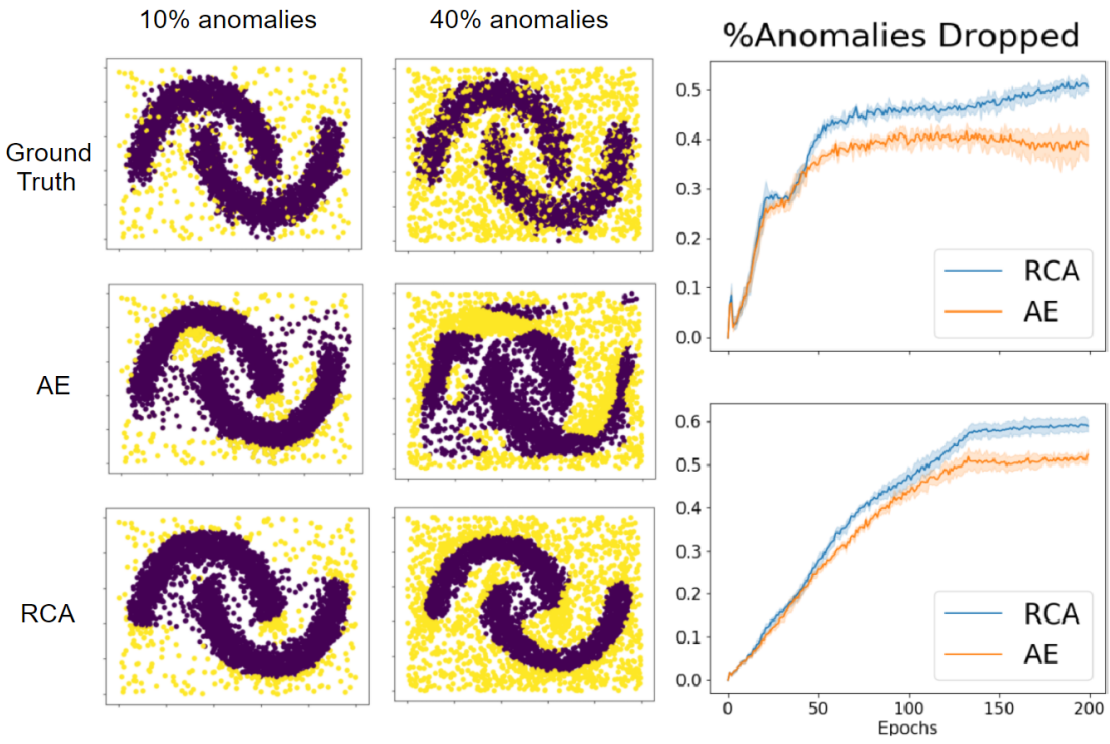


Figure 2.2: The first two columns are results for 10% and 40% anomaly ratio, respectively. The last column shows the fraction of points with highest reconstruction loss that are true anomalies. The top diagram is for 10% anomalies while the bottom is for 40% anomalies.

2.4 Experiment

We performed extensive experiments to compare the performance of RCA against various baseline methods and to investigate its robustness to noise due to missing value imputation. The code is attached with the supplementary materials.

2.4.0.1 Results on Synthetic Data

To better understand how RCA overcomes the limitations of conventional autoencoders (AE), we created a synthetic dataset containing a pair of crescent-shaped moons with Gaussian noise Pedregosa et al. (2011) representing the normal observations and anomalies generated from a uniform distribution. In this experiment, we vary the proportion of anomalies from 10% to 40% while fixing the sample size to be 10,000. Samples with the top-$[(1-\epsilon)n]$ highest anomaly scores are classified as anomalies, where ϵ is the anomaly ratio.

Table 2.1: Performance comparison of RCA against baseline methods in terms of their average and standard deviation of AUC scores across 10 random initializations.

	RCA	VAE	AE	SO_GAAL	DAGMM	Deep-SVDD	OCSVM	IF
vowels	**0.917±0.016**	0.503±0.045	0.879±0.020	0.637±0.197	0.340±0.103	0.206±0.035	0.765±0.036	0.768±0.013
pima	**0.711±0.016**	0.648±0.015	0.669±0.013	0.613±0.049	0.531±0.025	0.395±0.034	0.594±0.026	0.662±0.018
optdigits	0.890±0.041	**0.909±0.016**	0.907±0.010	0.487±0.138	0.290±0.042	0.506±0.024	0.558±0.009	0.710±0.041
sensor	**0.950±0.030**	0.913±0.003	0.866±0.050	0.557±0.224	0.924±0.085	0.614±0.073	0.939±0.002	0.948±0.002
letter	0.802±0.036	0.521±0.042	**0.829±0.031**	0.601±0.060	0.433±0.034	0.465±0.039	0.557±0.038	0.643±0.040
cardio	0.905±0.012	**0.944±0.006**	0.867±0.020	0.473±0.075	0.862±0.031	0.505±0.056	0.936±0.002	0.927±0.006
arrhythmia	0.806±0.044	**0.811±0.034**	0.802±0.044	0.538±0.042	0.603±0.095	0.635±0.063	0.782±0.028	0.802±0.024
breastw	0.978±0.003	0.950±0.006	0.973±0.004	0.980±0.011	0.976±0.000	0.406±0.037	0.955±0.006	**0.983±0.008**
musk	**1.000±0.000**	0.994±0.002	0.998±0.003	0.234±0.193	0.903±0.130	0.829±0.048	**1.000±0.000**	0.995±0.006
mnist	**0.858±0.012**	0.778±0.009	0.802±0.009	0.795±0.025	0.652±0.077	0.538±0.048	0.835±0.012	0.800±0.013
satimage-2	0.977±0.008	0.966±0.008	0.818±0.069	0.789±0.177	0.853±0.113	0.739±0.088	**0.998±0.003**	0.996±0.004
satellite	**0.712±0.011**	0.538±0.016	0.575±0.068	0.640±0.070	0.667±0.189	0.631±0.016	0.650±0.014	0.700±0.031
mammography	0.844±0.014	0.864±0.014	0.853±0.015	0.204±0.026	0.834±0.000	0.272±0.009	**0.881±0.015**	0.873±0.021
thyroid	0.956±0.008	0.839±0.011	0.928±0.020	**0.984±0.005**	0.582±0.095	0.704±0.027	0.960±0.006	0.980±0.006
annthyroid	0.688±0.016	0.589±0.021	0.675±0.022	0.679±0.022	0.506±0.020	0.591±0.014	0.599±0.013	**0.824±0.009**
ionosphere	**0.846±0.015**	0.763±0.015	0.821±0.010	0.783±0.080	0.467±0.082	0.735±0.053	0.812±0.039	0.843±0.020
pendigits	0.856±0.011	0.931±0.006	0.685±0.073	0.257±0.053	0.872±0.068	0.613±0.071	0.935±0.003	**0.941±0.009**
shuttle	0.935±0.013	0.987±0.001	0.921±0.013	0.571±0.316	0.890±0.109	0.531±0.290	0.985±0.001	**0.997±0.001**
glass	**0.998±0.000**	0.626±0.134	0.570±0.152	0.420±0.112	0.852±0.084	0.756±0.114	0.522±0.207	0.706±0.058

Figure 2.2 compares the performance of standard autoencoders (AE) against RCA for 10% (left column) and 40% (right column) anomaly ratio[1]. Although the performance for both methods degrades with increasing anomaly ratio, RCA is more robust compared to AE. In particular, when the anomaly ratio is 40%, AE fails to capture the true manifold of the normal data, unlike RCA. This result is consistent with the assertion in Theorem 2, which states that training the autoencoder with a subset of points selected by RCA is better than using all the data when anomaly ratio is large.

2.4.0.2 Results on Real-World Data

For evaluation, we used 19 benchmark datasets obtained from the Stony Brook ODDS library Rayana (2016)[2]. We reserve 60% of the data for training and the remaining 40% for testing. The performance of the competing methods are evaluated based on their Area under ROC curve (AUC) scores.

Baseline Methods We compared RCA against the following baseline methods: **Deep-SVDD** (deep one-class SVM) Ruff et al. (2018), **VAE** (Variational autoencoder) Kingma and Welling (2013a); An and Cho (2015), **DAGMM** (deep gaussian mixture model) Zong et al. (2018), **SO-GAAL** (Single-Objective Generative Adversarial Active Learning) Liu et al. (2019), **OCSVM** (one-class

[1]Results for 20% and 30% are in the supplementary materials.

[2]Additional experimental results on the CIFAR10 dataset are given in the supplementary materials.

SVM) Chen et al. (2001), and **IF** (isolation forest) Liu et al. (2008). Note that Deep-SVDD and DAGMM are two recent deep AD methods while OCSVM and IF are state-of-the-art unsupervised AD methods. In addition, we also perform an ablation study to compare RCA against its four variants: **AE** (standard autoencoders without collaborative networks) and **RCA-E** (RCA without ensemble evaluation), and **RCA-SS** (RCA without sample selection). To ensure fair comparison, we maintain similar hyperparameter settings for all the competing DNN-based approaches. Experimental results are reported based on their average AUC scores across 10 random initializations. More discussion about our experimental setting is given in supplementary materials.

Performance Comparison Table 2.1 summarizes the results of our experiments. Note that RCA outperforms all the deep unsupervised AD methods (SO-GAAL, DAGMM, Deep-SVDD) in 17 out of 19 datasets. RCA also performs better than both AE and VAE in 12 out of the 19 datasets, IF in 11 of the datasets, and OCSVM in 12 of the datasets. These results suggest that RCA clearly outperforms the baseline methods on majority of the datasets. Surprisingly, some of the complex DNN baselines such as SO-GAAL, DAGMM, and Deep-SVDD perform poorly on the datasets. This is because most of these DNN methods assume the availability of clean training data, whereas in our experiments, the training data are contaminated with anomalies to reflect a more realistic setting. Furthermore, we use the same network architecture for all the DNN methods (including RCA), since there is no guidance on how to best tune the network structure given that it is an unsupervised AD task.

RCA for Missing Values As real-world datasets are often imperfect, we compare the performance of RCA and other baseline methods in terms of their robustness to missing values. Mean imputation is a common approach to deal with missing values. In this experiment, we add missing values randomly in the features of each benchmark dataset and apply mean imputation to replace the missing values. Such imputation will likely introduce noise into the data. We vary the percentage of missing values from 10% to 50% and compare the average AUC scores of the competing methods. We omit the detailed results due to lack of space. Instead, we report only the number of wins, draws,

Table 2.2: Comparison of RCA against baseline methods in terms of (#win-#draw-#loss) on 19 benchmark datasets with different proportion of imputed missing values in the data. Results for RCA-E (no ensemble), RCA-SS (no sample selection), AE (no ensemble and no sample selection) are for ablation study.

Missing Ratio	RCA-E	RCA-SS	VAE	SO-GAAL	AE	DAGMM	Deep-SVDD	OCSVM	IF
0.0	12-2-5	17-0-2	13-0-6	17-0-2	16-0-3	18-0-1	19-0-0	12-1-6	11-0-8
0.1	13-1-5	15-1-3	17-1-1	17-0-2	15-0-4	18-0-1	19-0-0	14-1-4	13-0-6
0.2	12-1-6	14-3-2	15-2-2	18-0-1	14-0-5	19-0-0	19-0-0	16-0-3	10-0-9
0.3	10-3-6	14-1-4	16-0-3	18-1-0	14-0-5	19-0-0	19-0-0	17-0-2	15-1-3
0.4	11-0-8	13-2-4	15-0-4	16-0-3	13-0-6	18-0-1	19-0-0	17-0-2	16-0-3
0.5	9-3-7	11-1-7	12-1-6	15-0-4	10-0-9	16-0-3	18-0-1	15-1-3	14-0-5

and losses of RCA compared to each baseline method on the 19 benchmark datasets in Table 2.2. RCA was found to consistently outperform both DAGMM and Deep-SVDD by more than 80%, demonstrating the robustness of our algorithm compared to other deep unsupervised AD methods when training data is contaminated. Additionally, as the missing ratio increases to more than 30%, it outperforms IF and OCSVM by more than 70% on the datasets. The results suggest that our framework is better than the baselines on the majority of the datasets in almost all settings.

Ablation Study We have also performed an ablation study to investigate the effectiveness of using sample selection and ensemble evaluation. The results comparing RCA against its variants, RCA-E, RCA-SS, and AE are given in Table 2.2. Without missing value imputation, RCA outperformed all the variants in at least 12 of the datasets. The advantage of RCA over its variants, AE, RCA-SS, and RCA-E, reduces with increasing amount of noise due to missing value imputation but is still significant until the missing ratio is 50%.

Sensitivity Analysis RCA requires users to specify the anomaly ratio of the data. Since the true anomaly ratio ϵ is often unknown, we conducted experiments to evaluate the robustness of RCA when ϵ is overestimated or underestimated by 5% and 10%. from their true values on all datasets. The results in Table 2.3 suggest that the AUC scores for RCA do not change significantly even when the anomaly ratio was overestimated or underestimated by 10% on most of the datasets.

Table 2.3: Average and standard deviation of AUC scores (for 10 random initialization) as anomaly ratio parameter is varied from ϵ (i.e., true anomaly ratio of the data) to $\epsilon + \Delta\epsilon$. If ϵ is less than 0.05 or 0.1, then $\Delta\epsilon = 0.05$ and $\Delta\epsilon = 0.1$ will be truncated to 0.

	$\Delta\epsilon = 0.05$	$\Delta\epsilon = 0.1$	$\Delta\epsilon = 0$	$\Delta\epsilon = -0.05$	$\Delta\epsilon = -0.1$
vowels	0.918±0.016	0.920±0.015	0.917±0.016	0.908±0.019	0.908±0.019
pima	0.719±0.016	0.721±0.014	0.711±0.016	0.704±0.014	0.697±0.017
optdigits	0.973±0.009	0.980±0.010	0.890±0.041	0.861±0.040	0.861±0.040
sensor	0.876±0.000	0.876±0.000	0.950±0.030	0.913±0.000	0.913±0.000
letter	0.793±0.033	0.796±0.034	0.802±0.036	0.802±0.040	0.802±0.037
cardio	0.923±0.015	0.947±0.018	0.905±0.012	0.860±0.021	0.851±0.021
arrhythmia	0.807±0.044	0.807±0.044	0.806±0.044	0.806±0.043	0.806±0.043
breastw	0.981±0.002	0.983±0.002	0.978±0.003	0.973±0.006	0.970±0.006
musk	1.000±0.000	1.000±0.000	1.000±0.000	0.809±0.058	0.809±0.058
mnist	0.852±0.015	0.840±0.016	0.858±0.012	0.851±0.013	0.847±0.014
satimage-2	0.998±0.001	0.998±0.001	0.977±0.008	0.965±0.009	0.965±0.009
satellite	0.701±0.015	0.688±0.018	0.712±0.011	0.718±0.008	0.713±0.011
mammography	0.854±0.018	0.840±0.018	0.844±0.014	0.838±0.011	0.838±0.011
thyroid	0.959±0.008	0.957±0.007	0.956±0.008	0.949±0.011	0.949±0.011
annthyroid	0.693±0.019	0.689±0.022	0.688±0.016	0.683±0.016	0.669±0.019
ionosphere	0.844±0.012	0.841±0.013	0.846±0.015	0.855±0.017	0.862±0.016
pendigits	0.858±0.014	0.847±0.020	0.856±0.011	0.858±0.014	0.858±0.014
shuttle	0.949±0.037	0.994±0.001	0.935±0.013	0.956±0.009	0.958±0.006
glass	0.945±0.000	0.946±0.000	0.998±0.000	0.965±0.000	0.965±0.001

RCA with VAE The results reported for RCA in Tables 2.1 - 2.3 use autoencoders as the underlying DNN. To investigate whether our framework can benefit other DNN architectures, we compared our Robust Collaborative Variational Autoencoder (RCVA) against traditional VAE. The results in Figure 2.3 showed that RCVA outperformed VAE on most of the datasets. This suggests that our framework can improve the performance of other DNN architectures such as VAE for unsupervised AD.

RCA with Multiple Networks To extend RCA from 2 to multiple DNNs, we modified the shuffling step to allow each DNN to shuffle its selected data to any of the other DNNs. We varied the number of DNNs from 2 to 7 and plotted the results in Figure 2.4. The results suggest that adding more DNNs does not help significantly. This is not surprising since the shuffling step is designed to prevent the DNNs from converging too quickly rather than as an ensemble framework to improve

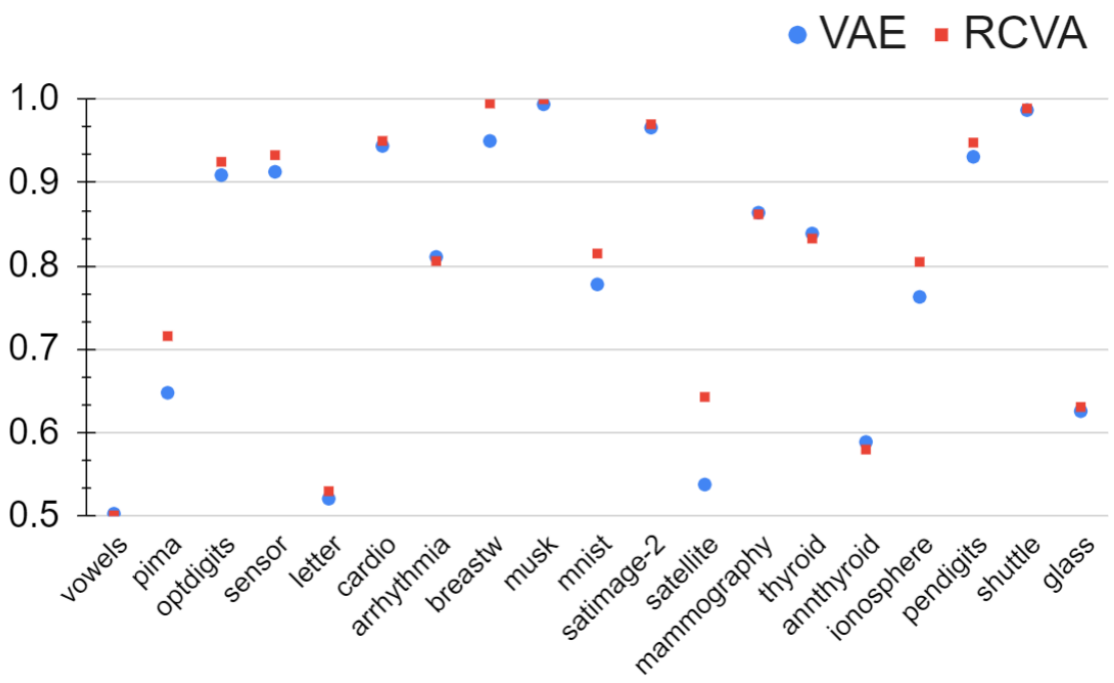


Figure 2.3: Comparison of RCVA and VAE in terms of AUC score. RCVA outperforms VAE on most datasets, suggesting that the RCA framework can improve the performance of other DNNs such as VAE.

performance. Increasing the number of DNNs also makes it more expensive to train, which reduces its benefits.

2.5 Conclusion

This chapter introduces a robust collaborative autoencoder framework for unsupervised AD. The framework is designed to overcome limitations of existing DNN methods for unsupervised AD. Theoretical analysis shows the effectiveness of RCA in eliminating corruption in training data due to anomalies. Our results showed that RCA outperforms various state-of-the-art algorithms under various experimental settings and is more robust to noise introduced by missing value imputation.

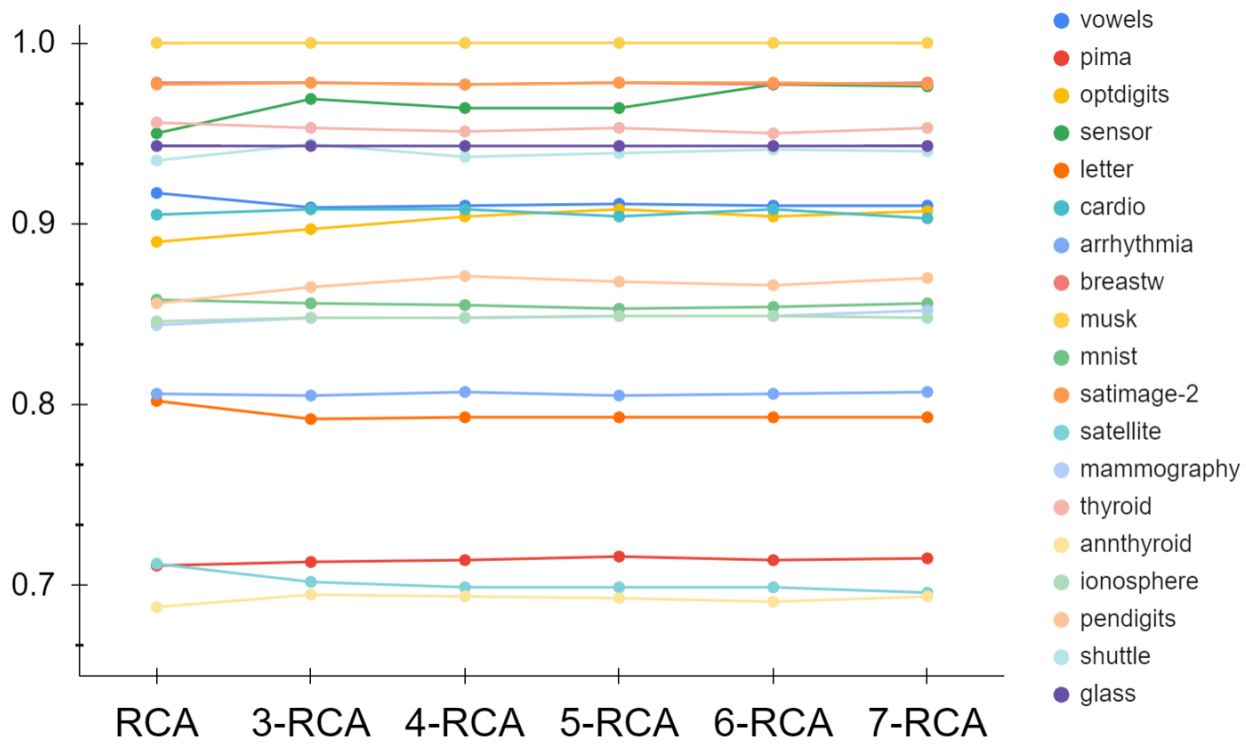


Figure 2.4: Effect of varying number of DNNs in RCA on AUC scores. RCA corresponds to a twin network while K-RCA has K DNNs.

2.6 Proofs of Theorem

2.6.1 Proof of Theorem 1

denote $M = n(1-\epsilon)L_{max}$ to be the smoothness of function $F = \sum_{i \notin O} f_i(\mathbf{w})$, denote the update rule $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \sum_i p_i^{(t)} \nabla f_i(\mathbf{w})$. For the normal stochastic gradient descent, by smoothness, we have:

$$
\begin{aligned}
F(\mathbf{w}^{(t+1)}) - F(\mathbf{w}^{(t)}) &\leq \langle \nabla F(\mathbf{w}^{(t)}), \mathbf{w}^{(t+1)} - \mathbf{w}^{(t)} \rangle + \frac{M}{2} \|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\|^2 \\
&\leq -\eta^{(t)} \langle \nabla F(\mathbf{w}^{(t)}), \nabla f_i(\mathbf{w}^{(t)}) \rangle + \frac{\eta^{(t)^2} M}{2} \|\nabla f_i(\mathbf{w}^{(t)})\|_2^2 \qquad (2.4)
\end{aligned}
$$

Take expectation on $\nabla f_i(\mathbf{w}^{(t)})$ by our sampling probability and applying triangle inequality on

the last term with inequality $\sum p_i^2 \leq \sum p_i$, we have

$$\mathbb{E}\left(F(\mathbf{w}^{t+1})\right) - F(\mathbf{w}^{(t)}) \leq -\eta^{(t)}\langle \nabla F(\mathbf{w}^{(t)}), \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)})\rangle$$
$$+ \sum_i p_i(\mathbf{w}^{(t)})\frac{\eta^{(t)^2}M}{2}\|\nabla f_i(\mathbf{w}^{(t)})\|^2$$
$$\leq -\eta^{(t)}\langle \nabla F(\mathbf{w}^{(t)}), \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)}) - \nabla F(\mathbf{w})\rangle - \eta^{(t)}\|\nabla F(\mathbf{w})\|^2 + \eta^{(t)^2}\frac{MG^2}{2}$$

complete the square and let $\hat{F}(\mathbf{w}^{(t)}) = \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)})$, we have

$$\leq \frac{\eta^{(t)}}{2}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)}}{2}\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2 - \eta^{(t)}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)^2}MG^2}{2}$$
$$\leq \frac{\eta^{(t)}}{2}\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2 - \frac{\eta^{(t)}}{2}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)^2}MG^2}{2}$$

Move the gradient norm to the left, and take total expectation, we have

$$\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq 2\left(\mathbb{E}\left(F(\mathbf{w}^{(t)})\right) - \mathbb{E}\left(F(\mathbf{w}^{(t+1)})\right)\right) +$$
$$\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2\right) + \eta^{(t)^2}MG^2$$

Sum it from $t = 0$ to $t = T$, we have:

$$\sum_{t=0}^{T}\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq 2\left(\mathbb{E}\left(F(\mathbf{w}^{(0)})\right) - \mathbb{E}\left(F(\mathbf{w}^{(T+1)})\right)\right) +$$
$$\sum_{t=0}^{T}\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2\right) + \eta^{(t)^2}MG^2$$

Assume $\left(\mathbb{E}\left(F(\mathbf{w}^{(0)})\right) - \mathbb{E}\left(F(\mathbf{w}^{(T+1)})\right)\right) = B$, we have

$$\sum_{t=0}^{T}\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq 2B + \sum_{t=0}^{T}\eta^{(t)}C + \sum_{t=0}^{T}\eta^{(t)^2}MG^2$$

$$\min_{t=0,1,2,\ldots,T}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \frac{2B}{\sum\eta^{(t)}} + MG^2\frac{\sum\eta^{(t)^2}}{\sum\eta^{(t)}} + C \tag{2.5}$$

By using the assumption of learning rate ($\sum\eta^{(t)} = \infty, \sum\eta^{(t)^2} \leq \infty$), the first two term can be ignored when T goes to infinity. We can get the convergence in theorem 1 (The convergence rate $\log(T)$ is get by assume learning rate is $\eta^{(t)} = 1/t$, which satisfy the above learning rate assumption).

We also provided a better convergence rate compared to submitted manuscript with a stricter learning rate setting. Start from equation 2.4, we have:

$$\mathbb{E}\left(F(\mathbf{w}^{t+1})\right) - F(\mathbf{w}^{(t)}) \leq -\eta^{(t)}\langle\nabla F(\mathbf{w}^{(t)}), \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)})\rangle + \sum_i p_i(\mathbf{w}^{(t)})\frac{\eta^{(t)^2}M}{2}\|\nabla f_i(\mathbf{w}^{(t)})\|^2$$

$$\leq -\eta^{(t)}\langle\nabla F(\mathbf{w}^{(t)}), \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)}) - \nabla F(\mathbf{w})\rangle - \eta^{(t)}\|\nabla F(\mathbf{w})\|^2 + \eta^{(t)^2}\frac{MG^2}{2}$$

complete square in a different way and let $\hat{F}(\mathbf{w}^{(t)}) = \sum_i p_i(\mathbf{w}^{(t)})\nabla f_i(\mathbf{w}^{(t)})$, we have

$$\leq \eta^{(t)}\frac{1}{2}\left(\eta^{(t)}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{1}{\eta^{(t)}}\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2\right)$$
$$- \eta^{(t)}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)^2}MG^2}{2}$$
$$\leq \frac{\eta^{(t)^2}}{2}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{1}{2}\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2 - \eta^{(t)}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)^2}MG^2}{2}$$
$$\leq \frac{1}{2}\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2 - \eta^{(t)}\|\nabla F(\mathbf{w}^{(t)})\|^2 + \frac{\eta^{(t)^2}(M+1)G^2}{2}$$

Move the gradient norm to the left, and take total expectation, we have

$$\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \mathbb{E}\left(F(\mathbf{w}^{(t)})\right) - \mathbb{E}\left(F(\mathbf{w}^{(t+1)})\right)$$
$$+ \frac{1}{2}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2\right) + \frac{\eta^{(t)^2}(M+1)G^2}{2}$$

Sum it from $t = 0$ to $t = T$, we have:

$$\sum_{t=0}^{T}\eta^{(t)}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \mathbb{E}\left(F(\mathbf{w}^{(0)})\right) - \mathbb{E}\left(F(\mathbf{w}^{(T+1)})\right)$$
$$+ \sum_{t=0}^{T}\frac{1}{2}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)}) - \nabla\hat{F}(\mathbf{w}^{(t)})\|^2\right) + \sum_{t=0}^{T}\frac{\eta_t^2(M+1)G^2}{2}$$
$$\leq B + \frac{TC}{2} + \sum_{t=0}^{T}\frac{\eta^{(t)^2}(M+1)G^2}{2}$$

$$\min_{t=0,1,2,\ldots,T}\mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \frac{B}{\sum\eta^{(t)}} + \frac{(M+1)G^2}{2}\frac{\sum\eta^{(t)^2}}{\sum\eta^{(t)}} + \frac{TC}{\sum\eta^{(t)}}$$

By assume the learning rate is constant, we write the RHS as a function of learning rate

$$\begin{aligned} f(\eta) &= \frac{B}{T\eta} + \frac{(M+1)G^2\eta}{2} + \frac{TC}{T\eta} \\ &= \frac{B}{T\eta} + \frac{(M+1)G^2\eta}{2} + \frac{C}{\eta} \\ &\text{Let } (M+1)G^2 = H, \text{ we have} \\ &= \frac{B}{T\eta} + \frac{H\eta}{2} + \frac{C}{\eta} \end{aligned}$$

We study the minima of function:

$$\begin{aligned} f(x) &= \frac{a}{x} + bx \\ x^* &= \sqrt{(a/b)}, f(x^*) = 2(\sqrt{ab}) \end{aligned}$$

Thus, to minimize the bound in RHS, we have the optimal learning rate $\eta^* = \sqrt{\frac{2B}{HT} + \frac{2C}{H}}$ by letting $a = \frac{B}{T} + C, b = \frac{H}{2}$. Then, the optimal value of RHS is:

$$f(\eta^*) = 2(\sqrt{(\frac{B}{T} + C)(\frac{H}{2})}) = \sqrt{\frac{2BH}{T} + 2CH} \leq \sqrt{\frac{2BH}{T}} + \sqrt{2CH} = \mathbb{O}(\frac{1}{\sqrt{T}}) + \mathbb{O}(\sqrt{C})$$

. Thus we can conclude that $\min_{t=0,1,2,\ldots,T} \mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \rightarrow \mathbb{O}(\frac{1}{\sqrt{T}}) + \mathbb{O}(\sqrt{C})$.

This is even a better results compared to theorem 1 in our paper, since we could achieve the convergence rate of $\sqrt{\frac{1}{T}}$ compared to $\log(T)$, while the error term is dependent on $\sqrt{C}$ instead of C. However, to achieve this rate, we need stricter condition on learning rate.

2.6.2 Proof of Theorem 2

We analysis the stationary point of using all data. Let $\mathbf{w}_{ns}^*$ denotes the stationary point by using the entire data, $\mathbf{w}^*$ denotes the stationary point by using the clean data, by the stationary condition of

$\mathbf{w}^*_{ns}$, we have

$$\sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*_{ns}) = -\sum_{j \in O}^{q} \nabla f_j(\mathbf{w}^*_{ns})$$

$$\| \sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*_{ns}) \| = \| \sum_{j \in O}^{q} \nabla f_j(\mathbf{w}^*_{ns}) \|$$

Upper bound for LHS

$$\| \sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*_{ns}) \| = \| \sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*_{ns}) - \sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*) \| \leq \sum_{i \notin O}^{p} \| \nabla f_i(\mathbf{w}^*_{ns}) - \nabla f_i(\mathbf{w}^*) \| \leq \sum_{i \notin O}^{p} L_i \| \mathbf{w}^*_{nsi} - \mathbf{w}^* \|$$

$$= (1 - \epsilon) n L_{max} \max_i \| \mathbf{w}^*_{nsi} - \mathbf{w}^* \|$$

$$\leq M\delta \quad (\delta \text{ is defined in equation 3 in the submitted manuscript})$$

Another upper bound for LHS from RHS

$$\| \sum_{i \notin O}^{p} \nabla f_i(\mathbf{w}^*_{ns}) \| = \| \sum_{j \in O}^{q} \nabla f_j(\mathbf{w}^*_{ns}) \| \leq n\epsilon G$$

Thus, we have

$$\| \nabla F(\mathbf{w}^*_{ns}) \| \leq \min(n\epsilon G, M\delta) \tag{2.6}$$

From theorem 1, by setting $\eta^* = \sqrt{\frac{2B}{HT} + \frac{2C}{H}}$, it is trivial to get

$$\min_{t=0,1,2,\ldots,T} \mathbb{E}\left(\| \nabla F(\mathbf{w}^{(t)}) \|^2 \right) \leq \sqrt{\frac{2BH}{T} + 2CH}$$

Or by assumption of $\sum \eta^{(t)} = \infty$, $\sum \eta^{(t)} \leq \infty$, from equation 2.5 we can get

$$\min_{t=0,1,2,\ldots,T} \mathbb{E}\left(\| \nabla F(\mathbf{w}^{(t)}) \|^2 \right) \leq \mathbb{E}\left(\| \nabla F(\mathbf{w}^{(t)}) \|^2 \right) \leq \mathbb{O}(\frac{1}{\sum \eta^{(t)}}) + \mathbb{O}(\frac{\sum \eta^{(t)^2}}{\sum \eta^{(t)}}) + C$$

We would like to study the worst case, when the upper bound of our algorithm is better than the upper bound without sample selection.

Thus, we want the following holds when t goes to infinity

$$\sqrt{\frac{2BH}{T} + 2CH} \leq \min(n\epsilon G, M\delta)^2$$

When t goes to infinity, by the assumption we have for learning rate, we have:

$$\sqrt{C} \leq \sqrt{1/2H}(\min(n\epsilon G, M\delta))^2(\text{fixed optimal lr})$$

Or similarly, in deminishing learning rate setting, we have

$$C \leq (\min(n\epsilon G, M\delta))^2(\sum \eta^{(t)} = \infty, \sum \eta^{(t)^2} \leq \infty)$$

Thus both LHS and RHS are the upper bound, thus we could get the conclusion that in worst cases, our solution is better than algorithm without sample selection, which gets our conclusion of existence.

2.6.3 Proof of Theorem 3

Now, we try to prove the correctness of the algorithm. We assume $\mathbf{w}^*$ satisfy $f_i(\mathbf{w}^*) < f_j(\mathbf{w}^*), \forall \mathbf{x}_i \notin O, \mathbf{x}_j \in O$.

Without loss of generality, we could define $\delta, \phi \geq 1$ as below:

$$\delta \leq \|\mathbf{w}_j^* - \mathbf{w}^*\| \leq \phi\delta, \forall j \in O \tag{2.7}$$

Now, we try to answer the first question, under what conditions, our solution is perfect. We define some neighbors around the optimal point. According to our anomaly detection setting, we with those anomalies should have higher loss compared to the normal data point:

$$\mathcal{B}_r(\boldsymbol{w}^*) = \left\{ \begin{array}{l} \boldsymbol{w} | f_i(\boldsymbol{w}) < f_j(\boldsymbol{w}), \forall i \notin \mathbb{O}, j \in \mathbb{O}, \\ \|\boldsymbol{w} - \boldsymbol{w}^*\| \leq r \end{array} \right\}. \tag{2.8}$$

We knew that such ball with radius r must be existed since the loss function above is strongly smooth.

Proof 1 *In order to better describe $\mathcal{B}_r(\boldsymbol{w}^*)$, we would like to analysis the boundary of the ball.*

Denote the set of intersection between the loss surface of the normal data and the loss surface of the abnormal data as:

$$\Omega_{\mathbf{w}} = \left\{ \mathbf{w}_{ij} \middle| f_{\mathbf{x}_i \notin O}(\mathbf{w}) = f_{\mathbf{x}_j \in O}(\mathbf{w}) \right\} \tag{2.9}$$

Then, we can write the boundary point of the ball as:

$$\mathbf{w}_B = \arg\min_{w \in \Omega_{\mathbf{w}}} \|\mathbf{w}_B - \mathbf{w}^*\| \tag{2.10}$$

At $\mathbf{w}_B$, by using smoothness and convexity, we have

$$f_i(\mathbf{w}_B) \leq f_i(\mathbf{w}^*) + \langle \mathbf{w}_B - \mathbf{w}^*, \nabla f_i(\mathbf{w}^*) \rangle + \frac{L_i}{2}\|\mathbf{w}_B - \mathbf{w}^*\|^2$$
$$f_j(\mathbf{w}_B) \geq f_j(\mathbf{w}_j^*) + \langle \mathbf{w}_B - \mathbf{w}_j^*, \nabla f_j(\mathbf{w}^*) \rangle + \frac{\mu_j}{2}\|\mathbf{w}_B - \mathbf{w}^*{}_j\|^2$$

By the definition of $\mathbf{w}_B$ and equal minimum assumption, without loss of generality, we could assume the minimum is 0, which does not affect the results. Then, we have

$$f_i(\mathbf{w}_B) \leq \frac{L_i}{2}\|\mathbf{w}_B - \mathbf{w}^*\|^2$$
$$-f_j(\mathbf{w}_B) \leq -\frac{\mu_j}{2}\|\mathbf{w}_B - \mathbf{w}_j^*\|^2$$

Adding two inequality, we have:

$$\|\mathbf{w}_B - \mathbf{w}^*\|^2 \geq \frac{\mu_j}{L_i}\|\mathbf{w}_B - \mathbf{w}_j^*\|^2$$

By triangle inequality, we have

$$\|\mathbf{w}_B - \mathbf{w}^*\| + \|\mathbf{w}_B - \mathbf{w}_j^*\| \geq \|\mathbf{w}_j^* - \mathbf{w}^*\|$$

Combining above two inequalities, we have

$$\|\mathbf{w}_B - \mathbf{w}^*\| \geq \sqrt{\frac{\mu_j}{L_i}}(\|\mathbf{w}_j^* - \mathbf{w}^*\| - \|\mathbf{w}_B - \mathbf{w}^*\|)$$
$$(1 + \sqrt{\frac{\mu_j}{L_i}})\|\mathbf{w}_B - \mathbf{w}^*\| \geq \sqrt{\frac{\mu_j}{L_i}}\delta$$
$$let\ \kappa = \sqrt{\frac{L_{max}^c}{\mu_{min}^o}}$$
$$\|\mathbf{w}_B - \mathbf{w}^*\| \geq \frac{1}{1 + \sqrt{\frac{L_i}{\mu_j}}}\delta \geq \frac{1}{1+\kappa}\delta$$

, where L_{max}^c denotes the maximum lipschitz smoothness in clean data and μ_{min}^o denotes the minimum convexity in anomalies.

Define $F(\mathbf{w}) = \sum_{\mathbf{x}_i \notin O} f_i(\mathbf{w})$*, we have that function* $F(\mathbf{w})$ *satisfy* $m = n(1-\epsilon)\mu_{min}$ *convexity. Similarly, we know that* $F(\mathbf{w})$ *also satisfy* $M = n(1-\epsilon)L_{max}$ *smoothness. Then, we can have:*

$$\begin{aligned}\|\mathbf{w}_{sr}^* - \mathbf{w}^*\| &\leq m\|\nabla F(\mathbf{w}_{sr}^*) - \nabla F(\mathbf{w}^*)\| \\ &= m\|\nabla F(\mathbf{w}_{sr}^*)\|\end{aligned}$$

$$\|\mathbf{w}_{sr}^* - \mathbf{w}^*\|^2 \leq m^2\|\nabla F(\mathbf{w}_{sr}^*)\|^2$$

Now, our goal is trying to upper bound the term $\|\nabla F(\mathbf{w}_{sr}^*)\|$*. According to theorem 1 for fixed optimal learning rate, we have*

$$\min_{t=0,1,2,\ldots,T} \mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \mathbb{E}\left(\|\nabla F(\mathbf{w}^{(t)})\|^2\right) \leq \sqrt{\frac{2BH}{T} + 2CH}$$

Now, we have:

$$\begin{aligned}\|\mathbf{w}_{sr}^* - \mathbf{w}^*\|^2 &\leq m^2\sqrt{\frac{2BH}{T} + 2CH} \\ \|\mathbf{w}_B - \mathbf{w}^*\|^2 &\geq (\frac{1}{1+\kappa}\delta)^2\end{aligned}$$

Thus, we could get the sufficient condition for $\|\mathbf{w}_{sr}^* - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_B - \mathbf{w}^*\|^2$ *as*

$$m^2\sqrt{\frac{2BH}{T} + 2CH} \leq (\frac{1}{1+\kappa}\delta)^2$$

by using the optimal fixed learning rate and assume T is sufficiently large, rearrange the term, we have:

$$\begin{aligned}\sqrt{C} &\leq \frac{1}{\sqrt{2H}}(\frac{\delta}{(1+\kappa)m})^2 = \mathbb{O}(\frac{\delta}{\kappa})^2 \\ C &\leq \mathbb{O}(\frac{\delta}{\kappa})^4\end{aligned}$$

Similarly, from the theorem 1 for the diminishing learning rate condition $\sum \eta^{(t)} = \infty, \sum \eta^{(t)} \leq \infty$*, we have*

$$C \leq (\frac{\delta}{(1+\kappa)m})^2 = \mathbb{O}(\frac{\delta}{\kappa})^2$$

We can conclude that as long as the above inequality holds, we can guarantee that our algorithm returns the correct answer.

CHAPTER 3

ROBUST DENSITY ESTIMATION FOR UNSUPERVISED ANOMALY DETECTION

In this chapter, we introduce the Robust Density Estimation, a robust learning density algorithm. We also empirically show that this algorithm could improve the performance of unsupervised anomaly detection.

3.1 Introduction

Anomaly detection (AD) is the task of finding unusual or abnormal observations in the data, whose characteristics are considerably different from the majority of the data. Application of AD can be found in diverse domains, including cybersecurity, finance, healthcare, and manufacturing process control. AD also plays an important role as a data preprocessing step since the anomalies may have a large impact on the predictive performance of various learning algorithms including deep neural networks (DNNs). By detecting and removing the anomalies early, we can prevent them from degrading the performance of subsequent downstream learning tasks.

While there have been extensive methods developed for unsupervised AD, density-based methods such as kernel density estimation (KDE), local outlier factor (LOF), and their variants have found success due to their simplicity, ease of use, and ability to detect diverse types of anomalies under various settings Chandola et al. (2009); Breunig et al. (2000). KDE-based methods estimate the probability distribution of the data over the input feature space and detect anomalies as points residing in abnormally low-density regions. Similarly, LOF identifies anomalies by measuring the deviation between the density of each data point with respect to the density of its local neighbourhood. Despite their success, current density-based AD methods are mostly designed to operate in their original feature space. This greatly limited their applicability to high-dimensional data, as density estimation has high sample complexity and incurs high computational costs as the number of dimensions increases Tsybakov (2008). Moreover, if the data lies in some low-dimensional, nonlinear, manifold, performing density estimation and anomaly detection in the original feature space may lead to

inferior performance. Finally, if the training data is contaminated by anomalies, which is common in many real-world applications, density estimation methods such as KDE may not be able to recover the true density of the normal data.

With the emergence of deep learning, DNNs have been widely used to perform density estimation in complex, high-dimensional datasets with nonlinear relationships among their underlying features. Deep density estimation parameterizes the density estimation function with a neural network, which is usually optimized by maximizing the likelihood of the training instances. For example, the recent flow-based model Dinh et al. (2016, 2014) employs the change of variable theorem to model the likelihood function. By carefully designing the network structure, it can successfully learn the underlying data distribution. Moreover, unlike other generative models such as generative adversarial networks Goodfellow et al. (2014a) and variational autoencoders Kingma and Welling (2013b), the flow-based model explicitly learns the density function, which makes it a powerful tool for detecting out-of-distribution instances.

However, due to the high capacity of the deep neural networks, the presence of anomalies in the training data may corrupt the density function estimated by the flow-based model. To illustrate the vulnerability of the flow-based model to training anomalies, in Figure 3.1 we provide a simple example where the normal data lies in a two-moon manifold. When the training data is clean, i.e., anomaly-free, the flow-based model successfully captures the true underlying density function. However, when training data is contaminated with anomalies, the flow-based model is biased towards learning the density of the anomalies as well. Thus, a more robust approach is needed to estimate the density function in order to enhance the capabilities of flow-based models in unsupervised AD tasks.

In this chapter, we propose a novel deep unsupervised anomaly detection algorithm called RobustRealNVP, inspired by the popular flow-based method, Real NVP Dinh et al. (2016). RobustRealNVP has two major advantages as compared to the vanilla Real NVP. It ignores low-density points during the density estimation process and uses the remaining data to update its model. We show that by using above strategy, the robustness of the RobustRealNVP highly depends on

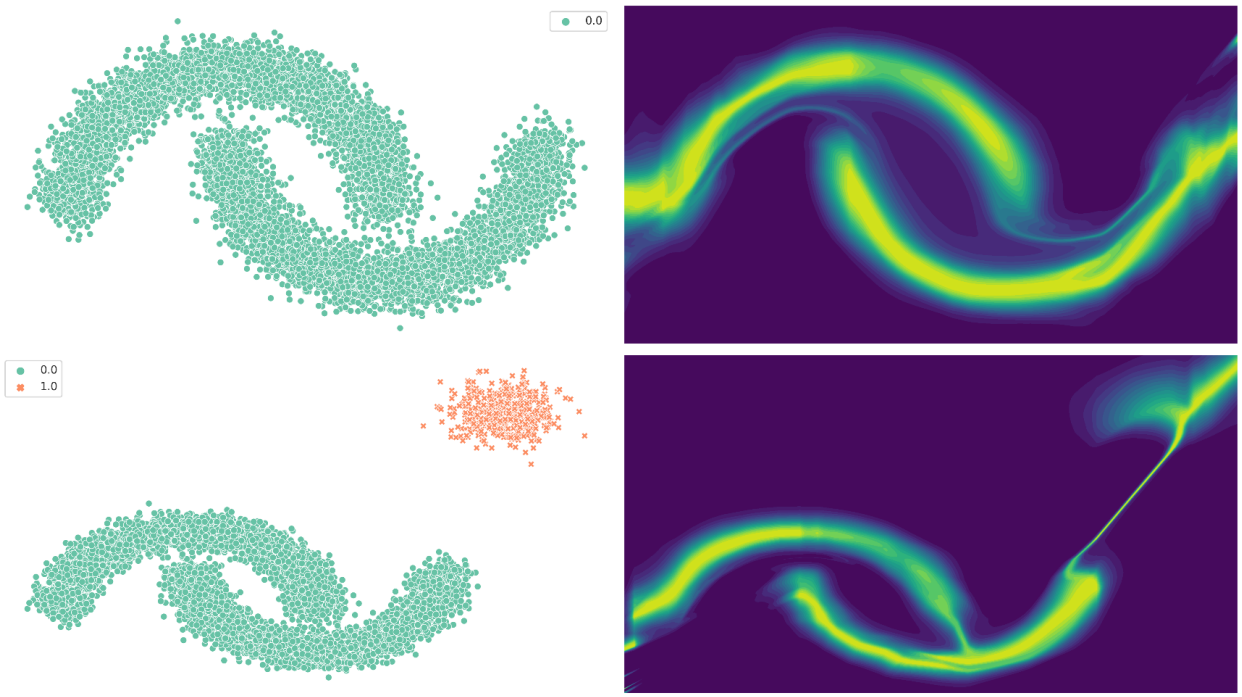


Figure 3.1: Effect of anomalies on density estimation for flow-based models. The upper left figure represents the clean data while the bottom left figure represents data contaminated with a small fraction of clustered anomalies (shown as orange points). The upper right figure shows the results of Real NVP results on the clean data while the bottom right figure shows the results for contaminated data. Observe that Real NVP assigns large density to the region of clustered anomalies, which leads to its suboptimal performance.

smoothness of the learned density function, especially when the anomalies are clustered. As a consequence, if we could guarantee the smoothness of the learned density function, the robustness of learned model could also be guaranteed. This is a significant enhancement over Real NVP, which has no performance guarantees when applied to contaminated data. In order to make the learned density smooth, RobustRealNVP imposes the Lipschitz regularization to enforce smoothness in the estimated density function. According to our best knowledge, no previous work utilized the lipschitz regularization in flow-based model or density estimation before.

In summary, the major contributions of this paper are as follows:

- We develop a robust flow-based density estimation method for unsupervised anomaly detection.
- We present theoretical analysis to demonstrate the robustness of RobustRealNVP against contaminated training data. Specifically, we show that RobustRealNVP converges to an

ϵ-approximated stationary point of the objective function for Real NVP trained on clean (anomaly-free) data.

- We perform extensive experiments on both synthetic and real-world data and show that our framework outperforms various state-of-the-art unsupervised anomaly detection algorithms.

3.2 Related Work

In this section, we will review previous works on density estimation, anomaly detection, and Lipschitz regularization.

3.2.1 Density Estimation and Anomaly Detection

Unsupervised anomaly detection Chandola et al. (2009) is more challenging compared to supervised or semi-supervised anomaly detection since it does not assume the availability of labeled or clean training data. Instead, it allows the model to be trained on large, unlabeled data without requiring any supervised information about which data instances had been corrupted. Density estimation is a well-known method in anomaly detection. By learning the density function of the normal instances, anomalies are detected when their densities fall below some minimum threshold. Most density-based AD approaches rely on non-parametric methods such as histogram, k-nearest-neighbor, and kernel density estimation Tsybakov (2008) to estimate the density function. While these methods may work well for low dimensional data, they are less effective for high-dimensional data due to the curse of dimensionality problem. Furthermore, such methods often assume the availability of clean training data since the presence of anomalies may alter the estimation of density function, especially when used with high capacity models such as deep neural networks. Current density-based methods also may not work well if the anomalies are concentrated or clustered since, by definition, the anomalies are assumed to have relatively lower densities than normal data. This assumption can be easily violated in practice.

There have been recent efforts to improve the robustness of density estimation. For example, the robust kernel density estimation approach was proposed in Humbert et al. (2020) to provide robustness guarantees by using the median of the means to replace the empirical mean of kernel sums. Another method is to replace the l2-loss with a huber loss in kernel density estimation Kim and Scott (2012). Although these approaches show promising robustness properties, they do not be scale well to high-dimensional data since they are all based on kernel density estimation, which is computationally expensive when applied to large-scale, high-dimensional data.

The success of deep learning has inspired many recent works using deep generative neural networks to approximate high-dimensional density functions. Three typical generative models in deep neural networks are generative adversarial networks (GAN) Goodfellow et al. (2014a), variational autoencoder (VAE) Kingma and Welling (2013b), and flow-based model Dinh et al. (2014). GAN can effectively learn how to sample from a density function, without explicitly inferring the sample density. Although VAE can perform both sampling and inferring the sample density, it does not directly optimize the likelihood function. Instead, VAE optimizes a lower bound of the likelihood function, which may lead to suboptimal performance. Compared to VAE, flow-based models can directly optimize the likelihood function by using the change of variable theorem. Many varieties of network architectures have been proposed to improve the efficiency and expressive power of flow-based models Dinh et al. (2014, 2016); Chen et al. (2019); Kingma and Dhariwal (2018). While these methods have been widely used for out-of-the-distribution detection, they are mostly applied in a semi-supervised learning setting Zisselman and Tamar (2020), which assumes the availability of clean training data. By restricting to clean training data, they may not be able to leverage the larger pool of unlabeled data, which has no guarantees of being anomaly-free.

3.2.2 Lipschitz Regularization

Lipschitz regularization aims to optimize the loss function while ensuring that the learned function is Lipschitz continuous (smooth), i.e., its gradient norm is upper bounded everywhere. Lipschitz regularization has been widely studied and applied to different scenarios. For example, it is used to

regularize the discriminator in GAN Gulrajani et al. (2017); Miyato et al. (2018a); Qin et al. (2020) and to defend against attacks in adversarial learning Hein and Andriushchenko (2017). There are many ways to constrain a learning function to be Lipschitz continuous. The simplest approach is to penalize the gradient norm Gulrajani et al. (2017). However, directly incorporating the gradient norm constraint into the optimization problem involves higher-order derivatives, which makes it harder to be applied to large scale problems. An alternative method is to use adversarial training for Lipschitz regularization Terjék (2019). Although it can well approximate the Lipschitz regularization, the method is not efficient because it requires finding adversarial samples for every data point. Another approach specifically designed for neural networks is spectral normalization, which tries to constrain the Lipschitz constant in a layer-wise fashion Miyato et al. (2018a). By reducing the upper bound of the Lipschitz constant, it can efficiently make the function smooth. The drawback of this method is that its upper bound can be loose, which makes the function susceptible to over-smoothing.

3.2.3 Robust Optimization

Another related field is robust optimization, which aims to optimize an objective function assuming a small fraction of the data is corrupted Huber (1992). Despite the extensive studies to improve robustness of an optimization algorithm in different problem settings, most of the previous works either focused on linear regression and its variants Bhatia et al. (2015, 2017); Shen and Sanghavi (2019) or are limited to convex optimization problems Prasad et al. (2018). Their results cannot be directly generalized to deep neural networks and their methods have not been studied under density estimation scenarios. SEVER Diakonikolas et al. (2019) is a highly generalizable, non-convex optimization algorithm with agnostic corruption guarantees. However, due to its high space complexity, it cannot be efficiently applied to deep neural networks given current hardware limitations.

3.3 Preliminaries

In this section, we first review the flow-based model before introducing our proposed framework in Section 3.4. Given n samples $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$, the flow-based model aims to maximize the likelihood function of the observed data, $\max_\theta p_\theta(\mathbf{X})$, where θ is the network parameter. Flow-based model tries to learn a mapping from a known distribution such as Gaussian to a more complex target distribution. Let $p(z)$ be some simple density function and $z = f(x)$, where f is a bijective transformation function. According to the change of variable formula:

$$p_X(x) = p_Z(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right|,$$

where $\left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right|$ is the determinant of the Jacobian matrix of f at x. By defining $g = f^{-1}$, we have the following sample generation process. First, we generate z from $p(z)$ and use the sample to generate $x = g(z)$. However, in order to maximize $p_X(x)$, we need to design a network architecture that satisfies the following two criteria. First, it must be easy to calculate the determinant to ensure backpropagation can be applied to optimize the network. Second, f must be invertible, so that f^{-1} can be used to generate the data x.

One such transformation is the affine coupling transformation proposed in Real NVP Dinh et al. (2016). Given a D-dimensional input x and $d < D$, the output y is defined as:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\left[s\left(x_{1:d}\right)\right] + t\left(x_{1:d}\right), \end{aligned}$$

where s and t are functions that map $\mathbb{R}^D \rightarrow \mathbb{R}^{D-d}$ and can be parameterized by complex neural networks. The transformation is invertible and its Jacobian matrix is a triangular matrix, whose determinant can be easily computed by multiplying the diagonal elements of the Jacobian matrix. Let $\mathcal{N}$ be the set of normal observations. The goal is to maximize the following objective:

$$\begin{aligned} &\max_f \log \left(\prod\nolimits_{x \in \mathcal{N}} p_Z(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| \right) \\ &= \max_f \sum\nolimits_{x \in \mathcal{N}} \log(p_Z(f(x)) + \log \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right|, \end{aligned}$$

which is equivalent to minimizing the following loss function:

$$\min_f -\left[\sum_{x \in \mathcal{N}} \log(p_Z(f(x)) + \log \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| \right].$$

After learning f, the density of a sample x can be inferred as follows:

$$\tilde{p}(x) = p_Z\left(f(x)\right) \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right|.$$

3.4 Methodology

Our goal is to develop a robust density-based unsupervised anomaly detection approach amenable to training data that contains both normal observations and anomalies. In this setting, our objective function is designed to optimize the following objective function:

$$-\min_f \sum_{x \in \mathcal{N} \cup \mathcal{A}} \log(p_Z(f(x)) + \log \left(\left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| \right), \tag{3.1}$$

where $\mathcal{A}$ is the set of anomalies. Directly optimizing Equation 3.1 may lead to poor results as the anomalies will likely impact the estimated density. To address this issue, we propose a robust density estimation approach to be described in the sections below.

3.4.1 Robustness Analysis from Optimization Perspective

We first define robustness for density estimation. Let $\epsilon \in [0, 0.5]$ be the anomaly ratio, i.e., proportion of anomalies in training data. Given a set of normal instances $\mathcal{N}$ from a density function p and a set of anomalies $\mathcal{A}$ from some arbitrary density function, our goal is to design a learning algorithm $\Psi : \mathcal{N} \cup \mathcal{A} \rightarrow \hat{p}$, such that the gradient norm of the negative log-likelihood for $\mathcal{N}$ is minimized. If $\hat{p}$ is parameterized by θ, then the learning objective is:

$$\min_\theta \left\| - \sum_{x \in \mathcal{N}} \nabla_\theta \log \hat{p}_\theta(x) \right\|.$$

Optimizing the preceding objective is challenging as we have no prior knowledge which instances belong to $\mathcal{N}$ or $\mathcal{A}$.

In this study, we address this problem by using a robust gradient estimation approach. Consider a gradient descent approach for minimizing the negative log-likelihood function. Given a data point, x_i, its corresponding gradient is given by

$$\mathbf{g}_i = -\nabla_\theta \log \hat{p}_\theta(\mathbf{x}_i) = -\frac{1}{\hat{p}_\theta(\mathbf{x}_i)} \frac{\partial \hat{p}(\mathbf{x}_i)}{\partial \theta}.$$

Given n data points, let $\mathbf{G} = [g_1^T, g_2^T ... g_n^T] \in \mathbb{R}^{n \times d}$ be the contaminated gradient matrix and $\Gamma : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^d$ be an aggregation function. The model parameters can be updated using gradient descent as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \Gamma(\mathbf{G}).$$

If $\Gamma(\cdot)$ corresponds to the empirical mean function, then the above update formula reduces to standard gradient descent. However, since the gradient matrix $\mathbf{G}$ includes both anomalies and the normal (clean) data, $\Gamma(G)$ cannot be guaranteed to be close to the true average gradient on clean training data. Thus, given a contaminated training data, $\mathcal{N} \cup \mathcal{A}$, our goal is to design a robust aggregation function Γ that optimizes the following objective to ensure that the estimated density function is not corrupted by anomalies:

$$\min_\Gamma \|\Gamma(G) - \mu\|^2,$$

where $\mu = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \mathbf{g}_i$ is the gradient of clean, anomaly-free data.

A natural question to ask is: If there exists a small constant ζ such that the gradient difference is upper bounded by ζ in every gradient descent step, i.e., $\forall t : \|\Gamma(G)^{(t)} - \mu^{(t)}\|^2 \leq \zeta$, can we guarantee that the final convergence point will have a bounded gradient norm? The following proposition gives such a guarantee on the bounded gradient norm of the final convergence:

Proposition 1 (Convergence of Biased SGD) *Let ϕ be the objective function and θ be the variable to be optimized. Under mild Lipschitz assumptions Diakonikolas et al. (2019); Ajalloeian and Stich (2020), denote ζ as the maximum l_2 norm of the difference between the clean mini-batch gradient μ and corrupted mini-batch gradient, $\Gamma(\mathbf{G})$: $\|\mu - \Gamma(\mathbf{G})\| \leq \zeta$. By using the biased gradient estimation $\Gamma(\mathbf{G})$, SGD converges to the ζ-approximated stationary points: $\mathbb{E}\left(\|\nabla \phi(\theta_t)\|\right) = \mathbb{O}(\zeta)$.*

Proposition 1 comes from a series of previous work Fan et al. (2020); Ajalloeian and Stich (2020); Bernstein et al. (2018); Hu et al. (2020); Diakonikolas et al. (2019), and has been widely studied in optimization community. For the sake of completeness, we include the above proposition here to show that it is enough to design a robust mean estimation method to guarantee that the final solution will have small gradient norm in terms of its clean (anomaly-free) objective. The proof of this proposition can be found in the Appendix section.

3.4.2 Robust Gradient Estimation Method for Density Estimation

Our robust aggregation approach for $\Gamma(G)$ works as follows. Given a mini-batch $\mathcal{B}$ of size m, we sort the individual losses of instances in $\mathcal{B}$ and discard the instances with the top-k largest loss. This simple strategy ensures robustness guarantees under Lipschitz condition.

Definition 2 (Lipschitz Continuous Function) *A function f is said to be L-smooth if there exists a constant L, such that for every x and y:*

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$$

The Lipschitz assumption states that if a function f is L-smooth, then the norm of gradient for f is upper bounded by L. Our first assumption is with regards to smoothness of the true density function for normal (clean) data $p(\mathbf{x})$ and the learned density function $\hat{p}(\mathbf{x})$ from corrupted data.

Assumption 5 *We assume the true density function for clean data $p(x)$ is L-smooth and the predicted density function $\hat{p}(x)$ is $\hat{L}$-smooth.*

Remark 4 *The Lipschitz condition is not that strong of an assumption for normal data since the density function for normal observations usually has some smoothness properties. Such an assumption has also been used in other studies for non-parametric density estimation Tsybakov (2008). However, the Lipschitz condition for predicted density function is a stronger assumption since our deep neural networks are trained to approximate the density function $\hat{p}$ for a mixture distribution of normal data and anomalies. As the anomalies can be introduced in arbitrary ways,*

Algorithm 2: Robust Gradient Estimation for Density Estimation

Input: corrupted gradient matrix $\mathbf{G} \in \mathbb{R}^{n \times d}$, where n = $|\mathcal{A}| + |\mathcal{N}|$, anomaly ratio ϵ, current parameter estimate, $\theta^{(t)}$

Output: Estimated mean of clean gradient, $\hat{\mu}^{(t)} \in \mathbb{R}^d$

1. For each row $\mathbf{g}_i$ in $\mathbf{G}$, calculate its predicted density, $\hat{p}_{\theta^{(t)}}(\mathbf{x}_i)$
2. Choose the ϵ-fraction rows in $\mathbf{G}$ with smallest $\hat{p}_{\theta^{(t)}}(\mathbf{x}_i)$
3. Remove the selected rows from $\mathbf{G}$
4. Return the empirical mean of the remaining rows as $\hat{\mu}^{(t)}$.

e.g., if they are clustered, this may produce extremely sharp density function. Thus, it is harder to guarantee that the function approximated by the neural network is L-smooth. In the next section, we will apply Lipschitz regularization to ensure $\hat{p}$ satisfies this assumption.

By applying the $\hat{L}$-smooth assumption on $\hat{p}$, we can bound the individual gradient norm as follows:

$$\|\mathbf{g}_i\| = \|-\nabla_\theta \log \hat{p}_\theta(\mathbf{x}_i)\| = \frac{1}{\hat{p}_\theta(\mathbf{x}_i)} \left\| \frac{\partial \hat{p}(\mathbf{x}_i)}{\partial \theta} \right\| \leq \frac{1}{\hat{p}_\theta} \hat{L}.$$

Since the loss $-\log \hat{p}_\theta(x)$ monotonically decreases with increasing $\frac{1}{\hat{p}_\theta(x)}$, thus, at each iteration, our algorithm simply needs to sort the current density estimates $\hat{p}_{\theta^{(t)}}(x)$ of every data point and discards those points with low predicted densities. It will then estimate the mean gradient, $\hat{\mu}^{(t)}$ for the remaining data points. This is summarized by the pseudocode given in Algorithm 2. The estimated gradient $\hat{\mu}$ is used to perform the gradient descent update: $\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \hat{\mu}^{(t)}$. We will first show that Algorithm 2 has robustness guarantee by using the following lemma:

Lemma 1 (Mean Gradient Estimation Error) *Let $\mathbf{g}_i^{(t)}$ be the gradient of the i_{th} datum at iteration t, $\mathcal{N}$ be the clean data, and $\hat{\mu}^{(t)}$ be the output of Algorithm 2 at iteration t. The following guarantee holds on the gradient estimation error:*

$$\left\| \frac{1}{|\mathcal{N}|} \sum\nolimits_{\mathcal{N}} \mathbf{g}_i^{(t)} - \hat{\mu}^{(t)} \right\|^2 = \mathbb{O}(\epsilon) \hat{L}.$$

For brevity, we only provide a sketch of the proof for the lemma. A more detailed proof is given in the Appendix. Assuming $\|\mathbf{g}\|_{i \in \mathcal{N}} \leq \frac{1}{\hat{p}_i} \hat{L}$, and let $\max_{i \in \mathcal{N}} \frac{1}{\hat{p}_i} = C$. Since we discard an ϵ-fraction of

the data with small $\hat{p}_i$, there are two possibilities: either all the training anomalies are discarded or some training anomalies may still remain as their $|g_i|$ values are less than $C\hat{L}$. In both cases, our approach will apply empirical mean estimation to the remaining gradient vectors, which have limited magnitude (i.e., bounded by $C\hat{L}$). When $\hat{L}$ is small, since the magnitude of the remaining gradients (which may include anomalies) are limited, the impact of each remaining anomaly on the mean gradient estimation is limited as well, not exceeding $C\hat{L}$. In the worst case, if no anomalies are discarded, this will introduce $\mathbb{O}(\epsilon)$ error. Putting them together, we obtain the above error rate of $\mathbb{O}(\epsilon)$L, assuming C is a constant.

The above lemma suggests that the gradient estimation error is controlled by the Lipschitz constant when ϵ is small. Combined with Proposition 1, if we can control the Lipschitz constant to be small (i.e. having a smooth loss surface), then the presence of anomalies in training data will not significantly alter the density estimation of clean data.

The analysis presented in this section suggests that the following two assumptions must be satisfied in order to make the gradient estimation robust. First, $\hat{p}_{i\in\mathcal{N}}$ must not be too small to ensure C is small. This is not a strong assumption as the data contain mostly normal observations and the neural network is capable of memorizing such observations Zhang et al. (2016). Second, the predicted density function should be smooth enough to ensure $\hat{L}$ is small. This is a much stronger assumption since neural networks can overfit the anomalies. If the training anomalies are extremely concentrated, the predicted density can have sharp peaks in a small region, which makes the Lipschitz constant $\hat{L}$ to be large. Thus, we need an approach to guarantee that the predicted density function has a small $\hat{L}$.

3.4.3 Spectral Normalization

As shown in Lemma 1, smoothness of the learned function is critical for robust training of the deep neural networks for density estimation. A small Lipschitz constant will ensure the error rate of Algorithm 2 is low. In this section, we describe our method to constrain the Lipschitz constant of our learned network. Our method is inspired by ideas from spectral normalization (SN) Miyato et al.

(2018a). For the sake of completeness, we first briefly review the method. We will extend SN so it can be adapted to our unsupervised AD method.

SN is motivated by the following inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$, where $\|.\|_{\text{Lip}}$ represents the Lipschitz constant. A neural network can be treated as a composition of a series of linear transformation and activation functions. Since most activation functions are Lipschitz continuous (e.g., RELU function has a Lipschitz constant of 1), if the Lipschitz constant for every linear layer is bounded, then the Lipschitz constant for the entire network is also bounded.

For a linear function $\phi : \mathbf{h} \rightarrow \mathbf{Wh}$, where $\mathbf{W}$ is the linear transformation matrix, its Lipschitz constant is upper bounded by the operator norm of the matrix $\mathbf{W}$, i.e., $\|\mathbf{W}\|_{op}$. Thus, the Lispchitz function for the whole network is upper bounded by $\prod_{i=1}^{v} \|\mathbf{W}_i\|_{op}$, where v is the number of layers (for RELU networks). In the original paper of SN, each layer is normalized as follows: $\mathbf{W}_{SN} = \dfrac{\mathbf{W}}{\|\mathbf{W}\|_{op}}$ to ensure that the function is 1-Lipschitz.

However, a 1-Lipschitz may over-smooth the function, which leads to underfitting of the density function. Thus, for our approach, we extend SN to satisfy the following k-Lipschitz constraint:

$$\mathbf{W}_{SN} = \frac{\mathbf{W}}{(\|\mathbf{W}\|_{op}/k)}.$$

In our experiments, we approximate the operator norm using power iterations, similar to the approach used in the original SN paper Miyato et al. (2018a).

3.4.4 Lipschitz Regularization for Flow-based Model

SN assumes that a deep neural network can be decomposed into a series of linear transformation and activation functions. However, for flow-based models, the network is no longer a simple decomposition of linear transformation and activation function. Recall the following transformation used by Real NVP:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) + t\left(x_{1:d}\right), \end{aligned} \tag{3.2}$$

where s and t are two fully-connected networks. While SN can be used to guarantee that the functions s and t are k-smooth, they is no assurance that the entire transformation is k-smooth.

In this section, we show that only constrain function s and t is enough to achieve the k-smoothness. Our rationale for this is based on the following two simple lemma on spectral norm:

Lemma 2 *If $f : \mathbb{R}^n \to \mathbb{R}^n$ is a differentiable function everywhere with a Jacobian matrix $\mathbf{J}$, then $\|\mathbf{J}\|_{op} \leq L$ if and only if f is L-smooth.*

The above lemma suggests that if a transformation function does not change the dimension and the function is L-smooth, then the spectral norm of this transformation is bounded by L.

Lemma 3 $\|[\mathbf{A}, \mathbf{B}]\|_{op} \leq \|\mathbf{A}\|_{op} + \|\mathbf{B}\|_{op}$, *where* $[\cdot, \cdot]$ *denotes matrix concatenation.*

We will analyze the applicability of SN to the transformation given in Equation 3.2 by using the preceding two lemmas. The Jacobian of the transformation is as follows:

$$\mathbf{J} = \frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}.$$

The matrix can be further decomposed into the sum of a diagonal and an off-diagonal matrix. Thus

$$\begin{aligned} \|\mathbf{J}\|_{op} &\leq \left\| \begin{bmatrix} \mathbb{I}_d & 0 \\ 0 & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix} \right\|_{op} + \left\| \begin{bmatrix} 0 & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & 0 \end{bmatrix} \right\|_{op} \\ &= \max(1, \max_{i \in [1,2,\ldots,D-d]} \exp[s(x_{1:d})]_i) + \left\| \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} \right\|_{op}. \end{aligned}$$

Observe that the first term is well bounded if the output of function s is well bounded. In our experiments, we use the tanh function as the last layer of function s. Thus, the first term will be bounded by $\exp(1)$. For the second term, according to Equation 3.2, we have

$$\begin{aligned} &\left\| x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) + t\left(x_{1:d}\right) \right\|_{\text{Lip}} \\ &= \left\| x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) \right\|_{\text{Lip}} + \left\| t\left(x_{1:d}\right) \right\|_{\text{Lip}}. \end{aligned}$$

Since s and t can be bounded by a Lipschitz constant using SN and assuming $\|x\|$ is bounded by B and the last layer of s is a tanh function, then, according to the inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$,

it is easy to see that the transformation $y_{d+1:D} = x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) + t\left(x_{1:d}\right)$ has a bounded Lipschitz constant $B * \exp(1) * |s|_{\text{Lip}} + |t|_{\text{Lip}}$. Thus the operator norm $\|\frac{\partial y_{d+1:D}}{\partial x_{1:d}^T}\|_{op}$ is also bounded. Finally, by the equivalence of operator norm of Jacobian matrix and Lipschitz continuity (Lemma 2), we conclude that application of SN to the functions s and t will guarantee that the flow model is Lipschitz regularized.

3.4.5 Anomaly Detection from Trained Network

By discarding data points with low estimated densities during training and adding spectral normalization to the flow-based model, we could robustly train our density estimation model. During the testing phase, the density for each test point can be inferred by applying the trained network. The anomaly score for each test point is then derived from the estimated density, in which the higher the density, the lower is its anomaly score.

3.5 Experiment

We have conducted empirical studies to validate the proposed approach using both synthetic and benchmark real-world data.

3.5.1 Experiments on Synthetic Data

The analysis given in Lemma 1 shows that the robustness of the estimated density function in the presence of contaminated data depends on the smoothness of the prediction function. Assuming the density function for the clean data is smooth, then the smoothness of the prediction function depends on the distribution of anomalies. If the anomalies are uniformly distributed, then spectral normalization (SN) is not necessary to ensure robustness of the estimated density function. However, if the anomalies are concentrated in a small, local region, then SN is needed to provide robustness guarantees. We validate these results with the following toy example.

We generate a two-dimensional synthetic data, where the normal observations have a two-moon shape distribution, whereas the anomalies are generated under the following two settings:

- **Uniform**: The anomalies are randomly generated from a uniform distribution, which means the Lipschitz constant for the anomaly density function is 0.
- **Clustered**: The anomalies are generated from a concentrated Gaussian distribution. As a result, the Lipschitz constant of the density function for anomalies is large.

In both settings, the anomaly ratio is set to be 0.05. We compared the performance of our algorithm, **RobustRealNVP**, against **Real NVP** Dinh et al. (2016). We also investigated a variation of our framework without Lipschitz regularization via spectral normalization. We termed this approach as **TrimRealNVP**. We expect the performance of Real NVP to be severely hampered by the presence of anomalies in both settings. For anomalies that are uniformly distributed, we expect both TrimRealNVP and RobustRealNVP to perform equally well since they are both designed to alleviate the effect of anomalies during density estimation. For clustered anomalies, we RobustRealNVP to perform the best since TrimRealNVP does not guarantee smoothness of the predicted density function.

The results shown in Figure 3.2 are consistent with our expectation. While TrimRealNVP successfully alleviates the effect of uniform anomalies, it is largely affected by clustered anomalies (shown in red), assigning high density to the anomaly region. The estimated density functions of Real NVP are also significantly impacted by both types of anomalies, unlike RobustRealNVP, which can effectively handle both types of anomalies. This is because the SN effectively increases the smoothness of the loss landscape, thereby imposing a small Lipschitz constant, which according to our theorem, makes the gradient estimation more robust.

3.5.2 Experiments on Real-World Data

Datasets To demonstrate the breadth of its applicability, we perform experiments on two benchmark datasets:

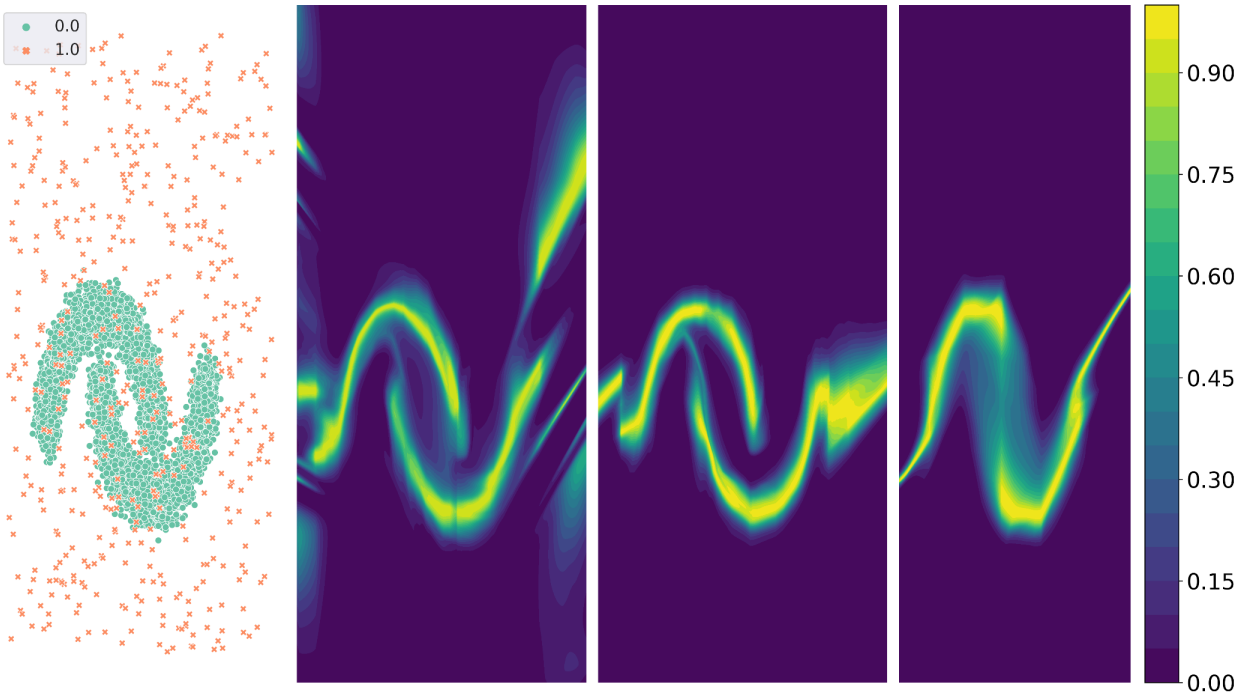


Figure 3.2: Effect of uniform anomalies on density estimation. Normal data are generated from a two-moon distribution while anomalies are from a uniform distribution (see left-most figure). Density estimation results of Real NVP, TrimRealNVP, and RobustRealNVP are shown from left to right.

a) Stony Brook ODDS library (Rayana, 2016): ODDS is a collection of different anomaly detection dataset, and we use 16 benchmark datasets in ODDS. A summary description of the data is given in the Appendix.

b) CIFAR10: This is a benchmark image data with high-dimensional features. Traditional algorithms such as LOF, isolation forest, etc., are expected to yield inferior results. Since the results on CIFAR10 are largely affected by the choice of backbone network structure, we preprocess the features for a fair comparison. Specifically, we first use the PyTorch official VGG19 network pretrained on ImageNet to extract the features from CIFAR10 and use the layer before final layer as the extracted features, which have 4096 dimensions. Finally, we apply PCA to reduce its dimensionality to 128 features.

Baseline Methods We compare the proposed **RobustRealNVP** framework against the following baselines:

- **AE** (autoencoder) and **VAE** (variational autoencoder), both of which use the reconstruction

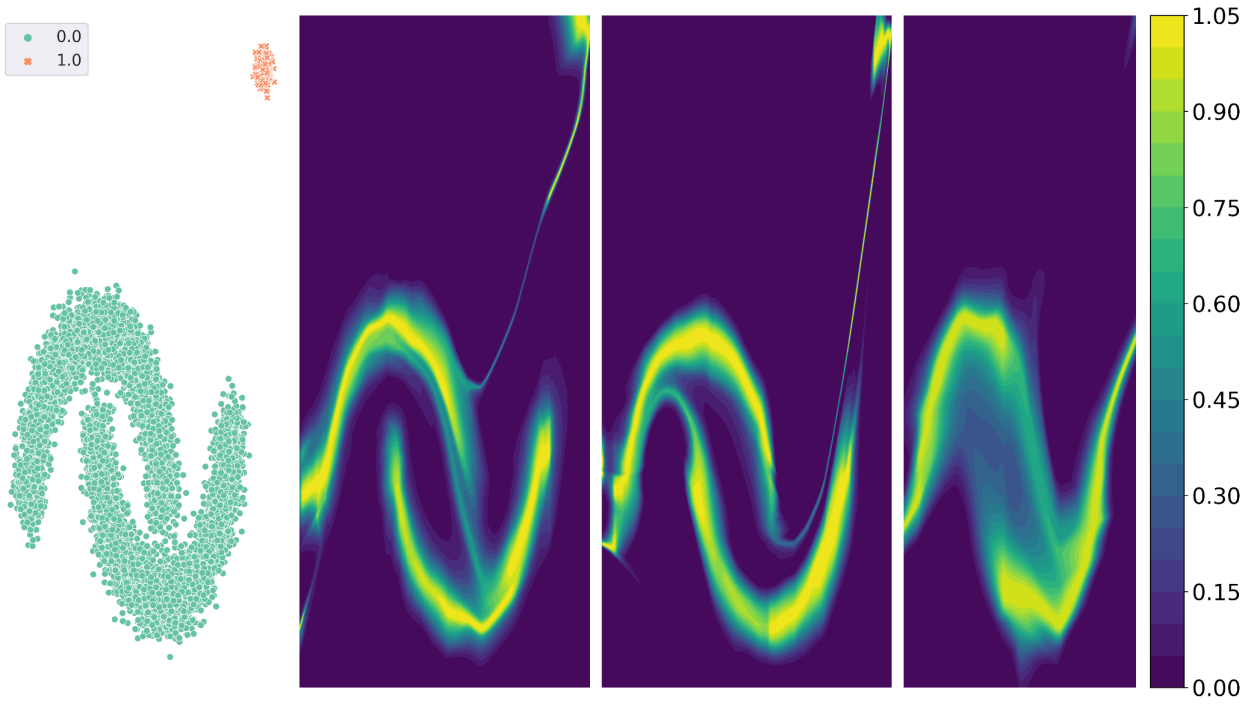


Figure 3.3: Effect of clustered anomalies on density estimation. Normal data are generated from a two-moon distribution while anomalies are from a concentrated Gaussian (see left-most figure). Density estimation results of Real NVP, TrimRealNVP, and RobustRealNVP are shown from left to right.

error as their anomaly scores.

- **Deep-SVDD Ruff et al. (2018)**: A deep implementation of one class SVM.
- **DAGMM Zong et al. (2018)**: Use Gaussian mixture model to estimate the density of the learned representation.
- **SO-GAAL Liu et al. (2019)**: Anomaly detection based on the generative adversarial networks.
- **OCSVM (Chen et al., 2001)**: A shallow one class SVM.
- **LOF Breunig et al. (2000)** (Local outlier factor): A local density-based anomaly detection method.
- **IF Liu et al. (2008)** (Isolation forest): An ensemble tree-based anomaly detection method.

We also report the performance of **Real NVP** and **TrimRealNVP**, with the latter for ablation study.

Experiment Settings For the ODDS dataset, we use 60% of the data for training and 40% for testing. For CIFAR10 dataset, 80% of the data are reserved for training while the remaining 20% for testing. CIFAR10 contains images from 10 classes. We create the anomalies as follows: First, we select one class as the normal class (with 5000 samples) and then create anomalies from the remaining classes in two ways: **i)** Anomalies are randomly sampled from other classes. **ii)** Anomalies are selected from one specific class. The training anomaly ratio is set to be 0.1. All the models are evaluated on a balanced CIFAR10 test set, containing 1000 samples for each class. All experiments are repeated with 5 different random seeds. For unsupervised AD, since we assume there is no clean data available, tuning the hyperparameter is trickier due to lack of evaluation metrics. Thus, we fixed the network structure for all methods across the different datasets. The details of the hyperparameters are given in the Appendix. Finally, we use AUC score as our evaluation metric since the metric does not require defining a threshold for anomaly.

3.5.3 Results for ODDS data

Table 3.1 shows the AUC scores for various methods on the ODDS datasets. While no method consistently outperforms all others given the diversity of the datasets, RobustRealNVP has the highest average rank compared to all other baselines. Isolation Forest also achieves highly competitive results. Surprisingly, all the sophisticated deep learning methods (i.e. SO-GAAL, DAGMM, Deep-SVDD) yield relatively poor results since most of them are designed for clean training data, which is unavailable in an unsupervised anomaly detection setting. Furthermore, they rely heavily on their hyperparameter tuning. We fix all the hyperparameters across the different datasets in our experiments since there are no clean validation data available. More discussion about the results are given in the Appendix.

3.5.4 Results for CIFAR10

Table 3.2 shows the AUC scores for various methods when different categories of images are chosen as normal class. RobustRealNVP outperforms all the baseline methods in 15 out of 20 experiments.

	RobustRealNVP	RealNVP	AE	VAE	SO-GAAL	DAGMM	Deep-SVDD	OCSVM	LOF	IF
vowels	**0.959±0.010**	0.889±0.038	0.879±0.020	0.503±0.045	0.637±0.197	0.340±0.103	0.206±0.035	0.765±0.036	0.947±0.014	0.776±0.017
pima	**0.678±0.024**	0.652±0.031	0.669±0.013	0.648±0.015	0.613±0.049	0.531±0.025	0.395±0.034	0.594±0.026	0.610±0.034	0.661±0.020
letter	**0.930±0.020**	0.915±0.017	0.829±0.031	0.521±0.042	0.601±0.060	0.433±0.034	0.465±0.039	0.557±0.038	0.845±0.026	0.621±0.030
cardio	0.737±0.028	0.712±0.039	0.867±0.020	**0.944±0.006**	0.473±0.075	0.862±0.031	0.505±0.056	0.936±0.002	0.684±0.027	0.925±0.009
arrhythmia	0.786±0.039	0.780±0.038	0.802±0.044	**0.811±0.034**	0.538±0.042	0.603±0.095	0.635±0.063	0.782±0.028	0.777±0.026	0.799±0.023
musk	0.991±0.011	0.794±0.074	0.998±0.003	0.994±0.002	0.234±0.193	0.903±0.130	0.829±0.048	**1.000±0.000**	0.353±0.054	0.997±0.003
mnist	0.818±0.009	0.820±0.015	0.802±0.009	0.778±0.009	0.795±0.025	0.652±0.077	0.538±0.048	**0.835±0.012**	0.698±0.013	0.805±0.007
satimage-2	0.943±0.014	0.915±0.021	0.818±0.069	0.966±0.008	0.789±0.177	0.853±0.113	0.739±0.088	**0.998±0.003**	0.428±0.109	0.996±0.005
satellite	**0.709±0.010**	0.690±0.007	0.575±0.068	0.538±0.016	0.640±0.070	0.667±0.189	0.631±0.016	0.650±0.014	0.570±0.005	0.706±0.026
mammography	0.841±0.018	0.855±0.013	0.853±0.015	0.864±0.014	0.204±0.026	0.187±0.000	0.272±0.009	**0.881±0.015**	0.768±0.024	0.873±0.019
thyroid	0.961±0.011	0.956±0.005	0.928±0.020	0.839±0.011	**0.984±0.005**	0.582±0.095	0.704±0.027	0.960±0.006	0.898±0.017	0.979±0.006
annthyroid	**0.880±0.025**	0.864±0.024	0.675±0.022	0.589±0.021	0.679±0.022	0.506±0.020	0.591±0.014	0.599±0.013	0.711±0.022	0.829±0.015
ionosphere	0.900±0.020	**0.914±0.015**	0.821±0.010	0.763±0.015	0.783±0.080	0.467±0.082	0.735±0.053	0.812±0.039	0.879±0.022	0.842±0.021
pendigits	0.748±0.023	0.730±0.024	0.685±0.073	0.931±0.006	0.257±0.053	0.872±0.068	0.613±0.071	0.935±0.003	0.472±0.029	0.943±0.013
shuttle	0.995±0.003	0.825±0.032	0.921±0.013	0.987±0.001	0.571±0.316	0.890±0.109	0.531±0.290	0.985±0.001	0.529±0.017	**0.997±0.001**
glass	0.786±0.039	0.747±0.056	0.570±0.152	0.626±0.134	0.420±0.112	0.852±0.084	0.756±0.114	0.522±0.207	0.756±0.141	**0.706±0.058**
Average rank	**2.937**	4.375	5	5.5	7.4375	7.375	8.34375	4.3125	6.6525	3.0625

Table 3.1: Average and standard deviation of AUC scores for the ODDS datasets (across 5 different random seeds). The last row corresponds to average rank of each method.

	RobustRealNVP	RealNVP	AE	VAE	SO-GAAL	DAGMM	SVDD	OCSVM	LOF	IF
airplane-u	**0.786±0.012**	0.784±0.012	0.582±0.010	0.583±0.012	0.534±0.036	0.462±0.024	0.484±0.088	0.582±0.007	0.743±0.008	0.582±0.010
automobile-u	**0.818±0.028**	0.782±0.022	0.486±0.010	0.505±0.010	0.490±0.072	0.393±0.041	0.565±0.067	0.486±0.008	0.750±0.008	0.502±0.008
bird-u	**0.694±0.070**	0.691±0.062	0.556±0.007	0.562±0.005	0.484±0.015	0.461±0.052	0.517±0.118	0.565±0.010	0.633±0.015	0.552±0.016
cat-u	**0.772±0.077**	0.726±0.076	0.484±0.009	0.499±0.015	0.526±0.021	0.433±0.034	0.493±0.098	0.517±0.008	0.582±0.005	0.503±0.010
deer-u	**0.801±0.006**	0.781±0.033	0.604±0.010	0.612±0.010	0.431±0.072	0.415±0.046	0.525±0.069	0.611±0.007	0.723±0.009	0.597±0.013
dog-u	**0.760±0.044**	0.704±0.050	0.444±0.003	0.450±0.006	0.535±0.065	0.387±0.061	0.439±0.124	0.490±0.017	0.587±0.008	0.441±0.010
frog-u	**0.807±0.022**	0.797±0.007	0.599±0.016	0.592±0.015	0.532±0.061	0.365±0.046	0.434±0.062	0.607±0.008	0.821±0.006	0.587±0.007
horse-u	**0.786±0.013**	0.769±0.057	0.504±0.014	0.503±0.011	0.510±0.052	0.468±0.032	0.510±0.069	0.534±0.008	0.670±0.018	0.511±0.017
truck-u	0.770±0.033	**0.838±0.027**	0.597±0.010	0.598±0.011	0.477±0.110	0.445±0.055	0.515±0.098	0.605±0.018	0.778±0.006	0.603±0.008
ship-u	0.798±0.006	**0.802±0.009**	0.493±0.016	0.489±0.006	0.410±0.055	0.461±0.039	0.495±0.061	0.505±0.006	0.738±0.009	0.506±0.016
airplane-c	**0.752±0.018**	0.748±0.023	0.611±0.013	0.619±0.010	0.512±0.035	0.437±0.057	0.464±0.116	0.616±0.008	0.687±0.009	0.619±0.021
automobile-c	**0.850±0.020**	0.843±0.014	0.469±0.016	0.466±0.007	0.447±0.054	0.420±0.044	0.496±0.113	0.448±0.012	0.748±0.017	0.469±0.017
bird-c	0.640±0.025	0.639±0.030	0.580±0.010	0.561±0.016	0.485±0.035	0.450±0.062	0.517±0.058	0.581±0.017	**0.644±0.006**	0.579±0.008
cat-c	0.462±0.017	0.451±0.027	0.427±0.008	0.417±0.005	**0.545±0.072**	0.425±0.019	0.489±0.059	0.449±0.004	0.450±0.005	0.426±0.015
deer-c	**0.825±0.019**	0.808±0.009	0.676±0.008	0.669±0.001	0.456±0.056	0.529±0.027	0.544±0.111	0.643±0.010	0.761±0.005	0.656±0.012
dog-c	0.500±0.021	0.487±0.023	0.410±0.012	0.404±0.005	**0.585±0.085**	0.372±0.043	0.481±0.055	0.429±0.006	0.416±0.017	0.418±0.016
frog-c	**0.826±0.015**	0.788±0.020	0.593±0.008	0.599±0.016	0.436±0.070	0.435±0.014	0.507±0.102	0.614±0.009	0.814±0.005	0.588±0.013
horse-c	**0.575±0.023**	0.550±0.025	0.462±0.005	0.460±0.003	0.512±0.114	0.386±0.039	0.451±0.090	0.490±0.002	0.558±0.013	0.475±0.022
truck-c	**0.657±0.030**	0.664±0.013	0.623±0.015	0.622±0.006	0.503±0.047	0.467±0.073	0.518±0.102	0.622±0.012	0.650±0.011	0.627±0.011
ship-c	**0.671±0.025**	0.653±0.023	0.450±0.012	0.446±0.013	0.492±0.064	0.433±0.070	0.515±0.132	0.464±0.002	0.598±0.005	0.472±0.003

Table 3.2: Average and standard deviation of AUC scores for the CIFAR10 dataset (across 5 different random seeds). The first column shows the category chosen as normal class. 'class'-c means the training anomalies are clustered whereas 'class'-u means the anomalies are uniformly sampled from other classes.

Similar to the ODDS results, the deep learning baseline do not perform quite as well. One possible reason is that, instead of using raw images, we use the preprocessed features as input for fair comparison against the non-deep baselines. Thus, the deep methods are unable to make use of convolutional neural networks, which degrade their performance. However, even compared with the results in the original Deep-SVDD paper Ruff et al. (2018), which is trained on the uniform anomalies by using convolutional neural networks, our results still perform better than their results. More discussion on DAGMM and Deep-SVDD is given in the Appendix.

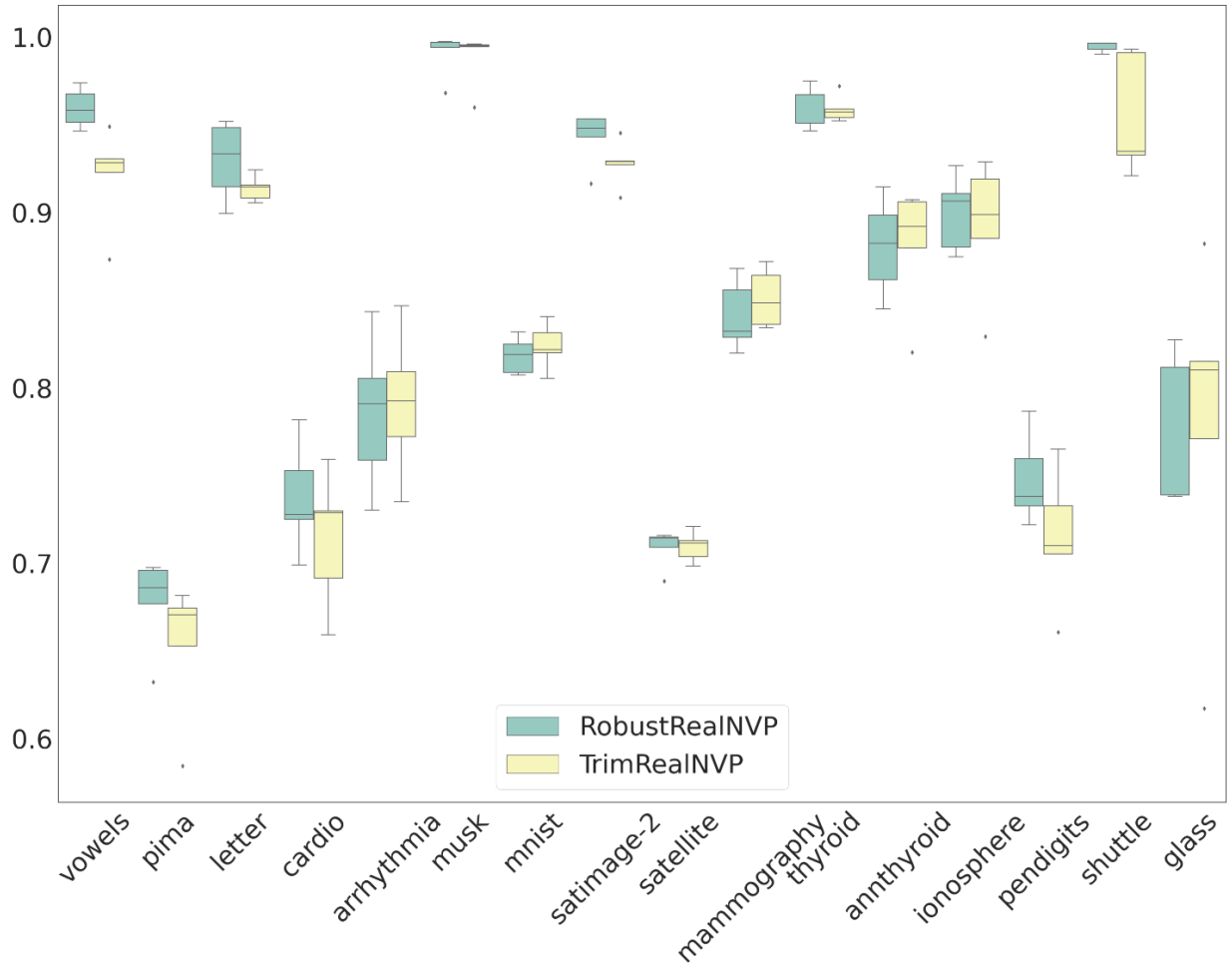


Figure 3.4: Ablation study for the spectral normalization module. The TrimRealNVP is the RobustRealNVP without the spectral normalization module.

3.5.5 Ablation Study for Spectral Normalization

We perform ablation study using the ODDS dataset to show the effectiveness of spectral normalization. In Figure 3.4, we see that RobustRealNVP outperforms TrimRealNVP in 8 out of the 16 datasets and tied in at least 4 of the remaining datasets. These results along with the synthetic experiment results demonstrated the effectiveness of using spectral normalization in our framework.

3.5.6 Sensitivity Analysis for ϵ

In real-world application, the true anomaly ratio (aka. contamination ratio) is often unknown. Therefore, it is hard to determine how many data points should be discarded in each gradient descent

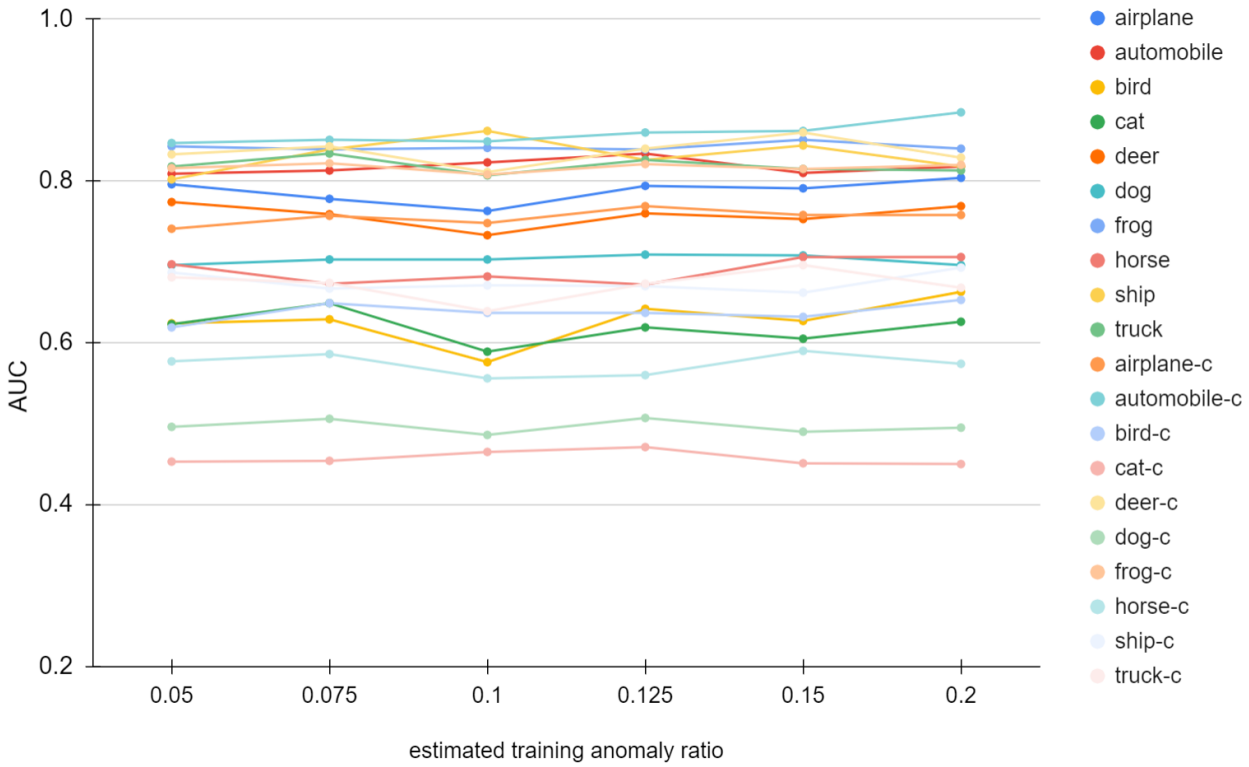


Figure 3.5: Sensitivity analysis for estimated ϵ. The ground truth ϵ is 0.1. We give the results of the estimated ϵ equals to 0.05, 0.075, 0.1, 0.125, 0.15, 0.2. The y-value is the averaged AUC across 5 random seeds.

step. To investigate how the performance of RobustRealNVP would vary when the anomaly ratio is over- or underestimated, we perform sensitivity analysis on the CIFAR10 datasets by varying the estimated contamination ratio from 0.05 to 0.2 (note that the true anomaly ratio is 0.1). The results shown in Figure 3.5 suggest that our method remains stable under most settings even though the anomaly ratio was overestimated or underestimated by as high as 15%. In addition, we also perform sensitivity analysis on the CIFAR10 datasets when the true anomaly ratio is varied from 0.1 to 0.4. The results for Real NVP and RobustRealNVP shown in Figure 3.6 suggest that RobustRealNVP performs consistently better than the Real NVP in most settings.

3.6 Conclusion

This paper presents RobustRealNVP, a deep, density-based unsupervised anomaly detection method that is robust against training contamination. We demonstrate the effectiveness of our framework from

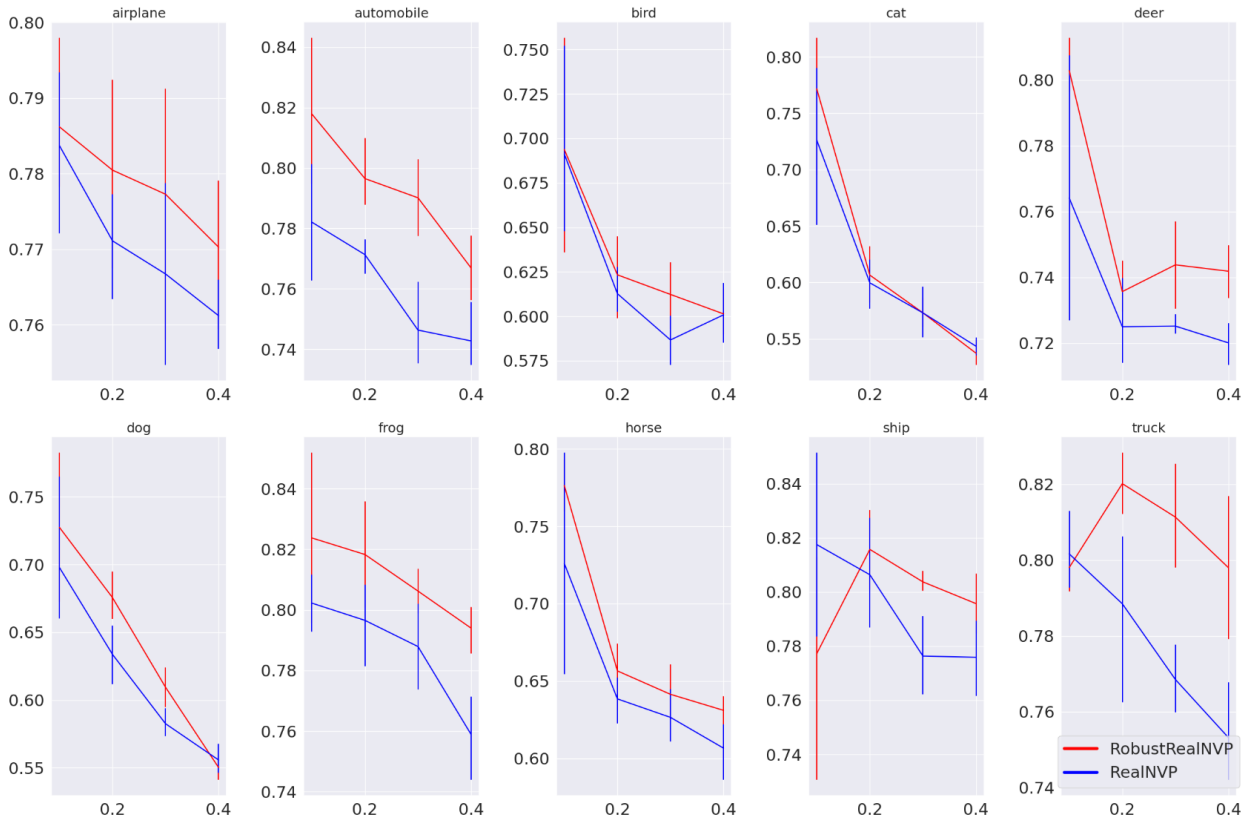


Figure 3.6: Sensitivity Analysis of ground-truth Training anomaly ratio for uniform anomalies from 0.1 to 0.4. Red is RobustRealNVP, blue is Real NVP.

both theoretical and empirical perspectives. For future work, we will investigate the applicability of the framework to other flow-based models beside Real NVP.

CHAPTER 4

ROBUST LEARNING DEEP NEURAL NETWORKS UNDER AGNOSTIC CORRUPTED SUPERVISION

In this chapter, we introduce the Provable Robust Learning (PRL), a robust learning algorithm which could tolerate the agnostic label corruption in training data.

4.1 Introduction

Corrupted supervision is a common issue in real-world learning tasks, where the learning targets are potentially noisy due to errors in the data collection or labeling process. Such corruptions can have severe consequences especially in deep learning models, whose large degree-of-freedom makes them easier to memorize the corrupted examples, and thus, susceptible to overfitting (Zhang et al., 2016).

There have been extensive efforts to achieve robustness against corrupted supervision. A natural approach to deal with corrupted supervision in deep neural networks (DNNs) is to reduce the model exposure to corrupted data points during training. By detecting and filtering (or re-weighting) the possible corrupted samples, the learning is expected to deliver a model that is similar to the one trained on clean data (without corruption) (Kumar et al., 2010; Han et al., 2018; Zheng et al., 2020). There are various criteria designed to identify the corrupted data points in training. For example, Kumar et al. (2010); Han et al. (2018); Jiang et al. (2018) leveraged the loss function values of the data points; Zheng et al. (2020) considered prediction uncertainty for filtering data; Malach and Shalev-Shwartz (2017) used the disagreement between two deep networks; while Reed et al. (2014) utilized the prediction consistency of neighboring iterations. The success of these methods highly depends on the effectiveness of the detection criteria in correctly identifying the corrupted data points. Since the true corrupted points remain unknown throughout the learning process, such “unsupervised" methods may not be effective, either lacking in theoretical guarantees of robustness (Han et al., 2018; Reed et al., 2014; Malach and Shalev-Shwartz, 2017; Li et al.,

2017) or providing guarantees only under the assumption that prior knowledge is available about the type of corruption present (Zheng et al., 2020; Shah et al., 2020; Patrini et al., 2017; Yi and Wu, 2019). Most existing theoretical guarantees under agnostic corruption during optimization are focused on convex losses (Prasad et al., 2018) or linear models (Bhatia et al., 2015, 2017), and thus cannot be directly extended to DNNs. (Diakonikolas et al., 2019) developed a generalized non-convex optimization algorithm against agnostic corruptions. However, it is not optimized for the label/supervision corruption and has a high space complexity, which may incur prohibitive costs when applied to typical DNNs with a large amount of parameters. Furthermore, many existing approaches are exclusively designed for classification problems (e.g., Malach and Shalev-Shwartz (2017); Reed et al. (2014); Menon et al. (2019); Zheng et al. (2020)); extending them to solve regression problems is not that straightforward.

To tackle these challenges, this chapter presents a unified optimization framework with robustness guarantees, without any assumptions on how supervisions are corrupted, and is applicable to both classification and regression problems. Instead of designing a criterion for accurate detection of corrupted samples, we focus on limiting the collective impact of corrupted samples during the learning process through robust mean estimation of the gradients. Specifically, if our estimated average gradient is close to the expected gradient from the clean data during the learning iterations, then the final model will be close to the model trained on clean data. As such, a corrupted data point can still be used during the training as long as it does not considerably alter the average gradient. This observation has remarkably impact our algorithm design: instead of explicitly quantifying (and identifying) individual corrupted data points, which is a hard problem in itself, we are now dealing with an easier task, i.e., eliminating training data points that significantly distort the mean gradient estimation. One immediate consequence of this design is that, even when a corrupted data point failed to be excluded by the proposed algorithm, the data point will likely have very limited impact on the overall gradient, as compared with state-of-the-art filtering data points based on loss values. Compared to state-of-the-art robust optimization methods (Prasad et al., 2018; Diakonikolas et al., 2019) that require the more expensive SVD computation on the gradient matrix, we fully utilize the

gradient structure when the corruptions are exclusively on the supervision to make our algorithm applicable to DNNs. Moreover, when only supervision are corrupted, we improve the error bound from $O(\sqrt{\epsilon})$ to $O(\epsilon)$, where ϵ is the corruption rate. We perform experiments on both regression and classification with corrupted supervision on multiple benchmark datasets. The results show that the proposed method outperforms state-of-the-art.

4.2 Related Work

Learning from corrupted data (Huber, 1992) has attracted considerable attention in the machine learning community (Natarajan et al., 2013). Many recent studies have investigated robustness of classification tasks with noisy labels. For example, Kumar et al. (2010) proposed a self-paced learning (SPL) approach, which assigns higher weights to examples with smaller loss. A similar idea was used in curriculum learning (Bengio et al., 2009), in which the model learns the easy concept before the harder ones. Alternative methods inspired by SPL include learning the data weights (Jiang et al., 2018) and collaborative learning (Han et al., 2018; Yu et al., 2019). Label correction (Patrini et al., 2017; Li et al., 2017; Yi and Wu, 2019) is another approach, which attempts to revise the original labels of the data to recover clean labels from corrupted ones. However, since we do not have access to which data points are corrupted, it is harder to obtain provable guarantees for label correction without strong assumptions about the corruption type.

Accurate estimation of the gradient is a key step for successful optimization. The relationship between gradient estimation and its final convergence has been widely studied in the optimization community. Since computing an approximated (and potentially biased) gradient is often more efficient than computing the exact gradient, many studies used approximated gradients to optimize their models and showed that they suffer from the biased estimation problem if there is no assumption on the gradient estimation (d'Aspremont, 2008; Schmidt et al., 2011; Bernstein et al., 2018; Hu et al., 2020; Ajalloeian and Stich, 2020).

A closely related topic is robust estimation of the mean. Given corrupted data, robust mean estimation aims at generating an estimated mean $\hat{\mu}$ such that the difference between the estimated

mean on corrupted data and the mean of clean data $\|\hat{\mu} - \mu\|_2$ is minimized. The median or trimmed mean have been shown to be optimal statistics for mean estimation in one-dimensional data (Huber, 1992). However, robustness in high dimension is more challenging since applying the coordinate-wise optimal robust estimator would lead to an error factor $O(\sqrt{d})$ that scales with dimensionality of the data. Although classical methods such as Tukey median (Tukey, 1975) have successfully designed algorithms to eliminate the $O(\sqrt{d})$ error, the algorithms cannot run in polynomial-time. More recently, Diakonikolas et al. (2016); Lai et al. (2016) successfully designed polynomial-time algorithms with dimension-free error bounds. The results have been widely applied to improve algorithmic efficiency in various scenarios (Dong et al., 2019; Cheng et al., 2020).

Robust optimization is designed to improve algorithm robustness in the presence of corrupted data. Most existing efforts have focused on linear regression and its variants (Bhatia et al., 2015, 2017; Shen and Sanghavi, 2019) or convex problems (Prasad et al., 2018). Thus, their results cannot be directly generalized to DNNs. Although Diakonikolas et al. (2019) presented a generalized non-convex optimization method with an agnostic corruption guarantee, the space complexity of the algorithm is high, and cannot be applied to DNNs due to large parameter sizes. We will discuss (Diakonikolas et al., 2019) in more details later.

4.3 Methodology

Before introducing our algorithm, we first present our corrupted supervision setting. To characterize agnostic corruptions, we assume there is an adversary that tries to corrupt the supervision of clean data. There is no restriction on how the adversary corrupts the supervision, which can either be randomly permuting the target, or in a way that maximizes negative impact (i.e., lower performance) on the model. The adversary can choose up to ϵ fraction of the clean target $\mathbf{D}_y \in \mathbb{R}^{n \times q}$ and alter the selected rows of $\mathbf{D}_y$ to arbitrary valid numbers, generating $\mathbf{D}_y^\epsilon \in \mathbb{R}^{n \times q}$. The adversary then returns the corrupted dataset $\mathbf{D}_x$, $\mathbf{D}_y^\epsilon$ to our learning algorithm $\mathcal{A}$. The adversary can have full knowledge of the data or even the learning algorithm $\mathcal{A}$. The only constraint on the adversary is the corruption rate, ϵ. A key question is: Given a dataset $\mathbf{D}_x \in \mathbb{R}^{n \times p}$, $\mathbf{D}_y^\epsilon \in \mathbb{R}^{n \times q}$, with ϵ-fraction of

corrupted supervision, and a learning objective $\phi : \mathbb{R}^p \times \mathbb{R}^q \times \mathbb{R}^d \to \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$, can we output the parameters θ such that $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$ is minimized?

When $\epsilon = 0$, $\mathbf{D}_y^\epsilon = \mathbf{D}_y$ and the learning is performed on clean data. The stochastic gradient descent algorithm may converge to a stationary point where $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\| = 0$. However, this is no longer the case when the supervision is corrupted as above due to the impact of the corrupted data on θ. We thus want an efficient algorithm to find a model θ that minimizes $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$. A robust model θ should have a small value of $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$, and we hypothesize that a smaller $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$ leads to better generalization.

4.3.1 Stochastic Gradient Descent with Biased Gradient

A direct consequence of corrupted supervision is biased gradient estimation. In this section, we will first analyze how such biased gradient estimation affects the robustness of learning. The classical analysis of stochastic gradient descent (SGD) requires access to the stochastic gradient oracle, which is an unbiased estimation of the true gradient. However, corrupted supervision leads to corrupted gradients, which makes it difficult to get unbiased gradient estimation without assumptions on how the gradients are corrupted.

The convergence of biased gradient has been studied via a series of previous works (Schmidt et al., 2011; Bernstein et al., 2018; Hu et al., 2020; Ajalloeian and Stich, 2020; Scaman and Malherbe, 2020). We show a similar theorem below for the sake of completeness. Before We gave the theorem of how biased gradient affect the final convergence of SGD. We introduce several assumptions and definition first:

Assumption 6 (L-smoothness) *The function ϕ: $\mathbb{R}^d \to \mathbb{R}$ is differentiable and there exists a constant $L > 0$ such that for all $\theta_1, \theta_2 \in \mathbb{R}^d$, we have $\phi(\theta_2) \leq \phi(\theta_1) + \langle \nabla\phi(\theta_1), \theta_2 - \theta_1 \rangle + \frac{L}{2}\|\theta_2 - \theta_1\|^2$*

Definition 3 (Biased gradient oracle) *A map $\mathbf{g} : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^d$, such that $\mathbf{g}(\theta, \xi) = \nabla\phi(\theta) + \mathbf{b}(\theta, \xi) + \mathbf{n}(\theta, \xi)$ for a bias $\mathbf{b} : \mathbb{R}^d \to \mathbb{R}^d$ and zero-mean noise $\mathbf{n} : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^d$, that is $\mathbb{E}_\xi(\mathbf{n}(\theta, \xi)) = 0$.*

Compared to standard stochastic gradient oracle, the above definition introduces the bias term **b**. In noisy-label settings, the **b** is generated by the data with corrupted labels.

Assumption 7 (σ-Bounded noise) *There exists constants $\sigma > 0$, such that $\mathbb{E}_\xi \|\mathbf{n}(\theta, \xi)\|^2 \leq \sigma, \quad \forall \theta \in \mathbb{R}^d$*

Assumption 8 (ζ-Bounded bias) *There exists constants $\zeta > 0$, such that for any ξ, we have $\|\mathbf{b}(\theta, \xi)\|^2 \leq \zeta^2, \quad \forall \theta \in \mathbb{R}^d$*

For simplicity, assume the learning rate is constant γ, then in every iteration, the biased SGD performs update $\theta_{t+1} \leftarrow \theta_t - \gamma_t \mathbf{g}(\theta_t, \xi)$. Then the following theorem showed the gradient norm convergence with biased SGD.

Theorem 4 (Convergence of Biased SGD(formal)) *Under assumptions 6, 7, 8, define $F = \phi(\theta_0) - \phi^*$ and step size $\gamma = \min\left\{\frac{1}{L}, (\sqrt{\frac{LF}{\sigma T}})\right\}$, denote the desired accuracy as k, then*

$$T = O\left(\frac{1}{k} + \frac{\sigma^2}{k^2}\right)$$

iterations are sufficient to obtain $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\theta_t)\|^2\right) = O(k + \zeta^2)$.

Remark 5 *Let $k = \zeta^2$, $T = O\left(\frac{1}{\zeta^2} + \frac{\sigma^2}{\zeta^4}\right)$ iterations is sufficient to get $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\theta_t)\|^2\right) = O(\zeta^2)$, and performing more iterations does not improve the accuracy in terms of convergence.*

The difference between the above theorem and the typical convergence theorem for SGD is that we are using a biased gradient estimation. According to Theorem 4, robust estimation of the gradient **g** is the key to ensure a robust model that converges to the clean solution. We also assume the loss function has the form of $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, in which many commonly used loss functions belong to this category.

4.3.2 Robust Gradient Estimation for General Data Corruption

Before discussing the corrupted supervision setting, we first review the general corruption setting, where the corruptions may be present in both the supervision and input features. A naïve approach is to apply a robust coordinate-wise gradient estimation approach such as coordinate-wise median for gradient estimation. However, by using the coordinate-wise robust estimator, the L2 norm of the difference between the estimated and ground-truth gradients contains a factor of $O(\sqrt{d})$, where d is the gradient dimension. This error term induces a high penalty for high dimensional models and thus cannot be applied to DNNs. Recently, (Diakonikolas et al., 2016) proposed a robust mean estimator with dimension-free error for general types of corruptions. (Diakonikolas et al., 2019) achieves an error rate of $O(\sqrt{\epsilon})$ for general corruption. This begs the question whether it is possible to further improve the $O(\sqrt{\epsilon})$ error rate if we consider only corrupted supervision.

To motivate our main algorithm (Alg. 4), we first introduce and investigate Alg. 3 for general corruption with dimension-dependent error. The algorithm excludes data points with large gradient norms and uses the empirical mean of the remaining points to update the gradient. Cor. 1 below describes its robustness property.

Algorithm 3: (***PRL(G)***) Provable Robust Learning for General Corrupted Data

input: dataset $\mathbf{D}_x, \mathbf{D}_y^\epsilon$ with corrupted supervision, learning rate γ_t;

for $t = 1$ to maxiter **do**

Randomly sample a mini-batch $\mathbf{M}$ from $\mathbf{D}_x, \mathbf{D}_y^\epsilon$

Calculate the individual gradient $\tilde{\mathbf{G}}$ for $\mathbf{M}$

For each row $\mathbf{z}_i$ in $\mathbf{G}$, calculate the l2 norm $\|\mathbf{z}_i\|$

Choose the ϵ-fraction rows with large $\|\mathbf{z}_i\|$

Remove those selected rows, and return the empirical mean of the rest points as $\hat{\mu}$.

Update model $\theta_{t+1} = \theta_t - \gamma_t \hat{\mu}$

end for

return model parameter θ

Corollary 1 (Robust Optimization For Corrupted Data) *Given the assumptions in Theorem 4, applying Algorithm 3 to ϵ-fraction corrupted data yields $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\mathbf{x}_t)\|\right) = O(\epsilon\sqrt{d})$ for large enough T, where d is the number of the parameters.*

Remark 6 *The term $\sqrt{d}$ is due to the upper bound of d-dimensional gradient norm of clean data. The term can be removed if we assume the gradient norm is uniformly bounded by L. However, this assumption is too strong for robust gradient estimation. We will show that later that the assumption can be relaxed (i.e. bounded maximum singular value of gradient) under the corrupted supervision setting.*

The error bound in the above corollary has several practical issues. First, the bound grows with increasing dimensionality, and thus, is prohibitive when working with DNNs, which have extremely large gradient dimensions due to their massive number of parameters. Even though one can improve the factor $\sqrt{\epsilon}$ Diakonikolas et al. (2019) to ϵ, the results remain impractical compared to the dimension-free $O(\sqrt{\epsilon})$ guarantee in (Diakonikolas et al., 2019), since above bound involves the dimension related term $\sqrt{d}$.

Efficiency is another main limitation of Alg. 3 since it requires computing individual gradients. Although there are advanced methods available to obtain the individual gradient, e.g., (Goodfellow, 2015), they are still relatively slow compared to the commonly used back-propagation algorithm. Moreover, many of them are not compatible with other components of DNN such as batch normalization (BN). Since the individual gradients are not independent within the BN, they will lose the benefits of parallelization. We will show below that the above issues can be addressed under the corrupted supervision setting and propose a practical solution that easily scales for DNNs.

4.3.3 Robust Gradient Estimation for One Dimensional Corrupted Supervision

In this section, we show that the robustness bound in Cor. 1 can be improved if we assume the corruption comes from the supervision only. In addition, by fully exploiting the gradient structure of the corrupted supervision, our algorithm is much more efficient and is compatible with batch normalization. We begin with a 1-dimensional supervision setting (e.g., binary classification or single-target regression) to illustrate this intuition and will extend it more general settings in the next section. Consider a supervised learning problem with input features $\mathbf{X} \in \mathbb{R}^{n \times p}$ and supervision

$\mathbf{y} \in \mathbb{R}^n$. The goal is to learn a function f, parameterized by $\theta \in \mathbb{R}^d$, by minimizing the following loss $\min_\theta \sum_{i=1}^n \phi_i = \min_\theta \sum_{i=1}^n \mathcal{L}(y_i, f(\mathbf{x}_i, \theta))$. The gradient for a data point i is $\nabla_\theta \phi_i = \frac{\partial l_i}{\partial f_i} \frac{\partial f_i}{\partial \theta} = \alpha_i \mathbf{g}_i$.

In general, if the corrupted gradients drive the gradient estimation away from the clean gradient, they are either large in magnitude or systematically change the direction of the gradient Diakonikolas et al. (2019). However, our key observation is that, when only the supervision is corrupted, the corruption contributes only to the term $\alpha_i = \frac{\partial l_i}{\partial f_i}$, which is a scalar in the one-dimensional setting. In other words, given the clean gradient of i^{th} point, $g_i \in \mathbb{R}^d$, the corrupted supervision only re-scales the gradient vector, changing the gradient from $\alpha_i \mathbf{g}_i$ to $\delta_i \mathbf{g}_i$, where $\delta_i = \frac{\partial l_i^\epsilon}{\partial f_i}$. As such, it is unlikely for the corrupted supervision to systematically change the gradient direction.

The fact that corrupted supervision re-scales the clean gradient can be exploited to reshape the robust optimization problem. Suppose we update our model in each iteration by $\theta^+ = \theta - \gamma \mu(\mathbf{G})$, where $\mu(\cdot)$ denotes the empirical mean function and $\mathbf{G} = [\nabla_\theta \phi_1^T, \ldots, \nabla_\theta \phi_m^T] \in \mathbb{R}^{m \times d}$ is the gradient matrix for a mini-batch of size m. We consider the following problem:

Problem 1 (Robust Gradient Estimation for One Dimensional Corrupted Supervision) *Given a clean gradient matrix $\mathbf{G} \in \mathbb{R}^{m \times d}$, an ϵ-corrupted matrix $\tilde{\mathbf{G}}$ with at most ϵ-fraction rows are corrupted from $\alpha_i \mathbf{g}_i$ to $\delta_i \mathbf{g}_i$, design an algorithm $\mathcal{A} : \mathbb{R}^{m \times d} \to \mathbb{R}^d$ that minimizes $\|\mu(\mathbf{G}) - \mathcal{A}(\tilde{\mathbf{G}})\|$.*

Note that when $\|\delta_i\|$ is large, the corrupted gradient will have a large effect on the empirical mean, and if $\|\delta_i\|$ is small, the corrupted gradient will have a limited effect on the empirical mean. This motivates us to develop an algorithm that filters out data points by the loss layer gradient $\|\frac{\partial l_i}{\partial f_i}\|$. If the norm of the loss layer gradient of a data point is large (in one-dimensional case, this gradient reduces to a scalar and the norm becomes its absolute value), we exclude the data point when computing the empirical mean of gradients for this iteration. Note that this algorithm is applicable to both regression and classification problems. Especially, when using the mean squared error (MSE) loss for regression, its gradient norm is exactly the loss itself, and the algorithm reduces to self-paced learning Kumar et al. (2010) or trim loss Shen and Sanghavi (2019). We summarize the procedure in Algorithm 4 and will extend it to the more general multi-dimensional case in the next section.

Algorithm 4: (***PRL(L)***) Efficient Provable Robust Learning for Corrupted Supervision

input: dataset $\mathbf{D}_x, \mathbf{D}_y^\epsilon$ with corrupted supervision, learning rate γ_t;

for $t = 1$ to maxiter **do**

Randomly sample a mini-batch $\mathbf{M}$ from $\mathbf{D}_x, \mathbf{D}_y^\epsilon$

Compute the predicted label $\hat{\mathbf{Y}}$ from $\mathbf{M}$

Calculate the gradient norm for the loss layer, (e.g., $\|\hat{\mathbf{y}} - \mathbf{y}\|$ for mean square error or cross entropy)

$\tilde{\mathbf{M}} \leftarrow \mathbf{M} - \mathbf{M}_\tau$, where $\mathbf{M}_\tau$ is the top-τ fraction of data points with largest $\|\hat{\mathbf{y}} - \mathbf{y}\|$

Update model $\theta_{t+1} = \theta_t - \gamma_t \hat{\mu}$, where $\hat{\mu}$ is the empirical mean of $\tilde{\mathbf{M}}$

end for

return model parameter θ;

4.3.4 Extension to Multi-Dimensional Corrupted Supervision

To extend our approach to multi-dimensional case, let q be the output dimension of y. The gradient for each data point i is $\nabla_\theta \phi_i = \frac{\partial l_i}{\partial f_i}\frac{\partial f_i}{\partial \theta}$, where $\frac{\partial l_i}{\partial f_i} \in \mathbb{R}^q$ is the gradient of the loss with respect to model output, and $\frac{\partial f_i}{\partial \theta} \in \mathbb{R}^{q\times d}$ is the gradient of the model output with respect to model parameters. When the supervision is corrupted, the corruption affects the term $\frac{\partial l_i}{\partial f_i}$, which is now a vector. Let $\delta_i = \frac{\partial l_i^\epsilon}{\partial f_i} \in \mathbb{R}^q$, $\alpha_i = \frac{\partial l_i}{\partial f_i} \in \mathbb{R}^q$, $\mathbf{W}_i = \frac{\partial f_i}{\partial \theta} \in \mathbb{R}^{q\times d}$, and m be the mini-batch size. Denote the clean gradient matrix as $\mathbf{G} \in \mathbb{R}^{m\times d}$, where the i_{th} row of gradient matrix $\mathbf{g}_i = \alpha_i \mathbf{W}_i$. The multi-dimensional robust gradient estimation problem is defined as follows.

Problem 2 (Robust Gradient Estimation for Multi-Dimensional Corrupted Supervision) *Given a clean gradient matrix* $\mathbf{G}$*, an* ϵ*-corrupted matrix* $\tilde{\mathbf{G}}$ *with at most* ϵ*-fraction rows corrupted from* $\alpha_i \mathbf{W}_i$ *to* $\delta_i \mathbf{W}_i$*, design an algorithm* $\mathcal{A} : \mathbb{R}^{m\times d} \to \mathbb{R}^d$ *that minimizes* $\|\mu(\mathbf{G}) - \mathcal{A}(\tilde{\mathbf{G}})\|$.

We begin our analysis by examining the effects of randomized filtering-base algorithms, i.e., using the empirical mean gradient of the random selected $(1-\epsilon)$-fraction subset to estimate clean averaged gradient. Randomized filtering-based algorithm does not serve a practical robust learning approach, but its analysis leads to important insights into designing one. We have the following lemma for ***any randomized filtering-based algorithm*** (proof is given in Appendix):

Lemma 4 (Gradient Estimation Error for Random Dropping ϵ-fraction Data) *Let* $\tilde{\mathbf{G}} \in \mathbb{R}^{m\times d}$ *be a corrupted matrix generated as in Problem 2, and* $\mathbf{G} \in \mathbb{R}^{m\times d}$ *be the original, clean gradient matrix. Suppose an arbitrary* $(1-\epsilon)$*-fraction rows are selected from* $\tilde{\mathbf{G}}$ *to form the matrix* $\mathbf{N} \in \mathbb{R}^{n\times d}$*. Let* μ *be the empirical mean function. Assume the clean gradient before loss layer has a bounded*

operator norm, i.e., $\|\mathbf{W}\|_{op} \leq C$*, the maximum clean gradient in loss layer* $\max_{i\in\mathbf{G}} \|\alpha_i\| = k$*, and the maximum corrupted gradient in loss layer* $\max_{i\in\mathbf{N}} \|\delta_i\| = v$*, then we have:*

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| \leq Ck\frac{3\epsilon - 4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon}.$$

Lemma 4 explains the factors affecting the robustness of filtering-based algorithm. Note that v is the only term that is related to the corrupted supervision. If v is large, then the bound is not safe since the right-hand side can be arbitrarily large (i.e. an adversary can change the supervision in such a way that v becomes extremely large). Thus controlling the magnitude of v provides a way to effectively reduce the bound. For example, if we manage to control $v \leq k$, then the bound is safe. This can be achieved by sorting the gradient norms at the loss layer, and then discarding the largest ϵ-fraction data points. Motivated by Lemma 4, we proposed Alg. 4, whose robustness guarantee is given in Thm. 5 and Cor. 2.

Theorem 5 (Robust Gradient Estimation For Supervision Corruption) *Let* $\tilde{\mathbf{G}}$ *be a corrupted matrix generated as in Problem 2,* q *be the output dimension, and* μ *be the empirical mean of the clean gradient matrix* $\mathbf{G}$*. Assuming the maximum clean gradient before loss layer has bounded operator norm:* $\|\mathbf{W}\|_{op} \leq C$*, then the output of gradient estimation in Algorithm 4,* $\hat{\mu}$*, satisfies* $\|\mu - \hat{\mu}\| = O(\epsilon\sqrt{q}) \approx O(\epsilon)$*.*

Thm. 5 can be obtained from Lemma 4 by substituting v by k. The following robustness guarantee can then be obtained by applying Thm. 4.

Corollary 2 (Robust Optimization For Corrupted Supervision Data) *Given the assumptions used in Thm. 4, applying Algorithm. 4 to any* ϵ*-fraction supervision corrupted data, yields* $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\mathbf{x}_t)\|\right) = O(\epsilon\sqrt{q})$ *for large enough* T*, where* q *is the dimension of the supervision.*

Comparing Cor. 1 and Cor. 2, we see that when the corruption only comes from supervision, the dependence on d is reduced to q, where $q \ll d$ in most deep learning problems.

4.3.5 Comparison against Other Robust Optimization Methods

SEVER (Diakonikolas et al., 2019) provides state-of-the-art theoretical results for general corruptions, with a promising $O(\sqrt{\epsilon})$ dimension-free guarantee. Compared to Diakonikolas et al. (2019), we have two contributions: **a)** When corruption comes only from the supervision, we show a better error rate if supervision dimension can be treated as a small constant. **b)** Our algorithm can scale to DNNs while Diakonikolas et al. (2019) cannot. This is especially critical as the DNN based models are currently state-of-the-art methods for noisy label learning problems.

Despite the impressive theoretical results in Diakonikolas et al. (2019), it cannot be applied to DNNs even with the current best hardware configuration. Diakonikolas et al. (2019) used dimension-free robust mean estimation techniques to design the learning algorithm, while most robust mean estimation approaches rely on filtering data by computing the score of projection to the maximum singular vector. For example, the approach in Diakonikolas et al. (2019) requires applying expensive SVD on $n \times d$ individual gradient matrix, where n is the sample size and d is the number of parameters. This method works well for smaller datasets and smaller models when both n and d are small enough for current memory limitation. However, for DNNs, this matrix size is far beyond current GPU memory capability. For example, in our experiment, n is 60,000 and d is in the order of millions (network parameters). It is impractical to store 60,000 copies of networks in a single GPU card. In contrast, our algorithm does not need to store the full gradient matrix. By only considering the loss-layer gradient norm, it can be easily extended to DNNs, and we show that this simple strategy works well in both theory and challenging empirical tasks.

We note that better robustness guarantee can be achieved in linear (Bhatia et al., 2015, 2017) or convex (Prasad et al., 2018) cases, but they cannot be directly applied to DNNs.

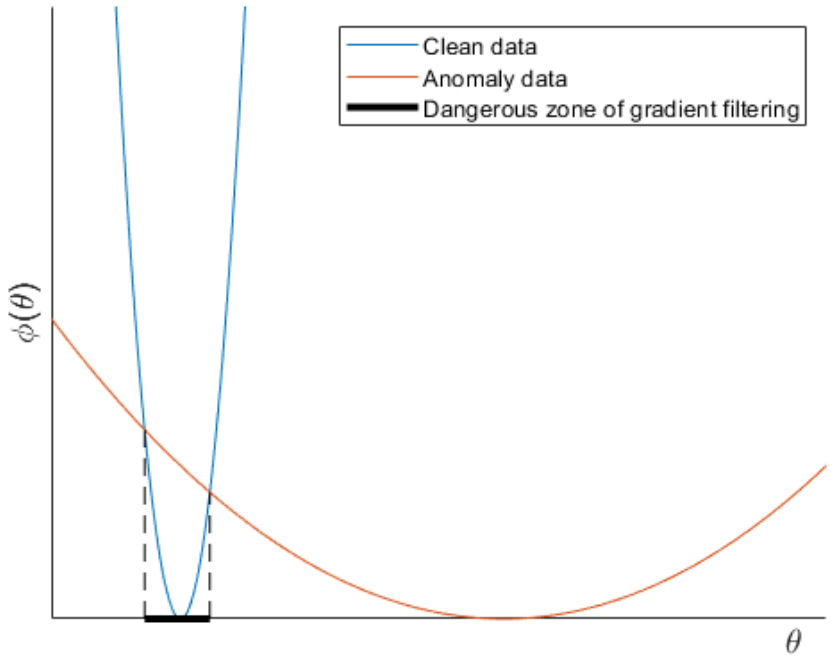
The strongest assumption behind our proof is that the maximum singular value of the gradient before loss layer is bounded, which is similar to the one used in Diakonikolas et al. (2019). We also treat the clean gradient loss layer norm (k in Lemma 4) as a constant, which is particularly true for DNNs due to their overparameterization. In practice, our algorithm slowly increase the dropping ratio τ at first few epochs, which guarantees that k is a small number.

4.3.6 Relationship to Self-Paced Learning (SPL)

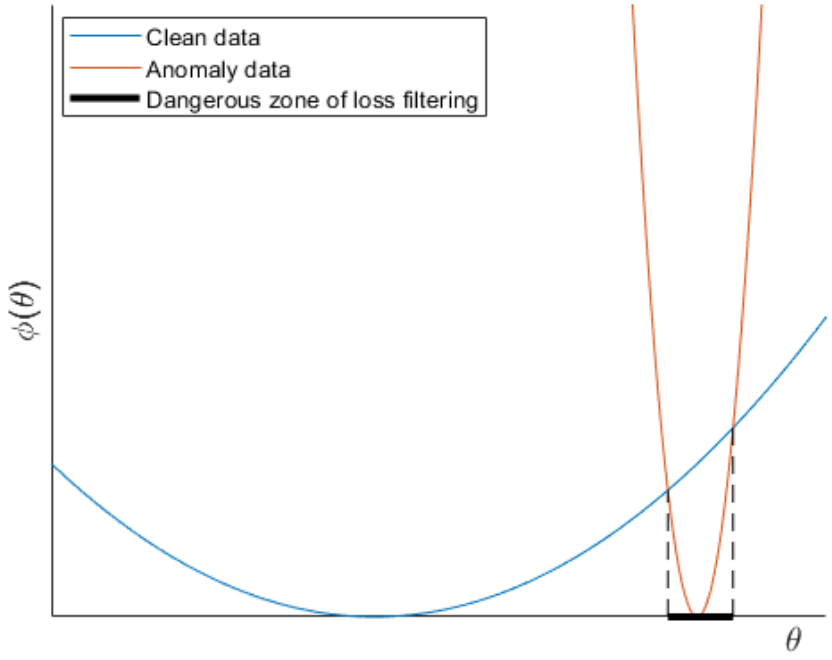
Many state-of-the-art methods with noisy labels depend on the SPL (Han et al., 2018; Song et al., 2019; Yu et al., 2019; Shen and Sanghavi, 2019; Wei et al., 2020; Sun et al., 2020). At first glance, our method looks very similar to SPL. Instead of keeping data points with small gradient norms, SPL tries to keep those points with small loss. The gradient norm and loss function are related via the famous Polyak-Łojasiewicz (PL) condition. The PL condition assumes there exists some constant $s > 0$ such that $\forall \mathbf{x} : \frac{1}{2}\|\nabla\phi(\mathbf{x})\|^2 \geq s\,(\phi(\mathbf{x}) - \phi^*)$. As we can see, when the neural network is highly over-parameterized, ϕ^* can be assumed to be equal across different samples since the neural networks can achieve zero training loss (Zhang et al., 2016). By sorting the error $\phi(\mathbf{x}_i)$ for every data point, SPL is actually sorting the lower bound of the gradient norm if the PL condition holds. However, the ranking of gradient norm and the ranking of the loss can be very different since there is no guarantee that the gradient norm is monotonically increasing with the loss value.

Here we show that the monotonic relationship can be easily violated even for the simple square loss function. One easy counter-example is $\phi(x_1, x_2) = 0.5x_1^2 + 50x_2^2$. Take two points (1000, 1) and (495, -49.5), we will find the monotonic relationship does not hold for these two points. Nocedal et al. (2002) showed that the monotonic relationship holds for ***square loss*** (i.e. $\phi(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\mathbf{Q}(\mathbf{x}-\mathbf{x}^*)$) if the condition number of Q is smaller than $3 + 2\sqrt{2}$, which is quite a strong assumption especially when $\mathbf{x}$ is in high-dimension. If we consider the more general type of loss function (e.g., neural network), the assumptions on condition number should only be stronger, thus breaking the monotonic relationship. Thus, although SPL sorts the lower bound of the gradient norm under mild assumptions, our algorithm is significantly different from SPL and its variations.

We also provide an illustration as to why SPL is not robust from the loss landscape perspective in figure 4.1. In order to have a more intuitive understanding of our algorithm, we could look at the Figure 4.1(b) and 4.1(d). Since we are in the agnostic label corruption setting, it is difficult to filtering out the correct corrupted data. We showed two situations when loss filtering failed and gradient filtering failed. As we could see that when loss filtering method failed, the remaining corrupted data could have large impact on the overall loss surface while when gradient filtering



(b) When gradient filtering method failed to pick out right corrupted data, the remaining corrupted data is relatively smooth, thus has limited impact on overall loss surface.



(d) When loss filtering method failed to pick out right corrupted data, the remaining corrupted data could be extremely sharp, thus has large impact on overall loss surface.

Figure 4.1: Geometric illustration of the difference between loss filtering and gradient norm filtering.

method failed, the remaining corrupted data only have limited impact on the overall loss surface, thus gaining robustness.

Next, we discuss the relationship between SPL and Algorithm 4 under corrupted supervision. SPL has the same form as Algorithm 4 when we are using mean square error to perform regression tasks since the loss layer gradient norm is equal to loss itself. However, in classification, Algorithm 4 is different from the SPL. In order to better understand the algorithm, we further analyze the difference between SPL and our algorithm for cross-entropy loss.

For cross entropy, denote the output logit as $\mathbf{o}$, we have

$$H(\mathbf{y}_i, \mathbf{f}_i) = -\langle \mathbf{y}_i, \log(\text{softmax}(\mathbf{o}_i)) \rangle = -\langle \mathbf{y}_i, \log(\mathbf{f}_i) \rangle.$$

The gradient norm of cross entropy with respect to $\mathbf{o}_i$ is: $\frac{\partial H_i}{\partial \mathbf{o}_i} = \mathbf{y}_i - \text{softmax}(\mathbf{o}_i) = \mathbf{f}_i - \mathbf{y}_i$. Thus, the gradient norm of loss layer is the MSE between $\mathbf{y}_i$ and $\mathbf{f}_i$. Next, we investigate when MSE and cross entropy give non-monotonic relationship. For simplicity, we only consider the sufficient condition for the non-monotonic relationship, which is given by Lemma 5.

Lemma 5 *Let $\mathbf{y} \in \mathbb{R}^q$, where $\mathbf{y}_k = 1$ and $\mathbf{y}_i = 0$ for $i \neq k$. Suppose α and β are two q-dimensional vectors in probability simplex. Without loss of generality, suppose α has a smaller cross entropy loss and $\alpha_k \geq \beta_k$, then the sufficient condition for $\|\alpha - \mathbf{y}\| \geq \|\beta - \mathbf{y}\|$ is $\text{Var}_{i \neq k}(\{\alpha_i\}) - \text{Var}_{i \neq k}(\{\beta_i\}) \geq \frac{q}{(q-1)^2}((\alpha_k - \beta_k)(2 - \alpha_k - \beta_k))$*

As $\alpha_k \geq \beta_k$, the term on right-hand-side of the inequality is non-negative. Thus, when MSE generates a result that differs from cross-entropy, the variance in the probability vector of the non-true class for the discarded data point is larger. For example, consider the ground-truth vector $\mathbf{y} = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$, and two prediction vectors, $\alpha = [0.08, 0.28, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08]$ and $\beta = [0.1, 0.3, 0.34, 0.05, 0.05, 0.1, 0.03, 0.03, 0, 0]$. α has a smaller MSE loss while β has a smaller cross-entropy loss. β will more likely be noisy data since it has two relatively large values of 0.3 and 0.34. Since cross entropy loss considers only one dimension, corresponding to the ground truth label, it cannot detect such a situation. Compared to cross-entropy,

the gradient (mse loss) considers all dimensions, and thus, will consider the distribution of the overall prediction.

4.3.7 Combining with Co-teaching Style Training

Co-teaching (Han et al., 2018) is one of the state-of-the-art deep methods for learning with noisy labels. Motivated by Co-teaching, we propose ***Co-PRL(L)***, which has the same framework as co-teaching but uses the loss-layer gradient to select the data. The key difference between ***Co-PRL(L)*** and algorithm 4 is that in ***Co-PRL(L)***, we optimize two network by ***PRL(L)***. Also in every iteration, two networks will exchange the selected data to update their own parameters. The algorithm is in 5.

Algorithm 5: Co-PRL(L), Collaborative Provable Robust Learning

input: initialize w_f and w_g, learning rate η, fixed τ, epoch T_k and T_{max}, iterations N_{max}

for $T = 1, 2, ..., T_{max}$ **do**

for $N = 1, ..., N_{max}$ **do**

random sample a minibatch $\mathbf{M}$ from $\mathbf{D}_x, \mathbf{D}_y^{\epsilon}$ (noisy dataset)

get the predicted label $\hat{\mathbf{Y}}_f$ and $\hat{\mathbf{Y}}_g$ from $\mathbf{M}$ by w_f. w_g

calculate the individual loss $l_f = \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_f)$, $l_g = \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_g)$

calculate the gradient norm of loss layer $score_f = \|\frac{\partial l_f}{\partial \hat{\mathbf{y}}_f}\|$, $score_g = \|\frac{\partial l_g}{\partial \hat{\mathbf{y}}_g}\|$.

sample $R(T)\%$ small-loss-layer-gradient-norm instances by $score_f$ and $score_g$ to get $\mathbf{N}_f$, $\mathbf{N}_g$

update $w_f = w_f - \eta \nabla_{w_f} \mathcal{L}(\mathbf{N}_f, w_f)$, $w_g = w_g - \eta \nabla_{w_g} \mathcal{L}(\mathbf{N}_g, w_g)$ (selected dataset)

update model $\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \hat{\mu}$

end for

Update $R(T) = 1 - \min\left\{\frac{T}{T_k}\tau, \tau\right\}$

end for

4.4 Experimental Results

We have performed our experiments on various benchmark regression and classification datasets. We compare ***PRL(G)***(Algorithm 3), ***PRL(L)*** (Algorithm 4), and ***Co-PRL(L)*** (Algorithm 5) to the following baselines.

- ***Standard***: standard training without filtering data (mse for regression, cross entropy for classification);

Corruption	Standard	Normclip	Huber	Min-sgd	Ignormclip	PRL(G)	PRL(L)	Co-PRL(L)
linadv: 10	-2.33±0.84	-2.22±0.74	0.868±0.01	0.103±0.03	0.68±0.07	**0.876±0.01**	**0.876±0.01**	**0.876±0.01**
linadv: 20	-8.65±2.1	-8.55±2.2	0.817±0.015	0.120±0.02	0.367±0.28	**0.871±0.01**	0.869±0.01	0.869±0.01
linadv: 30	-18.529±4.04	-19.185±4.31	0.592±0.07	0.146±0.03	-0.944±0.51	**0.865±0.01**	0.861±0.01	0.860±0.01
linadv: 40	-32.22±6.32	-32.75±7.07	-2.529±1.22	0.180±0.01	-1.60 ± 0.80	**0.857± 0.01**	0.847±0.02	0.847±0.02
signflip: 10	0.800±0.02	0.798±0.03	0.857±0.01	0.110±0.04	0.846±0.01	0.877±0.01	0.878±0.01	**0.879±0.01**
signflip: 20	0.641±0.05	0.638±0.04	0.786±0.02	0.105±0.07	0.82±0.02	0.875±0.01	0.875±0.01	**0.877±0.01**
signflip: 30	0.422±0.04	0.421±0.04	0.629±0.03	0.124±0.05	0.795±0.02	0.871±0.01	0.873±0.01	**0.875±0.01**
signflip: 40	0.193±0.043	0.190±0.04	0.379±0.05	-0.028±0.25	0.759±0.01	**0.872±0.01**	**0.872±0.01**	0.871±0.01
uninoise: 10	0.845±0.01	0.844±0.01	0.875±0.01	0.103±0.03	0.859±0.01	0.879±0.01	**0.881±0.01**	**0.881±0.01**
uninoise: 20	0.798±0.02	0.795±0.02	0.865±0.01	0.120±0.02	0.844±0.01	0.878±0.01	**0.880±0.01**	**0.880±0.01**
uninoise: 30	0.728±0.02	0.725±0.02	0.847±0.01	0.146±0.03	0.831±0.01	0.878±0.01	**0.879±0.01**	**0.879±0.01**
uninoise: 40	0.656±0.02	0.654±0.02	0.825±0.01	0.180±0.01	0.821±0.01	0.876± 0.01	**0.878±0.01**	**0.878±0.01**
pairflip: 10	0.852±0.02	0.851±0.02	0.870±0.01	0.110±0.04	0.867±0.01	0.877±0.01	0.876±0.01	**0.878±0.01**
pairflip: 20	0.784±0.03	0.783±0.03	0.841±0.02	0.120±0.03	0.849±0.01	**0.874±0.01**	0.873±0.01	**0.874±0.01**
pairflip: 30	0.688±0.04	0.686±0.04	0.770±0.02	0.133±0.02	0.828±0.01	0.870±0.01	0.872±0.01	**0.873±0.01**
pairflip: 40	0.556±0.06	0.553±0.06	0.642±0.06	0.134±0.03	0.810±0.02	0.863±0.01	**0.870±0.01**	**0.870±0.01**
mixture: 10	-0.212±0.6	-0.010±0.48	0.873±0.01	0.101±0.03	0.861±0.01	0.878±0.01	**0.880±0.01**	**0.880±0.01**
mixture: 20	-0.404±0.68	-0.463±0.67	0.855±0.01	0.119±0.03	0.855±0.01	0.877±0.01	0.878±0.01	**0.879±0.01**
mixture: 30	-0.716±0.57	-0.824±0.39	0.823±0.01	0.148±0.02	0.847±0.01	0.875±0.01	0.877±0.01	**0.878±0.01**
mixture: 40	-3.130±1.51	-2.69±0.84	0.763±0.01	0.175±0.02	0.835±0.01	0.872±0.01	0.875 ±0.01	**0.876±0.01**

Table 4.1: R-square on CelebA clean testing data, and the standard deviation is from last ten epochs and 5 random seeds.

Corruption	Standard	Normclip	Bootstrap	Decouple	Min-sgd	SPL	PRL(L)	Co-teaching	Co-PRL(L)
CF10-sym-30	63.22±0.18	62.41±0.06	63.67±0.24	70.73±0.51	13.31±2.24	77.77±0.34	79.40±0.19	79.90±0.13	**80.05±0.12**
CF10-sym-50	44.63±0.18	43.99±0.28	46.13±0.18	57.48±1.98	13.33±2.85	72.22±0.15	74.17±0.15	74.25±0.41	**75.43±0.09**
CF10-sym-70	24.12±0.09	24.17±0.37	25.13±0.39	40.11±4.62	9.08±0.94	56.19±0.33	58.36±0.62	58.41±0.33	**60.26±0.42**
CF10-pf-25	68.34±0.30	67.92±0.43	68.71±0.32	75.59±0.35	10.45±0.60	75.79±0.44	80.54±0.07	80.18±0.21	**81.51±0.13**
CF10-pf-35	58.68±0.28	58.27±0.18	58.19±0.12	66.38±0.44	12.29±1.92	70.40±0.27	77.61±0.35	77.97±0.03	**79.01±0.14**
CF10-pf-45	48.05±0.25	48.03±0.54	47.84±0.32	51.54±0.81	10.94±1.28	58.95±0.59	71.42±0.24	72.43±0.31	**73.78±0.17**
CF100-sym-30	32.83±0.39	32.10±0.64	34.47±0.22	32.95±0.44	2.94±0.61	44.37±0.44	46.40±0.18	45.02±0.29	**47.51±0.47**
CF100-sym-50	20.47±0.44	19.73±0.29	21.59±0.44	21.02±0.36	2.35±0.45	37.89±0.16	38.38±0.65	38.79±0.33	**40.64±0.11**
CF100-sym-70	9.93±0.07	9.93±0.23	10.59±0.17	12.55±0.46	2.32±0.24	24.10±0.44	25.38±0.56	24.94±0.53	**27.27±0.01**
CF100-pf-25	40.37±0.55	39.34±0.35	40.22±0.37	39.43±0.27	2.62±0.26	40.48±0.72	47.57±0.37	42.97±0.10	**48.06±0.26**
CF100-pf-35	34.07±0.19	32.88±0.10	34.53±0.23	33.14±0.07	2.30±0.07	34.17±0.46	43.32±0.16	36.69±0.23	**44.08±0.33**
CF100-pf-45	27.66±0.50	27.35±0.61	27.56±0.23	26.83±0.41	2.55±0.52	27.55±0.66	33.31±0.10	29.71±0.20	**34.43±0.05**

Table 4.2: Classification accuracy for clean testing data on CIFAR10 and CIFAR100 with training on ***symmetric*** and ***pairflip*** label corruption. The standard deviation is from last ten epochs and 3 random seeds.

- ***Normclip***: standard training with norm clipping; ***Huber***: standard training with huber loss (regression only);
- ***Decouple***: decoupling network, update two networks by using their disagreement (Malach and Shalev-Shwartz, 2017) (classification only);
- ***Bootstrap***: It uses a weighted combination of predicted and original labels as the correct labels, and then perform back propagation (Reed et al., 2014) (classification only);
- ***Min-sgd***: choosing the smallest loss sample in minibatch to update model (Shah et al., 2020);

- ***SPL*** Jiang et al. (2018): self-paced learning (also known as the trimmed loss), dropping the data with large losses (same as ***PRL(L)*** in regression setting with MSE loss);
- ***Ignormclip***: clipping individual gradient then average them to update model (regression only);
- ***Co-teaching***: collaboratively train a pair of SPL model and exchange selected data to another model (Han et al., 2018) (classification only).

Since it is hard to design experiments for ***agnostic corrupted supervision***, we analyzed the performance on a broad class of corrupted supervision settings:

- ***linadv***: the corrupted supervision is generated by random wrong linear relationship of features: $\mathbf{Y}_\epsilon = \mathbf{X} * \mathbf{W}_\epsilon$ (regression);
- ***signflip***: the supervision sign is flipped $\mathbf{Y}_\epsilon = -\mathbf{Y}$ (regression);
- ***uninoise***: random sampling from uniform distribution as corrupted supervision $\mathbf{Y}_\epsilon \sim [-5, 5]$ (regression);
- ***mixture***: mixture of above types of corruptions (regression);
- ***pairflip***: shuffle the coordinates (i.e. eyes to mouth in CelebA or cat to dog in CIFAR) (regression and classification);
- ***symmetric***: randomly assign wrong class label (classification).

For classification, we use accuracy as the evaluation metric, and R-square is used to evaluate regression experiments. We show the average evaluation score on testing data for the last 10 epochs. We also include the training curves to show how the testing evaluation metric changes during training phase. All experiments are repeated 5 times for regression experiments and 3 times for classification experiments. Main hyperparameters are showed in the Table 4.3. For Classification, we use the same hyperparameters in Han et al. (2018). For CelebA, we use 3-layer fully connected network with 256 hidden nodes in hidden layer and leakly-relu as activation function.

Data\HyperParameter	BatchSize	Learning Rate	Optimizer	Momentum
CF-10	128	0.001	Adam	0.9
CF-100	128	0.001	Adam	0.9
CelebA	512	0.0003	Adam	0.9

Table 4.3: Main Hyperparmeters

Data	$\epsilon - 0.1$	$\epsilon - 0.05$	ϵ	$\epsilon + 0.05$	$\epsilon + 0.1$
CF10-Pair-45%	65.07±0.83	70.07±0.67	73.78±0.17	77.56±0.55	79.36±0.43
CF10-Sym-50%	69.21±0.35	72.53±0.45	75.43 ± 0.09	77.65±0.27	78.10±0.31
CF10-Sym-70%	53.88±0.64	58.49±0.97	60.26 ± 0.42	60.89±0.43	54.91±0.68
CF100-Pair-45%	32.60±0.45	34.17±0.40	34.43 ± 0.05	36.87±0.41	38.34±0.78
CF100-Sym-50%	37.74±0.41	39.72±0.36	40.64 ± 0.11	43.02±0.36	43.92±0.61
CF100-Sym-70%	24.40±0.47	25.50±0.45	27.27 ± 0.10	27.80±0.50	28.20±0.97

Table 4.4: Sensitivity analysis for over-estimated/under-estimated ϵ.

4.4.1 Regression Results

For regression, we evaluated our method on the CelebA dataset, which contains 162,770 training images, 19,867 validation images, and 19,962 test images. Given a human face image, the goal is to predict the coordinates for 10 landmarks in the face image. Specifically, the target variable is a ten-dimensional vector of coordinates for the left eye, right eye, nose, left mouth, and right mouth. We added different types of corruption to the landmark coordinates. The CelebA dataset is preprocessed as follows: we use a three-layer CNN to train 162770 training images to predict clean coordinates (we use 19867 validation images to do the early stopping). We then apply the network to extract a 512-dimensional feature vector from the testing data. Thus, the final dataset after preprocessing consists of the feature sets $\mathbf{X} \in \mathbb{R}^{19962\times512}$ and the target variable $\mathbf{Y} \in \mathbb{R}^{19962\times10}$. We further split the data to the training and testing set, where training sets contain 80% of the data. We then manually add the ***linadv***, ***signflip***, ***uninoise***, ***pairflip***, and ***mixture*** corruptions to the target variable in the training set. The corruption rate for all types of corruptions is varied from 0.1 to 0.4. We use a 3-layer fully connected networks for our experiments. The results of averaged r-square for the last 10 epochs are shown in Table 4.1. The training curves could be found in the figure 4.2. Surprisingly, the performance of PRL(G) is comparable to PRL(L). This is partially due to the network structure and the initialization. Another possible reason is that for this task, the gradient

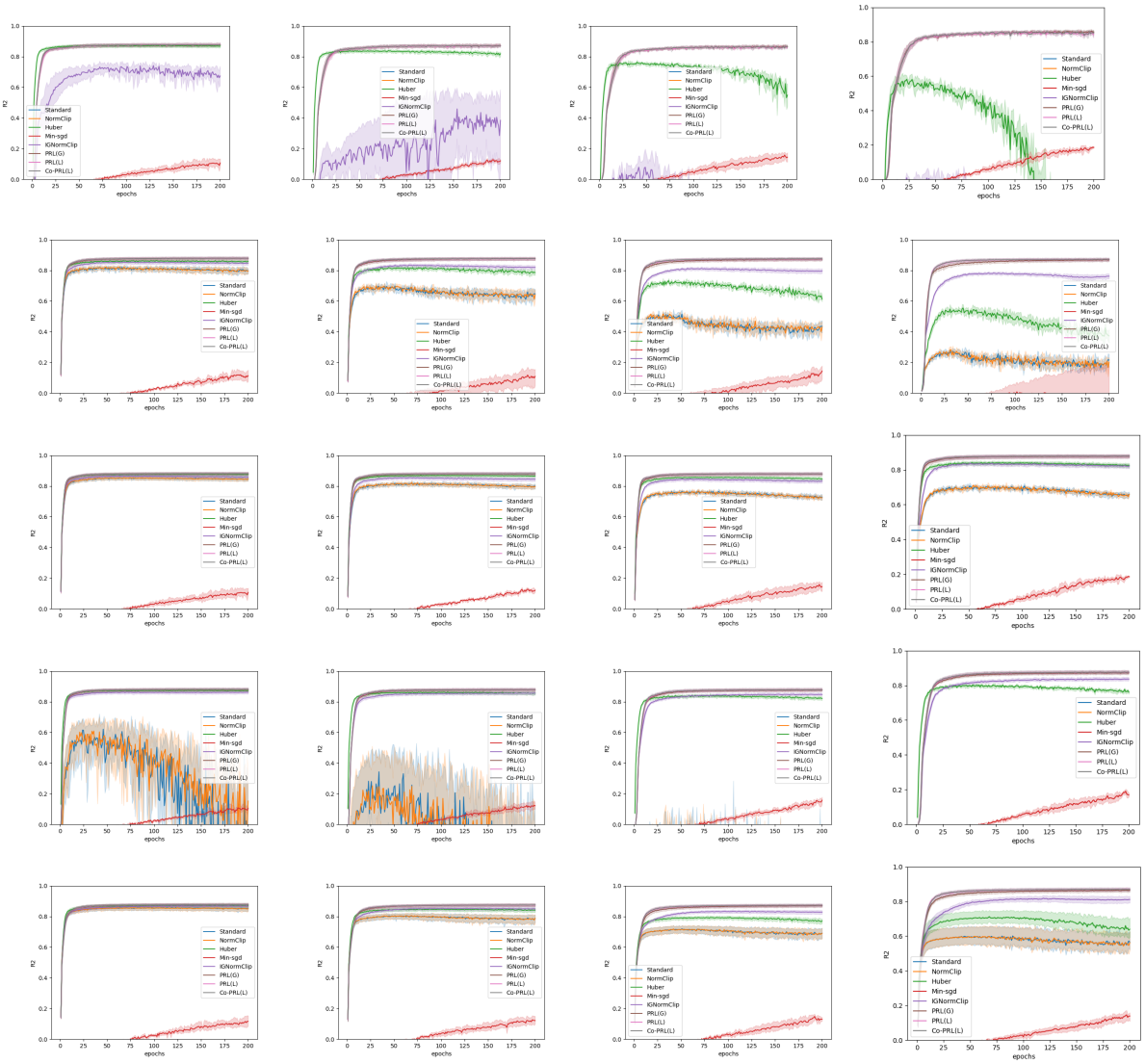


Figure 4.2: CelebA Testing Curve During Training. The corruption ratios are [0.1, 0.2, 0.3, 0.4] from left to right. The corruption types are [linadv, signflip, uninoise, mixture, pairflip] from up to bottom. X axis represents the epoch number, Y axis represents the testing r-square. In some experiment, there is no curve for Standard and NormClip since they gave negative r-square, which will effect the plotting scale. The shadow represents the confidence interval, which is calculated across 5 random seed. As we see, PRL(G), PRL(L), and Co-PRL(L) are robust against different types of corruptions.
norm is upper bounded by a small constant.

4.4.2 Classification Results

We perform our experiments on the CIFAR10 and CIFAR100 datasets to illustrate the effectiveness of our algorithm in classification setting. We use a 9-layer Convolutional Neural Network, similar to

the approach in Han et al. (2018). Since most baselines include batch normalization, it is difficult to get individual gradient efficiently, we exclude the ignormclip and PRL baselines. In the appendix, we attached the results if both co-teaching and Co-PRL(L) excludes the batch normalization module. Our results suggest that co-teaching cannot maintain robustness unlike our proposed method. The reason is discussed in the appendix. We consider ***pairflip*** and ***symmetric*** supervision corruptions in our experiments. Also, to compare with the current state of the art method, for ***symmetric*** noise, we use corruption rate beyond 0.5. Although our theoretical analysis assumes the noise rate is smaller than 0.5, when the noise type is not an adversary (i.e. symmetric), we empirically show that our method can also deal with such type of noise. The results for CIFAR10 and CIFAR100 are shown in Table 4.2. The results suggest that our method performs significantly better than the baselines irrespective of whether we are using one network (PRL vs SPL) or two networks (Co-PRL(L) vs Co-teaching). The training curves could be found in the figure 4.3.

4.4.3 Case Study on Limnology Dataset

We also performed a case study on Limnology Dataset. In this experiment, we use the variables from the LAGOSNE database (Soranno et al. (2017)) to predict the total phosphorus (TP) and total nitrogen (TN) in lakes. The predictor we used include chlorophyll, secchi, NO2NO3, etc. There are 10470 samples in the dataset, and we use 33% percent of the data as the testing data.

To study whether the PRL can defend against the noisy supervision in Limno data, we introduced two types of corruption to modifying the TN and TP value.

- **linadv:** the corrupted supervision is generated by random wrong linear relationship of features: $\mathbf{Y}_\epsilon = \mathbf{X} * \mathbf{W}_\epsilon$;
- **maximum:** the corrupted supervision is generated by assigning the maximum value of clean supervision;

We modifying the training data supervision by using above two corruption method with different corruption ratio while leave the testing data as untouched. For TN, we use a log transform on it

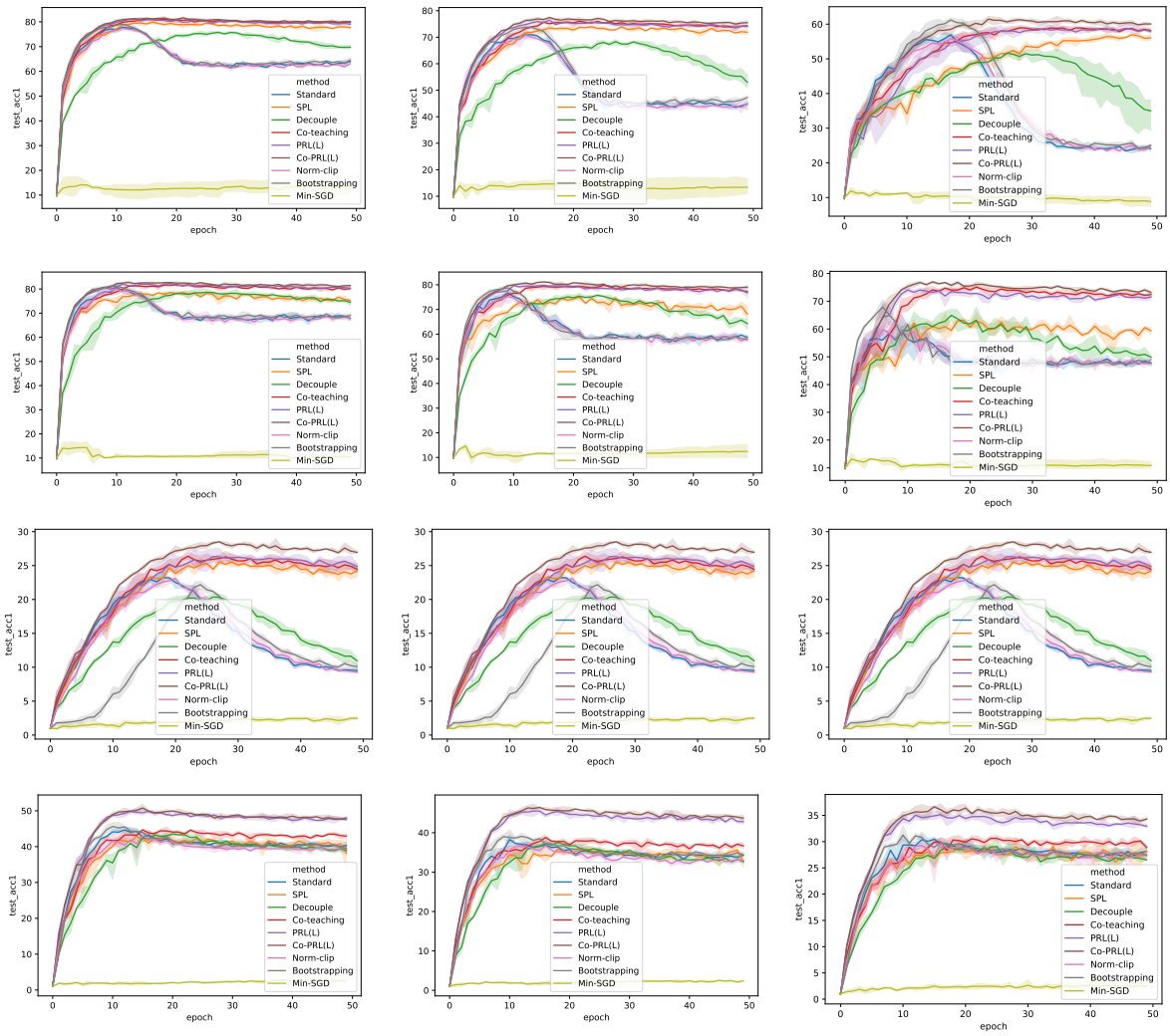


Figure 4.3: CIFAR10 and CIFAR100 Testing Curve During Training. X axis represents the epoch number, Y axis represents the testing accuracy. The shadow represents the confidence interval, which is calculated across 3 random seed. The first two rows are results for CIFAR10 while the last two rows are results for CIFAR100. For each dataset, the first row represents the symmetric noise while the second row represents the pairflip noise. The corruption ratios for symmetric noise are [0.3, 0.5, 0.7] from left to right while the corruption ratios for pairflip noise are [0.25, 0.35, 0.45] from left to right. As we see, PRL(L), and Co-PRL(L) are robust against different types of corruptions.

R-square	Corruption Ratio	maximum corruption		linear corruption	
		PRL	Standard	PRL	Standard
TP	0.05	0.31	-1.03	0.47	0.17
	0.15	0.36	-8.92	0.33	-0.16
	0.25	0.35	-23.55	0.31	-0.28
TN	0.05	0.37	0.15	0.31	0.15
	0.15	0.36	-0.98	0.15	-0.31
	0.25	0.32	-3.60	0.05	-1.07

Table 4.5: Prediction Results for Limonology Dataset. The numbers are averaged R-square across 3 random seeds. As we can see, standard training deep neural network cannot defend against both type of corruption while PRL showes robustness for such attack.

(i.e. $TN_{processed} = \log(TN + 1e - 3)$) as preprocessing. We compare the PRL and the standard training approach. The model we used for both methods are 3-layer neural networks with ReLU activation function. For each setting, we repeated the experiment for 3 random seeds, and we report the averaged R-square on testing set to compare the performance. The results are in the table 4.5.

4.4.4 Sensitivity Analysis

Since in real-world problems, it is hard to know that the ground-truth corruption rate, we also perform the sensitivity analysis in classification tasks to show the effect of overestimating and underestimating ϵ. The results are in Table 4.4. As we could see, the performance is stable if we overestimate the corruption rate, this is because only when we overestimate the ϵ, we could guarantee that the gradient norm of the remaining set is small. However, when we underestimate the corruption rate, in the worst case, there is no guarantee that the gradient norm of the remaining set is small. By using the empirical mean, even one large bad individual gradient would ruin the gradient estimation, and according to the convergence analysis of biased gradient descent, the final solution could be very bad in terms of clean data. That explains why to underestimate the corruption rate gives bad results. Also, from Table 4.4, we could see that using the ground truth corruption rate will lead to small uncertainty.

Method	Standard (Lower Bound)	PRL(G)	PRL(L)
CF10-Pair-45%	37.03s	145.55s	54.80s

Table 4.6: Execution Time of Single Epoch in CIFAR-10 Data

4.5 Conclusion

In this chapter, we proposed a simple yet effective algorithm to defend against agnostic supervision corruptions. Both the theoretical and empirical analysis showed the effectiveness of our algorithm. For future research, there are two questions that deserved further study. The first question is whether we can further improve $O(\epsilon)$ error bound or show that $O(\epsilon)$ is tight. The second question is how we can utilize more properties of neural networks, such as the sparse or low-rank structure in gradient to design better algorithms.

4.6 Supplementary Empirical Results on Running Time

As we claimed in previous section, the algorithm 2 (PRL(G)) is not efficient. In here we attached the execution time for one epoch for three different methods: ***Standard***, ***PRL(G)***, ***PRL(L)***. For fair comparison, we replace all batch normalization module to group normalization for this comparison, since it is hard to calculate individual gradient when using batch normalization. For PRL(G), we use opacus libarary (https://opacus.ai/) to calculate the individual gradient.

The results are showed in Table 4.6

4.7 Proofs of Theorem

4.7.1 Proof of Theorem 4

Since this is a standard results, similar results are showed in Bernstein et al. (2018); Devolder et al. (2014); Hu et al. (2020); Ajalloeian and Stich (2020). For the sake of completeness, we provide the proof sketch here. Details could be found on above literature

Proof: by L-smooth, we have:

$$\phi(\theta_2) \leq \phi(\theta_1) + \langle \nabla\phi(\theta_1), \theta_2 - \theta_1 \rangle + \frac{L}{2}\|\theta_2 - \theta_1\|^2$$

by using $\gamma \leq \frac{1}{L}$, we have

$$\begin{aligned}
\mathbb{E}\phi\left(\theta_{\mathbf{1}_{t+1}}\right) &\leq \phi\left(\theta_{\mathbf{1}_t}\right) - \gamma\left\langle\nabla\phi\left(\theta_{\mathbf{1}_t}\right), \mathbb{E}\mathbf{g}_t\right\rangle + \frac{\gamma^2 L}{2}\left(\mathbb{E}\left\|\mathbf{g}_t - \mathbb{E}\mathbf{g}_t\right\|^2 + \mathbb{E}\left\|\mathbb{E}\mathbf{g}_t\right\|^2\right) \\
&= \phi\left(\theta_{\mathbf{1}_t}\right) - \gamma\left\langle\nabla\phi\left(\theta_{\mathbf{1}_t}\right), \nabla\phi\left(\theta_{\mathbf{1}_t}\right) + \mathbf{b}_t\right\rangle + \frac{\gamma^2 L}{2}\left(\mathbb{E}\left\|\mathbf{n}_t\right\|^2 + \mathbb{E}\left\|\nabla\phi\left(\theta_{\mathbf{1}_t}\right) + \mathbf{b}_t\right\|^2\right) \\
&\leq \phi\left(\theta_{\mathbf{1}_t}\right) + \frac{\gamma}{2}\left(-2\left\langle\nabla\phi\left(\theta_{\mathbf{1}_t}\right), \nabla\phi\left(\theta_{\mathbf{1}_t}\right) + \mathbf{b}_t\right\rangle + \left\|\nabla\phi\left(\theta_{\mathbf{1}_t}\right) + \mathbf{b}_t\right\|^2\right) + \frac{\gamma^2 L}{2}\mathbb{E}\left\|\mathbf{n}_t\right\|^2 \\
&= \phi\left(\theta_{\mathbf{1}_t}\right) + \frac{\gamma}{2}\left(-\left\|\nabla\phi\left(\theta_{\mathbf{1}_t}\right)\right\|^2 + \left\|\mathbf{b}_t\right\|^2\right) + \frac{\gamma^2 L}{2}\mathbb{E}\left\|\mathbf{n}_t\right\|^2
\end{aligned}$$

According to the assumption, we have $\|\mathbf{b}_t\|^2 \leq \zeta^2$, $\|\mathbf{n}_t\|^2 \leq \sigma^2$, by plug in the learning rate constraint, we have

$$\begin{aligned}
\mathbb{E}\phi\left(\theta_{\mathbf{1}_{t+1}}\right) &\leq \phi\left(\theta_{\mathbf{1}_t}\right) - \frac{\gamma}{2}\left\|\nabla\phi\left(\theta_{\mathbf{1}_t}\right)\right\|^2 + \frac{\gamma}{2}\zeta^2 + \frac{\gamma^2 L}{2}\sigma^2 \\
\mathbb{E}\phi\left(\theta_{\mathbf{1}_{t+1}}\right) - \phi\left(\theta_{\mathbf{1}_t}\right) &\leq -\frac{\gamma}{2}\left\|\nabla\phi\left(\theta_{\mathbf{1}_t}\right)\right\|^2 + \frac{\gamma}{2}\zeta^2 + \frac{\gamma^2 L}{2}\sigma^2
\end{aligned}$$

Then, removing the gradient norm to left hand side, and sum it across different iterations, we could get

$$\frac{1}{2T}\sum_{t=0}^{T-1}\mathbb{E}\|\phi\left(\theta_{\mathbf{1}_t}\right)\| \leq \frac{F}{T\gamma} + \frac{\zeta^2}{2} + \frac{\gamma L\sigma^2}{2}$$

Take the minimum respect to t and substitute the learning rate condition will directly get the results.

4.7.2 Proof of Corollary 1

We first prove the gradient estimation error.

Denote $\tilde{\mathbf{G}}$ to be the set of corrupted mini-batch, $\mathbf{G}$ to be the set of original clean mini-batch and we have $|\mathbf{G}| = |\tilde{\mathbf{G}}| = m$. Let $\mathbf{N}$ to be the set of remaining data and according to our algorithm, the remaining data has the size $|\mathbf{N}| = n = (1-\epsilon)m$. Define $\mathbf{A}$ to be the set of individual clean gradient, which is not discarded by algorithm 1. $\mathbf{B}$ to be the set of individual corrupted gradient, which is not discarded. According to our definition, we have $\mathbf{N} = \mathbf{A} \cup \mathbf{B}$. $\mathbf{AD}$ to be the set of individual good

gradient, which is discarded, **AR** to be the set of individual good gradient, which is replaced by corrupted data. We have $\mathbf{G} = \mathbf{A} \cup \mathbf{AD} \cup \mathbf{AR}$. **BD** is the set of individual corrupted gradient, which is discarded by our algorithm. Denote the good gradient to be $\mathbf{g}_i = \alpha_i \mathbf{W}_i$, and the bad gradient to be $\tilde{\mathbf{g}}_i$, according to our assumption, we have $\|\tilde{\mathbf{g}}_i\| \leq L$.

Now, we have the l2 norm error:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &= \|\frac{1}{m}\sum_{i\in\mathbf{G}}^{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i=1}^{m}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i\| + \|\frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\sum_{\mathbf{A}}\frac{m-n}{nm}\mathbf{g}_i + \sum_{\mathbf{AD}}\frac{1}{m}\mathbf{g}_i + \sum_{\mathbf{AR}}\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\| \\
&\leq \sum_{\mathbf{A}}\|\frac{m-n}{nm}\mathbf{g}_i\| + \sum_{\mathbf{AD}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{AR}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\|
\end{aligned}$$

By using the filtering algorithm, we could guarantee that $\|\tilde{\mathbf{g}}_i\| \leq L$. Let $|\mathbf{A}| = x$, we have $|\mathbf{B}| = n - x = (1-\epsilon)m - x$, $|\mathbf{AR}| = m - n = \epsilon m$, $|\mathbf{AD}| = m - |\mathbf{A}| - |\mathbf{AR}| = m - x - (m-n) = n - x = (1-\epsilon)m - x$. Thus, we have:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq x\frac{m-n}{nm}L + (n-x)\frac{1}{m}L + (m-n)\frac{1}{m}L + (n-x)\frac{1}{n}L \\
&\leq x(\frac{m-n}{nm} - \frac{1}{m})L + n\frac{1}{m}L + (m-n)\frac{1}{m}L + (n-x)\frac{1}{n}L \\
&= \frac{1}{m}(\frac{2\epsilon-1}{1-\epsilon})xL + L + L - \frac{1}{n}xL \\
&= xL(\frac{2\epsilon-2}{n}) + 2L
\end{aligned}$$

To minimize the upper bound, we need x to be as small as possible since $2\epsilon - 2 < 1$. According to

our problem setting, we have $x = n - m\epsilon \leq (1 - 2\epsilon)m$, substitute back we have:

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq (1 - 2\epsilon)Lm(\frac{2\epsilon - 2}{n}) + 2L \\ &= \frac{1 - 2\epsilon}{1 - \epsilon}2L + 2L \\ &= 4L - \frac{\epsilon}{1 - \epsilon}2L\end{aligned}$$

Since $\epsilon < 0.5$, we use tylor expansion on $\frac{\epsilon}{1 - \epsilon}$, by ignoring the high-order terms, we have

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| = O(\epsilon L)$$

Note, if the Lipschitz continuous assumption does not hold, then L should be dimension dependent (i.e. $\sqrt{d}$).

Combining above gradient estimation error upper bound and Theorem 1, we could get the results in Corollary 1.

4.7.3 Proof of Lemma 4

Denote $\tilde{\mathbf{G}}$ to be the set of corrupted mini-batch, $\mathbf{G}$ to be the set of original clean mini-batch and we have $|\mathbf{G}| = |\tilde{\mathbf{G}}| = m$. Let $\mathbf{N}$ to be the set of remaining data and according to our algorithm, the remaining data has the size $|\mathbf{N}| = n = (1 - \epsilon)m$. Define $\mathbf{A}$ to be the set of individual clean gradient, which is not discarded by any filtering algorithm. $\mathbf{B}$ to be the set of individual corrupted gradient, which is not discarded. According to our definition, we have $\mathbf{N} = \mathbf{A} \cup \mathbf{B}$. $\mathbf{AD}$ to be the set of individual good gradient, which is discarded, $\mathbf{AR}$ to be the set of individual good gradient, which is replaced by corrupted data. We have $\mathbf{G} = \mathbf{A} \cup \mathbf{AD} \cup \mathbf{AR}$. $\mathbf{BD}$ is the set of individual corrupted gradient, which is discarded by our algorithm. Denote the good gradient to be $\mathbf{g}_i = \alpha_i \mathbf{W}_i$, and the bad gradient to be $\tilde{\mathbf{g}}_i = \delta_i \mathbf{W}_i$, according to our assumption, we have $\|\mathbf{W}_i\|_{op} \leq C$.

Now, we have the l2 norm error:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &= \|\frac{1}{m}\sum_{i\in\mathbf{G}}^{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i=1}^{m}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i\| + \|\frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \qquad (4.1)
\end{aligned}$$

Let $|\mathbf{A}| = x$, we have $|\mathbf{B}| = n - x = (1-\epsilon)m - x$, $|\mathbf{AR}| = m - n = \epsilon m$, $|\mathbf{AD}| = m - |\mathbf{A}| - |\mathbf{AR}| = m - x - (m-n) = n - x = (1-\epsilon)m - x$. Thus, we have:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq \|\sum_{\mathbf{A}}\frac{m-n}{nm}\mathbf{g}_i + \sum_{\mathbf{AD}}\frac{1}{m}\mathbf{g}_i + \sum_{\mathbf{AR}}\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\| \\
&\leq \sum_{\mathbf{A}}\|\frac{m-n}{nm}\mathbf{g}_i\| + \sum_{\mathbf{AD}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{AR}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\|
\end{aligned}$$

For individual gradient, according to the label corruption gradient definition in problem 2, assuming the $\|\mathbf{W}\|_{op} \leq C$, we have $\|\mathbf{g}_i\| \leq \|\alpha_i\|\|\mathbf{W}_i\|_{op} \leq C\|\alpha_i\|$. Also, denote $\max_i \|\alpha_i\| = k$, $\max_i \|\delta_i\| = v$, we have $\|\mathbf{g}_i\| \leq Ck$, $\|\tilde{\mathbf{g}}_i\| \leq Cv$.

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| \leq Cx\frac{m-n}{nm}k + C(n-x)\frac{1}{m}k + C(m-n)\frac{1}{m}k + C(n-x)\frac{1}{n}v$$

Note the above upper bound holds for any x, thus, we would like to get the minimum of the upper bound respect to x. Rearrange the term, we have

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Cx(\frac{m-n}{nm} - \frac{1}{m})k + Cn\frac{1}{m}k + C(m-n)\frac{1}{m}k + C(n-x)\frac{1}{n}v \\
&= C\frac{1}{m}(\frac{2\epsilon-1}{1-\epsilon})xk + Ck + Cv - \frac{1}{n}Cxv \\
&= Cx\left(\frac{k(2\epsilon-1)}{m(1-\epsilon)} - \frac{v}{n}\right) + Ck + Cv \\
&= Cx\left(\frac{k(2\epsilon-1)-v}{m(1-\epsilon)}\right) + Ck + Cv
\end{aligned}$$

Since when $\epsilon < 0.5$, $\frac{k(2\epsilon-1)-v}{m(1-\epsilon)} < 0$, we knew that x should be as small as possible to continue the bound. According to our algorithm, we knew $n-m\epsilon = m(1-\epsilon)-m\epsilon = (1-2\epsilon)m \leq x \leq n = (1-\epsilon)m$. Then, substitute $x = (1-2\epsilon)m$, we have

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Ck(1-2\epsilon)\frac{2\epsilon-1}{1-\epsilon} + Ck + Cv - Cv\frac{1-2\epsilon}{1-\epsilon} \\ &= Ck\frac{3\epsilon-4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon}\end{aligned}$$

4.7.4 Proof of Theorem 5

According to algorithm2, we could guarantee that $v \leq k$. By lemma 1, we will have:

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Ck\frac{3\epsilon-4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon} \\ &\leq Ck\frac{4\epsilon-4\epsilon^2}{1-\epsilon} \\ &= 4\epsilon Ck \\ &\approx O(\epsilon\sqrt{q})(\text{C is constant, k is the norm of } q\text{-dimensional vector})\end{aligned}$$

4.7.5 Proof of Lemma 2

Assume we have a d class label $\mathbf{y} \in \mathcal{R}^d$, where $y_k = 1, y_i = 0, i \neq k$. We have two prediction $\mathbf{p} \in \mathcal{R}^d$, $\mathbf{q} \in \mathcal{R}^d$.

Assume we have a d class label $\mathbf{y} \in \mathbb{R}^d$, where $y_k = 1, y_i = 0, i \neq k$. With little abuse of notation, suppose we have two prediction $\mathbf{p} \in \mathbb{R}^d$, $\mathbf{q} \in \mathbb{R}^d$. Without loss of generality, we could assume that $\mathbf{p}_1$ has smaller cross entropy loss, which indicates $\mathbf{p}_k \geq \mathbf{q}_k$

For MSE, assume we have opposite result

$$\begin{aligned}&\|\mathbf{p}-\mathbf{y}\|^2 \geq \|\mathbf{q}-\mathbf{y}\|^2 \\ \Rightarrow &\sum_{i\neq k} p_i^2 + (1-p_k)^2 \geq \sum_{i\neq k} q_i^2 + (1-q_k)^2\end{aligned} \tag{4.2}$$

For each $p_i, i \neq k$, We have

$$Var(p_i) = E(p_i^2) - E(p_i)^2 = \frac{1}{d-1}\sum_{i\neq k} p_i^2 - \frac{1}{(d-1)^2}(1-p_k)^2 \tag{4.3}$$

Then

$$
\begin{aligned}
&\sum_{i\neq k} p_i^2 + (1-p_k)^2 \geq \sum_{i\neq k} q_i^2 + (1-q_k)^2 \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) + \frac{d}{(d-1)^2}(1-p_k)^2 \geq Var_{i\neq k}(\mathbf{q}_i) + \frac{d}{(d-1)^2}(1-q_k)^2 \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) - Var_{i\neq k}(\mathbf{q}_i) \geq \frac{d}{(d-1)^2}\left((1-q_k)^2 - (1-p_k)^2\right) \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) - Var_{i\neq k}(\mathbf{q}_i) \geq \frac{d}{(d-1)^2}\left((p_k-q_k)(2-p_k-q_k)\right)
\end{aligned}
\tag{4.4}
$$

CHAPTER 5

DEFENDING BACKDOOR ATTACKS VIA ROBUSTNESS AGAINST NOISY LABEL

In this chapter, we introduce a robust meta-algorithm defending against backdoor attacks by exploring the connection between noisy label attacks and backdoor data poisoning attacks.

5.1 Introduction

Deep neural networks (DNN) have achieved significant success in a variety of applications such as image classification (Krizhevsky et al., 2012), autonomous driving (Major et al., 2019), and natural language processing (Devlin et al., 2018), due to their powerful generalization ability. However, DNN can be highly susceptible to even small perturbations of training data, which has raised considerable concerns about their trustworthiness (Liu et al., 2020). One representative perturbation approach is backdoor attack, which undermines the DNN performance by modifying a small fraction of the training samples with specific triggers injected into their input features, whose ground-truth labels are altered accordingly to be the attacker-specified ones. It is unlikely such backdoor attacks will be detected by monitoring the model training performance since the trained model can still perform well on the benign validation samples. Consequently, during testing phase, if the data is augmented with the trigger, it would be mistakenly classified as the attacker-specified label. Subtle yet effective, backdoor attacks can pose serious threats to the practical application of DNNs.

Another typical type of data poisoning attack is noisy label attacks (Han et al., 2018; Patrini et al., 2017; Yi and Wu, 2019; Jiang et al., 2017), in which the labels of a small fraction of data are altered deliberately to compromise the model learning, while the input features of the training data remain untouched. Backdoor attacks share a close connection to noisy label attacks, in that during a backdoor attack, the feature can only be altered insignificantly to put the trigger in disguise, which makes the corrupted feature (e.g. images with the trigger) highly similar to the uncorrupted ones. Prior efforts have been made to effectively address *noisy label attacks*. For instance, there are algorithms that can tolerate a large fraction of label corruption, with up to 45% noisy labels (Han

et al., 2018; Jiang et al., 2018). However, to the best of our knowledge, most algorithms defending against *backdoor attacks* cannot deal with a high corruption ratio even if the features of corrupted data are only slightly perturbed. Observing the limitation of prior state-of-the-art, we aim to answer one key question: Can one train a deep neural network that is robust against a large number of backdoor attacks? Moreover, given the resemblance between noisy label attacks and backdoor attacks, we also investigate another intriguing question: Can one leverage algorithms initially designed for handling noisy label attacks to defend against backdoor attacks more effectively?

The contributions of this chapter are multi-fold. First, we provide a novel and principled perspective to decouple the challenges of defending backdoor attacks into two components: one induced by the corrupted input features, and the other induced by the corrupted labels, based on which we can draw a theoretical connection between the noisy-label attacks and backdoor data attacks. Second, we propose a meta-algorithm to address both challenges by a novel minimax optimization. Specifically, the proposed approach takes any noisy-label defense algorithm as its input and outputs a reinforced version of the algorithm that is robust against backdoor poisoning attacks, even if the initial form of the algorithm fails to provide such protection. Moreover, we also propose a robust meta-algorithm in semi-supervised setting based on our theorem, leveraging more data information to boost the robustness of the algorithm. Extensive experiments show that the proposed meta-algorithm improves the robustness of DNN models against various backdoor attacks on a variety of benchmark datasets with up to 45% corruption ratio, while most previous study on backdoor attack only provide robustness against small corruption ratio. Furthermore, we propose a systematic, meta-framework to solve backdoor attacks, which can effectively join existing knowledge in noisy label attack defenses and provides more insights to future development of defense algorithms.

5.2 Related Work

5.2.1 Robust Deep Learning Against Adversarial Attack

Although DNNs have shown high generalization performance on various tasks, it has been observed that a trained DNN model would yield different results even by perturbing the image in an invisible manner (Goodfellow et al., 2014b; Yuan et al., 2019). Prior efforts have been made to tackle this issue, among which one natural defense strategy is to change the empirical loss minimization into a minimax objective. By solving the minimax problem, the model is guaranteed a better worst-case generalization performance (Duchi and Namkoong, 2021). Since exactly solving the inner maximization problem can be computationally prohibitive, different strategies have been proposed to approximate the inner maximization optimization, including heuristic alternative optimization, linear programming Wong and Kolter (2018), semi-definite programming Raghunathan et al. (2018), etc. Besides minimax optimization, another approach to improve model robustness is imposing a Lipschitz constraint on the network. Work along this line includes randomized smoothing Cohen et al. (2019); Salman et al. (2019), spectral normalization Miyato et al. (2018a), and adversarial Lipschitz regularization Terjék (2019). Although there are algorithms that are robust against adversarial samples, they are not designed to confront backdoor attacks, in which clean training data is usually inaccessible. There are also studies that investigated the connection between adversarial robustness and robustness against backdoor attack (Weber et al., 2020). However, to our best knowledge, there is no literature studying the relationship between label flipping attack and backdoor attack.

5.2.2 Robust Deep Learning Against Noisy Labels

Many recent studies have investigated the robustness of classification tasks with noisy labels. For example, Kumar et al. (2010) proposed the Self-Paced Learning (**SPL**) approach, which assigns higher weights to examples with a smaller loss. A similar idea was used in Curriculum Learning (Bengio et al., 2009), in which a model is trained on easier examples before moving to the harder ones. Other methods inspired by SPL include learning the data weights (Jiang et al., 2018) and

collaborative learning (Han et al., 2018; Yu et al., 2019). An alternative approach to defending noisy label attacks is label correction (Patrini et al., 2017; Li et al., 2017; Yi and Wu, 2019), which attempts to revise the original labels of the data to recover clean labels from corrupted ones. However, since we do not have the knowledge of which data points have been corrupted, it is nontrivial to obtain provable guarantees for label corrections, unless strong assumptions have been made on the corruption type.

5.2.3 Data Poisoning Backdoor Attack and its Defense

Robust learning against backdoor attacks has been widely studied recently. Gu et al. (2017) showed that even a small patch of perturbation can compromise the generalization performance when data is augmented with a backdoor trigger. Other types of attacks include the blend attacks (Chen et al., 2017), clean label attacks (Turner et al., 2018; Shafahi et al., 2018), latent backdoor attacks (Yao et al., 2019), etc. While there are various types of backdoor attacks, some attack requires that the adversary not only has access to the data but also has limited control on the training and inference process. Those attacks include trojan attacks and blind backdoor attacks (Pang et al., 2020). We refer readers to Pang et al. (2020) for a comprehensive survey on different types of backdoor attacks.

Various defense mechanisms have been proposed to defend against backdoor attacks. One defense category is to remove the corrupted data by using anomaly detection (Tran et al., 2018; Chen et al., 2018). Another category of approach is based on model inspection (Wang et al., 2019a), which aims to inspect and modify the backdoored model to make it robust against the trigger. In addition, there are other methods to tackle the backdoor attacks, such as randomized smoothing (Cohen et al., 2019; Weber et al., 2020), and the median of means (Levine and Feizi, 2020). However, they are either inefficient or cannot defend against backdoor attacks with a large ratio of corrupted data. Some of the above methods also hinge on having a clean set of validation data, which is impractical since it is unlikely we can guarantee the existence of clean validation data given that the validation data is usually a subset of the training data. To the best of our knowledge, there is no existing backdoor defense algorithm that is motivated from the label corruption perspective.

5.3 Preliminaries

5.3.1 Learning with Noisy Labels

There are two representative approaches for defending against noisy-labels: 1) Filtering-based approach is one of the most effective strategies for defending against noisy labels, which works by selecting or weighting the training samples based on indicators such as sample losses (Jiang et al., 2017; Han et al., 2018; Jiang et al., 2020) or gradient norms of the loss-layer (Liu et al., 2021). For instance, Jiang et al. (2017) proposed to assign higher probabilities to samples with lower losses to be selected for model training. 2) Consistency-based approach modifies data labels during model training. Specifically, the Bootstrap approach (Reed et al., 2014) encourages model predictions to be consistent between iterations, by modifying the labels as a linear combination of the observed labels and previous predictions.

Although the initial forms of these approaches can be vulnerable to backdoor attacks, we propose a meta-algorithm that empowers them to effectively counter against backdoor attacks. In this chapter, we examine two filtering-based noisy label algorithms, namely, Self-Paced Learning (**SPL**) Jiang et al. (2017); Kumar et al. (2010) and Provable Robust Learning (**PRL**) Liu et al. (2021), and one consistency-based algorithm, the **Bootstrap** Reed et al. (2014), to investigate the efficacy of the proposed meta algorithm. We briefly summarize the main idea of the above algorithms in Table 5.5 in Appendix section. The empirical results in Section 5.5 strongly suggest that our meta framework can readily benefit the existing robust noisy-label algorithms.

5.3.2 Problem Setting of Backdoor Attacks

We follow the standard setting for backdoor attacks and assume that there is an adversary that tries to perform the backdoor attack. Firstly, the adversary can choose up to ϵ fraction of clean labels $\mathbf{Y} \in \mathbb{R}^{n \times q}$ and modify them to arbitrary valid numbers to form the corrupted labels $\mathbf{Y}_b \in \mathbb{R}^{\lfloor n\epsilon \rfloor \times q}$. Let $\mathbf{Y}_r$ represent the remaining untouched labels. The final training labels can be denoted as $\mathbf{Y}_\epsilon = [\mathbf{Y}_b, \mathbf{Y}_r]$. Accordingly, the corresponding original feature are denoted as

$\mathbf{X} = [\mathbf{X}_o \in \mathbb{R}^{\lfloor n\epsilon \rfloor \times d}, \mathbf{X}_r \in \mathbb{R}^{(n-\lfloor n\epsilon \rfloor) \times d}]$. The adversary can design a trigger $\mathbf{t} \in \mathbb{R}^d$ to form the corrupted feature set $\mathbf{X}_b \in \mathbb{R}^{\lfloor n\epsilon \rfloor \times d}$ such that for any $\mathbf{b}_i$ in $\mathbf{X}_b$, $\mathbf{o}_i$ in $\mathbf{X}_o$, it satisfies $\mathbf{b}_i = \mathbf{o}_i + \mathbf{t}$. Finally, the training features are denoted as $\mathbf{X}_\epsilon = [\mathbf{X}_b \in \mathbb{R}^{\lfloor n\epsilon \rfloor \times d}, \mathbf{X}_r \in \mathbb{R}^{(n-\lfloor n\epsilon \rfloor) \times d}]$. Assuming $\mathbf{T} = [\mathbf{t}, \mathbf{t}, ..., \mathbf{t}] \in \mathbb{R}^{\lfloor n\epsilon \rfloor \times d}$, therefore $\mathbf{X}_o + \mathbf{T} = \mathbf{X}_b$[1]. Before analyzing the algorithm, we make following assumptions about the adversary attack:

Assumption 9 (Bounded Corruption Ratio) *The overall corruption ratio and the corruption ratio in each class is bounded. Specifically,*

$$\mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{y}_b)\in(\mathbf{X},\mathbf{Y},\mathbf{Y}_b)}\left[\frac{\mathbf{I}(\mathbf{y}_b = c|\mathbf{y} \neq c)}{\mathbf{I}(\mathbf{y} = c)}\right] \leq \epsilon = 0.5 \; \forall c \in \triangle\mathbf{Y}.$$

Assumption 10 (Small Trigger) *The backdoor trigger satisfies $\|\mathbf{t}\|_p \leq \tau$, which subtly alters the data within a small radius-τ ball without changing its ground-truth label.*

We also assume that there exists at least one black-box robust algorithm $\mathcal{A}$ which can defend noisy label attacks so long as the noisy-label ratio is bounded by ϵ. Note that the assumption of noisy label algorithm is mild, since a variety of existing algorithm can handle noisy labels attacks with a large corruption rate (e.g. 45%) (Jiang et al., 2017; Han et al., 2018; Reed et al., 2014; Liu et al., 2021).

5.4 Methodology

Given an ϵ-backdoor attacked dataset $(\mathbf{X}^\epsilon, \mathbf{Y}^\epsilon)$, a clean distribution $p^* := (\mathbf{X}, \mathbf{Y})$, and a loss function $\mathcal{L}$, our goal is to learn a network function f that minimizes the generalization error under the **corrupted** distribution, i.e. $\mathbb{E}_{(x,y)\sim p^*}[\mathcal{L}(f(x+\mathbf{t}), y)]$ and **clean** distribution, i.e. $\mathbb{E}_{(x,y)\sim p^*}[\mathcal{L}(f(x), y)]$. Next, we elaborate our meta-approach for defending against backdoor attacks in order to achieve our goal.

[1]Some backdoor attack algorithms design instance-specific trigger. In this chapter, we only focus on the static trigger case and leave the instance-specific trigger case for our future study.

5.4.1 A black-box robust algorithm against noisy labels

The ultimate goal for defending against backdoor attacks is to learn a network function f to minimize its risk given some corrupted input features:

$$\min_f J(f) := \mathbb{E}_{(x,y)\sim p^*} \left[\mathcal{L}(f(x+\mathbf{t}), y)\right]. \tag{5.1}$$

However, Equation 5.1 is not directly optimizable for two reasons: 1) we only have access to the corrupted *inputs* and the corrupted *labels* $\mathbf{Y}^\epsilon$, and 2) the trigger $\mathbf{t}$ is unknown. As such, we consider an surrogate objective that optimizes the worst-case of Equation 5.1:

$$\min_f \max_{\|\mathbf{c}\|_p \leq \tau} \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \left[\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})\right]. \tag{5.2}$$

Since the trigger satisfies $\|\mathbf{t}\|_p \leq \tau$, it is easy to see that Equation 5.2 minimizes an *upper-bound* of the ground-truth loss, in that: $\frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \mathcal{L}(f(\mathbf{x}+\mathbf{t}), \mathbf{y}) \leq \max_{\|\mathbf{c}\|_p\leq\tau} \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \left[\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})\right]$. To this end, directly optimizing the surrogate objective in Equation 5.2 is still intractable, since we do not have access to clean $\mathbf{X}$ and $\mathbf{Y}$, which prevent us from using adversarial training to solve the minimax objective. To tackle this challenge, we will first assume that the clean label $\mathbf{Y}$ is available, and then relax this assumption by using any learning algorithms that are robust against noisy labels. Specifically, by assuming that $\phi_{\mathbf{w}} = \mathcal{L} \circ f$ has a Lipschitz constant L w.r.t. $\mathbf{x}$, we further obtain a new upper bound (see Appendix for derivation details):

$$\frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \left[\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})\right] \leq \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^\epsilon,\mathbf{y}\in\mathbf{Y}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \epsilon\tau L, \tag{5.3}$$

which draws a principled connection between the risk of using corrupted data and that of using clean data:

$$\begin{aligned} &\min_f \max_{\|\mathbf{c}\|_p \leq \tau} \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \left[\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})\right] \\ &\approx \left\{ \min_f \max_{\|\mathbf{c}\|_p \leq \tau} \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^\epsilon,\mathbf{y}\in\mathbf{Y}} \left[\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})\right] + \epsilon\tau L \right\}, \end{aligned} \tag{5.4}$$

where the first term on the RHS of Equation 5.4 involves optimization on the *corrupted* features $\mathbf{X}^\epsilon$ and *clean* labels $\mathbf{Y}$, while the second term on the RHS requires minimizing the Lipschitz constant

L w.r.t. $\mathbf{x}$. Recall that minimizing the maximum gradient norm is equivalent to minimizing the Lipschitz constant (Terjék, 2019). Therefore, optimizing the first term naturally regulates the maximum change of the loss function within a small ball, which hence constrains the magnitude of the gradient and has negligible effects on the Lipschitz regularization. The relationship between Lipschitz regularization and adversarial training has been well discussed in the literature (Terjék, 2019; Miyato et al., 2018b). We defer this discussion to the Appendix section.

Equation 5.4 indicates that if the target labels are not corrupted and the learned function has a small Lipschitz constant, learning with corrupted features is feasible to achieve a low risk. Up to now, the remaining challenge of optimizing the surrogate objective in Equation 5.4 is the inaccessible clean label set $\mathbf{Y}$. Fortunately, a variety of algorithms are at hand for handling noisy labels during learning (Jiang et al., 2017; Liu et al., 2021; Kumar et al., 2010), which we can directly apply to our minimax optimization scheme. Specifically, for the outer minimization, one can have: $\min_f \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^\epsilon, \mathbf{y}\in\mathbf{Y}^\epsilon} [\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})]$, and we can perform the noisy-label update for the above optimization objective. For instance, given the mini-batch $\mathbf{M}_x$, $\mathbf{M}_y$ with batch size m, if we use SPL to perform the update, we can get the top $(1-\epsilon)m$ data with a small risk $\mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y})$ to perform one-step gradient descent. If we use the PRL to perform the update, assuming $\mathcal{L}$ is the cross-entropy loss, the top $(1-\epsilon)m$ data with small loss-layer gradient norm $\|f(\mathbf{x}+\mathbf{c}) - \mathbf{y}\|$ can be used to perform one-step gradient descent. If we apply the bootstrap method, we can add a bootstrap regularization to update the above objective.

Meanwhile, it is non-trivial to directly solve the inner maximization, since adversarial learning $\mathbf{c}$ in Equation 5.4 still faces the threat of noisy labels. To tackle this issue, we can leverage the same robust noisy label algorithm. Specifically, we first approximate the inner optimization using the first-order Tyler expansion: $\mathbf{c}^* = \arg\max_{\|\mathbf{c}\|_p \leq \tau} \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}, \mathbf{y}\in\mathbf{Y}} \mathcal{L}(f(\mathbf{x}+\mathbf{c}), \mathbf{y}) \approx \arg\max_{\|\mathbf{c}\|_p \leq \tau} \mathbf{c}^T \nabla_{\mathbf{x}} \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}, \mathbf{y}\in\mathbf{Y}} \mathcal{L}(f(\mathbf{x}), \mathbf{y})$. The preceding optimization is a linear programming problem. With the l_∞ norm ball constraint on the perturbation, the optimization problem can be efficiently solved by the fast gradient sign method (FGSM). Given a minibatch $\mathbf{M}_x, \mathbf{M}_y$ with

batchsize m, we have the following closed-form solution:

$$\tilde{\mathbf{c}} = \text{Clip}_{\mathbf{c}} \left\{ \frac{\tau}{m} \cdot \sum_{\mathbf{x} \in \mathbf{M}_x, \mathbf{y} \in \mathbf{M}_y} \text{sign} \left(\nabla_{\mathbf{x}} \mathcal{L} \left(f(\mathbf{x}), \mathbf{y} \right) \right) \right\}. \tag{5.5}$$

To relax the prerequisite of having a clean label set $\mathbf{y}$ in Equation 5.5, we will use a noisy-label algorithm to perform the update. For instance, if we use a loss-filtering based algorithm (e.g. SPL), then for each mini-batch, only the top $(1-\epsilon)m$ data with small $\mathcal{L}\left(f(\mathbf{x}), \mathbf{y}\right)$ would be included in the update. If we adopt a gradient-based filtering algorithm (e.g. PRL), given that $\mathcal{L}$ is the cross-entropy loss, then only the top $(1-\epsilon)m$ data with small $\|f(\mathbf{x}) - \mathbf{y}\|$ will be included. The outside clipping ensures that the feature value of the corrupted image is in the valid range. Based on the above discussion, we now introduce our meta-algorithm in Algorithm 6 that is robust against backdoor attacks, given an arbitrary noisy-label robust algorithm $\mathcal{A}$ as its input. We also provided an illustration in Figure 5.1 of the Appendix.

Algorithm 6: Meta Algorithm for Robust Learning Against Backdoor Attacks

input: Corrupted training data $\mathbf{X}^\epsilon$, $\mathbf{Y}^\epsilon$, perturbation limit: τ, learning with noisy label algorithm $\mathcal{A}$ (e.g. PRL, SPL, Bootstrap).

while epoch $\leq$ max_epoch **do**
  for sampled minibatch $\mathbf{M}_x, \mathbf{M}_y$ in $\mathbf{X}^\epsilon$,$\mathbf{Y}^\epsilon$ **do**
    initialize $\mathbf{c}$ as 0 vector.
    optimize the objective $\max_{\|\mathbf{c}\| \leq \tau} \mathcal{L}(f(\mathbf{M}_x + \mathbf{c}), \mathbf{M}_y)$ w.r.t. to $\mathbf{c}$ by using robust algorithm $\mathcal{A}$ for one step
    optimize the objective $\min_f \mathcal{L}(f(\mathbf{M}_x + \mathbf{c}), \mathbf{M}_y)$ w.r.t. f by using robust algorithm $\mathcal{A}$ for one step
  end for
end while

5.4.2 Theoretical Justification

Our ultimate goal is to learn $\mathbf{w}$ that achieves a low expected risk $\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p*}\phi_{\mathbf{w}}(\mathbf{x}+\mathbf{t},\mathbf{y})$. To study the generalization performance on the ground-truth distribution $p*$, we first define the following risks: $\mathcal{R}_t^{emp} = \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \phi_{\mathbf{w}}(\mathbf{x}+\mathbf{t},\mathbf{y})$, $\mathcal{R}_t = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p*}\phi_{\mathbf{w}}(\mathbf{x}+\mathbf{t},\mathbf{y})$, $\mathcal{R}_c^{emp} = \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X}^\epsilon,\mathbf{y}\in\mathbf{Y}} \phi_{\mathbf{w}}(\mathbf{x}+\mathbf{c},\mathbf{y})$, Next, we focus on the gap between $\mathcal{R}_t$ and $\mathcal{R}_c^{emp}$.

Theorem 6 *Let $\mathcal{R}_c^{emp}, \mathcal{R}_t, \epsilon, \tau$ defined as above. Assume that the prior distribution of the network parameter $\mathbf{w}$ is $\mathcal{N}(0,\sigma)$, and the posterior distribution of parameter is $\mathcal{N}(\mathbf{w},\sigma)$ which is learned*

from the training data. Let k be the number of parameters, n be the sample size, and $\Gamma = \sqrt{\frac{\frac{1}{4}k\log\left(1+\frac{\|\boldsymbol{w}\|_2^2}{k\sigma^2}\right)+\frac{1}{4}+\log\frac{n}{\delta}+2\log(6n+3k)}{n-1}}$. *If the objective function $\phi_{\mathbf{w}} = \mathcal{L} \circ f$ is L_ϕ-Lipschitz smooth, then with probability at least 1-δ, one can have:*

$$\mathcal{R}_t \leq \mathcal{R}_c^{emp} + L_\phi(2\tau + \epsilon\tau) + \Gamma. \tag{5.6}$$

We hereby present the skeleton of the proof and defer more details to the Appendix. First, we decompose the error into two terms: 1) the generalization gap on the triggered data, and 2) the difference of performance loss between the trigger $\mathbf{t}$ and worst case perturbation $\mathbf{c}$: $\mathcal{R}_t - \mathcal{R}_c^{emp} = (\mathcal{R}_t - \mathcal{R}_t^{emp}) + (\mathcal{R}_t^{emp} - \mathcal{R}_c^{emp})$. The first component can be bounded by Γ, which is derived by following the uniform convergence PAC-Bayes framework (Foret et al., 2020). For the second term, the gap is introduced by two sources. The first source is the difference between $\mathbf{c}$ and $\mathbf{t}$, and the second is from the difference between $\mathbf{X}$ and $\mathbf{X}^\epsilon$. Since the objective is L_ϕ Lipschitz, and $\|\mathbf{t} - \mathbf{c}\| \leq 2\tau$ according to our constraint to the adversary, it is easy to upper bound the error as $2\tau L_\phi$. Meanwhile, there is ϵ-fraction of difference between $\mathbf{X}$ and $\mathbf{X}^\epsilon$, which is bounded by $\|\mathbf{t}\| < \tau$ and leads to the other difference term $L_\phi\epsilon\tau$.

Theorem 6 presents an upper-bound of the gap $\mathcal{R}_t - \mathcal{R}_c^{emp}$. The first term in Equation 5.6 can be minimized by using a noisy label algorithm. The second term, which is the error induced by the adversarial trigger, is jointly constrained by the Lipschitz constant L_ϕ, perturbation limit τ, and the corruption ratio ϵ. We can regularize the L_ϕ whereas the τ and ϵ are controlled by the unknown adversary. Note that existing literature has also shown that adversarial training plays a similar role as Lipschitz regularization. The last term, the normal generalization error on the clean data, is difficult to minimize directly. The bound in Theorem 6 emphasizes the importance of involving both the noisy label algorithm and the adversarial training. The noisy label algorithm can reduce the $\mathcal{R}_c^{emp}$ while the adversarial training regularize the Lipschitz constant $\mathcal{L}_\phi$.

5.4.3 A semi-supervised algorithm

In backdoor attacks, most attacking algorithms require modifying the label to successfully deploy the attacks. If we could leverage the knowledge from unlabeled data (i.e. via semi-supervised learning), the model performance will likely improve. In this section, we extend Theorem 6 to a semi-supervised learning setting and show that utilizing more data can benefit the model robustness. Our motivation is from the following property of Lipschitz functions. If h is a composition of two functions, f and g ($h = f \circ g$), then $\|h\|_{lip} \leq \|f\|_{lip}\|g\|_{lip}$. Recall in Eq. 5.6, the Lipschitz constant L_ϕ depends on the loss function ϕ, which can be decomposed into the representation function $h : X \to Z$, a linear prediction layer $q : Z \to \tilde{Y}$, and a cross entropy layer $CE : Y \times \tilde{Y} \to \mathcal{R}$. We then have the following proposition:

Proposition 2 *With the assumptions in Theorem 6, let the network be a composition of representation extraction h and linear classifier q. Let σ_{max} be the maximum singular value of the last layer linear prediction weight matrix (i.e. fine-tuning layer). If the representation extraction is L_h Lipschitz, then with probability at least 1-δ, we have:*

$$\mathcal{R}_t \leq \mathcal{R}_c^{emp} + L_h \sigma_{max} \sqrt{2}(2\tau + \epsilon\tau) + \Gamma.$$

The proof is provided in the Appendix. The advantage of decomposing the Lipschitz constant of the objective function into the Lipschitz constants of the representation and prediction functions is that controlling L_h does not require access to the labels. This suggests that we can leverage the unlabeled data to control L_h and let the supervised learning part to control σ_{max}. Let the representation of the last layer be $\mathbf{Z} = h(\mathbf{X})$, we have L_h defined as $\|h(\mathbf{X}_1) - h(\mathbf{X}_2)\| \leq L_h\|\mathbf{X}_1 - \mathbf{X}_2\|, \forall \mathbf{X}_1, \mathbf{X}_2$. Then, our goal is to leverage more data to improve L_h, and fine-tune the last linear layer to control the σ_{max} with labeled data.

In this work, we use the SimCLR to learn the representation function h. The SimCLR first defines some random transformation set $\mathcal{T}$ (i.e. cropping, color jittering, flipping, rotation, i.e.), and then samples two random transformations, $\mathcal{T}_1$ and $\mathcal{T}_2$, to generate two views $\mathcal{T}_1(\mathbf{X})$ and $\mathcal{T}_2(\mathbf{X})$ for each image. Then, the model is trained to maximize their cosine similarity. An important

component of SimCLR is that those transformations usually makes images after transformation $\mathcal{T}(\mathbf{X})$ to be semantically close to $\mathbf{X}$, and we assume that there exists some distance metric d (i.e. Wasserstein distance) so that the distance between original image and transformed one is small (i.e. $d(\mathbf{X}, \mathcal{T}_i(\mathbf{X})) \leq \frac{\tau}{2}$, $\forall \mathcal{T}_i \sim \mathcal{T}$). Then, by triangle inequality, we have $d(\mathcal{T}_i(\mathbf{X}), \mathcal{T}_j(\mathbf{X})) \leq \tau$. Thus, the SimCLR actually samples two images which are closed in some distance metric, and then maximizes the cosine similarity, which is equivalent to minimizing the normalized l2-distance. This process can be viewed as enforcing a Lipschitz regularization for the representation learning since SimCLR minimizes the normalized l2 distance in representation space for two random images that are close in Wasserstein distance. The remaining part that needs to be controlled is the maximum singular value of the last linear layer, which can be enforced by using spectral normalization Miyato et al. (2018a). Motivated by this observation, we propose the following semi-supervised robust algorithm to defend against backdoor attacks:

Algorithm 7: Semi-Supervised Algorithm for Robust Learning Against Backdoor Attacks

input: Corrupted training data $\mathbf{X}^\epsilon$, $\mathbf{Y}^\epsilon$, Clean Augmented Dataset $\mathbf{X}_{aug}$, perturbation limit: τ, learning with noisy label algorithm $\mathcal{A}$ (e.g. PRL, SPL, Bootstrap).

Use SimCLR on $[\mathbf{X}^\epsilon, \mathbf{X}_{aug}]$ to learn the representation function h, then fine-tune the linear layer with spectral normalization using noisy label algorithm as following
while epoch $\leq$ max_epoch **do**
  for sampled minibatch $\mathbf{M}_x, \mathbf{M}_y$ in $\mathbf{X}^\epsilon$,$\mathbf{Y}^\epsilon$ **do**
    $\min_f \mathcal{L}(f_{lin}(h(\mathbf{M}_x)), \mathbf{M}_y)$ w.r.t. f_{lin} by using robust algorithm $\mathcal{A}$ for one step
    use spectral normalization to truncate the largest singular value of last linear layer.
  end for
end while

5.4.4 How to choose the noisy label algorithm

One key question regarding our framework is how to choose the noisy label algorithm. In practice, we found PRL gives consistent robustness against both badnet and blending attacks on different settings. This might be because PRL is designed for agnostic corrupted supervision, which is suitable for a variety of noisy label attack types.

From a theoretical view, analyzing how different noisy label algorithms minimize the first term of RHS in Eq. 5.6 depends on the noisy label algorithm used. Here we present a high-level analysis for

PRL. PRL guarantees convergence to the ϵ-approximated stationary point, where ϵ is the corrupted ratio. Formally, we have the following proposition:

Proposition 3 *Given the assumptions used in Theorem 6, assume the objective function $\phi_{\mathbf{w}} = \mathcal{L} \circ f$ is L_ϕ-Lipschitz smooth and satisfying the PL condition $\frac{1}{2}\|\nabla\phi_{\mathbf{w}}\| \geq \mu(\phi_{\mathbf{w}} - \phi_{\mathbf{w}^*})$. Then, with the assumption of bounded operator norm of gradient before loss layer, we have with probability at least 1-δ, by applying PRL-AT, we have:*

$$\mathcal{R}_t \leq \frac{1}{\mu}O(\epsilon) + L_\phi(2\tau + \epsilon\tau) + \Gamma.$$

The proof is in the Appendix. In general, considering ϕ is a deep neural network, the first term is more difficult to analyze without further assumptions (i.e. PL condition). Nevertheless, empirical study shows that many noisy label algorithms can effectively minimize the first term, noisy label loss, even though some of them have theoretical guarantees while do not. This motivates us to treat these algorithms as black-box algorithms.

Backdoor Attack Defense Accuracy.

Dataset	ϵ	AT	BootStrap	Bootstrap-AT	PRL	PRL-AT	SPL	SPL-AT	Standard	Fine-Pruning	SpecSig
CIFAR10 with **Patch** Attack, **Poison** Accuracy	0.15	66.64 ± 5.28	2.09 ± 0.13	3.05 ± 0.47	81.71 ± 0.37	80.15 ± 0.42	34.60 ± 1.57	77.60 ± 3.81	2.10 ± 0.10	56.67 ± 0.23	35.90 ± 2.13
	0.25	63.98 ± 7.16	2.01 ± 0.23	2.75 ± 0.17	45.94 ± 25.19	78.14 ± 0.48	10.87 ± 2.13	22.17 ± 10.51	2.13 ± 0.15	60.85 ± 0.42	29.02 ± 5.34
	0.35	60.19 ± 1.35	1.98 ± 0.15	2.66 ± 0.16	31.27 ± 17.63	75.04 ± 0.29	11.74 ± 1.24	15.40 ± 7.56	2.01 ± 0.09	56.84 ± 0.15	51.59 ± 3.24
	0.45	51.25 ± 1.81	1.94 ± 0.12	2.53 ± 0.20	17.50 ± 1.66	58.90 ± 12.52	12.32 ± 1.20	14.00 ± 5.35	1.88 ± 0.04	44.21 ± 3.24	24.10 ± 6.23
CIFAR10 with **Patch** Attack, **Clean** Accuracy	0.15	66.77 ± 5.17	85.22 ± 0.48	82.62 ± 0.26	82.06 ± 0.16	80.25 ± 0.43	77.35 ± 2.76	77.70 ± 3.78	85.40 ± 0.37	80.34 ± 0.37	80.32 ± 0.26
	0.25	63.98 ± 7.16	85.25 ± 0.19	81.90 ± 0.25	78.57 ± 1.03	78.22 ± 0.56	69.52 ± 2.38	68.49 ± 2.76	85.20 ± 0.26	79.50 ± 0.15	80.40 ± 0.15
	0.35	60.31 ± 1.37	84.86 ± 0.13	81.75 ± 0.25	73.63 ± 0.75	75.10 ± 0.31	60.23 ± 3.14	58.88 ± 3.46	84.73 ± 0.13	79.10 ± 0.27	72.01 ± 0.31
	0.45	51.25 ± 1.81	1.94 ± 0.12	2.53 ± 0.20	17.50 ± 1.66	58.90 ± 12.52	50.82 ± 1.48	14.00 ± 5.35	1.88 ± 0.04	78.73 ± 0.16	24.01 ± 0.34
CIFAR10 with **Blend** Attack, **Poison** Accuracy	0.15	65.15 ± 0.94	2.17 ± 0.17	24.98 ± 10.01	6.41 ± 3.91	79.71 ± 0.33	11.60 ± 6.56	74.77 ± 3.53	2.29 ± 0.10	34.38 ± 0.13	70.74 ± 0.28
	0.25	56.98 ± 0.72	2.06 ± 0.10	33.33 ± 20.03	6.77 ± 2.81	76.99 ± 0.37	11.60 ± 8.59	52.36 ± 10.57	2.03 ± 0.18	13.94 ± 0.24	75.40 ± 0.35
	0.35	47.84 ± 1.49	1.86 ± 0.07	13.13 ± 7.11	9.42 ± 5.28	73.17 ± 0.96	12.71 ± 9.33	50.79 ± 7.92	1.97 ± 0.07	23.71 ± 0.43	66.87 ± 0.14
	0.45	34.66 ± 1.49	1.83 ± 0.11	6.12 ± 2.86	8.13 ± 4.50	49.88 ± 8.43	8.69 ± 4.41	35.06 ± 4.00	1.88 ± 0.06	16.36 ± 0.26	41.32 ± 0.36
CIFAR10 with **Blend** Attack, **Clean** Accuracy	0.15	66.14 ± 0.98	85.54 ± 0.58	81.44 ± 0.58	77.51 ± 1.20	80.06 ± 0.34	76.25 ± 2.78	75.65 ± 3.11	85.28 ± 0.34	79.53 ± 0.15	83.60 ± 0.37
	0.25	58.91 ± 5.70	84.95 ± 0.30	80.89 ± 0.65	71.45 ± 1.40	77.82 ± 0.26	67.86 ± 2.58	65.08 ± 0.82	85.06 ± 0.39	79.32 ± 0.26	81.23 ± 0.26
	0.35	50.07 ± 13.26	84.72 ± 0.58	80.63 ± 0.57	66.22 ± 1.15	74.34 ± 1.01	60.52 ± 2.26	60.16 ± 2.39	84.72 ± 0.28	78.28 ± 0.17	76.63 ± 0.19
	0.45	38.03 ± 15.42	84.36 ± 0.38	80.35 ± 0.39	55.78 ± 2.09	57.17 ± 9.02	49.48 ± 2.19	46.74 ± 0.71	84.07 ± 0.17	76.70 ± 0.24	62.53 ± 0.29

Table 5.1: Performance on CIFAR10. ϵ is the corruption rate.

5.5 Experiment

We perform experiments on CIFAR10, CIFAR100, and STL10 benchmark data to validate our approach. We use ResNet-32 (He et al., 2016) as the backbone network structure for the experiments.

Backdoor Attack Defense Accuracy.											
Dataset	ϵ	AT	BootStrap	Bootstrap-AT	PRL	PRL-AT	SPL	SPL-AT	Standard	Fine-Pruning	SpecSig
CIFAR100 with **Patch** Attack, **Poison** Accuracy	0.15	23.70 ± 1.39	5.23 ± 0.81	44.74 ± 4.05	15.15 ± 9.17	47.11 ± 0.58	24.87 ± 5.27	42.24 ± 0.76	5.28 ± 0.50	12.50 ± 0.51	30.01 ± 0.23
	0.25	21.84 ± 1.17	3.07 ± 0.23	44.09 ± 1.10	17.53 ± 18.06	43.81 ± 0.41	8.48 ± 1.13	35.46 ± 1.13	3.10 ± 0.60	13.00 ± 0.53	33.82 ± 0.18
	0.35	17.16 ± 1.09	2.85 ± 0.12	40.14 ± 0.20	20.83 ± 10.03	39.76 ± 0.72	7.37 ± 0.59	28.41 ± 1.72	3.24 ± 1.04	25.80 ± 0.12	29.07 ± 0.21
	0.45	13.61 ± 0.74	10.60 ± 10.49	31.21 ± 0.30	23.98 ± 9.32	29.76 ± 1.11	7.26 ± 0.76	20.43 ± 1.69	10.51 ± 11.21	32.33 ± 0.04	16.83 ± 0.43
CIFAR100 with **Patch** Attack, **Clean** Accuracy	0.15	34.08 ± 0.40	52.39 ± 0.38	47.76 ± 0.14	50.50 ± 0.41	47.21 ± 0.56	46.38 ± 0.41	42.38 ± 0.73	52.42 ± 0.59	43.42 ± 0.12	44.23 ± 0.16
	0.25	31.72 ± 0.75	50.54 ± 0.25	44.82 ± 0.52	47.49 ± 0.91	43.89 ± 0.35	39.98 ± 0.80	35.65 ± 1.14	50.53 ± 0.55	41.11 ± 0.03	39.64 ± 0.25
	0.35	29.50 ± 1.73	48.41 ± 0.42	40.38 ± 0.18	44.21 ± 0.21	39.80 ± 0.67	34.11 ± 1.10	28.52 ± 1.70	48.75 ± 0.71	39.34 ± 0.08	29.23 ± 0.39
	0.45	23.93 ± 3.43	41.46 ± 5.00	31.48 ± 0.38	34.34 ± 0.91	29.79 ± 1.13	27.87 ± 2.28	20.55 ± 1.75	41.02 ± 6.06	36.32 ± 0.13	16.94 ± 0.14
CIFAR100 with **Blend** Attack, **Poison** Accuracy	0.15	33.65 ± 0.54	2.19 ± 0.28	46.65 ± 0.33	2.10 ± 0.43	46.01 ± 0.50	6.14 ± 1.12	41.57 ± 0.74	2.09 ± 0.20	19.09 ± 0.48	35.64 ± 0.44
	0.25	30.95 ± 0.42	1.17 ± 0.08	41.84 ± 0.59	1.45 ± 0.21	41.78 ± 0.76	2.95 ± 0.56	33.54 ± 1.76	1.12 ± 0.20	8.80 ± 0.32	33.61 ± 0.36
	0.35	27.30 ± 0.45	1.05 ± 0.06	31.88 ± 1.26	1.51 ± 0.17	34.51 ± 1.60	2.00 ± 0.49	25.71 ± 2.31	1.08 ± 0.16	6.12 ± 0.05	27.13 ± 0.17
	0.45	20.79 ± 4.97	0.99 ± 0.07	23.61 ± 1.07	2.68 ± 1.17	22.00 ± 1.95	2.39 ± 0.17	18.62 ± 1.21	0.92 ± 0.11	8.13 ± 0.02	18.35 ± 0.32
CIFAR100 with **Blend** Attack, **Clean** Accuracy	0.15	34.22 ± 0.58	52.65 ± 0.19	47.77 ± 0.36	48.61 ± 0.18	46.92 ± 0.47	46.01 ± 0.40	42.40 ± 0.70	52.60 ± 0.59	43.30 ± 0.11	45.54 ± 0.16
	0.25	33.65 ± 0.55	51.12 ± 0.37	44.75 ± 0.45	45.23 ± 0.34	42.87 ± 0.72	40.47 ± 1.47	35.71 ± 1.10	50.98 ± 0.43	41.11 ± 0.08	41.02 ± 0.24
	0.35	28.14 ± 0.48	49.80 ± 0.24	40.85 ± 0.37	40.46 ± 0.17	36.30 ± 1.24	35.70 ± 1.68	28.56 ± 2.05	49.65 ± 0.49	39.84 ± 0.06	32.13 ± 0.35
	0.45	22.03 ± 0.49	48.46 ± 0.53	34.78 ± 1.39	34.98 ± 0.83	24.71 ± 1.37	29.91 ± 1.40	21.82 ± 1.21	48.07 ± 0.52	37.83 ± 0.08	19.40 ± 0.27

Table 5.2: Performance on CIFAR100. ϵ is the corruption rate.

The initial learning rates for all the methods are set to be 3e-4. We also use AdamW (Loshchilov and Hutter, 2017) as the optimizer for all methods. The evaluation metric is the top-1 accuracy for both clean testing data and testing data with backdoor trigger.

For backdoor attacks, we use simple badnet attack (Gu et al., 2017) and Gaussian blending attack (Chen et al., 2017), since these two attacks do not require any information about the model or training procedure (Pang et al., 2020). Examples of the poisoned samples can be found in the later section. We deploy the multi-target backdoor attack in this chapter. Our data poisoning approach is as follows: we first systematically flip the label to perform a label-flipping attack. We then add triggers to the features associated with the attacked samples. Without adding the trigger, the problem would have reduced to the noisy label problem.

We use the following two evaluation metrics: (1) **top-1 clean accuracy**, which is calculated from the clean test examples without any triggers and (2) **top-1 poison accuracy**, which is calculated by comparing the predicted class of the poisoned test examples against their ground truth clean labels. The first metric evaluates how well the model performs on benign (uncorrupted) data while the second metric assesses how well the model performs on the corrupted data. We vary the training data poisoning rate as $[15\%, 25\%, 35\%, 45\%]$ to investigate how the algorithms perform for different corruption ratios. All the methods are trained for 100 epochs, Furthermore, we assume there is no clean validation data available. Thus, it is difficult to perform early stopping or decide which epoch result to use. We report the average accuracy across the last 10 epochs for each method.

We study three noisy label algorithms by comparing the performance of the original and reinforced methods. Specifically, we choose SPL, PRL, and Bootstrap as our original noisy label algorithm and denote their corresponding reinforced algorithm with adversarial training as SPL-AT, PRL-AT, Bootstrap-AT. We also compare our method against adversarial training only (AT), which uses adversarial training without a noisy label algorithm. To measure the success of the attack, we also include the training results.

We also evaluate the performance of other backdoor defense algorithms. Note that a large fraction of them are either designed for single target attacks (Liu et al., 2018) or require clean data (Liu et al., 2018; Wang et al., 2019a; Li et al., 2021). In this chapter, we compare our framework against the following two baselines: (1) **spectral signature** (Tran et al., 2018): which filters the data by examining the score of projecting to singular vector, and (2) **fine-pruning** (Liu et al., 2018), which prunes the model by deleting non-activated neurons. Note this method uses 5% clean training data.

How well do existing robust noisy label algorithms defend against backdoor attacks? To answer this, we evaluate the performance of PRL, SPL, and Bootstrap on the CIFAR10 and CIFAR100 datasets. The results are given in Tables 5.1 and 5.2. Observe that the existing algorithms perform well on the benign testing data (i.e. high clean accuracy) but poorly on the corrupted data (i.e. low poison accuracy) especially when the corruption ratio is high. This suggests the ability of backdoor attacks to compromise the defense mechanism of existing robust noisy label algorithms.

How adversarial training improves noisy label algorithms? To investigate whether adversarial training can enhance the robustness of existing noisy label algorithms against backdoor attacks, we evaluate the performance of our proposed reinforced algorithms, SPL-AT, PRL-AT and Bootstrap-AT, on both the clean and corrupted test examples. The results shown in Tables 5.1 and 5.2 suggest that the performance of the reinforced noisy label algorithms on the triggered data is largely boosted, with significant improvement in the poison accuracy. The improvement is observed for all three noisy label algorithms, which indicates the effectiveness of the proposed method on improving the robustness of the existing algorithms against backdoor attacks. Also, compared to adversarial

Backdoor Attack Defense Accuracy.				**CIFAR100→ CIFAR10**		**STL10 → CIFAR10**	
Dataset	ϵ	Standard	PRL-AT	PRL-SimCLR	PRL-SimCLR-SN	PRL-SimCLR	PRL-SimCLR-SN
	0.15	26.66 ± 0.07	64.43 ± 8.37	83.72 ± 0.04	82.99 ± 0.05	80.73 ± 0.06	82.96 ± 0.05
Patch Attack,	0.25	5.67 ± 0.02	60.94 ± 0.88	26.91 ±0.05	80.78 ± 0.12	78.07 ± 0.08	77.92 ± 0.23
Poison Accuracy	0.35	5.20 ± 0.13	55.53 ± 0.60	36.12 ± 0.06	77.90 ± 0.23	26.91 ± 0.14	45.99 ± 0.27
	0.45	5.28 ± 0.24	46.46 ± 0.33	16.97 ± 1.04	32.94 ± 0.31	45.40 ± 0.25	45.39 ± 0.31
	0.15	67.16 ± 0.09	64.44 ± 0.21	83.65 ± 0.03	83.08 ± 0.05	80.79 ± 0.05	83.01 ± 0.04
Patch Attack,	0.25	67.34 ± 0.07	60.92 ± 0.27	81.54 ± 0.42	80.95 ± 0.13	78.14 ± 0.09	78.05 ± 0.31
Clean Accuracy	0.35	65.44 ± 0.17	55.62 ± 0.47	79.17 ± 0.41	78.23 ± 0.21	71.87 ± 0.16	63.99 ± 0.25
	0.45	63.70 ± 0.13	46.48 ± 0.34	73.97 ± 1.02	72.93 ± 0.33	45.36 ± 0.22	45.41 ± 0.29
	0.15	6.55 ± 0.05	64.22 ± 0.26	82.83 ± 0.43	81.96 ± 0.04	79.82 ± 0.08	83.82 ± 0.08
Blend Attack,	0.25	5.44 ± 0.07	59.88 ± 0.98	81.22 ± 0.59	80.21 ± 0.17	77.34 ± 0.15	82.33 ± 0.18
Poison Accuracy	0.35	4.56 ± 0.14	52.66 ± 2.02	78.18 ± 1.71	77.47 ± 0.22	72.70 ± 0.19	80.31 ± 0.23
	0.45	4.82 ± 0.27	35.62 ± 0.92	69.81 ± 2.19	71.45 ± 0.39	47.36 ± 0.23	76.03 ± 0.29
	0.15	69.46 ± 0.04	63.60 ± 0.29	83.43 ± 0.65	82.62 ± 0.04	80.64 ± 0.07	84.50 ± 0.08
Blend Attack,	0.25	68.02 ± 0.05	54.54 ± 0.31	81.80 ± 0.36	80.76 ± 0.16	78.09 ± 0.13	81.09 ± 0.16
Clean Accuracy	0.35	66.64 ± 0.08	54.54 ± 0.41	78.70 ± 0.80	78.13 ± 0.23	71.92 ± 0.18	82.99 ± 0.24
	0.45	65.34 ± 0.12	40.25 ± 1.13	71.13 ± 1.37	72.51 ± 0.37	47.93 ± 0.23	75.23 ± 0.26

Table 5.3: Accuracy on CIFAR10 in semi-supervised setting. ϵ is the corruption rate.

training only (AT), adding noisy labels does indeed improve the performance, particularly, when comparing PRL-AT to AT. We also found that compared to consistency-based noisy-label algorithm (i.e., Bootstrap), the filtering based algorithms (i.e., SPL and PRL) are more easier to be boosted by adversarial training. The potential reason behind this could be that the filtering-based methods are more efficient compared to consistency-based algorithms Han et al. (2018); Jiang et al. (2017); Liu et al. (2021). Finally, we observe that PRL-AT has higher poisoned accuracy compared to spectral signature and fine-pruning under most settings while its clean accuracy is still high, which indicates the advantage of PRL-AT. For high corruption ratio, the robustness of spectral signature and fine-pruning significantly decreases while PRL-AT still gives reasonable poison accuracy.

Semi-supervised learning. For this experiment, we test our algorithm on backdoored CIFAR10 data with CIFAR100 or STL10 (unlabeled part) as augmented data. For the semi-supervised setting, we only use 20% backdoored data as labeled training data (i.e. in backdoored CIFAR10, when $\epsilon = 0.2$, we have 7500 clean labeled images, 2500 backdoored images, and clean augmented data without label). To investigate the advantage of decoupling the Lipschitz constant of the objective function and to determine whether semi-supervised learning helps improve robustness, we compare **standard training**, **PRL-SimCLR** (i.e. algorithm 7 with PRL as the noisy label algorithm without spectral normalization), **PRL-SimCLR-SN** (i.e. algorithm 7 with PRL as the noisy label algorithm) and **PRL-AT**. The results are given in Table 5.3. As we can see, PRL-AT provides

ϵ	PRL-AT (patch)	PRL-AT (blend)
0.15	68.14/68.00	67.97/68.48
0.25	71.78/71.74	71.28/71.87
0.35	74.26/74.15	74.17/74.32
0.45	69.91/27.02	64.78/54.19

Table 5.4: Sensitivity analysis of ϵ. Average top-1 accuracy across three random seeds. The first number is the clean accuracy while the second number is the poisoned accuracy. The hyperparameter ϵ is fixed to be 0.5 while the ground truth ϵ is varied.

consistent robustness against both patch and blending backdoor attacks. If we utilize more data, both PRL-SimCLR and PRL-SimCLR-SN can achieve better performance for the blending attack. For the patch attack, PRL-SimCLR and PRL-SimCLR-SN show robustness when the corruption ratio is small. For large corruption ratio, PRL-SimCLR fails to achieve its robustness against patch attack while PRL-SimCLR-SN still maintains good performance, which indicates the necessity of adding spectral normalization to regularize the maximum singular value of the last layer.

Ablation study for ϵ. Since the degree of corruption is often unknown, we perform experiments to investigate how well our algorithm performs without knowing the true corruption ratio. Specifically, we provide the worst-case result by setting $\epsilon = 0.5$ for our algorithm regardless of the ground truth ϵ. We choose this as it would be impossible to learn a reasonable classifier when the corruption ratio is more than 0.5. We evaluate the performance of PRL-AT for CIFAR10 on both badnet and blending attacks. The results in Table 5.4 suggest that our algorithm is still robust despite using the highly-overestimated ϵ when compared to the standard training results in Table 5.1. Besides ϵ, we also provide more results of ablation studies of the inner-maximization and outer-minimization in the Appendix section.

5.6 Proof of Theorems

5.6.1 Proof of Inequality in Eq. 5.3

In this section we provide a formal proof of the inequality in Eq. 5.3 in the main paper:

$$\frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}}[\mathcal{L}(f(\mathbf{x}+\mathbf{c}),\mathbf{y})] \leq \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X}^{\epsilon},\mathbf{y}\in\mathbf{Y}}\phi_w(\mathbf{x}_i+c,\mathbf{y})+\epsilon\tau L$$

Proof 2 *let $\mathcal{G}$ denote the initially clean sample set (i.e. $(\mathbf{X}, \mathbf{Y})$), and $\mathcal{B}$ the corrupted sample set (i.e. the training set corrupted with a trigger whereas the labels are untouched). Let $\mathcal{R}$ denote the clean sample set which is replaced by the adversary (i.e. $\mathcal{R}$ is the subset of $\mathcal{G}$, and is replaced by $\mathcal{B}$, i.e. $\mathcal{G}^{'} = \mathcal{G} \setminus \mathcal{R} \cup \mathcal{B} = (\mathbf{X}^{\epsilon}, \mathbf{Y})$), and let ϕ_w denote the function $\mathcal{L} \circ f$.*

One can decompose the inner part of our mini-max objective in Equation 5.2 as follows,

$$
\begin{aligned}
\frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} [\mathcal{L}(f(\mathbf{x}+\mathbf{c}),\mathbf{y})] &= \frac{1}{n} \sum_{i\in\mathcal{G}'\setminus\mathcal{B}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \frac{1}{n} \sum_{i\in\mathcal{R}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) \\
&= \frac{1}{n} \sum_{i\in\mathcal{G}'\setminus\mathcal{B}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \frac{1}{n} \sum_{i\in\mathcal{R}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \frac{1}{n} \sum_{i\in\mathcal{B}} \phi_w(\mathbf{x}_i + \mathbf{t} + \mathbf{c}, \mathbf{y}) \\
&- \frac{1}{n} \sum_{i\in\mathcal{B}} \phi_w(\mathbf{x}_i + \mathbf{t} + \mathbf{c}, \mathbf{y}) \\
&= \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^{\epsilon},\mathbf{y}\in\mathbf{Y}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \left(\frac{1}{n} \sum_{i\in\mathcal{R}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) - \frac{1}{n} \sum_{i\in\mathcal{B}} \phi_w(\mathbf{x}_i + \mathbf{c} + \mathbf{t}, \mathbf{y}) \right) \\
&\leq \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^{\epsilon},\mathbf{y}\in\mathbf{Y}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \left| \left(\frac{1}{n} \sum_{i\in\mathcal{R}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) - \frac{1}{n} \sum_{i\in\mathcal{B}} \phi_w(\mathbf{x}_i + \mathbf{c} + \mathbf{t}, \mathbf{y}) \right) \right| \\
&\leq \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^{\epsilon},\mathbf{y}\in\mathbf{Y}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \epsilon L \|t\| \leq \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{X}^{\epsilon},\mathbf{y}\in\mathbf{Y}} \phi_w(\mathbf{x}_i + c, \mathbf{y}) + \epsilon\tau L,
\end{aligned}
$$

This concludes the proof.

5.6.2 Proof of theorem 1 and corollary 2

Theorem 7 *Let $\tilde{\mathcal{R}}_c^{emp}, \mathcal{R}_c^{emp}, \mathcal{R}_t, \epsilon, \tau$, is defined as above. Assume the prior distribution of the network parameter $\mathbf{w}$ is $\mathcal{N}(0, \sigma)$, and the posterior distribution of parameter is $\mathcal{N}(\mathbf{w}, \sigma)$ is the posterior parameter distribution, where $\mathbf{w}$ is learned according to training data. Let k to be the number of parameters, n to be the sample size, assume the objective function $\phi_{\mathbf{w}} = \mathcal{L} \circ f$ is L_ϕ-lipschitz smooth, then, with probability at least 1-δ, we have:*

$$
\mathcal{R}_t \leq \mathcal{R}_c^{emp} + L_\phi(2\tau + \epsilon\tau) + \sqrt{\frac{\frac{1}{4}k \log\left(1 + \frac{\|\boldsymbol{w}\|_2^2}{k\sigma^2}\right) + \frac{1}{4} + \log\frac{n}{\delta} + 2\log(6n + 3k)}{n-1}}.
$$

Proof 3 *we first decompose the gap as following*

$$\mathcal{R}_t - \mathcal{R}_c^{emp} = (\mathcal{R}_t - \mathcal{R}_t^{emp}) + (\mathcal{R}_t^{emp} - \mathcal{R}_c^{emp}) \leq |(\mathcal{R}_t - \mathcal{R}_t^{emp})| + |(\mathcal{R}_t^{emp} - \mathcal{R}_c^{emp})|$$

We bound the second part first.

$$\begin{aligned}
&\mathcal{R}_t^{emp} - \mathcal{R}_c^{emp} \leq \|\mathcal{R}_t^{emp} - \mathcal{R}_c^{emp}\| \\
&= \frac{1}{n}\| \sum_{\mathbf{x}\in\mathbf{X}_r,\mathbf{y}\in\mathbf{Y}_r} [\phi(\mathbf{x}+\mathbf{t},\mathbf{y}) - \phi(\mathbf{x}+\mathbf{c},\mathbf{y})] + \left[\sum_{\mathbf{x}\in\mathbf{X}_o,\mathbf{y}\in\mathbf{Y}_o} \phi(\mathbf{x}+\mathbf{t},\mathbf{y}) - \sum_{\mathbf{x}\in\mathbf{X}_b,\mathbf{y}\in\mathbf{Y}_o} \phi(\mathbf{x}+\mathbf{c},\mathbf{y}) \right] \| \\
&\leq \frac{1}{n}\| \sum_{\mathbf{x}\in\mathbf{X}_r,\mathbf{y}\in\mathbf{Y}_r} [\phi(\mathbf{x}+\mathbf{t},\mathbf{y}) - \phi(\mathbf{x}+\mathbf{c},\mathbf{y})] \| + \frac{1}{n}\| \sum_{\mathbf{x}\in\mathbf{X}_o,\mathbf{y}\in\mathbf{Y}_o} \phi(\mathbf{x}+\mathbf{t},\mathbf{y}) - \sum_{\mathbf{x}\in\mathbf{X}_b,\mathbf{y}\in\mathbf{Y}_o} \phi(\mathbf{x}+\mathbf{c},\mathbf{y}) \| \\
&\leq (1-\epsilon)L_\phi\|\mathbf{t}-\mathbf{c}\| + \epsilon L_\phi\|\mathbf{t}-\mathbf{c}\| + L_\phi \max_{\mathbf{x}_o,\mathbf{x}_b} \|\mathbf{x}_o - \mathbf{x}_b\| \\
&\leq (1-\epsilon)L_\phi\|\mathbf{t}-\mathbf{c}\| + \epsilon L_\phi\|\mathbf{t}-\mathbf{c}\| + \epsilon L_\phi\|\mathbf{t}\| \\
&= L_\phi\|\mathbf{t}-\mathbf{c}\| + \epsilon L_\phi\|\mathbf{t}\| \\
&\leq L_\phi 2\tau + \epsilon L_\phi\|t\| \\
&\leq L_\phi(2\tau + \epsilon\tau)
\end{aligned}$$

Now, we bound the second term. Note the second term is a typical gap term between empirical loss and generalization loss, and there are many approaches to bound this term like VC dimension. Since we aimed to focus the deep neural network, we follow the PAC-Bayes framework McAllester (1999) to analyze the generalization bound. Specifically, we use results from Foret et al. (2020), which gives $\sqrt{\frac{\frac{1}{4}k\log\left(1+\frac{\|\boldsymbol{w}\|_2^2}{k\sigma^2}\right)+\frac{1}{4}+\log\frac{n}{\delta}+2\log(6n+3k)}{n-1}}$ *under the assumption of gaussian prior and posterior. The proof for this can be found in the appendix of Foret et al. (2020) (i.e. equation 13 on the paper).*

As for the corollary 2, the proof is straightforward. By decomposing the Lipschitz constant of the loss function to the Lipschitz constant of representation network, last linear layer, and cross-entropy loss, respectively. Since the cross-entropy loss gradient is $\|\mathbf{y}^2 - \mathbf{y}\|$*, where* $\mathbf{y}$ *is a one-hot vector and* $\mathbf{y}^2$ *is a probability vector. Thus, the maximum gradient (i.e. Lipschitz constant) is* $\sqrt{2}$*. As for the linear layer, according to the definition of the operator norm, the Lipschitz constant is exactly the maximum singular value of that linear layer. This concludes the proof.*

5.6.3 Proof of Proposition 3

We first introduct the property of PRL in the following corollary:

Corollary 3 (Convergence of PRL to clean objective (Liu et al., 2021)) *Assuming the maximum clean gradient before loss layer has bounded operator norm:* $\|W\|_{op} \leq C$*, applying PRL to any* ϵ*-fraction supervision corrupted data, yields* $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\mathbf{w}_t)\|\right) = O(\epsilon\sqrt{q})$ *for large enough* T*, where* q *is the dimension of the supervision.*

Details can be found in Liu et al. (2021). According to above corollary, let $\mathbf{w}_{PRL}$ is the solution get by PRL algorithm, we can have $\|\nabla_{\mathbf{w}_{PRL}}\mathcal{R}_c^{emp}\| = O(\epsilon)$ (i.e. assume q is small). With Polyak-Lojasiewicz (PL) condition with some constant μ such that $\frac{1}{2}\|\nabla f(x)\| \geq \mu(f(x) - f^*)$ holds, we have $\mu(\mathcal{R}_c^{emp} - \mathcal{R}_c^{emp*}) \leq \frac{1}{2}\|\nabla_{\mathbf{w}_{PRL}}\mathcal{R}_c^{emp}\| = O(\epsilon)$. For a highly-overparameterized deep neural network, the global optima $\mathcal{R}_c^{emp*}$ is usually 0. Thus, we can conclude that with PL condition, using PRL as the noisy label algorithm in our framework can guarantee $\mathcal{R}_c^{emp}$ can be minimized to the order $\frac{1}{\mu}O(\epsilon)$.

5.7 More Discussions about the Proposed Framework

In this section, we provided more discussion about our proposed framework.

5.7.1 Noisy Label Algorithm

The details of PRL, SPL, and Bootstrap is showed in table 5.5.

	Mini-batch
PRL	Keep data with small loss-layer gradient norm and perform back-propagation
SPL	Keep data with small loss and perform back-propagation
Bootstrap	change the label by using $y_{true} = \alpha y_{true} + (1-\alpha)y_{pred}$ and perform back-propagation

Table 5.5: Overview of noisy-label defending algorithms, which achieve robustness against up to 45% of pairwise flipping label noises.

5.7.2 Ablation Study of Inner Maximization and Outer Minimization

In this section, we aim to explore more about the proposed framework. Since our algorithm use the noisy-label solver for both inner and outer optimization. A interesting question to ask is that whether both inner and outer noisy-label solver plays an important role in defense the backdoor attack. Thus, we have two variants. One is we only use noisy label algorithm to update the model for outer minimization and another one is we only use the noisy label algorithm to update the model for inner maximization. The results can be found at table 5.6 and table 5.7 in the appendix. As we could see in these two tables, using noisy label algorithm to perform the inner maximization is more important compared to using noisy label algorithm to perform out minimization.

		Backdoor Attack Defense Accuracy.					
Dataset	ϵ	BootStrap-inner	Bootstrap-outer	PRL-inner	PRL-outer	SPL-inner	SPL-outer
CIFAR10 with **Patch** Attack, **Poison** Accuracy	0.15	3.15 ± 0.61	3.20 ± 0.63	80.78 ± 0.31	2.84 ± 0.23	65.50 ± 16.22	3.09 ± 0.35
	0.25	2.74 ± 0.10	2.73 ± 0.09	79.07 ± 0.20	2.50 ± 0.10	18.95 ± 9.91	2.58 ± 0.19
	0.35	2.70 ± 0.24	2.67 ± 0.15	76.06 ± 0.37	2.39 ± 0.26	13.45 ± 5.40	2.38 ± 0.14
	0.45	2.32 ± 0.08	2.51 ± 0.11	67.87 ± 2.63	2.24 ± 0.10	12.10 ± 4.46	2.23 ± 0.26
CIFAR10 with **Patch** Attack, **Clean** Accuracy	0.15	82.58 ± 0.33	82.45 ± 0.25	80.86 ± 0.31	83.09 ± 0.12	76.48 ± 3.03	83.02 ± 0.49
	0.25	82.14 ± 0.28	81.87 ± 0.23	79.10 ± 0.17	83.13 ± 0.21	69.33 ± 2.57	83.30 ± 0.13
	0.35	81.71 ± 0.46	81.55 ± 0.54	76.08 ± 0.34	82.83 ± 0.38	59.76 ± 3.59	83.05 ± 0.38
	0.45	81.53 ± 0.16	81.00 ± 0.47	69.96 ± 0.37	82.78 ± 0.18	49.31 ± 0.53	82.84 ± 0.27
CIFAR10 with **Blend** Attack, **Poison** Accuracy	0.15	29.85 ± 10.65	40.79 ± 13.27	80.38 ± 0.15	46.29 ± 18.09	72.89 ± 6.14	48.21 ± 14.90
	0.25	14.81 ± 10.42	27.57 ± 10.93	78.44 ± 0.19	27.34 ± 18.42	54.46 ± 10.45	21.18 ± 11.85
	0.35	6.52 ± 3.80	17.41 ± 10.13	71.93 ± 2.69	11.25 ± 5.92	46.12 ± 13.77	14.58 ± 7.12
	0.45	11.58 ± 14.94	9.01 ± 4.66	64.98 ± 2.74	5.90 ± 2.28	42.30 ± 5.85	5.16 ± 1.64
CIFAR10 with **Blend** Attack, **Clean** Accuracy	0.15	81.54 ± 0.25	81.18 ± 0.75	80.73 ± 0.18	82.51 ± 0.36	76.11 ± 3.32	82.35 ± 0.22
	0.25	80.99 ± 1.12	80.37 ± 0.94	78.23 ± 0.46	82.35 ± 0.65	66.64 ± 2.31	82.15 ± 0.37
	0.35	81.04 ± 0.81	79.54 ± 1.32	71.62 ± 2.65	82.55 ± 0.47	57.44 ± 1.78	81.81 ± 1.00
	0.45	81.06 ± 0.25	78.93 ± 0.84	62.34 ± 2.51	82.15 ± 0.48	48.82 ± 0.94	81.81 ± 1.06

Table 5.6: Ablation study on CIFAR10. ϵ is the corruption rate.

5.7.3 Discussion about Lipschitz regularization and adversarial training

As seen from the above theorem that a small Lipschitz constant could bring robustness against backdoor attack. In this section, we elaborate why we claim adversarial training helps Lipschitz regularization. The definition of Lipschitz function is $\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y}$. Since the Lipschitz constant shows in the upper bound of the error, we would like to get the minimum Lipschitz constant to tighten the bound. Follow (Terjék, 2019), the minimum Lipschitz constant

		Backdoor Attack Defense Accuracy.					
Dataset	ϵ	BootStrap-inner	Bootstrap-outer	PRL-inner	PRL-outer	SPL-inner	SPL-outer
	0.15	45.76 ± 2.65	41.66 ± 8.37	47.34 ± 0.44	44.32 ± 5.45	43.05 ± 0.39	43.41 ± 6.36
CIFAR100 with **Patch** Attack,	0.25	44.41 ± 1.55	43.65 ± 0.88	44.71 ± 0.40	41.13 ± 5.82	35.69 ± 1.05	41.81 ± 3.89
Poison Accuracy	0.35	40.02 ± 0.19	38.72 ± 0.60	40.19 ± 0.39	38.75 ± 0.60	24.98 ± 6.34	38.97 ± 1.10
	0.45	31.12 ± 0.40	29.46 ± 0.33	31.49 ± 1.04	29.74 ± 0.35	20.13 ± 1.80	30.19 ± 0.31
	0.15	48.01 ± 0.28	47.91 ± 0.21	47.43 ± 0.43	48.41 ± 0.40	43.29 ± 0.21	48.12 ± 0.36
CIFAR100 with **Patch** Attack,	0.25	45.27 ± 0.82	44.68 ± 0.27	44.83 ± 0.38	45.58 ± 0.39	36.21 ± 0.80	45.14 ± 0.64
Clean Accuracy	0.35	40.49 ± 0.23	38.98 ± 0.47	40.40 ± 0.41	39.46 ± 0.30	29.58 ± 1.49	39.89 ± 0.50
	0.45	31.32 ± 0.48	29.81 ± 0.34	31.49 ± 1.02	30.39 ± 0.27	20.70 ± 1.51	30.77 ± 0.55
	0.15	46.72 ± 0.23	46.56 ± 0.26	46.59 ± 0.43	46.83 ± 1.00	42.15 ± 0.68	46.80 ± 0.70
CIFAR100 with **Blend** Attack,	0.25	41.64 ± 1.54	40.60 ± 0.98	43.43 ± 0.59	40.02 ± 1.44	34.10 ± 1.60	39.30 ± 3.16
Poison Accuracy	0.35	31.18 ± 2.83	30.91 ± 2.02	35.84 ± 1.71	28.86 ± 2.82	25.36 ± 2.65	28.94 ± 3.49
	0.45	22.98 ± 1.18	23.37 ± 0.92	24.60 ± 2.19	22.16 ± 3.73	19.57 ± 1.64	24.17 ± 2.70
	0.15	48.05 ± 0.33	47.83 ± 0.29	47.24 ± 0.65	48.43 ± 0.44	42.94 ± 0.55	48.37 ± 0.68
CIFAR100 with **Blend** Attack,	0.25	44.85 ± 0.52	44.59 ± 0.31	44.17 ± 0.36	45.19 ± 0.26	36.18 ± 1.20	45.06 ± 0.18
Clean Accuracy	0.35	40.80 ± 0.56	40.08 ± 0.41	38.23 ± 0.80	41.84 ± 0.51	30.23 ± 1.25	41.18 ± 0.69
	0.45	35.32 ± 1.77	34.13 ± 1.13	27.33 ± 1.37	39.06 ± 0.45	24.22 ± 1.66	38.62 ± 1.30

Table 5.7: Ablation study on CIFAR100. ϵ is the corruption rate.

can be written as:

$$\|f\|_L = \sup_{x,y \in X; x \neq y} \frac{d_Y(f(x), f(y))}{d_X(x, y)}.$$

Rewrite y as $x + c$, we get:

$$\|f\|_L = \sup_{x, x+r \in X; 0 < d_X(x, x+c)} \frac{d_Y(f(x), f(x+r))}{d_X(x, x+r)}.$$

Minimizing the above objective respect to function f reduces to the adversarial learning:

$$\inf_f \|f\|_L = \inf_f \sup_{x, x+c \in X; 0 < d_X(x, x+c)} \frac{d_Y(f(x), f(x+c))}{d_X(x, x+c)}.$$

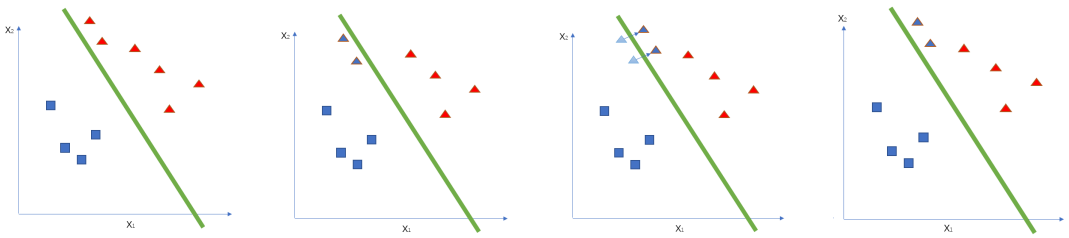
If we treat the denominator as a constant, then this is exactly the same as our minimax objective. More details can be found in (Terjék, 2019).

5.7.4 Supplementary Experiment Results

We provided the experiment hyperparameters, and supplementary results for the experiment. We provided the code in the supplementary materials.

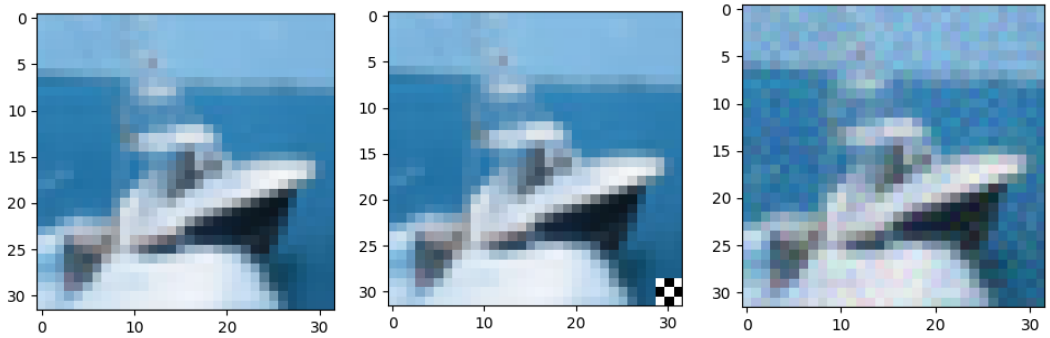
5.7.4.1 Experiment Hyperparameters

We list the details of experiment in this section. All the methods use Resnet-32 as the backbone network. AdamW is used as the optimizer for all methods. The perturbation limit τ is set to be 0.05



(a) clean data (b) two samples are added with small trigger and flipped label (c) Inner maximization by noisy label algorithm (d) Reduce the backdoor attack to label flipping attack

Figure 5.1: Illustration of our meta algorithm. By combining the minimax objective and noisy label algorithm, we could reduce a backdoor attack problem to a label flipping attack. The left most is the clean original data. The second shows corrupted samples. The third figure shows the inner maximization step while the last figure shows the outer minimization step.



(a) clean example (b) badnet attack (c) blend attack

Figure 5.2: Example of clean and various poisoned samples.badnet patch attack: trigger is a 3×3 black-white checkerboard and it is added to the right bottom corner of the image. blending attack: trigger is a fixed Gaussian noise which has the same dimension as the image. The corrupted image generated by $\mathbf{x}_i^\epsilon = (1-\alpha)\mathbf{x}_i + \alpha\mathbf{t}$. In our experiment, we set the α as 0.1.
for all methods requiring τ. All methods are repeated for three different random seeds to calculate the standard deviation. For the SimCLR, we train the network by 500 epochs.

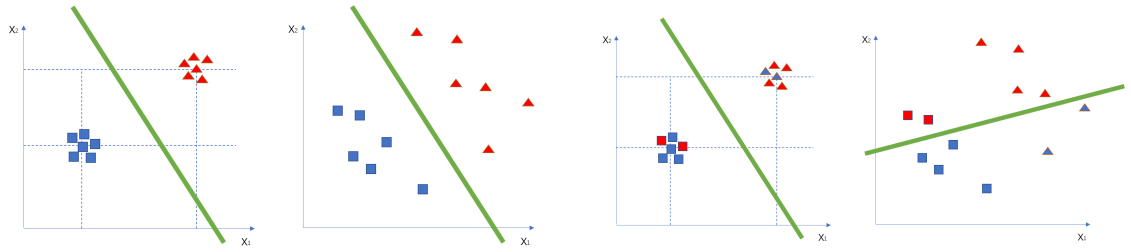
The trigger for badnet attack and blending attack can be found in figure 5.2.

5.7.4.2 Experiment on MNIST

Our experiments showed interesting results on MNIST. In MNIST, we found adversarial training itself sometimes gives robustness to the backdoor attack. We hypothesize that this is because that learning from MNIST is potentially an easier task than that from CIFAR. Here, we show the performance of adversarial training and PRL-AT on MNIST. The results can be found at Table 5.8.

		Backdoor Attack Defense Accuracy.						
Dataset	ϵ	AT	BootStrap	Bootstrap-AT	PRL	PRL-AT	SPL	SPL-AT
MNIST with **Patch** Attack, **Poison** Accuracy	0.15	0.30 ± 0.07	0.04 ± 0.01	3.17 ± 3.23	97.96 ± 0.21	98.44 ± 0.05	59.24 ± 33.30	85.61 ± 12.56
	0.25	0.26 ± 0.17	0.04 ± 0.02	0.17 ± 0.10	89.91 ± 7.53	97.04 ± 1.07	25.25 ± 1.38	30.70 ± 6.62
	0.35	0.10 ± 0.02	0.08 ± 0.06	0.14 ± 0.01	77.91 ± 10.41	97.71 ± 0.18	13.54 ± 0.76	26.03 ± 5.27
	0.45	0.11 ± 0.01	0.04 ± 0.02	0.42 ± 0.34	43.42 ± 11.66	76.42 ± 8.65	12.85 ± 1.90	10.97 ± 2.16
MNIST with **Patch** Attack, **Clean** Accuracy	0.15	98.17 ± 0.69	99.49 ± 0.05	95.48 ± 1.56	98.08 ± 0.26	98.44 ± 0.05	93.23 ± 4.72	97.74 ± 0.37
	0.25	98.59 ± 0.22	99.48 ± 0.07	98.83 ± 0.19	97.46 ± 0.07	97.11 ± 1.06	86.98 ± 0.72	85.89 ± 1.76
	0.35	94.48 ± 4.87	99.48 ± 0.04	98.45 ± 0.32	97.40 ± 0.46	97.86 ± 0.11	73.09 ± 4.34	77.49 ± 0.96
	0.45	98.27 ± 0.43	99.42 ± 0.02	96.44 ± 1.36	75.69 ± 0.99	92.32 ± 4.25	60.58 ± 1.90	57.20 ± 0.37
MNIST with **Blend** Attack, **Poison** Accuracy	0.15	63.42 ± 35.24	0.04 ± 0.01	96.66 ± 2.58	96.81 ± 1.30	96.74 ± 1.03	97.43 ± 0.13	96.16 ± 0.19
	0.25	70.43 ± 28.61	0.04 ± 0.01	97.83 ± 0.91	77.68 ± 20.34	97.20 ± 0.66	6.43 ± 1.27	83.86 ± 2.74
	0.35	58.32 ± 40.59	0.05 ± 0.03	97.94 ± 0.57	78.79 ± 17.74	97.59 ± 0.12	11.05 ± 2.70	69.69 ± 6.59
	0.45	97.66 ± 1.04	0.03 ± 0.03	98.16 ± 0.58	27.18 ± 19.53	95.17 ± 1.83	4.49 ± 1.08	64.78 ± 3.14
MNIST with **Blend** Attack, **Clean** Accuracy	0.15	64.78 ± 33.81	99.44 ± 0.02	98.29 ± 0.81	97.93 ± 0.25	96.18 ± 1.44	97.30 ± 0.22	95.93 ± 0.20
	0.25	74.00 ± 25.25	99.46 ± 0.05	97.44 ± 0.99	97.30 ± 0.63	97.10 ± 0.74	77.75 ± 0.74	83.34 ± 2.81
	0.35	58.62 ± 40.18	99.44 ± 0.02	97.43 ± 0.94	96.25 ± 1.84	97.39 ± 0.19	72.41 ± 3.80	67.92 ± 8.38
	0.45	96.78 ± 1.42	99.42 ± 0.06	97.89 ± 0.61	76.47 ± 8.66	95.07 ± 1.59	63.63 ± 4.47	63.82 ± 4.12

Table 5.8: Performance on MNIST. ϵ is the corruption rate.



(a) clean data for binary feature value and continuous feature value

(b) label flipping attack on both binary feature value and continuous feature value

Figure 5.3: Example of label flipping attacks on both binary feature values and continuous feature values.

As seen for the MNIST, especially for the blend attack, the poison accuracy for adversarial training does show good performance with a large standard deviation. This is because that some random seeds work while some random seeds failed. We hypothesize that this is because MNIST dataset has almost binary feature values. When adding a small Gaussian noise on feature x, the label flipping attack cannot change the decision boundary much. That is why the noisy label algorithm seems is not as important as the noisy label algorithm in CIFAR dataset. We plot a two dimensional toy example in figure 5.3 to illustrate label flipping attack on continuous features and binary features. As seen in the figure, for the binary-valued features, the label flipping attack is not easy to change the decision boundary too much, while it can easily change the decision boundary in the continuous feature value scenario. However, this is a very rough conjecture for the reason why in MNIST,

adversarial training sometimes works. We leave the investigation of this phenomenon in the future work.

CHAPTER 6

CONCLUSION & FUTURE WORK

6.0.1 Conclusion

In this thesis, we present four robust learning algorithms for DNN. Specifically, we proposed RCA and RobustRealNVP dealing with natural corruption, PRL for agnostic adversarial label corruption, as well as a robust meta-algorithm to deal with the backdoor attack.

As we can see, when the corruption is not designed by an adversary, unsupervised anomaly detection is an effective approach, since, with high probability, those injected anomalies would be detected. However, when the corruption is designed by the adversary, it requires us to design a robust algorithm that can tolerate such noise.

6.0.2 Future Work

In this thesis, we mainly focused on the robustness against training corruption. However, many other types of corruption can happen. For example, the adversarial attack is a well-known testing-phase corruption, which can successfully fool the classifier even the classifier is trained on clean data. The key challenge to defend against adversarial attacks is efficiently regularizing the Lipschitz constant of the learned model. In chapter 5, we use empirical min-max optimization to regularize the Lipschitz constant and many other approaches can be used to achieve Lipschitz regularization. Developing more effective/efficient algorithms to reduce the Lipschitz constant of the learned model is one important research problem for studying adversarial robustness. Another type of corruption would be the distributional shift, which can also degrade the classifier generalization performance. In distributional shift, the testing distribution is no longer the same as training distribution, which violates the i.i.d assumption in classical machine learning. To tackle this problem, distribution robust optimization tries to minimize the worst-case loss instead of empirical loss. However, distribution robust optimization itself still faces many challenges such as not being efficient and stable enough.

Studying how to develop an efficient/scalable distribution robust optimization algorithm is also an important research direction.

Besides handling different types of corruption, there are also many other fields such as privacy and fairness, which are highly related to robustness. It turns out in some cases, those problems would finally reduce to a robustness problem. Thus, studying the connections between those fields is also very important in the applied machine learning field. For example, the individual fairness requires the model prediction will not be altered if we change the protected attribute such as race/gender, which is very similar to the lipschitz constant concept in robustness. The difference is in fairness, most protected attribute is discrete instead of continuous. Thus, a key challenge of achieving the individual fairness is to extent the lipschitz regularization to discrete case, which is still an open problem.

Also, there are many theoretical hard-core robustness problems that need to be solved. For example, the overparameterization is empirically shown to be helpful to robustness, however, it is still a mystery for us how overparameterization helps us in terms of robustness. Many papers are trying to describe the relationship between overparameterization and robustness, but according to my best knowledge, none of them completely tackled this problem. Another example would be the efficient robust mean estimation algorithm. Although recent papers already give the subgaussian error rates estimator, the computational efficiency is still unacceptable if we want to use them in the deep neural network. It would be also interesting to study how to design a more efficient robust mean estimation algorithm, which can be used in many other fields including deep learning.

Nowadays, machine learning impacts people's daily life in a dramatic way. The application of machine learning is around us in this era. However, the algorithm behind the applications is not trusted by people for a variety of reasons. Thus, it is highly important for us to study the trustworthy machine learning algorithm, and this requires us to keep exploring the robustness property of the machine learning algorithm.

BIBLIOGRAPHY

BIBLIOGRAPHY

Aggarwal, C. C. and Sathe, S. (2017). Outlier ensembles: An introduction. Springer.

Ajalloeian, A. and Stich, S. U. (2020). Analysis of sgd with biased gradient estimators. arXiv preprint arXiv:2008.00051.

An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE, 2(1):1–18.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning, pages 41–48.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. (2018). signsgd: Compressed optimisation for non-convex problems. arXiv preprint arXiv:1802.04434.

Bhatia, K., Jain, P., Kamalaruban, P., and Kar, P. (2017). Consistent robust regression. In Advances in Neural Information Processing Systems, pages 2110–2119.

Bhatia, K., Jain, P., and Kar, P. (2015). Robust regression via hard thresholding. In Advances in Neural Information Processing Systems, pages 721–729.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pages 93–104.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3):1–58.

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. (2018). Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728.

Chen, R. T., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. arXiv preprint arXiv:1906.02735.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526.

Chen, Y., Zhou, X. S., and Huang, T. S. (2001). One-class svm for learning in image retrieval. In Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205), volume 1, pages 34–37. IEEE.

Cheng, Y., Diakonikolas, I., Ge, R., and Soltanolkotabi, M. (2020). High-dimensional robust mean estimation via gradient descent. arXiv preprint arXiv:2005.01378.

Cohen, J., Rosenfeld, E., and Kolter, Z. (2019). Certified adversarial robustness via randomized smoothing. In International Conference on Machine Learning, pages 1310–1320. PMLR.

d'Aspremont, A. (2008). Smooth optimization with approximate gradient. SIAM Journal on Optimization, 19(3):1171–1183.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Devolder, O., Glineur, F., and Nesterov, Y. (2014). First-order methods of smooth convex optimization with inexact oracle. Mathematical Programming, 146(1-2):37–75.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A. (2019). Sever: A robust meta-algorithm for stochastic optimization. In International Conference on Machine Learning, pages 1596–1606.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. (2016). Robust estimators in high dimensions without the computational intractability. In 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pages 655–664. IEEE.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

Dong, Y., Hopkins, S., and Li, J. (2019). Quantum entropy scoring for fast robust mean estimation and improved outlier detection. In Advances in Neural Information Processing Systems, pages 6067–6077.

Duchi, J. C. and Namkoong, H. (2021). Learning models with uniform performance via distributionally robust optimization. The Annals of Statistics, 49(3):1378–1406.

Emmott, A., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2015). A meta-analysis of the anomaly detection problem. arXiv preprint arXiv:1503.01158.

Fan, J., Zhang, Q., Zhu, J., Zhang, M., Yang, Z., and Cao, H. (2020). Robust deep auto-encoding gaussian process regression for unsupervised anomaly detection. Neurocomputing, 376:180–190.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412.

Goodfellow, I. (2015). Efficient per-example gradient computations. arXiv preprint arXiv:1510.01799.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial networks. arXiv preprint arXiv:1406.2661.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Gu, T., Dolan-Gavitt, B., and Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In Advances in neural information processing systems, pages 8527–8537.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Hein, M. and Andriushchenko, M. (2017). Formal guarantees on the robustness of a classifier against adversarial manipulation. arXiv preprint arXiv:1705.08475.

Hu, Y., Zhang, S., Chen, X., and He, N. (2020). Biased stochastic gradient descent for conditional stochastic optimization. arXiv preprint arXiv:2002.10790.

Huang, C., Cao, J., Ye, F., Li, M., Zhang, Y., and Lu, C. (2019). Inverse-transform autoencoder for anomaly detection. arXiv preprint arXiv:1911.10676.

Huber, P. J. (1992). Robust estimation of a location parameter. In Breakthroughs in statistics, pages 492–518. Springer.

Huber, P. J. et al. (1973). Robust regression: asymptotics, conjectures and monte carlo. Annals of statistics, 1(5):799–821.

Humbert, P., Bars, B. L., Minvielle, L., and Vayatis, N. (2020). Robust kernel density estimation with median-of-means principle. arXiv preprint arXiv:2006.16590.

Jiang, L., Huang, D., Liu, M., and Yang, W. (2020). Beyond synthetic noise: Deep learning on controlled noisy labels. In International Conference on Machine Learning, pages 4804–4815. PMLR.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2017). Mentornet: Learning data-driven

curriculum for very deep neural networks on corrupted labels. arXiv preprint arXiv:1712.05055.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In International Conference on Machine Learning, pages 2304–2313.

Kim, J. and Scott, C. D. (2012). Robust kernel density estimation. The Journal of Machine Learning Research, 13(1):2529–2565.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. arXiv preprint arXiv:1807.03039.

Kingma, D. P. and Welling, M. (2013a). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kingma, D. P. and Welling, M. (2013b). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25:1097–1105.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In Advances in neural information processing systems, pages 1189–1197.

Lai, K. A., Rao, A. B., and Vempala, S. (2016). Agnostic estimation of mean and covariance. In 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pages 665–674. IEEE.

Levine, A. and Feizi, S. (2020). Deep partition aggregation: Provable defense against general poisoning attacks. arXiv preprint arXiv:2006.14768.

Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. (2021). Neural attention distillation: Erasing backdoor triggers from deep neural networks. arXiv preprint arXiv:2101.05930.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L.-J. (2017). Learning from noisy labels with distillation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1910–1918.

Liu, B., Sun, M., Wang, D., Tan, P.-N., and Zhou, J. (2021). Learning deep neural networks under agnostic corrupted supervision. arXiv preprint arXiv:2102.06735.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, pages 413–422. IEEE.

Liu, K., Dolan-Gavitt, B., and Garg, S. (2018). Fine-pruning: Defending against backdooring attacks on deep neural networks. In International Symposium on Research in Attacks, Intrusions, and Defenses, pages 273–294. Springer.

Liu, Y., Li, Z., Zhou, C., Jiang, Y., Sun, J., Wang, M., and He, X. (2019). Generative adversarial active learning for unsupervised outlier detection. IEEE Transactions on Knowledge and Data Engineering.

Liu, Y., Ma, X., Bailey, J., and Lu, F. (2020). Reflection backdoor: A natural backdoor attack on deep neural networks. In European Conference on Computer Vision, pages 182–199. Springer.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Major, B., Fontijne, D., Ansari, A., Teja Sukhavasi, R., Gowaikar, R., Hamilton, M., Lee, S., Grzechnik, S., and Subramanian, S. (2019). Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pages 0–0.

Malach, E. and Shalev-Shwartz, S. (2017). Decoupling" when to update" from" how to update". In Advances in Neural Information Processing Systems, pages 960–970.

McAllester, D. A. (1999). Pac-bayesian model averaging. In Proceedings of the twelfth annual conference on Computational learning theory, pages 164–170.

Menon, A. K., Rawat, A. S., Reddi, S. J., and Kumar, S. (2019). Can gradient clipping mitigate label noise? In International Conference on Learning Representations.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018a). Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018b). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 41(8):1979–1993.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013). Learning with noisy labels. In Advances in neural information processing systems, pages 1196–1204.

Nocedal, J., Sartenaer, A., and Zhu, C. (2002). On the behavior of the gradient norm in the steepest descent method. Computational Optimization and Applications, 22(1):5–35.

Pang, R., Zhang, Z., Gao, X., Xi, Z., Ji, S., Cheng, P., and Wang, T. (2020). Trojanzoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask). arXiv preprint arXiv:2012.09302.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1944–1952.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Prasad, A., Suggala, A. S., Balakrishnan, S., and Ravikumar, P. (2018). Robust estimation via robust gradient estimation. arXiv preprint arXiv:1802.06485.

Qin, Y., Mitra, N., and Wonka, P. (2020). How does lipschitz regularization influence gan training? In European Conference on Computer Vision, pages 310–326. Springer.

Raghunathan, A., Steinhardt, J., and Liang, P. (2018). Semidefinite relaxations for certifying robustness to adversarial examples. arXiv preprint arXiv:1811.01057.

Rayana, S. (2016). ODDS library. Stony Brook University, Department of Computer Sciences, `http://odds.cs.stonybrook.edu`.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. arXiv preprint arXiv:1412.6596.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In International conference on machine learning, pages 4393–4402.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pages 4–11.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I., and Bubeck, S. (2019). Provably robust deep learning via adversarially trained smoothed classifiers. arXiv preprint arXiv:1906.04584.

Scaman, K. and Malherbe, C. (2020). Robustness analysis of non-convex stochastic gradient descent using biased expectations. Advances in Neural Information Processing Systems, 33.

Schmidt, M., Roux, N. L., and Bach, F. R. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. In Advances in neural information processing systems, pages 1458–1466.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. In Proceedings of the

32nd International Conference on Neural Information Processing Systems, pages 6106–6116.

Shah, V., Wu, X., and Sanghavi, S. (2020). Choosing the sample with lowest loss makes sgd robust. arXiv preprint arXiv:2001.03316.

Shen, Y. and Sanghavi, S. (2018). Learning with bad training data via iterative trimmed loss minimization. arXiv preprint arXiv:1810.11874.

Shen, Y. and Sanghavi, S. (2019). Learning with bad training data via iterative trimmed loss minimization. In International Conference on Machine Learning, pages 5739–5748. PMLR.

Song, H., Kim, M., and Lee, J.-G. (2019). Selfie: Refurbishing unclean samples for robust deep learning. In International Conference on Machine Learning, pages 5907–5915. PMLR.

Soranno, P. A., Bacon, L. C., Beauchene, M., Bednar, K. E., Bissell, E. G., Boudreau, C. K., Boyer, M. G., Bremigan, M. T., Carpenter, S. R., Carr, J. W., et al. (2017). Lagos-ne: A multi-scaled geospatial and temporal database of lake ecological context and water quality for thousands of us lakes. GigaScience, 6(12):gix101.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958.

Sun, M., Xing, J., Chen, B., and Zhou, J. (2020). Robust collaborative learning with noisy labels. arXiv preprint arXiv:2012.13670.

Terjék, D. (2019). Adversarial lipschitz regularization. In International Conference on Learning Representations.

Tran, B., Li, J., and Madry, A. (2018). Spectral signatures in backdoor attacks. arXiv preprint arXiv:1811.00636.

Tsybakov, A. B. (2008). Introduction to nonparametric estimation. Springer Science & Business Media.

Tukey, J. W. (1975). Mathematics and the picturing of data. In Proceedings of the International Congress of Mathematicians, Vancouver, 1975, volume 2, pages 523–531.

Turner, A., Tsipras, D., and Madry, A. (2018). Clean-label backdoor attacks.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(Dec):3371–3408.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. (2019a). Neural cleanse:

Identifying and mitigating backdoor attacks in neural networks. In 2019 IEEE Symposium on Security and Privacy (SP), pages 707–723. IEEE.

Wang, S., Zeng, Y., Liu, X., Zhu, E., Yin, J., Xu, C., and Kloft, M. (2019b). Effective End-to-end Unsupervised Outlier Detection via Inlier Priority of Discriminative Network. In Advances in Neural Information Processing Systems, pages 5960–5973.

Weber, M., Xu, X., Karlas, B., Zhang, C., and Li, B. (2020). Rab: Provable robustness against backdoor attacks. arXiv preprint arXiv:2003.08904.

Wei, H., Feng, L., Chen, X., and An, B. (2020). Combating noisy labels by agreement: A joint training method with co-regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13726–13735.

Wong, E. and Kolter, Z. (2018). Provable defenses against adversarial examples via the convex outer adversarial polytope. In International Conference on Machine Learning, pages 5286–5295. PMLR.

Yao, Y., Li, H., Zheng, H., and Zhao, B. Y. (2019). Latent backdoor attacks on deep neural networks. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 2041–2055.

Yi, K. and Wu, J. (2019). Probabilistic end-to-end noise correction for learning with noisy labels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7017–7025.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. (2019). How does disagreement help generalization against label corruption? In International Conference on Machine Learning, pages 7164–7173. PMLR.

Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems, 30(9):2805–2824.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

Zhao, Y., Nasrullah, Z., and Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. arXiv preprint arXiv:1901.01588.

Zheng, S., Wu, P., Goswami, A., Goswami, M., Metaxas, D., and Chen, C. (2020). Error-bounded correction of noisy labels. In International Conference on Machine Learning.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 665–674.

Zisselman, E. and Tamar, A. (2020). Deep residual flow for out of distribution detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13994–14003.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection.