

CERTIFIED AND FORENSIC DEFENSES AGAINST POISONING AND
BACKDOOR ATTACKS

by

ZAYD HAMMOUDEH

A DISSERTATION

Presented to the Department of Computer Science
and the Division of Graduate Studies of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2023

DISSERTATION APPROVAL PAGE

Student: Zayd Hammoudeh

Title: Certified and Forensic Defenses against Poisoning and Backdoor Attacks

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer Science by:

Daniel Lowd	Chair
Thien Nguyen	Core Member
Humphrey Shi	Core Member
Luca Mazzucato	Institutional Representative

and

Krista Chronister	Vice Provost for Graduate Studies
-------------------	-----------------------------------

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded December 2023

© 2023 Zayd Hammoudeh
All rights reserved.

DISSERTATION ABSTRACT

Zayd Hammoudeh

Doctor of Philosophy

Department of Computer Science

December 2023

Title: Certified and Forensic Defenses against Poisoning and Backdoor Attacks

Data poisoning and backdoor attacks manipulate model predictions by inserting malicious instances into the training set. Most existing defenses against poisoning and backdoor attacks are empirical and easily evaded by an adaptive attacker. In addition, existing empirical defenses provide, at best, minimal insights into an attacker’s identity, goals, and methods. In contrast, this work proposes two classes of poisoning and backdoor defenses: (1) *certified defenses*, which provide provable guarantees on their robustness and (2) *forensic defenses*, which provide actionable, human-interpretable insights into an attack’s goals so as to stop the attack via intervention outside the ML system. We focus on certified defenses for regression, where the model predicts a continuous value, and sparse (ℓ_0) attacks, where the adversary controls an unknown subset of the training and test features. Our forensic defense identifies the target of poisoning and backdoor attacks while simultaneously mitigating the attack; we validate our forensic defense on a wide range of data modalities, including speech, text, and vision.

This dissertation includes previously published and unpublished coauthored material.

CURRICULUM VITAE

NAME OF AUTHOR: Zayd Hammoudeh

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR
University of California Santa Cruz, Santa Cruz, CA
San José State University, San Jose, CA
Drexel University, Philadelphia, PA

DEGREES AWARDED:

Doctor of Philosophy, Computer Science, 2023, University of Oregon
Master of Science, Computer Science, 2016, San José State University
Master of Science, Computer Engineering, 2006, Drexel University
Bachelor of Science, Computer Engineering, 2006, Drexel University

AREAS OF SPECIAL INTEREST:

Certified Adversarial Defenses
Training Data Influence Analysis
Data Poisoning
Adversarial Machine Learning
Positive-Unlabeled Learning

PROFESSIONAL EXPERIENCE:

ML Applied Scientist, Qualtrics, 2023–Present
Graduate Researcher, University of Oregon, 2018–2023
Graduate Researcher, University of California Santa Cruz, 2017–2018
Wireless Power Engineer, Integrated Device Technology, 2011–2017
Applications Development Engineer, Teradyne, 2006–2011
Undergrad. Researcher & Teaching Assistant, Drexel University, 2003–2006

GRANTS, AWARDS, AND HONORS:

Gurdeep Pall Graduate Student Fellowship, University of Oregon 2022
J. Donald Hubbard Family Scholarship, University of Oregon, 2021
Travel Award, International Joint Conference on Artificial Intelligence (IJCAI) 2019
Travel Award, SAT Association 2018
Travel Award, Federated Logic Conference (FLoC) 2018
Best Student Paper, International Conference on Theory and Applications of Satisfiability Testing (SAT) 2018
Chancellor’s Fellowship, University of California, Santa Cruz 2017
Undergraduate Student Research Award, Drexel University 2005
Arnold H. Kaplan Stochastic Achievement and Academic Excellence Scholarship, Drexel University 2005
Alvin W. Wene Engineering Scholarship, Drexel University 2004
Teaching Assistant Excellence Award, Drexel University 2004

PUBLICATIONS:

- W. You, Z. Hammoudeh, and D. Lowd. Large Language Models Are Better Adversaries: Exploring Generative Clean-Label Backdoor Attacks Against Text Classifiers”. In: *Findings of the Association for Computational Linguistics*. ELMNLP’23. 2023.
- Z. Hammoudeh** and D. Lowd. Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*, AdvML-Frontiers’23, 2023.
- W. You, **Z. Hammoudeh**, and D. Lowd. Large Language Models are Better Adversaries: Exploring Generative Clean-Label Backdoor Attacks Against Text Classifiers. In *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*, AdvML-Frontiers’23, 2023.
- J. Brophy, **Z. Hammoudeh**, and D. Lowd. Adapting and Evaluating Influence-Estimation Methods for Gradient-Boosted Decision Trees. In: *Journal of Machine Learning Research* vol. 24 (2023), pp. 1–48.

- Z. Hammoudeh** and D. Lowd. Reducing Certified Regression to Certified Classification for General Poisoning Attacks. In *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning, SaTML'23*, 2023.
- Z. Hammoudeh** and D. Lowd. Identifying a Training-Set Attack's Target using Renormalized Influence Estimation. In *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security, CCS'22*, 2022.
- Z. Hammoudeh** and D. Lowd. Simple, Attack-Agnostic Defense Against Targeted Training Set Attacks Using Cosine Similarity. In *Proceedings of the 3rd ICML Workshop on Uncertainty and Robustness in Deep Learning, UDL'21*, 2021.
- Z. Xie, J. Brophy, A. Noack, W. You, K. Asthana, C. Perkins, S. Reis, **Z. Hammoudeh**, D. Lowd, and S. Singh. What Models Know About Their Attackers: Deriving Attacker Information From Latent Representations. In *Proceedings of the 4th BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2021.
- Z. Hammoudeh** and D. Lowd. Learning from Positive and Unlabeled Data with Arbitrary Positive Shift. In *Proceedings of the 34th Conference on Neural Information Processing Systems, NeurIPS'20*, 2020.
- S. Jamshidi, **Z. Hammoudeh**, R. Durairajan, D. Lowd, R. Rejaie, and W. Willinger. On the practicality of learning models for network telemetry. In *Proceedings of the 4th Network Traffic Measurement and Analysis Conference, TMA'20*, 2020.
- Z. Hammoudeh** and D. Lowd. Positive-Unlabeled Learning with Arbitrarily Non-Representative Labeled Data. In *Proceedings of the 37th International Conference on Machine Learning's Workshop on Uncertainty & Robustness in Deep Learning, UDL'20*, 2020.
- D. Achlioptas, **Z. Hammoudeh**, and P. Theodoropoulos. Fast Sampling of Perfectly Uniform Satisfying Assignments. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing, SAT'2018*, 2018. (**Best Student Paper Award**. Authors alphabetical)

Z. Hammoudeh and C. Pollett. Clustering-Based, Fully Automated Mixed-Bag Jigsaw Puzzle Solving. In *Proceedings of 17th International Conference on Computer Analysis of Images and Patterns, CAIP'17*, 2017.

To my mother.

TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION	34
2. PRELIMINARIES	37
2.1. Nomenclature	37
2.2. On Attacker Threat Models	39
2.3. On the Defender Objectives	39
3. RELATED WORK	40
3.1. Defenses Against Evasion Attacks	41
3.1.1. Empirical Defenses	42
3.1.2. Certified Evasion Defenses	43
3.2. Defenses Against Poisoning and Backdoor Attacks	44
3.2.1. Empirical Classification Defenses	45
3.2.2. Certified Pointwise Classifiers	46
3.2.3. Robust Regression	48
3.2.3.1. Resilient Regression	48
3.2.3.2. Certified Regression	49
3.3. Defenses Outside the ML System	50
4. REDUCING CERTIFIED REGRESSION TO CERTIFIED CLASSIFICATION	51
4.1. Preliminaries	52
4.1.1. One-Sided vs. Two-Sided Certification Bounds	53
4.1.2. Relating Regression and Binary Classification	53
4.2. Warmup: Perturbing a Set’s Median	54
4.2.1. Unweighted Swap Paradigm	55

Chapter	Page
4.2.2. Insertion/Deletion Paradigm	55
4.2.3. Weighted Swap Paradigm	57
4.3. Reducing Regression to Voting-Based Binary Classification	58
4.4. Certified Instance-Based Regression	61
4.4.1. Fixed-Population Neighborhood	62
4.4.2. Region-Based Neighborhood	63
4.4.3. Computational Complexity	64
4.5. Certified Regression for General Models	65
4.5.1. Partitioned Certified Regression	65
4.5.2. Weighted Partitioned Certified Regression	67
4.5.3. Computational Complexity	68
4.6. Certified Regression Using Overlapping Training Data	69
4.6.1. Overlapping Certified Regression	69
4.6.2. Weighted Overlapping Certified Regression	71
4.6.3. Computational Cost	72
4.7. Certifying Any Model Beyond Unit Cost	73
4.7.1. Combining Instance-Based Learners & Ensembles	74
4.7.2. Certifying Non-Unit Costs by Construction	74
4.7.3. More Submodels vs. Weighted Costs	76
4.8. Evaluation	76
4.8.1. Experimental Setup	77
4.8.2. Analyzing the Certified Accuracy	79
4.9. Conclusions	81
5. CERTIFIED DEFENSE AGAINST A UNION OF ℓ_0 ATTACKS	85
5.1. Preliminaries	87

Chapter	Page
5.2. Related Work	89
5.3. Certifying Feature Robustness	91
5.3.1. Feature Robustness Under Plurality Voting	92
5.3.2. Feature Robustness Under Run-Off Elections	93
5.3.3. Advantages of Feature Partition Aggregation	96
5.4. Feature Partitioning Strategies	96
5.4.1. Feature Partitioning Paradigms	97
5.5. Evaluation	98
5.5.1. Experimental Setup	98
5.5.2. Main Results	101
5.6. Conclusions	104
6. IDENTIFYING POISONING AND BACKDOOR ATTACK TARGETS WHILE MITIGATING THE ATTACK	105
6.1. Preliminaries	109
6.2. Review of Influence Analysis and Estimation	110
6.3. Why Influence Estimation Often Fails and How to Fix It	115
6.3.1. A Simple Experiment	115
6.3.2. Why Influence Estimation Performs Poorly	116
6.3.3. Renormalizing Influence Estimation	120
6.3.4. Renormalization and More Advanced Attacks	122
6.3.5. Renormalization and Non-Adversarial Data	125
6.4. Identifying Attack Targets	127
6.4.1. Measuring (Renormalized) Influence	128
6.4.2. Identifying Anomalous Influence	130
6.4.3. Target Driven Attack Mitigation	133
6.5. Evaluation	134

Chapter	Page
6.5.1. Training-Set Attacks Evaluated	135
6.5.2. Identifying Adversarial Set \mathcal{D}_{adv}	137
6.5.3. Identifying Attack Targets	139
6.5.4. Target-Driven Mitigation	141
6.6. Adaptive Attacks	142
6.7. Discussion and Conclusions	146
7. CONCLUSIONS AND FUTURE DIRECTIONS	149
 APPENDICES	
A. NOMENCLATURE REFERENCE	152
B. PROOFS	159
B.1. Proofs for Chapter 4	159
B.2. Proofs for Chapter 5	169
B.3. Proof for Chapter 6	174
C. DETAILED EMPIRICAL RESULTS	177
C.1. Chapter 4 Detailed Results	178
C.1.1. Baseline Accuracy	178
C.1.2. Numerical Results	178
C.1.3. k NN-CR Full Certified Accuracy Plots	185
C.2. Chapter 5 Detailed Results	187
C.2.1. Non-Robust Accuracy	187
C.2.2. Detailed Median Certified Robustness Results	188
C.2.3. Feature Partition Aggregation vs. Randomized Ablation Certified Accuracy Detailed Comparison	193
C.2.3.1. Numerical Comparison of Feature Partition Aggregation and Randomized Ablation	194

Chapter	Page
C.2.3.2. Graphical Comparison of Feature Partition Aggregation and Randomized Ablation	199
C.3. Chapter 6 Detailed Results	203
C.3.1. Speech Recognition Backdoor Full Results	203
C.3.2. Vision Backdoor Full Results	205
C.3.3. Natural Language Poisoning Full Results	208
C.3.4. Vision Poisoning Full Results	211
C.4. Convex Polytope Poisoning and GAS Joint Optimization	214
C.4.1. Adversarial-Set Identification of the Jointly Optimized Poisoning Attack	218
C.4.2. Target Identification of the Jointly Optimized Poisoning Attack	220
C.4.3. Target-Driven Attack Mitigation of the Jointly Optimized Poisoning Attack	222
D. EVALUATION SETUPS	223
D.1. Evaluation Setup for the Experiments in Chapter 4	224
D.1.1. Dataset Configuration	224
D.1.2. Dataset Target Value Statistics	225
D.1.3. Hyperparameters	226
D.2. Evaluation Setup for the Experiments in Chapter 5	228
D.2.1. Hardware Setup	228
D.2.2. Baselines	228
D.2.3. Datasets	230
D.2.4. Network Architectures	231
D.2.5. Hyperparameters	233
D.3. Evaluation Setup for the Experiments in Chapter 6	235
D.3.1. Dataset Configurations	235

Chapter	Page
D.3.1.1. Training Set Sizes	237
D.3.1.2. Target Set Sizes	237
D.3.2. Hyperparameters	238
D.3.2.1. Model Training	238
D.3.2.2. Upper-Tail Heaviness Hyperparameters	238
D.3.2.3. Target-Driven Mitigation Hyperparameters	239
D.3.2.4. Adversarial Set \mathcal{D}_{adv} Crafting	240
D.3.2.5. Baselines	242
D.3.3. Network Architectures	244
REFERENCES CITED	247

LIST OF FIGURES

Figure	Page
<p>1. Unweighted Median Perturbation: (1a) Blue denotes elements in subset \mathcal{V}_1, i.e., elements in \mathcal{V} with value at most $\xi = 5.4$. \mathcal{V}_u's values are red. Each “swap” (1b) switches a value in \mathcal{V}_1 with an arbitrarily large replacement. Deletions (1d) and insertions (1c) are interchangeable (suppl. Lemma B.1), with both yielding the same median value in the same number of modifications made to \mathcal{V}. In Figs. 1b to 1d above, any additional modifications to the set would perturb the median.</p>	56
<p>2. Weighted Swap Paradigm: Extension of Fig. 1 to weighted costs. For simplicity and w.l.o.g., let $\mathcal{R} = \{3, \dots, 7\}$, i.e., $\forall_l r_l = \nu_l + 1$. Fig. 2a is identical to Fig. 1a except below each element ν_l is its corresponding weight r_l. Observe $\Delta = 1$ and $\tilde{\mathcal{R}}_1 = \{3, 4\}$. Fig. 2b shows that for $R = 6$ (visualized below each element), it is impossible to perturb the median, and any additional weight would be sufficient to swap out $\nu_2 = 3$.</p>	57
<p>3. Certified Regression to Certified Classification Reduction: For $\mathbf{x}_{te} \in \mathcal{X}$, the decision function is $f(\mathbf{x}_{te}) := \text{med } \mathcal{V}$ – just like voting-based certified classification. Certified regression binarizes \mathcal{V} into $\mathcal{V}_{\pm 1}$, which is used by the robustness certifier (optionally with weights \mathcal{R}) to determine R.</p>	59
<p>4. Certified Instance-Based Regression: Fig. 4a visualizes an unperturbed IBL model. Test instance \mathbf{x}_{te}'s neighborhood is visualized as a dashed line with neighborhood $\mathcal{N}(\mathbf{x}_{te})$ identical to \mathcal{V} in Fig. 1a. Fig. 4b shows an attack on a kNN-m model where the neighborhood's cardinality ($L = 5$) is fixed, and the one attack instance (\blacksquare) replaces one instance in \mathcal{V}_1 (\circ) (source Fig. 1b). A rNN-median model is shown in Fig. 4c, where the two inserted instances (\blacksquare) do not change the neighborhood's radius (source Fig. 1c).</p>	63
<p>5. Overlapping Certified Ensemble: Simple visualization of the ensemble architecture for (weighted) overlapping certified regression. Function h_{tr} partitions training set \mathcal{D} into ($m = 7$) blocks. Function h_f defines each of the $L = 5$ submodel training sets, $\mathcal{D}_1, \dots, \mathcal{D}_5$. The ensemble prediction is the median submodel prediction, i.e., $f(\mathbf{x}_{te}) := \text{med } \{f_1(\mathbf{x}_{te}; 1), \dots, f_L(\mathbf{x}_{te}; L)\}$.</p>	68

6. **Overlapping Certified Regression Integer Linear Program:** Adapted from the partial set (multi)cover integer linear program. Calculates certified robustness R for both OCR and W-OCR with indicator variable σ adjusting the program to account for weighted costs. For arbitrary feature vector \mathbf{x}_{te} , \mathcal{T}_1 is the set of submodels that predict $f_l(\mathbf{x}_{te}) \leq \xi$. Variable $\omega^{(j)}$ contains the number of modifications made to training set block $D^{(j)}$. Binary variable $\delta_l = 1$ if submodel f_l has been sufficiently modified for $f_l(\mathbf{x}_{te}) > \xi$ and 0 otherwise. 73
7. **Certified Accuracy:** Mean certified accuracy (larger is better) for our five primary certified regressors. k NN-CR is always trained on all of training set \mathcal{D} (i.e., $q = 1$). Ensemble submodels are trained on $\frac{1}{q}$ -th of \mathcal{D} , with three q values tested per dataset. The x-axis is clipped to enhance readability; see suppl. Sec. C.1.3 for k NN-CR’s full results. The best performing method depends on the target certified robustness R . For smaller R values, W-OCR achieves the best certified accuracy. For larger R values, k NN-CR outperforms the ensemble methods. This result aligns with previous findings on certified classification [Jia+22a]. Sec. 4.8.2 summarizes these experiments’ primary takeaways. *Figure continued on the next page.* 83
7. **Certified Accuracy (cont.):** Mean certified accuracy (larger is better) for our five primary certified regressors. k NN-CR is always trained on all of training set \mathcal{D} (i.e., $q = 1$). Ensemble submodels are trained on $\frac{1}{q}$ -th of \mathcal{D} , with three q values tested per dataset. The x-axis is clipped to enhance readability; see suppl. Sec. C.1.3 for k NN-CR’s full results. The best performing method depends on the target certified robustness R . For smaller R values, W-OCR achieves the best certified accuracy. For larger R values, k NN-CR outperforms the ensemble methods. This result aligns with previous findings on certified classification [Jia+22a]. Sec. 4.8.2 summarizes these experiments’ primary takeaways. See Sec. C.1 for the numerical results, including variance. 84

8. **Feature partition aggregation example** prediction for: test instance $\mathbf{x} \in \mathcal{X}$, $n = 3$, $d = 4$, and $|\mathcal{Y}| = 3$. Feature partitioning across $L = 4$ submodels, where the l -th submodel uses only feature dimensions $\mathcal{S}_l = \{l\} \subset [4]$ and training set D_l , i.e., the tuple containing the l -th column of feature matrix \mathbf{X} (denoted \mathbf{X}_l) and label vector $\mathbf{y} := [y_1, y_2, y_3]$. $\mathbf{x}_{\mathcal{S}_l}$ denotes the subvector of \mathbf{x} restricted to the feature dimensions in \mathcal{S}_l . Plurality label $y_{\text{pl}} = 0$; runner-up label $y_{\text{ru}} = 1$; and run-off label $y_{\text{RO}} = 0$. Under the plurality voting decision function (Sec. 5.3.1), $f(\mathbf{x})$ has certified feature robustness $R_{\text{pl}} = 0$. With run-off (Sec. 5.3.2), $f(\mathbf{x})$'s certified feature robustness is $R_{\text{RO}} = 1$ 89

9. **Renormalized Influence: CIFAR10 & MNIST joint, binary classification** for [frog] vs. [airplane & MNIST 0] with $|\mathcal{D}_{\text{cl}}| = 10,000$ & $|\mathcal{D}_{\text{adv}}| = 150$. Existing influence estimators (upper half) consistently failed to rank \mathcal{D}_{adv} 's MNIST training instances as highly influential on MNIST test instances. In contrast, all of our renormalized influence estimators (Section 6.3.3) outperformed their unnormalized version – with AUPRC improving up to $25\times$. Results averaged across 30 trials. 111

10. **CIFAR10 and MNIST Intra-training Loss Tracking:** \mathcal{D}_{adv} 's (-) and \mathcal{D}_{cl} 's (-) median cross-entropy losses (\mathcal{L}) at each training checkpoint for binary classification – frog vs. airplane & MNIST 0. The shaded regions correspond to each training set loss's interquartile range. MNIST's training losses are generally several orders of magnitude smaller than CIFAR10's losses. Gradient norm ratio (-) shows the tight coupling of loss and training gradient magnitude. 117

11. **Layerwise Decomposition of an Attack Target's Intra-Training Gradient Magnitude:** One-pixel and blend backdoor adversarial triggers (dashed and solid lines respectively) trained separately on CIFAR10 binary classification ($y_{\text{targ}} = \text{airplane}$ and $y_{\text{adv}} = \text{bird}$) using ResNet9. The network's first convolutional (Conv1) and final linear layers are a small fraction of the parameters (0.03% and 0.01% resp.) but constitute most of the target's gradient magnitude ($\|\hat{\mathbf{g}}_{\text{targ}}\|$) with the dominant layer attack dependent. Results are averaged over 20 trials. 124

12. **Effect of Removing Influential, Non-Adversarial Training Data:** Test example z_{filt} 's misclassification rate (larger is better) when filtering the training set using influence rankings based on influence functions (top) and TracIn (bottom). Renormalization (Rn.) always improved mean performance across all training set filtering percentages. Results are averaged across five CIFAR10 class pairs with 30 trials per class pair and 20 models trained per method per trial. Results are separated by the reference influence estimator. 126
13. GAS renormalized influence, \mathbf{v} , density distributions for two training set attacks: CIFAR10 vision poisoning [Zhu+19] ($y_{\text{targ}} = \text{dog}$ and $y_{\text{adv}} = \text{bird}$) and speech-recognition backdoor [Liu+18] ($y_{\text{targ}} = 4$ and $y_{\text{adv}} = 5$). Theoretical normal (\cdot) is w.r.t. $\mathcal{D} := \mathcal{D}_{\text{adv}} \cup \mathcal{D}$. Observe that target examples (Figs. 13a and 13d) have significant \mathcal{D}_{adv} mass ($-$) well to the right of \mathcal{D}_{cl} 's mass ($-$). This upper-mass phenomenon is absent in non-targets (Figs. 13b and 13e). Training example gradient norms (Fig. 13c and 13f) are poorly correlated with whether the training example is adversarial. For example, speech recognition has \mathcal{D}_{cl} mass well to the right of even the right-most \mathcal{D}_{adv} mass, necessitating renormalization. See Sections 6.5.1 and D.3 for more details on these attacks. 129
14. **Adversarial-Set Identification:** Mean AUPRC identifying adversarial set \mathcal{D}_{adv} using a randomly selected target for Sec. 6.5.1's four attacks. Results averaged across related setups with ≥ 10 trials per setup. See supplemental Section C.3 for the full granular results. 138
15. **Static Influence Adversarial-Set Identification:** Comparing the mean adversarial-set identification AUPRC of the static influence estimators and their corresponding renormalized (Rn.) versions. For all attacks, renormalization improved the static estimators' mean performance by up to a factor of $>600\times$. These experiments also highlight layerwise renormalization's performance gains, e.g., influence functions on natural-language poison. Results are averaged across related experimental setups with ≥ 10 trials per setup. 139
16. **Target Identification:** Mean target identification AUPRC for Sec. 6.5.1's four attacks. "FIT w/ GAS" denotes GAS was FIT's influence estimator with matching notation for GAS-L. Results averaged across setups with ≥ 10 trials per setup. See Sec. C.3 for the full granular results. 140

17.	Adversarial-Set Identification for the Adaptive Vision Poison Attack: Mean AUPRC identifying the adversarial set where Zhu et al.’s vision poison attack is adapted to jointly minimize the adversarial loss and the GAS influence. The baseline results (orange) used Zhu et al.’s standard attack. Our jointly-optimized attack reduced the GAS similarity by 7% at the cost of a 19% decrease in ASR w.r.t. Table 6. See suppl. Sec. C.4 for the granular results.	148
18.	Target Identification for the Adaptive Vision Poison Attack: Mean target identification AUPRC where Zhu et al.’s vision poison attack is jointly optimized with minimizing GAS. FIT with GAS’s mean target identification AUPRC declined only 9% versus the baseline – an average change in target rank of 1.16 to 1.28 – still strong performance. Results are averaged across related setups with ≥ 10 trials per setup. See suppl. Sec. C.4 for the full results.	148
C.19.	kNN-CR vs. W-OCR Certified Accuracy: Full plots of the mean certified accuracy for Sec. 4.8’s six datasets. The shaded regions visualize one standard deviation of the certified accuracy for each R value. W-OCR’s q value for each dataset is in Table C.19.	186
C.20.	Classification certified accuracy envelope for datasets CIFAR10 ($d = 1024$) and MNIST ($d = 784$) for feature partition aggregation (FPA) and baseline randomized ablation (RA). Each method’s envelope considers the corresponding hyperparameters in Tables C.25 and C.26, emulating a certified defense where the hyperparameters are roughly tuned to maximize the certified accuracy at each robustness level. Subfigures C.20a and C.20b visualize each method’s certified accuracy envelope (larger is better); also shown in these subfigures is a naive baseline where the decision function always predicts label $f(\mathbf{x}) = 1$. Subfigures C.20c and C.20d visualize the improvement in certified accuracy when using FPA with the run-off decision function over the two randomized ablation baselines from Levine and Feizi [LF20b] and Jia et al. [Jia+22b]. The envelope plots’ underlying numerical values are provided in Table C.25 for CIFAR10 and Table C.26 for MNIST.	201

C.21. Regression certified accuracy envelope for the Weather [Mal+21] ($d = 128$) and Ames [Coc11] ($d = 352$) datasets for feature partition aggregation (FPA) and baseline randomized ablation (RA). Each method’s envelope considers the corresponding hyperparameters in Tables C.27 and C.28, emulating a certified defense where the hyperparameters are tuned to maximize each robustness level’s certified accuracy. Subfigures C.21a and C.21b visualize each method’s certified accuracy envelope (larger is better); also shown in these subfigures is a naive baseline that always predicts the median training data target value. Subfigures C.21c and C.21d visualize the improvement in certified accuracy when using FPA (with plurality voting) as the decision function over the two randomized ablation baselines from Levine and Feizi [LF20b] and Jia et al. [Jia+22b]. FPA outperforms randomized ablation for smaller certified robustness values, while Jia et al.’s [Jia+22b] version of RA marginally outperformed both FPA and the naive baseline at larger robustness values. The envelope plots’ underlying numerical values are provided in Table C.27 for Weather and Table C.28 for Ames.	202
C.22. Speech Backdoor Adversarial Set Identification: Mean backdoor set (\mathcal{D}_{adv}) identification AUPRC across 30 trials for all 10 class pairs with $21 \leq \mathcal{D}_{\text{adv}} \leq 28$ (varies by class pair, see Tab. D.57). GAS and GAS-L outperformed all baselines in all experiments, with GAS-L the overall top performer on 6/10 class pairs. See Table C.29 for the numerical results.	203
C.23. Speech Backdoor Target Identification: See Table C.30 for numerical results.	204
C.24. Vision Backdoor Adversarial-Set Identification: Backdoor set, \mathcal{D}_{adv} , identification mean AUPRC across >30 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns with a randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $ \mathcal{D}_{\text{adv}} = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. See Table C.32 for the numerical results.	205
C.25. Vision Backdoor Target Identification: Mean target identification AUPRC across 15 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns and randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $ \mathcal{D}_{\text{adv}} = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. See Table C.33 for the numerical results.	206

Figure	Page
C.26. Natural Language Poisoning Adversarial-Set Identification: See Table C.35 for the numerical results.	208
C.27. Natural Language Poisoning Target Identification: See Table C.36 for the numerical results.	209
C.28. Vision Poisoning Adversarial-Set Identification: Adversarial set (\mathcal{D}_{adv}) identification mean AUPRC across >15 trials for four CIFAR10 class pairs with $ \mathcal{D}_{\text{adv}} = 50$. Our renormalized influence estimators, GAS and GAS-L, using just initial parameters θ_0 and with 5 subepoch checkpointing outperformed all baselines for all class pairs.	211
C.29. Vision Poisoning Target Identification: See Table C.39 for the numerical results.	213
C.30. Adversarial-Set Identification for the Adaptive Vision Poison Attack: Mean AUPRC identifying the adversarial set where Zhu et al.’s vision poison attack is jointly optimized with minimizing GAS with ≥ 10 trials per setup as described in Section C.4. Section 6.6’s baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. This joint optimization reduces the GAS similarity by 7% at the cost of a 19% decrease in ASR w.r.t. Table 6. See Table C.42 (below) for the numerical results.	219
C.31. Target Identification for the Adaptive Vision Poison Attack: Mean target identification AUPRC where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing GAS. Section 6.6’s baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. See Table C.43 (below) for the numerical results.	221

LIST OF TABLES

Table		Page
1.	<p>Evaluation Dataset Summary: Training set size (n), data dimension, overlapping spread degree (d), error threshold (ξ), and submodel architecture for the six datasets. Error thresholds that are a percentage of each instance’s true target value are denoted $X\% \cdot y$. Alternate ξ values are evaluated in the original paper [HL23c, Fig. 9].</p>	79
2.	<p>Median certified robustness. Each dataset’s best performing method is in bold. Our median robustness was 20–30% larger for classification and 3 to 4× larger for regression while simultaneously providing stronger guarantees. For detailed results, see Sec. C.2.2.</p>	101
3.	<p>Classification accuracy (% – larger is better). We report FPA’s accuracy at both RA’s (middle, bold) and FPA’s (blue) best median robustness levels. At RA’s best median robustness, FPA had better classification accuracy for all four datasets. For full results, see Sec. C.2.2.</p>	101
4.	<p>CIFAR10 certified patch accuracy (% – larger is better) for FPA, RA, and three dedicated patch defenses. FPA is competitive despite making fewer assumptions and providing stronger guarantees than patch defenses.</p>	103
5.	<p>Mean certification time in seconds for FPA and Jia et al.’s [Jia+22b] randomized ablation (RA). FPA is 2 to 3 orders of magnitude faster than baseline RA.</p>	103
6.	<p>Target Driven Attack Mitigation: Alg. 6’s target-driven, iterative data sanitization applied to Sec. 6.5.1’s four attacks for randomly selected targets. The attacks were neutralized with few clean instances removed and little change in test accuracy. Attack success rate (ASR) is w.r.t. the analyzed target. Results are averaged across related setups with ≥ 10 trials per setup. Detailed results appear in Sec. C.3.1–C.3.4.</p>	143

Table	Page
7. Attack Mitigation for the Adaptive Vision Poison Attack: Algorithm 6’s target-driven data sanitization where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing the GAS influence. The results below consider exclusively the jointly-optimized attack with $\beta = 10^{-2}$. Clean-data removal remains low, and test accuracy either improved or stayed the same for in but one setup. The performance is comparable to the results with Zhu et al.’s [Zhu+19]’s standard vision poisoning attack (see Table C.40). Bold denotes the best mean performance with ≥ 10 trials per class pair.	146
A.8. General Nomenclature Reference: This table contains symbols that are relevant to one or more chapters in this dissertation. Related symbols are grouped together with groups separated by dotted lines.	152
A.9. Chapter 4 Nomenclature Reference: Notation specific to Chapter 4 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages. . .	153
A.9. Chapter 4 Nomenclature Reference (Continued): Notation specific to Chapter 4 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.	154
A.10. Chapter 5 Nomenclature Reference: Notation specific to Chapter 5 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages. . .	155
A.10. Chapter 5 Nomenclature Reference (Continued): Notation specific to Chapter 5 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.	156
A.11. Chapter 6 Nomenclature Reference: Notation specific to Chapter 6 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages. . .	157
A.11. Chapter 6 Nomenclature Reference (Continued): Notation specific to Chapter 6 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.	158

- C.12. **Baseline Accuracy:** Summary of the baseline (i.e., uncertified) accuracy mean and standard deviation for Sec. 4.8’s six datasets. Submodels were trained on all of training set \mathcal{D} (i.e., $q = 1$). Beside each dataset’s name is the submodel architecture used by the ensemble. Threshold ξ matches values in Table 1. 178
- C.13. **Ames Housing Full Results:** Certified accuracy mean and standard deviation for the Ames Housing [Coc11] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R ’s best mean certified accuracy in **bold**. 179
- C.14. **Austin Housing Full Results:** Certified accuracy mean and standard deviation for the Austin Housing [Pie21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R ’s best mean certified accuracy in **bold**. 180
- C.15. **Diamonds Full Results:** Certified accuracy mean and standard deviation for the Diamonds [Wic16] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R ’s best mean certified accuracy in **bold**. 181

- C.16. **Weather Full Results:** Certified accuracy mean and standard deviation for the Weather [Mal+21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**. 182
- C.17. **Life Full Results:** Certified accuracy mean and standard deviation for the Life [Raj21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**. . . . 183
- C.18. **Spambase Full Results:** Certified accuracy mean and standard deviation for the Spambase [Hop+17] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**. 184
- C.19. **W-OCR q Values:** As detailed in Sec. 4.8.1, ensemble submodels were trained on $\frac{1}{q}$ -th of the training data where q varies by dataset. Below are the W-OCR q values used in Fig. C.19. 185
- C.20. **Non-Robust Accuracy:** Prediction accuracy when training a single model on all model features, i.e., $L = 1$. These values represent an upper bound on the potential accuracy of our method given the training set, model architecture, and hyperparameters. . . . 187

C.21. CIFAR10 Detailed Results: Classification accuracy (%) and median certified robustness (larger is better) for the CIFAR10 [KNH14] dataset ($d = 1024$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. Each certification method’s hyperparameter setting with the best median robustness is shown in bold . The best overall median robustness is shown in blue	189
C.22. MNIST Detailed Results: Classification accuracy (%) and median certified robustness (larger is better) for the MNIST [LeC+98] dataset ($d = 784$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. Each certification method’s hyperparameter setting with the best median robustness is shown in bold . The best overall median robustness is shown in blue	190
C.23. Weather Detailed Results: Classification accuracy (%) and median certified robustness (larger is better) for the Weather [Mal+21] dataset ($d = 128$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. FPA considers only plurality voting-based certification (Sec. 5.3.1) since the reduction is from certified regression to certified <i>binary</i> classification. FPA results are reported using both GBDTs [Ke+17] and linear submodels. Median robustness “ $-\infty$ ” denotes that the classification accuracy was less than 50%. Each approach’s hyperparameter setting with the best median robustness is shown in bold . The best overall median robustness is shown in blue	191
C.24. Ames Detailed Results: Classification accuracy (%) and median certified robustness (larger is better) for the Ames [Coc11] dataset ($d = 352$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. FPA considers only plurality voting-based certification (Sec. 5.3.1) since the reduction is from certified regression to certified <i>binary</i> classification. FPA results are reported using both GBDTs [Ke+17] and linear submodels. Median robustness “ $-\infty$ ” denotes that the classification accuracy was less than 50%. Each approach’s hyperparameter setting with the best median robustness is shown in bold . The best overall median robustness is shown in blue	192

C.25. CIFAR10 ($d = 1024$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “Plurality” denotes FPA with plurality voting as the decision function while “Run-Off” denotes FPA using run-off elections as the decision function. “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. We also consider an additional naive baseline that always predicts $f(\mathbf{x}) = 1$. For each certified robustness level, each method’s best performing hyperparameter setting is shown in bold with the overall best performing method shown in blue	195
C.26. MNIST ($d = 784$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “Plurality” denotes FPA with plurality voting as the decision function while “Run-Off” denotes FPA using run-off elections as the decision function. “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. We also consider an additional naive baseline that always predicts $f(\mathbf{x}) = 1$. For each certified robustness level, each method’s best performing hyperparameter setting is shown in bold with the overall best performing method shown in blue	196
C.27. Weather [Mal+21] dataset ($d = 128$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. Hammoudeh and Lowd’s [HL23c] reduction is from certified regression to certified binary classification. Run-off is identical to plurality voting under binary classification, so we report only the plurality voting results below. We also consider an additional naive baseline that always predicts the median training set target value (i.e., $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$). For each certified robustness level, each method’s best performing hyperparameter setting is shown in bold with the overall best performing method shown in blue . These numerical results are visualized graphically as envelope plots in Figure C.21.	197

C.28. Ames [Coc11] dataset ($d = 352$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. Hammoudeh and Lowd’s [HL23c] reduction is from certified regression to certified binary classification. Run-off is identical to plurality voting under binary classification, so we report only the plurality voting results below. We also consider an additional naive baseline that always predicts the median training set target value (i.e., $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$). For each certified robustness level, each method’s best performing hyperparameter setting is shown in bold with the overall best performing method shown in blue . These numerical results are visualized graphically as envelope plots in Figure C.21.	198
C.29. Speech Backdoor Adversarial Set Identification: Mean AUPRC across 30 trials for speech backdoor dataset [Liu+18] with $21 \leq \mathcal{D}_{\text{adv}} \leq 28$. GAS(-L) always outperformed the baselines. Bold denotes the best mean performance. Mean results are shown graphically in Figs. 14 and C.22. Variance results appear in the original paper [HL22a, Sec. F.1.1].	203
C.30. Speech Backdoor Target Identification: Bold denotes the best mean performance. Mean results are shown graphically in Figures 16 and C.23. Variance results appear in the original paper [HL22a, Sec. F.1.1].	204
C.31. Speech Backdoor Attack Mitigation: Bold denotes the best mean performance with 10 trials per class pair. Aggregated results are shown in Table 6.	204
C.32. Vision Backdoor Adversarial-Set Identification: Backdoor set, \mathcal{D}_{adv} , identification mean AUPRC across >30 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns with a randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $ \mathcal{D}_{\text{adv}} = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. Bold denotes the best mean performance. Mean results are shown graphically in Figures 14 and C.24. Variance results appear in the original paper [HL22a, Sec. F.1.2].	205

Table	Page
C.33. Vision Backdoor Target Identification: Target identification mean AUPRC across 15 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns and randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. Bold denotes the best mean performance. Mean results are shown graphically in Figures 16 and C.25. Variance results appear in the original paper [HL22a, Sec. F.1.2].	207
C.34. Vision Backdoor Attack Mitigation: Bold denotes the best mean performance with 15 trials per setup. Aggregated results are shown in Table 6.	207
C.35. Natural Language Poisoning Adversarial-Set Identification: Poison identification mean AUPRC across 10 trials for 4 positive and 4 negative sentiment SST-2 movie reviews [Soc+13] with $ \mathcal{D}_{\text{adv}} = 50$. GAS-L perfectly identified all poison in all but one trial. Bold denotes the best mean performance. Mean results are shown graphically in Figures 14 and C.26. Variance results appear in the original paper [HL22a, Sec. F.1.3].	208
C.36. Natural Language Poisoning Target Identification: Bold denotes the best mean performance with 10 trials per review. Mean results are shown graphically in Figures 16 and C.27. Variance results appear in the original paper [HL22a, Sec. F.1.3].	210
C.37. Natural Language Poisoning Attack Mitigation: Bold denotes the best mean performance with 10 trials per review. Aggregated results are shown in Table 6.	210
C.38. Vision Poisoning Adversarial-Set Identification: Adversarial set (\mathcal{D}_{adv}) identification mean AUPRC across >15 trials for four CIFAR10 class pairs with $ \mathcal{D}_{\text{adv}} = 50$. Our renormalized influence estimators, GAS and GAS-L, using just initial parameters θ_0 and with 5 subepoch checkpointing outperformed all baselines for all class pairs. Bold denotes the best mean performance. Mean results are shown graphically in Figure 14 and C.28. Variance results appear in the original paper [HL22a, Sec. F.1.4].	212
C.39. Vision Poisoning Target Identification: Bold denotes the best mean performance with ≥ 15 trials per class pair. Mean results are shown graphically in Figures 16 and C.29. Variance results appear in the original paper [HL22a, Sec. F.1.4].	213

Table	Page
C.40. Vision Poisoning Attack Mitigation: Bold denotes the best mean performance with ≥ 15 trials per class pair. Aggregated results are shown in Table 6.	213
C.41. Effect of joint-optimization hyperparameter β on the attacker’s success rate (ASR). Observe that even at $\beta = 0$, the attack success rate is significantly lower than the 77.9% ASR in Table 6 due to the fewer surrogate models that could be used during jointly-optimized poison crafting as explained above.	217
C.42. Adversarial-Set Identification for the Adaptive Vision Poison Attack: Adversarial-set identification mean AUPRC with ≥ 10 trials per setup as described in Section C.4. Section 6.6’s baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. Bold denotes the best mean performance. Mean results are shown graphically in Figures 17 and C.30. Variance results appear in the original paper [HL22a, Sec. F.2.1].	218
C.43. Target Identification for the Adaptive Vision Poison Attack: Target identification mean AUPRC where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing GAS. Section 6.6’s baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. Bold denotes the best mean performance with ≥ 10 trials per class pair. Mean results are shown graphically in Figures 18 and C.31. Variance results appear in the original paper [HL22a, Sec. F.2.2].	220
C.44. Target-Driven Attack Mitigation for the Adaptive Vision Poison Attack: Algorithm 6’s target-driven data sanitization where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing the GAS influence. The results below consider exclusively the jointly-optimized attack with $\beta = 10^{-2}$. Clean-data removal remains low, and test accuracy either improved or stayed the same for in but one setup. The performance is comparable to the results with Zhu et al.’s [Zhu+19]’s standard vision poisoning attack (see Table C.40). Bold denotes the best mean performance with ≥ 10 trials per class pair.	222
D.45. Target Value Test Distribution Statistics: Mean (\bar{y}), standard deviation (σ_y), minimum value (y_{\min}) and maximum value (y_{\max}) for the test instances’ target y value for Sec. 4.8’s five regression datasets.	225

Table	Page
D.46. Ridge Regression Hyperparameters: Hyperparameter settings for the three datasets that used ridge regression as the ensemble submodel architecture. Hyperparameters are reported for the three q values used in Fig. 7 and Sec. C.1. We also report the hyperparameters for uncertified accuracy when $q = 1$	228
D.47. XGBoost Hyperparameters: Hyperparameter settings for the three datasets that used XGBoost as the ensemble submodel architecture. Hyperparameters are reported for the three q values used in Fig. 7 and Sec. C.1. We also report the hyperparameters for uncertified accuracy when $q = 1$	229
D.48. Evaluation dataset information	231
D.49. Target Value Test Distribution Statistics: Mean (\bar{y}), standard deviation (σ_y), minimum value (y_{\min}) and maximum value (y_{\max}) for the test instances' target y value for regression datasets Weather and Ames.	231
D.50. ResNet9 neural network architecture	232
D.51. Network-in-Network neural network architecture	233
D.52. FPA's neural network training hyperparameters	234
D.53. Regression datasets LightGBM submodel training hyperparameters	234
D.54. Regression datasets linear submodel training hyperparameters	235
D.55. SST-2 movie reviews selected by Wallace et al.'s [Wal+21] poisoning attack implementation.	236
D.56. Chapter 6 target identification dataset sizes	237
D.57. Number of backdoor training examples for each speech backdoor digit pair. As detailed above, Liu et al.'s [Liu+18] dataset provides 30 backdoored instances for each digit pair. The remainder of the 30 instances for each digit pair are part of the fixed, validation set.	237
D.58. Target and non-target set sizes used in Section 6.5.3's target identification experiments.	238
D.59. Renormalized influence model training hyperparameter settings	238
D.60. Training-set attack model training hyperparameter settings	239
D.61. Upper-tail heaviness cutoff count (κ)	239

Table	Page
D.62. Target-driven attack mitigation hyperparameters	240
D.63. CIFAR10 vision backdoor adversarial trigger maximum ℓ_2 -norm perturbation distance	241
D.64. Convex polytope poison crafting [Zhu+19] hyperparameter settings . .	242
D.65. SST-2 sentiment analysis poison crafting hyperparameter settings. These are identical to Wallace et al.'s [Wal+21] hyperparameter settings.	242
D.66. Influence functions hyperparameter settings	243
D.67. Simplified ResNet9 neural network architecture used for Sec. 6.5's CIFAR10 binary classification	245
D.68. Speech recognition convolutional neural network	246

CHAPTER 1

INTRODUCTION

Machine learning systems are increasingly being applied in domains critical to human safety and well-being [HL22b; Awa+18]. Simultaneously, algorithmic decisions are increasingly black boxes where the factors that led to a model prediction are not human interpretable – in particular for neural networks. Exacerbating this is neural networks’ propensity to learn spurious correlations or “shortcuts” [DAm+20; Gei+20]. Robust machine learning models are urgently needed because *when* (not if) today’s brittle models fail, society will have to carry the burden of that failure.

Spurious correlations occur naturally in most training data [Ily+19] and are often non-malicious [Fel20]. For example, model misbehavior can arise due to training outliers drawn from the tails of the data distributions. Similarly, measurement or labeling noise can also introduce spurious relationships into the training set. Rather than focusing on these disparate causes of spurious training data, this dissertation focuses on *adversarial attacks*, where an adversary introduces or exploits *worst-case* spurious correlations [Car+23]. Making a model robust against worst-case training modifications simultaneously makes the model robust against less severe (e.g., benign) training set issues, including those mentioned above. Today’s models are highly susceptible to numerous different types of adversarial attacks [Li+22; LXL23; Kum+20]. However, defenses against adversarial attacks remain relatively primitive and “lack fundamental security rigor” [Kum+20]. The current defense landscape has even been likened to “crypto Pre-Shannon” [Car19].

This dissertation focuses on two related types of adversarial attacks. First, *poisoning attacks* manipulate model predictions on pristine or “natural” test instances by adversarially modifying the training sets. Second, *backdoor attacks* manipulate

predictions by combining perturbations to the training set and with perturbations to the test set. A recent survey of governmental and corporate organizations [Kum+20] found that poisoning and backdoor attacks were the first and third biggest ML security concerns, respectively, due to previously successful attacks [Lee16; Mur16]. Where applicable, this dissertation also considers *evasion attacks*, which adversarially perturb only test instances.

This dissertation proposes three novel defenses to make ML systems more robust against poisoning and backdoor attacks. We focus on two primary strategies to stop an attacker. We first consider two *certified defenses* which provide guaranteed robustness given a specific *threat model* – i.e., definition of the attacker’s capability. We then propose a *forensic defense* that provides insights into the identity, goals, and methods of an attacker so as to stop the attack via intervention outside the ML system. In practice, these two types of defenses are complementary and can be deployed together to enhance their effectiveness.

Below we briefly summarize this dissertation’s primary contributions. Chapters 4 to 6 each provide a more detailed list of the chapter’s corresponding contributions.

- A reduction from certified regression to certified classification (Chapter 4). Our reduction allows regression tasks to directly reuse methods that were previously used only for classification.
- A unified, certified defense against sparse¹ (ℓ_0) poisoning, backdoor, and evasion attacks (Chapter 5) – ℓ_0 or otherwise. To the extent of our knowledge, our method is the first to provide non-trivial guarantees over this union of attack types.

¹A “*sparse*” or ℓ_0 attacker arbitrarily controls an unknown subset of the training and/or test features [Sch+19; LF21; Jia+22b].

- A defense that simultaneously identifies the target(s) of poisoning and backdoor attacks while also mitigating the attack (Chapter 6).

Note that all proofs appear in the appendix (Chapter B).

Before detailing our specific contributions, Chapter 2 first reviews general nomenclature. Chapter 3 then reviews related defenses against adversarial attacks.

Note that Chapters 3, 4, 5, and 6 as well as Appendices B, D, and C contained published and unpublished material coauthored with Daniel Lowd.

CHAPTER 2

PRELIMINARIES

This section introduces our primary nomenclature and includes a brief discussion of the attacker threat models and defender objectives.

2.1 Nomenclature

In cases where a specific chapter uses specialized nomenclature, the custom notation is introduced at the beginning of that chapter. See Chapter A in the appendix for a full nomenclature reference.

Let $[a]$ denote integer set $\{1, \dots, a\}$, and denote the corresponding *power set* $2^{[a]}$. $\mathbb{1}[q]$ is the *indicator function* which equals 1 if predicate q is true and 0 otherwise. Let $H(a) := \sum_{i=1}^a \frac{1}{i}$ denote the a -th *harmonic number*. Denote (multi)set A 's *median* as $\text{med } A$. In cases where A 's cardinality is even, the median is the midpoint between A 's $\frac{|A|}{2}$ -th and $(\frac{|A|}{2} + 1)$ -th largest values.

Let $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ denote a *feature vector*, where $d := |\mathbf{x}|$ denotes the *feature dimension*. The *feature set* is denoted $[d]$. $y \in \mathcal{Y} \subseteq \mathbb{R}$ denotes a dependent *target value*; we consider both discrete and continuous target values. Let $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ denote the *instance space*. *Training set* $\mathcal{D} := \{z_i\}_{i=1}^n \subset \mathcal{Z}^n$ consists of n instances where the i -th training instance is tuple $z_i := (\mathbf{x}_i, y_i)$.

Model $f : \mathcal{X} \rightarrow \mathcal{Y}$ is trained on \mathcal{D} . Given arbitrary test instance $(\mathbf{x}_{\text{te}}, y_{\text{te}})$, the model prediction is denoted $\hat{y}_{\text{te}} := f(\mathbf{x}_{\text{te}})$. This dissertation considers both ensemble and singleton models. We introduce the notation for these two types of models below.

Singleton Model We consider both parametric and non-parametric singleton models. In the case of parametric models, let $\theta \in \mathbb{R}^p$ denote f 's *model parameters*. Let $f(\mathbf{x}; \theta)$ denote a parameterized prediction for $\mathbf{x} \in \mathcal{X}$. We usually drop parameter vector θ for brevity and to enhance readability.

Parametric models are trained using any iterative, first-order optimization algorithm (e.g., gradient descent, Adam [KB15]). Let $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ denote the *loss function*. Denote instance z 's empirical *risk* w.r.t. θ as $\mathcal{L}(z; \theta) := \mathcal{L}(f(\mathbf{x}; \theta), y)$. θ_0 denotes f 's initial parameters with θ_0 randomly set and/or pre-trained. During each training iteration $t \in \{1, \dots, T\}$, the optimizer updates parameters θ_t using loss \mathcal{L} , parameters θ_{t-1} , and batch $\mathcal{B}_t \subseteq \mathcal{D}$, where $b := |\mathcal{B}_t|$. Gradients are denoted $g_i^{(t)} := \nabla_{\theta} \mathcal{L}(z_i; \theta_t)$; the gradient's superscript “(t)” is dropped when the iteration is clear from context.

Ensemble Model In cases where f is an ensemble, let L denote the number of submodels, where $f_l : \mathcal{X} \rightarrow \mathcal{Y}$ is the l -th submodel ($l \in [L]$). Ensemble submodels are *deterministic*, meaning given a fixed submodel training set and \mathbf{x}_{te} , submodel prediction $f_l(\mathbf{x}_{\text{te}})$ is always the same.

A *decision function* aggregates the L submodel predictions to form ensemble f 's overall prediction; f 's decision function is *voting-based*. Let

$$\dot{c}_y(\mathbf{x}_{\text{te}}) := \sum_{l=1}^L \mathbb{1}[f_l(\mathbf{x}) = y] \quad (2.1)$$

be the number of ensemble submodels that predict label $y \in \mathcal{Y}$ for $\mathbf{x}_{\text{te}} \in \mathcal{X}$. The *plurality label* for \mathbf{x}_{te} is defined as

$$y_{\text{pl}} = \arg \max_{y \in \mathcal{Y}} \dot{c}_y(\mathbf{x}_{\text{te}}). \quad (2.2)$$

The *runner-up label* (i.e., the label that receives the second-most votes) is defined as

$$y_{\text{ru}} = \arg \max_{y \in \mathcal{Y} \setminus y_{\text{pl}}} \dot{c}_y(\mathbf{x}). \quad (2.3)$$

All ties are broken by selecting the smallest class indices.

2.2 On Attacker Threat Models

A *threat model* defines the assumptions made regarding an attacker’s capabilities. Chapters 4, 5, and 6 detail this dissertation’s primary theoretical contributions. Each chapter considers a different poisoning or backdoor threat model, and we discuss the corresponding threat model towards the beginning of each of these three chapters.

2.3 On the Defender Objectives

Differences in the attacker’s threat model lead to differences in the defender’s objective(s). At the beginning of Chapters 4, 5, and 6, we detail the chapter’s corresponding defender objective.

CHAPTER 3

RELATED WORK

This chapter draws on previously published, coauthored material [HL22a; HL23c; HL23a]. Hammoudeh wrote this entire section, including adding new related work that did not appear in the coauthored material. Lowd provided supervision and editorial suggestions in the original papers [HL22a; HL23c].

Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security*. CCS’22. Los Angeles, CA: Association for Computing Machinery, 2022. URL: <https://arxiv.org/abs/2201.10055>

Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning*. SaTML’23. 2023. URL: <https://arxiv.org/abs/2208.13904>

Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023. URL: <https://arxiv.org/abs/2302.11628>

Defenses against adversarial attacks partition into two broad categories. First, *empirical defenses* derive from understandings and observations about the underlying mechanisms adversarial attacks exploit to change a network’s predictions. Empirical defenses provide no guarantees of their effectiveness, and adversaries can adapt their attacks to bypass the defense – often with very little effort [Tra+20]. In contrast, *certified defenses* provide formal guarantees on their effectiveness given a specific set

of assumptions (i.e., threat model). These two defense categories are complementary and can be deployed together for better performance.

The threat model defines the types of attacks against which the defense is directed. Generally, most adversarial defenses target a single type of attack, e.g., poisoning, backdoor, evasion, etc. Very few defenses – certified or empirical – are robust across attack types [Web+23; HL23a]. We, therefore, organize the discussion of related adversarial defenses below based on the type of attack they consider.

3.1 Defenses Against Evasion Attacks

Recall from Chapter 1 that *evasion attacks* manipulate model predictions by perturbing test instances. Formally, given some test instance $(\mathbf{x}_{te}, y_{te})$, the adversary attempts to find some perturbation $\delta \in B$ such that $y_{te} \neq f(\mathbf{x}_{te} + \delta)$, where $B \subset \mathcal{X}$ defines the *perturbation neighborhood*. Often B is constrained to limit the perturbation’s perceptibility. Perturbation δ is typically constructed iteratively using either the target model or a surrogate. Common perturbation optimization algorithms include *fast gradient sign method* (FGSM) [GSS15] and *projected gradient descent* (PGD) [Mad+18].

What constitutes an imperceptible adversarial perturbation is subjective and implicitly human-centric [LSF21]. For continuous domains like vision, the perturbation neighborhood is usually constrained based on some ℓ_p norm, where $p \in \mathbb{N} \cup \infty$. Formally, an ℓ_p -bounded attack defines the $B_{p,\epsilon}(\mathbf{x}_{te})$ neighborhood w.r.t. $\mathbf{x}_{te} \in \mathcal{X}$ as

$$B_{p,\epsilon}(\mathbf{x}_{te}) := \{\mathbf{x} - \mathbf{x}_{te} : \|\mathbf{x} - \mathbf{x}_{te}\|_p \leq \epsilon\}, \quad (3.1)$$

where $\epsilon \in \mathbb{R}$ is the *perturbation radius* [LXL23]. Chapter 5 considers a *sparse*, or ℓ_0 , attacker that arbitrarily controls an unknown subset of the features. For discrete inputs (e.g., text), an ℓ_p perturbation model generally does not apply;

instead, text perturbation models focus on preserving syntactic structure and semantic meaning [Ebr+18; Jin+20].

Below we discuss empirical and certified defenses against evasion attacks.

3.1.1 Empirical Defenses. Empirical evasion defenses can be broadly categorized into two classes: adversarial training and gradient obfuscation methods. We discuss both of these defense strategies below.

First, *adversarial training* is perhaps the best-known method to improve a model’s robustness against adversarial attack [Bai+21]. Formally, empirical risk minimization (without regularization) defines the optimal model parameters as

$$\theta^* := \arg \min_{\theta} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \mathcal{L}(z_i; \theta). \quad (3.2)$$

Adversarial training considers a minimax constraint that, ideally, minimizes the empirical risk over worst-case perturbations given B where the adversarially optimized model parameters are

$$\theta_{\text{adv}}^* := \arg \min_{\theta} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \max_{\delta_i \in B} \mathcal{L}((\mathbf{x}_i + \delta_i, y_i); \theta). \quad (3.3)$$

Typically, adversarial training considers ℓ_p -bounded attacks for one or more definitions of p [TB19; MWK20; SGF22].

Finding the worst-case adversarial perturbation is provably hard [Wal+21]. First-order and second-order gradient-based methods often only approximate Eq. (3.3)’s inner maximization [LXL23]. While adversarial training generally improves robustness, the robustness improvement is generally not verifiable against worst-case perturbations, meaning adversarial training is only an empirical defense.

Gradient Obfuscation Methods The second class of empirical evasion defenses (implicitly) exploit the reality that, in practice, most adversarial defenses are evaluated

against gradient-based attacks (e.g., FGSM, PGD) [KGB16; CW17; Mad+18]. These defenses [Buc+18; Ma+18; Guo+18; Dhi+18; XZZ20] (unknowingly) rely on *obfuscated gradients* [ACW18], which reduce the utility of the gradients used to construct the adversarial example. Without useful gradients, traditional iterative, gradient-based attacks fail. However, obfuscation-based defenses provide a false sense of security [ACW18] since they are easily bypassed by an adaptive attacker [TB19]. The brittleness of empirical evasion defenses was a primary impetus that spurred the emergence of certified defenses, which we discuss next.

3.1.2 Certified Evasion Defenses. Recall that a certified defense provides provable robustness guarantees under a specific threat model. Numerous orthogonal strategies to certify evasion robustness have been proposed. For example, some methods bound a network’s curvature or Lipschitz constant to prove the impossibility of an adversarial example under the threat model [Wen+18; LLP20]. Other methods use a linear relaxation of a ReLU network to prove that a prediction is robust w.r.t. $B_{p,\epsilon}(\mathbf{x}_{te})$ [Gow+19]. A complete review of certified evasion defenses is well beyond the scope of this work. For a detailed review of existing certified evasion defenses including a taxonomy of the methods, we refer the reader to the excellent survey by Li et al. [LXL23].

The subclass of certified evasion defenses most relevant to this work are based on *randomized smoothing* [Li+19; CRK19; Léc+19]. Formally, given some \mathbf{x}_{te} , smoothing-based methods create a *smoothed classifier* \tilde{f} whereby prediction $\tilde{f}(\mathbf{x}_{te})$ is the most probable label within some predefined region around \mathbf{x}_{te} . For example, for ℓ_2 smoothing, the evaluated region is based on isotropic Gaussian $\mathcal{N}(\mathbf{x}_{te}, \sigma^2 I)$, where $\sigma > 0$ is a user-specified hyperparameter. In most situations, it is intractable for randomized smoothing to measure each class’s probability exactly; instead, class probabilities

are estimated using Monte Carlo methods [CRK19]. Smoothing-based methods then commonly use the Neyman-Pearson lemma [NP33] to bound the certified radius.

In terms of smoothing-based evasion defenses, the method most relevant to this dissertation is *randomized ablation* (RA) – a specialized form of randomized smoothing [CRK19] for sparse (ℓ_0) evasion attacks¹ where the adversary arbitrarily controls an unknown subset of the features [LF20b]. RA creates a smoothed classifier by repeatedly evaluating different *ablated inputs*, each of which *keeps* a small random subset of the features unchanged and masks out (*ablates*) all other features. Randomized smoothing certifies ℓ_0 -norm robustness, where the attacker arbitrarily controls a subset of \mathbf{x}_{te} 's features.

3.2 Defenses Against Poisoning and Backdoor Attacks

Evasion attacks are just one means to manipulate a model's predictions. Poisoning and backdoor attacks are an alternative approach whereby an attacker manipulates predictions by modifying the training set. Poisoning and backdoor attacks differ only in that the latter allows test instance (\mathbf{x}_{te}) perturbations while the former considers pristine test instances.

Poisoning attacks can be subclassified based on their effect on the model. An *indiscriminate* or *availability* poisoning attack degrades the model's *overall* performance, e.g., accuracy [BNL12; Xia+15; Fow+21]. In contrast, *targeted* attacks seek to manipulate an ML system's prediction on specific *target* instances [YHL23a; YHL23b]. A *single-target* attack seeks to influence one specific test prediction [Che+17], while *multitarget* attacks manipulate multiple test predictions – usually with some shared property (e.g., instances related to a specific individual or company) [Jag+21;

¹Sec. 5.2 formalizes ℓ_0 -norm robustness.

Lin+20]. Generally, all backdoor attacks are multitarget, where the backdoor trigger can be added to any (related) test instance.

Below we describe both empirical and certified defenses for poisoning attacks. In practice, these two defense categories are complementary and can be deployed together for better performance.

3.2.1 Empirical Classification Defenses. Like empirical evasion defenses, empirical defenses against backdoor and poisoning attacks provide no guarantees of their effectiveness. Empirical poisoning and backdoor defenses are generally founded on insights into the mechanisms and characteristics of specific attacks. As such, most existing empirical poisoning and backdoor defenses assume highly restricted threat models, including specific data modalities (e.g., only vision [Gao+19; Ude+19; VB20; Zhu+21]), model architectures (e.g, CNNs [Kol+19]), optimizers [HNM19], or training paradigms [Sor+20].

Below we review a few common categories of empirical poisoning and backdoor defenses. For a more comprehensive review, we direct the reader to the survey by Li et al. [Li+22].

Sanitization-Based Defenses These methods seek to identify and remove (i.e., sanitize) the adversarially perturbed instances in the training set. Sanitization-based methods all generally follow the same paradigm. They first identify training instances meeting some “outlier” criteria. The outliers are then removed from training set \mathcal{D} , and the model retrained [Per+20]. For example, Tran et al. [TLM18] found that backdoor attacks tend to leave a “spectral signature,” i.e., a detectable trace in the spectrum of the covariance of the feature representation. Tran et al. score and filter training instances based on their variance from typical feature representations. Similarly, Chen et al. [Che+19] cluster training instances based on the principal

components of the last linear layer; any training instances that are far away in feature space from other instances with the same label are sanitized from the training set.

A primary limitation of sanitization-based defenses is determining how many training instances to remove. Oversanitization results in excess removal of clean training instances, degrading the model’s clean performance. Undersanitization means that the attack may still succeed, albeit at a lower rate.

Model Disinfectant Defenses Rather than cleaning the training set, model disinfectant defenses try to directly repair the poisoned model itself. For example, Liu et al. [LXS17] neutralize any corrupted model weights by finetuning the model on known-clean data hoping that any corruption is deactivated through deliberate catastrophic forgetting. Another common disinfectant strategy relies on the insight that poisoning and backdoor attacks activate rarely-used neurons. Pruning-based defenses identify and disable these “unimportant” neurons expecting this will stop an attack.

Trigger Synthesis Defenses These methods seek to reconstruct any backdoor trigger(s) a model learned [Gao+19; Ude+19; VB20; Zhu+21]. The identified triggers are added to known-clean data and the model retrained in the expectation catastrophic forgetting deactivates the trigger. Note that trigger-synthesis defenses are specific to backdoor attacks since poisoning attacks are triggerless.

3.2.2 Certified Pointwise Classifiers. Recent years have seen a marked shift away from empirical poisoning and backdoor defenses to certified methods [SKL17; JCG21; Web+23; WLF22b; Rez+23]. Certified defenses differ concerning the assumptions they make about the attacker’s ability to perturb the training set. For example, Rosenfeld et al. [Ros+20] consider an attack that is only able to perturb

(i.e., “flip”) training labels. Weber et al. [Web+23] propose an alternate threat model that bounds the total ℓ_2 perturbation distance of the training set.

The *instance-wise poisoning* threat model allows the attacker to arbitrarily insert or delete entire *instances* in the training set and provide *pointwise* guarantees, i.e., certify the robustness w.r.t. individual predictions. These general-purpose certified poisoning classifiers are voting-based and derive their guarantees by lower bounding the number of training set modifications required to flip the predicted label. The primary difference between these certified classifiers is in the mechanism used to generate the “votes” multiset. We briefly review a few of these methods below.

Jia et al.’s [Jia+22a] certified poisoning classifier based on nearest neighbor methods is the simplest certified poisoning classifier, where the multiset of “votes” is the training labels from the test instance’s neighborhood. Formally, let $\mathcal{N}(\mathbf{x}_{te})$ denote the multiset of labels for the k training instances nearest to \mathbf{x}_{te} . Given plurality label $y_{pl} = f(\mathbf{x}_{te})$, the pointwise certified poisoning robustness is

$$R = \left\lfloor \frac{\sum_{y \in \mathcal{N}(\mathbf{x}_{te})} \mathbb{1}[y_{pl} = y] - \sum_{y \in \mathcal{N}(\mathbf{x}_{te})} \mathbb{1}[y_{ru} = y] + \mathbb{1}[y_{pl} > y_{ru}]}{2} \right\rfloor - 1, \quad (3.4)$$

where the indicator function breaks ties deterministically by choosing whichever label is assigned the *larger* index.

The second class of certified poisoning classifiers is ensemble based. *Deep partition aggregation* (DPA) was the first such method [LF21]. Levine and Feizi [LF21, Thm. 1] specify DPA’s certified robustness bound as

$$R = \left\lfloor \frac{\dot{c}_{y_{pl}}(\mathbf{x}_{te}) - (\dot{c}_{y_{ru}}(\mathbf{x}_{te}) + \mathbb{1}[y_{ru} < y_{pl}])}{2} \right\rfloor, \quad (3.5)$$

where the indicator function breaks deterministically ties by choosing whichever label is assigned the *smaller* index.² Implicitly, Eq. (3.5) assumes the worst-case that a single

²Observe that other than how ties are broken, Eqs. (3.4) and (3.5) calculate instance-wise robustness R in functionally the same way.

perturbation to any submodel’s training set can change that submodel’s prediction arbitrarily. We formalize this assumption below.

Def. 3.1. *Unit-Cost Assumption:* Any modification to a submodel’s training set changes the submodel arbitrarily.

In practice, there are limits to how much a single training set modification will alter a submodel and its predictions – in particular for models with strong inductive biases (e.g., linear models). Therefore, the unit-cost assumption’s pessimism can cause methods like DPA to underestimate a prediction’s true robustness. Nonetheless, this assumption greatly simplifies certifying ensemble classifier robustness by reducing the task to just submodel vote counting.

Multiple improvements to DPA have been proposed. Wang et al. [WLF22a] modify DPA’s ensemble so that submodels can be trained on overlapping data, which (slightly) improves the ensemble’s certification bounds. More recently, Rezaei et al. [Rez+23] propose *run-off elections*, a novel decision function for DPA that can improve DPA’s certified robustness by several percentage points.

3.2.3 Robust Regression. So far, we have focused on exclusively robust methods for classification, where the label space is finite. Many of the methods proposed above do not (directly) generalize to cases where target space \mathcal{Y} is continuous or has unbounded cardinality. Below we discuss previous methods to improve the robustness of regression.

3.2.3.1 Resilient Regression. Early methods were rooted in *robust statistics* and focused on mitigating the effect of training set outliers. For example, various trimmed loss functions (e.g., Huber [Hub64], Tukey [BT74]) cap a training outlier’s influence on a model [JW78; Lec89]. Methods like RANSAC [FB81] are based on data sanitization [TZ00; RH11].

3.2.3.2 Certified Regression. The above robust regressors primarily target random noise/outliers. As explained in Chapter 1, adversarial training instances can be much more insidious since they are crafted to avoid detection by appearing uninfluential and may only affect a very small fraction of test predictions [Che+17; Wal+21]. These factors can combine to make adversarial training instances difficult for resilient methods to fully detect and correct [Li+22].

Some existing poisoning and backdoor regression defenses do provide pointwise robustness guarantees, albeit under strong assumptions about the underlying data distribution [KKM18]. For example, some work assumes that the training set follows a linear data distribution with arbitrary white, Gaussian noise [CCM13; Liu+20a]. Others assume the data distribution’s feature matrix is low rank [Liu+17]. Conditioning a guarantee on a specific data distribution is inherently precarious – in particular if the strong distributional assumption rarely holds and cannot be easily verified. If the distributional assumption does not hold, any guarantee is no guarantee at all.

Note that there are some poisoning defenses for regression that provide guarantees without making distributional assumptions [Jag+18; KKM18]. However, their robustness guarantees are themselves distributional. For example, Jagielski et al. [Jag+18] bound the clean training data’s mean error but provide no pointwise guarantees. In other words, such methods do not provide insight into each prediction’s robustness.

In summary, while certified classification methods has seen numerous promising advances in recent years, certified regression still largely lags behind. Better certified regressors that make fewer strong assumptions are sorely needed.

3.3 Defenses Outside the ML System

All of the ideas above seek to improve an ML system’s robustness by improving the model itself. A motivated attacker will search for and exploit the weakest point in the ML system, which may not be the model. For example, human failures are often a common cause of security breaches [Col22; Ric22].

The best way to defend an ML system may lie outside of the ML system. For example, email spammers can be stopped by blocking their access to payment processors [Lev+11]. These outside defenses require information about an attacker’s goals, and methods [Tur20]. Knowledge about an attack, including its target, enables forensic and security analysts to reason about an attacker’s identity. Furthermore, insight into an attacker and their motivations helps anticipate future attacks [Pit+09] and build cost-effective, targeted defenses [Aga+19].

CHAPTER 4

REDUCING CERTIFIED REGRESSION TO CERTIFIED CLASSIFICATION

This chapter contains previously published, coauthored material [HL23c]. Hammoudeh developed the primary method, developed all code, conducted all experiments, and wrote the manuscript. Lowd provided supervision, editorial suggestions, and proposed some supplemental experiments.

Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning*. SaTML’23. 2023. URL: <https://arxiv.org/abs/2208.13904>

Section 3.2.3.2 explains that multiple certifiably-robust classifiers have recently been proposed. These methods certify robustness against the insertion and deletion of arbitrary instances in the training set. For example, Levine and Feizi [LF21] propose *deep partition aggregation*, which uses an ensemble of L deterministic, independent submodels trained on disjoint training sets. In addition, Jia et al. [Jia+22a] propose a certified classifier based on nearest-neighbor classification. Certified regression has not kept pace with these rapid advances in certified classification.

Formally, a problem Q is *reducible* to a different problem Q' if an efficient algorithm to solve Q' can also efficiently solve Q [DPV08]. Our *key insight* is that certified regression is reducible to voting-based certified classification. Mapping certified regression to certified classification requires only minimal changes to the certified classifier’s architecture, with the robustness certification function identical. Given reducibility, an important takeaway is that certified regression can be viewed as no harder than certified classification.

Coupling our reduction with existing certified classifiers [Jia+22a; LF21; WLF22a], we propose six new certifiably-robust regressors. To the extent of our knowledge, our methods are the first to provide pointwise regression robustness guarantees against poisoning without both distributional and model assumptions.

This chapter’s primary contributions are enumerated below.

1. We formalize three paradigms based on median perturbation to map certified regression to certified classification. All of this chapter’s certified regressors apply one of these paradigms.
2. We propose two provably-robust instance-based regressors – one based on k -nearest neighbors and the other based on all training instances within a feature-space region.
3. We separately propose four ensemble-based certified regressors, where one pair of regressors trains submodels on disjoint data while the other pair allows submodels to be trained on overlapping data.
4. We significantly improve the certification performance of our ensemble-based regressors *and* existing certified classifiers via a tighter analysis of submodel prediction stability.
5. We demonstrate our methods’ effectiveness on both regression and classification datasets, where we certify significant fractions of the training set and even outperform state-of-the-art certified classifiers on binary classification.

4.1 Preliminaries

Below, we formalize this chapter’s threat model and defender objective. We then review certification bounds for regression and how a certified regressor can be reused as certified binary classifier.

Threat Model For arbitrary *test instance* $(\mathbf{x}_{te}, y_{te})$, the adversary’s objective is to alter the model so that the *prediction error* $|f(\mathbf{x}_{te}) - y_{te}|$ is as large as possible. Our primary threat model considers an adversary that can insert arbitrary instances into training set \mathcal{D} and arbitrarily delete instances from \mathcal{D} .¹ The attacker has perfect knowledge of the learner and our method. We make *no assumptions about the underlying data distribution or adversarial training instances*.

Our Objective Determine *certified robustness* R – a guarantee on the number of training instances that can be inserted into or deleted from training set \mathcal{D} without the model prediction ever violating the requirement that $\xi_l \leq f(\mathbf{x}_{te}) \leq \xi_u$, where $\xi_l, \xi_u \in \mathbb{R}$ are user-specified and application dependent. Note that robustness R is *pointwise*, meaning each prediction $f(\mathbf{x}_{te})$ is certified individually.

4.1.1 One-Sided vs. Two-Sided Certification Bounds. For simplicity, the remaining sections exclusively describe how to certify a *one-sided upper bound*, $f(\mathbf{x}) \leq \xi$, since all other bounds reduce to this base case. For example, certifying a *one-sided lower bound* reduces to certifying an upper bound via negation as $f(\mathbf{x}) \geq \xi \Leftrightarrow -f(\mathbf{x}) \leq -\xi$. Likewise, a *two-sided bound* is equivalent to the worst one-sided robustness as

$$\begin{aligned} \xi_l \leq f(\mathbf{x}) \leq \xi_u &\Leftrightarrow (f(\mathbf{x}) \geq \xi_l) \wedge (f(\mathbf{x}) \leq \xi_u) \\ &\Leftrightarrow (-f(\mathbf{x}) \leq -\xi_l) \wedge (f(\mathbf{x}) \leq \xi_u). \end{aligned} \tag{4.1}$$

4.1.2 Relating Regression and Binary Classification. Binary classification can be viewed as a simple form of regression where $\mathcal{Y} = \{\pm 1\}$. The model’s decision function becomes $\text{sgn } f(\mathbf{x}_{te})$ where $\text{sgn } a = +1$ if $a > 0$ and

¹Sec. 4.7 considers a somewhat restricted threat model where attackers only make arbitrary deletions but no insertions. This allows us to empirically evaluate our method despite few base models fully utilizing our threat model.

−1 otherwise. While our primary focus is regression, our methods also achieve state-of-the-art results for binary classification.

4.2 Warmup: Perturbing a Set’s Median

Traditional center statistics such as mean have a *breakdown point* of 0, i.e., altering a single value in a set can shift the mean arbitrarily. In contrast, median has maximum robustness, i.e., a breakdown of 50%. A high breakdown point entails that a statistic is stable and resistant to change. We formalize changes to median below.

Def. 4.1. *Median Perturbation: The task of altering a set’s contents so that its median exceeds some specified $\xi \in \mathbb{R}$.*

Throughout this work, determining pointwise robustness R simplifies to quantifying the number of changes that can be made to a set without perturbing its median. To better foster intuitions, we first formalize robustness R w.r.t. simply perturbing a multiset’s median and unrelated to any model. Later sections apply these ideas to link certified regression and certified classification.

Formally, let \mathcal{V} be a multiset of cardinality $L := |\mathcal{V}|$. Denote the subset of elements in \mathcal{V} that are at most ξ as $\mathcal{V}_\ell := \{\nu_l \in \mathcal{V} : \nu_l \leq \xi\}$ and denote its complement $\mathcal{V}_u := \mathcal{V} \setminus \mathcal{V}_\ell$.

Below we define three different paradigms that constrain how \mathcal{V} is modified. Figure 1 visualizes our first two unweighted paradigms. Note that Fig. 1’s values are repeatedly used throughout this chapter, including in Fig. 2 for our third median perturbation paradigm and later in Figs. 4 and 5. In all cases below, consider when $\text{med } \mathcal{V} \leq \xi$ since the degenerate case of $\text{med } \mathcal{V} > \xi$ is by definition non-robust.

4.2.1 Unweighted Swap Paradigm. Here, set \mathcal{V} has fixed, odd-valued² cardinality L . All modifications to \mathcal{V} take the form of “swaps” where a single value in \mathcal{V} is replaced with any real number. Fig. 1b visualizes the unweighted swap paradigm on a simple set $\mathcal{V} = \{2, \dots, 6\}$ of $L = 5$ values. Lemma 4.2 tightly bounds the number of arbitrary swaps R that can be made to \mathcal{V} without perturbing its median.

Lemma 4.2. *For $\xi \in \mathbb{R}$, real multiset \mathcal{V} where $\text{med } \mathcal{V} \leq \xi$ with $L := |\mathcal{V}|$ odd, and $\mathcal{V}_1 := \{\nu_i \in \mathcal{V} : \nu_i \leq \xi\}$, let $\tilde{\mathcal{V}}$ be a multiset formed from \mathcal{V} where elements have been arbitrarily replaced. If the number of elements replaced in $\tilde{\mathcal{V}}$ does not exceed*

$$R = |\mathcal{V}_1| - \left\lceil \frac{L}{2} \right\rceil, \quad (4.2)$$

it is guaranteed that $\text{med } \tilde{\mathcal{V}} \leq \xi$.

Proof sketch. For a set of odd cardinality L , the median is always the set’s $\lceil \frac{L}{2} \rceil$ -th largest value. For \mathcal{V} ’s median to be at most ξ , at least $\lceil \frac{L}{2} \rceil$ items in \mathcal{V} cannot exceed ξ . Each swap reduces the number of elements not exceeding ξ by at most one. If there are $|\mathcal{V}_1|$ elements less than or equal to ξ in \mathcal{V} and there must be at least $\lceil \frac{L}{2} \rceil$ such elements to avoid perturbing the median, then at most $|\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$ swaps can be performed.

4.2.2 Insertion/Deletion Paradigm. For the second paradigm, \mathcal{V} is no longer fixed cardinality (it may expand or contract), and L may be even or odd. Each modification of \mathcal{V} takes the form of either a single deletion or insertion but not both. Figs. 1c and 1d visualize median perturbation under insertions and deletions resp. with certified robustness R following Lem. 4.3. Suppl. Sec. B.1 proves that worst-case insertions and deletions perturb a set’s median in exactly the same way and thus

²Fixing L as odd simplifies the overall formulation and presentation since it ensures that \mathcal{V} ’s median is always an element in \mathcal{V} . In all cases here where L is fixed as odd, L is always a user-selected hyperparameter. Extending our formulation to consider even L is not challenging but is verbose.

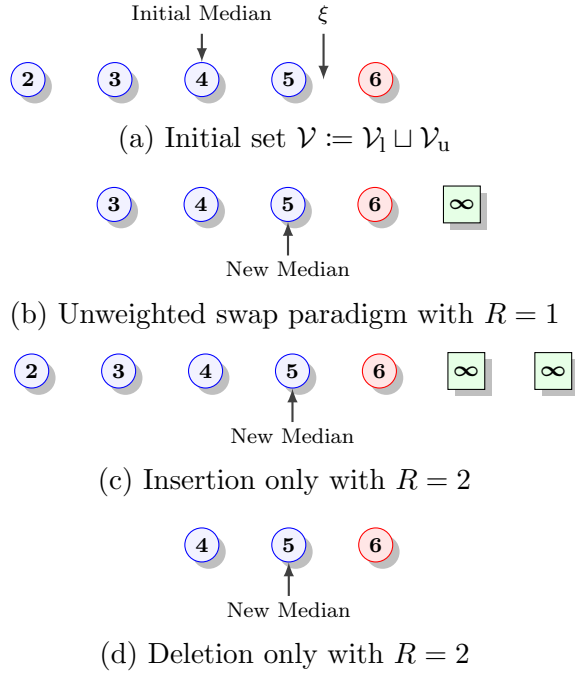


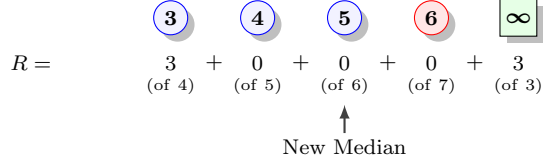
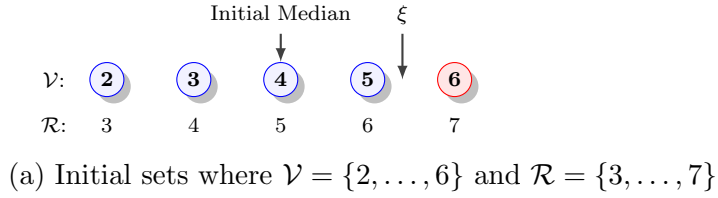
Figure 1. Unweighted Median Perturbation: (1a) Blue denotes elements in subset \mathcal{V}_l , i.e., elements in \mathcal{V} with value at most $\xi = 5.4$. \mathcal{V}_u 's values are red. Each “swap” (1b) switches a value in \mathcal{V}_l with an arbitrarily large replacement. Deletions (1d) and insertions (1c) are interchangeable (suppl. Lemma B.1), with both yielding the same median value in the same number of modifications made to \mathcal{V} . In Figs. 1b to 1d above, any additional modifications to the set would perturb the median.

are interchangeable. That is why Figs. 1c and 1d have identical certified robustness ($R = 2$).

Lemma 4.3. *For $\xi \in \mathbb{R}$ and real multiset \mathcal{V} where $\text{med } \mathcal{V} \leq \xi$, define $L := |\mathcal{V}|$ and $\mathcal{V}_l := \{\nu_l \in \mathcal{V} : \nu_l \leq \xi\}$. Let $\tilde{\mathcal{V}}$ be any multiset formed from \mathcal{V} where elements have been arbitrarily deleted and/or inserted. Then, if the total number of inserted and deleted elements in $\tilde{\mathcal{V}}$ does not exceed*

$$R = 2|\mathcal{V}_l| - L - 1, \quad (4.3)$$

it is guaranteed that $\text{med } \tilde{\mathcal{V}} \leq \xi$.



(b) Weighted swap paradigm with $R = 6$

Figure 2. Weighted Swap Paradigm: Extension of Fig. 1 to weighted costs. For simplicity and w.l.o.g., let $\mathcal{R} = \{3, \dots, 7\}$, i.e., $\forall_l r_l = \nu_l + 1$. Fig. 2a is identical to Fig. 1a except below each element ν_l is its corresponding weight r_l . Observe $\Delta = 1$ and $\tilde{\mathcal{R}}_1 = \{3, 4\}$. Fig. 2b shows that for $R = 6$ (visualized below each element), it is impossible to perturb the median, and any additional weight would be sufficient to swap out $\nu_2 = 3$.

Eq. (4.2)’s bound may be non-tight by 1. We did this for consistency with other ideas.

Comparing Eqs. (4.2) & (4.3), the insertion/deletion paradigm’s robustness R is about twice that of the unweighted swap paradigm. Intuitively, this is because one swap entails two separate operations – both an insertion and a deletion.

4.2.3 Weighted Swap Paradigm. The two median-perturbation paradigms above assume that each modification to \mathcal{V} has equivalent cost. Consider a generalized swap paradigm where each value $\nu_l \in \mathcal{V}$ has an associated weight/cost $r_l \in \mathbb{N}$. We seek to tightly bound the budget an attacker could spend with it remaining guaranteed that $f(\mathbf{x}_{te}) \leq \xi$; we still denote this budget R .

Given \mathcal{V}_1 as above, $\mathcal{R}_1 := \{r_l : \nu_l \in \mathcal{V}_1\}$ contains \mathcal{V}_1 ’s corresponding weights/costs. Define $\Delta := |\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$, and let multiset \mathcal{R}_Δ be the Δ smallest values in \mathcal{R}_1 (i.e., $|\mathcal{R}_\Delta| = \Delta$). Directly applying Lem. 4.2, an obvious but non-optimal bound is

$$R \geq \sum_{r \in \mathcal{R}_\Delta} r. \quad (4.4)$$

Recall Fig. 1a where $\mathcal{V} = \{2, \dots, 6\}$ and $\xi = 5.4$. Consider its weighted extension where for simplicity and w.l.o.g. $\mathcal{R} = \{3, \dots, 7\}$, i.e., $\forall_l r_l = \nu_l + 1$. Eq. (4.4) certifies robustness $R = 3$ for this example. However, Fig. 2b shows $R = 6$ since the budget of the second (i.e., $(\Delta + 1)$ -th) largest value in \mathcal{R}_1 can be partially used. Lemma 4.4 formalizes this insight into a tight bound for median perturbation under weighted swaps.

Lemma 4.4. *For $\xi \in \mathbb{R}$ and real multiset \mathcal{V} where $\text{med } \mathcal{V} \leq \xi$, let \mathcal{R} be \mathcal{V} 's corresponding integral weight multiset where $L := |\mathcal{V}| = |\mathcal{R}|$ is fixed and odd. Define $\mathcal{R}_1 := \{r_l \in \mathcal{R} : \nu_l \leq \xi\}$, and let $\tilde{\mathcal{R}}_1$ be the smallest $(|\mathcal{V}| - \lceil \frac{L}{2} \rceil + 1)$ values in \mathcal{R}_1 . Then the cost to perturb \mathcal{V} 's median exceeds*

$$R = \sum_{r \in \tilde{\mathcal{R}}_1} r - 1. \quad (4.5)$$

4.3 Reducing Regression to Voting-Based Binary Classification

We now show how methods used to certify binary classification can be adapted to certify regression. During inference, all voting-based certified methods (both classifiers and regressors) follow the same basic procedure.

First, the model generates a multiset of votes, which for binary classification we denote $\mathcal{V}_{\pm 1}$. Certified classifiers only differ in how $\mathcal{V}_{\pm 1}$ is constructed and in the consequences that construction has on certifying R . For example, $\mathcal{V}_{\pm 1}$ could be a k NN neighborhood or the submodel predictions in an ensemble. Nonetheless, for binary classification, $\mathcal{V}_{\pm 1}$ contains at most two unique values ($+1$ and -1), meaning $\mathcal{V}_{\pm 1}$'s majority label is also its median. In other words, $f(\mathbf{x}_{\text{te}}) = \text{med } \mathcal{V}_{\pm 1}$.

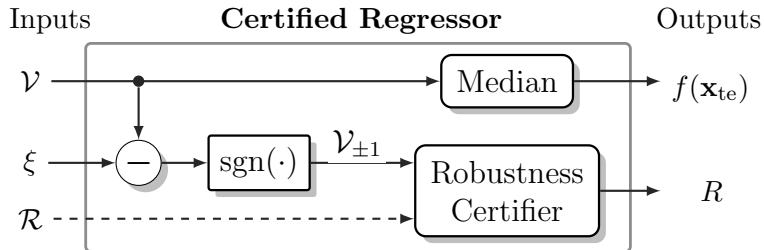


Figure 3. Certified Regression to Certified Classification Reduction: For $\mathbf{x}_{te} \in \mathcal{X}$, the decision function is $f(\mathbf{x}_{te}) := \text{med } \mathcal{V}$ – just like voting-based certified classification. Certified regression binarizes \mathcal{V} into $\mathcal{V}_{\pm 1}$, which is used by the robustness certifier (optionally with weights \mathcal{R}) to determine R .

To certify robustness R , existing methods rely on a function we term the *robustness certifier*. The function’s inputs are votes $\mathcal{V}_{\pm 1}$ and optionally weights/costs \mathcal{R} . Implicitly, the certifier knows how the votes were generated and how changes to training set \mathcal{D} could affect $\mathcal{V}_{\pm 1}$. Generally, a simple procedure to construct $\mathcal{V}_{\pm 1}$ entails a simple certifier, and complex construction implies a complex certifier. Fundamentally, for voting-based, binary classification, robustness certification always reduces to the same core idea. If $f(\mathbf{x}_{te}) = \text{med } \mathcal{V}_{\pm 1}$, then for the runner-up label to overtake the majority label, $\mathcal{V}_{\pm 1}$ ’s median must be perturbed. Therefore, *certifying voting-based, binary classification is simply certifying median perturbation*.

To generalize a voting-based, certified classifier to certify regression, two primary modifications are required; we visualize our regression to classification reduction in Fig. 3.

First, the model is modified from generating binary votes $\mathcal{V}_{\pm 1}$ to generating real-valued ones denoted \mathcal{V} . The changes necessary to make this switch are specific to the underlying certified classifier. In some cases, no change is required [Jia+22a]; for others, ensemble submodel classifiers are simply replaced with submodel regressors [LF21; WLF22a].

The second modification is more subtle. If \mathcal{V} is real-valued, a robustness certifier expecting binary votes cannot be directly applied. That is where $\xi \in \mathbb{R}$ fits in; it partitions \mathcal{V} into two subsets: \mathcal{V}_l containing all “votes” at most ξ and \mathcal{V}_u containing all “votes” exceeding ξ . We can think of these subsets as two different classes where if $f(\mathbf{x}_{te}) \leq \xi$, \mathcal{V}_l is the majority class and \mathcal{V}_u runner-up. For any prediction $f(\mathbf{x}_{te}) := \text{med } \mathcal{V}$, the robustness certifier’s output R equals the number of training set modifications that can be made without ever perturbing $\text{med } \mathcal{V}$ beyond ξ .

Lemma 4.5 formalizes the connection between real-valued and binarized robustness. This symmetry in robustness derives from both tasks’ (implicit) shared reliance on median.

Lemma 4.5. *For $\xi' \in \mathbb{R}$ and real multiset \mathcal{V}' where $\text{med } \mathcal{V}' \leq \xi'$, let*

$$\mathcal{V}_{\pm 1} := \{\text{sgn}(\nu_l - \xi') : \nu_l \in \mathcal{V}'\}. \quad (4.6)$$

Let \mathcal{R} be the corresponding integral weight multiset of \mathcal{V}' where $|\mathcal{V}'| = |\mathcal{R}|$. Then, under the (un)weighted swap and insertion/deletion paradigms, both $\mathcal{V} = \mathcal{V}'$ with $\xi = \xi'$ and $\mathcal{V} = \mathcal{V}_{\pm 1}$ with $\xi = 0$ have equivalent robustness R .

By binarizing \mathcal{V} , Lem. 4.5 enables us to directly *reuse robustness certifiers from binary classification to certify regression*.

Our reduction to certified classification entails two primary benefits. First, it allows us to repurpose for regression the diverse set of voting-based, certified classifiers that already exist [Jia+22a; LF21; WLF22a]. Moreover, as new voting-based, certified classifiers are proposed in the future, these yet undiscovered methods can also be reformulated as certified regressors.

Although this work focuses on certified poisoning defenses, other types of certified defenses also rely on voting-based schemes, including randomized smoothing methods

for evasion attacks [LF20b; Jia+22b]. Our certified regression to certified classification reduction can also be applied to these other types of voting-based defenses as well.

As mentioned above, the procedure to construct the set of votes and to certify robustness is unique to each classifier. The next three sections describe how to certify regression using progressively more complex models, with each method based on a reduction to an existing voting-based certified classifier.

4.4 Certified Instance-Based Regression

For the first method, recall from Sec. 3.2.2 that Jia et al. [Jia+22a] propose a state-of-the-art certified classifier based on k NN. Nearest-neighbor methods are a specific type of *instance-based learner* (IBL), where predictions are made using memorized training instances [AKA91]. IBLs generally rely on the intuition that instances close together in *feature space* (\mathcal{X}) have similar target values (\mathcal{Y}). Specifically, IBLs search for stored training instances most similar to \mathbf{x}_{te} and derive the model prediction from these retrieved *neighbors*.

We partition IBLs into two subcategories:

- *Fixed-population neighborhood* methods specify the exact number of “neighbors” when making a prediction.
- *Region-based neighborhood* methods define a neighborhood as all training instances in a specific feature-space region.

These two subcategories calculate certified robustness differently and are discussed separately below.

All IBLs considered here use the same decision rule. Formally, given $\mathbf{x}_{te} \in \mathcal{X}$ and real multiset neighborhood $\mathcal{N}(\mathbf{x}_{te})$ returned by the IBL, the model’s prediction is the neighborhood’s median, i.e., $f(\mathbf{x}_{te}) := \text{med} \mathcal{N}(\mathbf{x}_{te})$. Recall that our goal is to certify

that if at most R arbitrary insertions or deletions are made to \mathcal{D} , it is guaranteed that $f(\mathbf{x}_{te}) \leq \xi$.

4.4.1 Fixed-Population Neighborhood. As the name indicates, fixed-population neighborhood IBLs make predictions using a fixed number of training instances, i.e., $\forall \mathbf{x}_{te} L = |\mathcal{N}(\mathbf{x}_{te})|$. k -nearest neighbors is perhaps the best-known fixed-population method. Traditionally, k NN returns the neighborhood’s mean value. For clarity, we will refer to the version of k NN that uses the neighborhood’s median value as *k-Nearest Neighbors Median*, or simply k NN-m.

Our threat model allows the adversary to insert arbitrary training instances and/or delete any existing instances. Fig. 4b visualizes an example attack on a k NN-m regressor. Since k is fixed, inserting a new instance (■) into the neighborhood causes one neighborhood instance to be ejected; in other words, insertions are simply instance swaps. As a worst-case, we assume that the ejected element equals at most threshold ξ , meaning each insertion always maximally increases the neighborhood’s median. Under this simplifying assumption, adversarial insertions are always at least as harmful as deletions for fixed-population neighborhood IBLs.

Neighborhood size k is a user-specified hyperparameter so let k be odd-valued. Therefore, these fixed-population neighborhood IBL regressors satisfy all of the criteria of median perturbation under the unweighted swap paradigm where $L = k$. Theorem 4.6 then follows directly from Lemma 4.2.

Theorem 4.6. *Let f be an instance-based regressor trained on set \mathcal{D} . Given $\xi \in \mathbb{R}$ and $\mathbf{x}_{te} \in \mathcal{X}$, let real multiset $\mathcal{N}(\mathbf{x}_{te})$ be \mathbf{x}_{te} ’s neighborhood under f with fixed, odd-valued cardinality $L := |\mathcal{N}(\mathbf{x}_{te})|$. Define $\mathcal{V}_1 := \{y \in \mathcal{N}(\mathbf{x}_{te}) : y \leq \xi\}$. Given $f(\mathbf{x}_{te}) := \text{med} \mathcal{N}(\mathbf{x}_{te}) \leq \xi$, then if model f is trained on a modified \mathcal{D} where the total number of inserted and deleted training instances does not exceed*

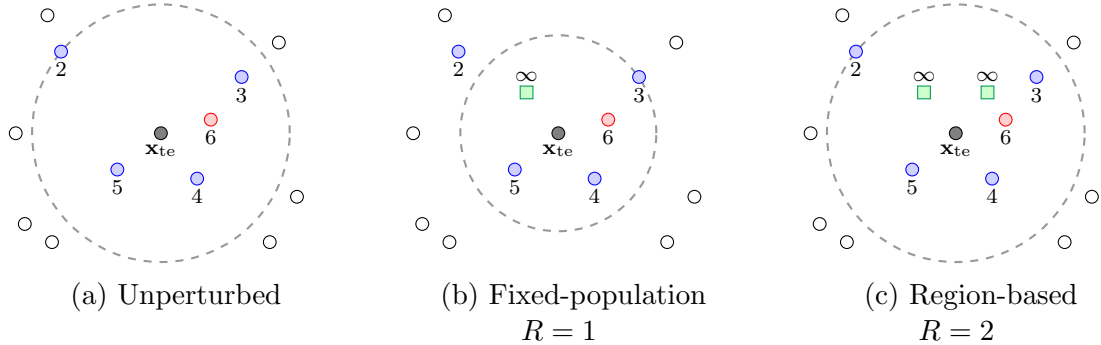


Figure 4. Certified Instance-Based Regression: Fig. 4a visualizes an unperturbed IBL model. Test instance \mathbf{x}_{te} 's neighborhood is visualized as a dashed line with neighborhood $\mathcal{N}(\mathbf{x}_{te})$ identical to \mathcal{V} in Fig. 1a. Fig. 4b shows an attack on a k NN- m model where the neighborhood's cardinality ($L = 5$) is fixed, and the one attack instance (\square) replaces one instance in \mathcal{V}_1 (\circ) (source Fig. 1b). A r NN-median model is shown in Fig. 4c, where the two inserted instances (\square) do not change the neighborhood's radius (source Fig. 1c).

$$R = |\mathcal{V}_1| - \left\lceil \frac{L}{2} \right\rceil, \quad (4.7)$$

it is guaranteed that $f(\mathbf{x}_{te}) \leq \xi$.

We denote k NN- m certified regression as k NN-CR. We defer the reader to the supplement of the original paper [HL23c, Lemma 15] for the proof that when under binary classification, k NN-CR and Jia et al.'s [Jia+22a] k NN classifier yield identical robustness guarantees.

4.4.2 Region-Based Neighborhood. Neighborhood membership does not need to be tied to the number of neighbors. Rather, a neighborhood can be defined by specific criteria, with all stored training instances satisfying those criteria included in the neighborhood. For instance, *radius nearest neighbors* (r NN) defines \mathbf{x}_{te} 's neighborhood as all training instances within a given distance of \mathbf{x}_{te} [Ben75]. Alternatively, *fully-random decision trees* recursively partition the feature space into

disjoint regions, and a neighborhood is defined as all instances within the same feature region [GEW06].

Fig. 4c visualizes an attack on an r NN-median learner, where the adversary inserts malicious instances (\square) to perturb the median prediction. Unlike fixed-population neighborhoods, the inserted instances do not cause any existing training instances to be ejected. Rather, inserting and deleting training instances are distinct operations.

It is easy to see that region-based IBLs with median as the decision operator follow Sec. 4.2.2’s insertion/deletion paradigm. Theorem 4.7 then follows directly from Lemma 4.3.

Theorem 4.7. *Let f be an instance-based regressor trained on \mathcal{D} that partitions \mathcal{X} into disjoint regions. Given $\mathbf{x}_{te} \in \mathcal{X}$, let real multiset $\mathcal{N}(\mathbf{x}_{te})$ be \mathbf{x}_{te} ’s neighborhood under f where $L := |\mathcal{N}(\mathbf{x}_{te})|$. For $\xi \in \mathbb{R}$, define $\mathcal{V}_1 := \{y \in \mathcal{N}(\mathbf{x}_{te}) : y \leq \xi\}$. If model f is trained on a modified \mathcal{D} where the total number of inserted and deleted training instances does not exceed*

$$R = 2|\mathcal{V}_1| - L - 1, \tag{4.8}$$

it is guaranteed that $f(\mathbf{x}_{te}) \leq \xi$.

Jia et al. propose an r NN-based certified classifier, with the robustness certifier identical to their k NN method. By using our insertion/deletion paradigm for the robustness certifier instead of Jia et al.’s approach, Eq. (4.8)’s R roughly doubles.

4.4.3 Computational Complexity. Eqs. (4.7) and (4.8) require determining \mathcal{V}_1 ’s cardinality, which has complexity $\mathcal{O}(L)$. However, constructing neighborhood $\mathcal{N}(\mathbf{x}_{te})$ can require scanning the entire training set and has

complexity $\mathcal{O}(n)$. Therefore, certifying each IBL regression prediction’s robustness is in $\mathcal{O}(n)$ – the same as Jia et al.’s certified k NN and r NN classifiers.

4.5 Certified Regression for General Models

Instance-based learners lend themselves to robustness certification. However, there are many applications where IBLs perform poorly. This section explores reducing certified regression to a second certified classifier, which will now allow us to use whichever model architecture has the best performance.

Recall from Sec. 3.2.2 that Levine and Feizi’s [LF21] certified classifier, DPA, uses an ensemble trained on partitioned training data. In this section, we first reduce certified regression to certified classification using DPA. We then improve the certification performance of DPA and by extension our certified regressor by using tighter, weighted analysis. All certified regression ensembles we consider have L submodels denoted f_1, \dots, f_L , and the ensemble decision function uses median, i.e., $f(\mathbf{x}_{te}) := \text{med} \{f_l(\mathbf{x}_{te}; 1), \dots, f_l(\mathbf{x}_{te}; L)\}$.

Since ensemble size L is always a user-specified hyperparameter, select odd L . For arbitrary $\mathbf{x}_{te} \in \mathcal{X}$, let $\mathcal{V} := \{f_l(\mathbf{x}_{te}) : l \in [L]\}$. Our goal remains to determine R – a pointwise guarantee on the total number of training set modifications where it remains guaranteed that $f(\mathbf{x}_{te}) \leq \xi$.

4.5.1 Partitioned Certified Regression. Here, the L submodel regressors are *fully-independent*, meaning their training sets are disjoint, and each submodel prediction provides no direct insight into any other submodel’s behavior. This simple framework makes *no assumptions about the submodel architecture*; the submodels may be non-parametric or parametric, deep or shallow, etc. The only requirement is that each submodel returns a deterministic prediction given its training set and feature vector \mathbf{x}_{te} .

Levine and Feizi enforce disjoint submodel training sets by using deterministic function h_{tr} to partition training set \mathcal{D} into L disjoint blocks, $D^{(1)}, \dots, D^{(L)}$. Formally, for all $l \in [L]$, submodel f_l 's training set is $\mathcal{D}_l = D^{(l)}$.

Since each training instance is assigned to exactly one submodel, any training set modification can only affect one submodel. Under the *unit-cost assumption* (Def. 3.1), each training set modification changes the corresponding submodel's prediction from $f_l(\mathbf{x}_{te})$ to ∞ in the worst case. Thus, perturbing a partitioned ensemble's median prediction follows Sec. 4.2.1's unweighted swap paradigm where, as explained above, *each perturbed submodel entails one training set modification*.

Via reduction to DPA, Theorem 4.8 directly applies Lemma 4.2 to certify unit-cost, partitioned regression's robustness under arbitrary training set insertions and deletions.

Theorem 4.8. *For $\mathbf{x}_{te} \in \mathcal{X}$, $\xi \in \mathbb{R}$, and deterministic function h_{tr} that partitions set \mathcal{D} into disjoint blocks $D^{(1)}, \dots, D^{(L)}$, let f be an ensemble of L submodels where L is odd, and each deterministic submodel f_l is trained on block $D^{(l)}$. Define $\mathcal{V}_1 := \{f_l(\mathbf{x}_{te}) : f_l(\mathbf{x}_{te}) \leq \xi\}$. Given $f(\mathbf{x}_{te}) := \text{med} \{f_l(\mathbf{x}_{te}; 1), \dots, f_l(\mathbf{x}_{te}; L)\} \leq \xi$, if model f is trained on a modified \mathcal{D} where the total number of inserted and deleted training instances does not exceed*

$$R = |\mathcal{V}_1| - \left\lceil \frac{L}{2} \right\rceil, \tag{4.9}$$

it is guaranteed that $f(\mathbf{x}_{te}) \leq \xi$.

We denote this disjoint ensemble regressor as *partitioned certified regression* (PCR). The original paper [HL23c, Lemma 16] proves that when regression is used for binary classification, PCR and DPA yield identical robustness guarantees (R).

4.5.2 Weighted Partitioned Certified Regression.

Levine and Feizi only consider the maximally pessimistic unit-cost assumption. For a feature vector \mathbf{x}_{te} , it may take multiple training set insertions/deletions to corrupt a submodel’s prediction. For example, Theorems 4.6 and 4.7 prove that IBL predictions are robust to multiple training set modifications.

Fixing the regressor’s overall architecture, one obvious way to improve certified robustness R is to improve the robustness certifier. Below, we introduce tighter analysis of each PCR submodel’s pointwise robustness so as to move beyond unit cost. Let $r_l \in \mathbb{N}$ denote the minimum number of insertions/deletions required to change³ the submodel enough where $f_l(\mathbf{x}_{te}) > \xi$. By definition, if $f_l(\mathbf{x}_{te}) > \xi$ without any training set modifications, $r_l = 0$. When $\exists_l r_l > 1$, better certified guarantees are possible through a weighted framework. Theorem 4.9 directly applies Lemma 4.4’s weighted swap paradigm to adapt PCR (and DPA) to weighted perturbation costs. We denote this extension *weighted partitioned certified regression* (W-PCR).

Theorem 4.9. *For $\mathbf{x}_{te} \in \mathcal{X}$, $\xi \in \mathbb{R}$, and function h_{tr} that partitions set \mathcal{D} into disjoint blocks $D^{(1)}, \dots, D^{(L)}$, let f be an ensemble of L submodels where L is odd. Each deterministic submodel f_l is trained on block $D^{(l)}$ and requires at least $r_l \in \mathbb{Z}_+$ modifications to $D^{(l)}$ for $f_l(\mathbf{x}_{te}) > \xi$. For $\mathcal{R} := \{r_l : f_l(\mathbf{x}_{te}) \leq \xi\}$, let $\tilde{\mathcal{R}}_1$ be \mathcal{R} ’s smallest $|\mathcal{R}| - \lceil \frac{L}{2} \rceil + 1$ values. Given $f(\mathbf{x}_{te}) := \text{med} \{f_l(\mathbf{x}_{te}; 1), \dots, f_l(\mathbf{x}_{te}; L)\} \leq \xi$, if model f is trained on a modified \mathcal{D} where the total number of inserted and deleted training instances does not exceed*

³Certified robustness R is the total number of training set modifications that can be made with it remaining guaranteed that $f(\mathbf{x}_{te}) \leq \xi$. In contrast, r_l is minimum the number of modifications needed to *perturb* submodel l ’s prediction enough that $f_l(\mathbf{x}_{te}) > \xi$. If R_l were the certified robustness of just *submodel* l , then $r_l = R_l + 1$. r_l ’s definition here follows related work [Ran+21].

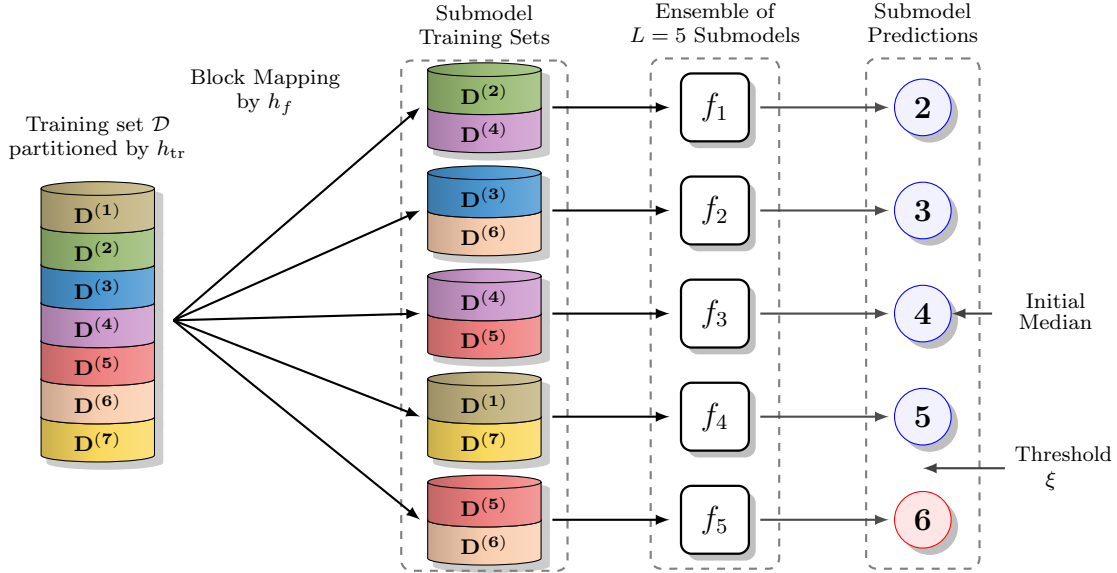


Figure 5. Overlapping Certified Ensemble: Simple visualization of the ensemble architecture for (weighted) overlapping certified regression. Function h_{tr} partitions training set \mathcal{D} into ($m = 7$) blocks. Function h_f defines each of the $L = 5$ submodel training sets, $\mathcal{D}_1, \dots, \mathcal{D}_5$. The ensemble prediction is the median submodel prediction, i.e., $f(\mathbf{x}_{\text{te}}) := \text{med} \{f_l(\mathbf{x}_{\text{te}}; 1), \dots, f_l(\mathbf{x}_{\text{te}}; L)\}$.

$$R = \sum_{r \in \tilde{\mathcal{R}}_1} r - 1, \quad (4.10)$$

it is guaranteed that $f(\mathbf{x}_{\text{te}}) \leq \xi$.

It can be easily shown that W-PCR always yields certified robustness at least as good as PCR. Although proposed in the context of regression, our weighted formulation also notably improves certified classification as shown in Sec. 4.8.2.

4.5.3 Computational Complexity. Both PCR and W-PCR require training $\mathcal{O}(L)$ models. As established by Lemmas 4.2 and 4.4, the computational complexity of PCR and W-PCR (resp.) to certify each ensemble prediction is $\mathcal{O}(L)$ [Blu+73] – the same complexity as DPA.⁴

⁴Not included in W-PCR’s complexity is the time to determine r_1, \dots, r_L .

4.6 Certified Regression Using Overlapping Training Data

This section reduces certified regression to a third certified classifier, specifically Wang et al.’s [WLF22a] reformulation of DPA where the submodels are trained on overlapping data. This makes the submodels interdependent, meaning one training set modification may alter multiple submodel predictions. Fig. 5 visualizes an ensemble trained on overlapping training sets. Again, L is the number of submodels.⁵ Function $h_{\text{tr}} : \mathcal{Z} \rightarrow [m]$ still partitions the instance space into m disjoint blocks, where $m \geq L$. Following Wang et al., a second deterministic function $h_f : [m] \rightarrow 2^{[L]}$ maps each training set block to one or more submodel training sets. Formally, submodel f_l ’s training set is $\mathcal{D}_l := \bigsqcup_{l \in h_f(j)} D^{(j)}$. Let $d^{(j)} := |h_f(j)|$ denote $D^{(j)}$ ’s *spread degree*, i.e., the number of models that use $D^{(j)}$ during training. Denote the *maximum spread degree* as $d_{\max} := \max\{d^{(1)}, \dots, d^{(m)}\}$. The ensemble’s decision function is still the median submodel prediction.

Below, we first consider certified regression on overlapping data under the unit-cost assumption. We then improve overlapping regression by leveraging our weighted reformulation.

4.6.1 Overlapping Certified Regression. Irrespective of whether the submodels are trained on disjoint or overlapping data, under the unit-cost assumption, at least $|\mathcal{V}_l| - \lceil \frac{L}{2} \rceil$ submodel predictions must exceed ξ to perturb the ensemble’s median. Observe that each submodel training set $\mathcal{D}_l \subset \mathcal{D}$ is composed of one or more dataset blocks. Perturbing *any* block in \mathcal{D}_l is sufficient to perturb the submodel’s prediction, with an optimal attacker *minimizing the number of training set (block) modifications*.

⁵In practice, for overlapping certified regression to guarantee better robustness than (W-)PCR the number of submodels generally must increase by several folds over partitioned regression.

If the goal were to perturb all L submodels, then for arbitrary block mapping function h_f , determining the minimum number of blocks that need to be modified reduces to *minimum set cover*, which is NP-hard [Sla97a]. Specifically, the set to cover is $\mathcal{T}_1 := \{l : f_l(\mathbf{x}_{te}) \leq \xi\}$, i.e., the submodels predicting at most ξ , and the collection of subsets is $\mathbf{S} := \{\{j : l \in h_f(j)\} : f_l(\mathbf{x}_{te}) \leq \xi\}$, which contains the dataset blocks each relevant submodel is trained on.

However, recall that for median perturbation under unweighted swaps, we only need to perturb (i.e., cover) $|\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$ submodels – not all of them. Therefore, rather than mapping to set cover, our problem reduces to the related problem of *partial set cover*, where only a constant fraction of the instances (i.e., submodels) need to be covered. For arbitrary block mapping function h_f , Lemma 4.10 below establishes that finding the optimal R here is NP-hard [Sla97b; EK10].

Lemma 4.10. *Finding optimal certified robustness R for overlapping certified regression is NP-hard.*

Although our problem is NP-hard, it is polynomial-time approximable. Specifically, the approximation uses the famous greedy set-cover algorithm where in each iteration, the subset (training block $D^{(j)}$) covering the most remaining elements (submodels) is selected [Chv79; Sla97b]. Let G denote the bound found by this greedy method, and define $\Delta := |\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$. Then for the non-naive case where $\Delta \geq 2$,

$$R \geq \left\lceil \frac{G}{\min\{H(d_{\max}), \ln \Delta - \ln \ln \Delta + 3 + \ln \ln 32 - \ln 32\}} \right\rceil, \quad (4.11)$$

where $H(d_{\max})$ is the d_{\max} -th harmonic number. This bound follows directly from partial set cover *approximation factor* analysis ([Sla97a, Thm. 4]; [Sla97b, Thm. 3]).⁶

⁶Eq. (4.11)'s bound is tighter (often significantly so) than the much more famous approximation factor, $H(\Delta)$, of Johnson [Joh74] and Lovász [Lov75].

Slavík [Sla97a] shows that the difference between this approximation factor’s overall lower and upper bound is only roughly 1.1, meaning this general approximation is quite good overall.

However, in most cases, the performance advantage of overlapping versus disjoint unit-cost regressors is small enough that the greedy optimality gap wipes out all gains. Instead, we rely on Fig. 6’s integer linear program (ILP) to bound R in the overlapping case.⁷ This ILP is directly adapted from standard partial set-cover, where for unit costs $\forall_l r_l = 1$.

While the ILP is still NP-hard in the worst case, modern LP solvers often find a (near) optimal solution in reasonable time (e.g., a few seconds) [Gur22]. In cases where finding true robustness R is computationally expensive, these solvers generally return guaranteed bounds on R that are (much) better than the greedy approximation [Van14].⁸ We refer to this unit-cost, ILP-based approach as *overlapping certified regression* (OCR).

4.6.2 Weighted Overlapping Certified Regression. Recall that Sec. 4.5.2 improves certified regressor PCR by reformulating DPA so as not to be restricted by the unit-cost assumption. Here, we follow the same approach of improving certified regressor OCR by generalizing Wang et al.’s [WLF22a] certified classifier to non-unit costs.

As with W-PCR earlier, $r_l > 1$ entails that submodel f_l ’s training set must be modified at least r_l times for $f_l(\mathbf{x}_{te}) > \xi$. This prevents weighted overlapping regression from applying partial set cover since each submodel f_l now has a *coverage* requirement. Instead, *partial set multicover* (PSMC) generalizes partial set cover

⁷Fig. 6 jointly formulates calculating R under unit and weighted costs.

⁸Sec. 4.8’s experiments use a fixed time limit to ensure tractability.

to support coverage requirements $r_l \geq 0$ [Shi+19; Ran+21], and we adapt PSMC to weighted, overlapping regression. PSMC, and by extension our task, is provably hard.

Corollary 4.10.1. *Finding the optimal certified robustness R for weighted overlapping certified regression is NP-hard.*

PSMC is far less studied than (partial) set cover. PSMC is polynomial-time approximable – albeit with worse known bounds than partial set cover. Ran et al. [Ran+21] provide the best-known PSMC bounds; their method is much more complicated than greedy partial set cover and relies on a reduction to another NP-hard problem, minimum densest subcollection. Let G be the solution generated by Ran et al.’s algorithm, then

$$R \geq \left\lceil \frac{G}{4 \lg LH(d_{\max}) \ln \Delta + 2 \lg L\sqrt{L}} \right\rceil. \quad (4.12)$$

Like with unweighted overlapping regression, Eq. (4.12)’s approximation factor is large enough that it usually wipes out the performance gains derived from weighted costs. Instead, we use Fig. 6’s ILP to bound R in accordance with Lem. 4.4. In the ILP, $\sigma = 1$ in the weighted case and 0 otherwise. Hence, at least $(|\mathcal{V}_1| - \lceil \frac{L}{2} \rceil + 1)$ submodels must be covered (i.e., perturbed) in the weighted case. Following Eq. (4.5), sum $\sum_{j=1}^m \omega^{(j)}$ is decremented by one in the ILP.

We refer to this overlapping ILP-based approach as *weighted overlapping certified regression* (W-OCR).

4.6.3 Computational Cost. See the supplement of the original paper [HL23c, Sec. I.E] for an empirical evaluation and extended discussion of the OCR and W-OCR ILP execution time.

$$\min R = \sum_{j=1}^m \omega^{(j)} - \sigma \quad (4.13a)$$

$$\text{s.t. } \mathcal{T}_1 = \{l : f_l(\mathbf{x}_{\text{te}}) \leq \xi\}, \quad (4.13b)$$

$$r_{\max} = \max\{r_l : l \in [L]\} \quad (4.13c)$$

$$\sigma = \mathbb{1}[r_{\max} > 1] \quad (4.13d)$$

$$\sum_{l \in \mathcal{T}_1} \delta_l \geq |\mathcal{V}_1| - \left\lceil \frac{L}{2} \right\rceil + \sigma, \quad (\text{Median perturb.}) \quad (4.13e)$$

$$r_l \delta_l \leq \sum_{D^{(j)} \subseteq \mathcal{D}_l} \omega^{(j)}, \quad l \in \mathcal{T}_1 \quad (4.13f)$$

$$\delta_l \in \{0, 1\}, \quad l \in \mathcal{T}_1 \quad (4.13g)$$

$$\omega^{(j)} \in \{0, \dots, r_{\max}\}, \quad j \in [m] \quad (4.13h)$$

Figure 6. **Overlapping Certified Regression Integer Linear Program:** Adapted from the partial set (multi)cover integer linear program. Calculates certified robustness R for both OCR and W-OCR with indicator variable σ adjusting the program to account for weighted costs. For arbitrary feature vector \mathbf{x}_{te} , \mathcal{T}_1 is the set of submodels that predict $f_l(\mathbf{x}_{\text{te}}) \leq \xi$. Variable $\omega^{(j)}$ contains the number of modifications made to training set block $D^{(j)}$. Binary variable $\delta_l = 1$ if submodel f_l has been sufficiently modified for $f_l(\mathbf{x}_{\text{te}}) > \xi$ and 0 otherwise.

4.7 Certifying Any Model Beyond Unit Cost

The preceding sections describe the benefits of having more robust ensemble components (i.e., $r_l > 1$) but do not address how to find r_l . Apart from IBLs and ensembles, the two methods we focus on in this work, we know of no general method for computing insertion/deletion robustness efficiently. We attribute this scarcity to the task’s difficulty. Nonetheless, we believe this work shows that certification beyond unit cost merits future study. This section explores certifying beyond unit cost from two perspectives. First, we consider the obvious idea of combining IBLs with ensembles and explain why that performs poorly. Next, we propose a simple, general approach to certify any (sub)model beyond unit cost, albeit with a (slightly) more restricted threat model.

4.7.1 Combining Instance-Based Learners & Ensembles. The points raised below apply to both fixed-population and region-based IBLs. We exclusively discuss k NN-CR here with the extension to other certified IBLs straightforward.

In practice, function h_{tr} partitions instance space \mathcal{Z} uniformly at random (u.a.r.) into m approximately equal-sized regions. For simplicity and w.l.o.g., consider an ensemble of k NN-CR submodels trained on disjoint subsets where $L = m$.

Let k' and R denote the neighborhood size and certified robustness (resp.) of a k NN-CR model trained on i.i.d. training set \mathcal{D} . If \mathcal{D} is partitioned u.a.r. to train L k NN-CR submodels each with $k \approx \frac{k'}{L}$, then each submodel's expected robustness is roughly $\frac{R}{L}$. In the best case for the defender ($\forall_l f_l(\mathbf{x}_{\text{te}}) \leq \xi$), an adversary only needs to perturb at most $\lceil \frac{L}{2} \rceil$ submodels. Combining the above with Theorem 4.9 for W-PCR, this k NN-CR ensemble's expected certified robustness is approximately

$$\left\lceil \frac{L}{2} \right\rceil \left(\frac{R}{L} \right) - 1 = \frac{R}{2} + \frac{R}{2L} - 1 < R. \quad (4.14)$$

As $n, L \rightarrow \infty$, then by Eq. (4.14), a k NN-CR ensemble's expected robustness *decreases by 50% versus the single k NN-CR model baseline*. Intuitively, for ensembles, an adversary only needs to directly attack about half of the submodels and by extension half of the training data. In contrast, when there is only a single k NN-CR model trained on all of \mathcal{D} , the adversary must attack the whole training set.

4.7.2 Certifying Non-Unit Costs by Construction. Since IBLs are a poor candidate to marry with ensembles, we need an alternative approach to certify a model's robustness beyond $r = 1$. Given the dearth of existing methods (known to us), we fill in the gap and propose a simple, general-purpose method to *certify robustness against arbitrary deletions*.

To be clear, this is a (slightly) restricted version of the full threat model considered so far, which allows arbitrary insertions and deletions. Nonetheless, this restricted threat model still has broad applicability. For example, an adversary may only be able to insert poisoned instances into a training set but not delete clean ones [Che+17; Liu+17; Sha+18; Wal+21; HL22a].

Motivated by Cook and Weisberg’s [CW82] classic *case deletion diagnostics*, we use a constructive proof to certify a (sub)model’s pointwise robustness under instance deletions. Consider training $(n + 1)$ deterministic models – one model using full set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and another n models on each of the leave-one-out subsets $\mathcal{D} \setminus (\mathbf{x}_i, y_i)$ for all $i \in [n]$. If all $(n + 1)$ trained models make the same prediction (e.g., a value not exceeding ξ for some \mathbf{x}_{te}), then by construction, the model trained on all of \mathcal{D} has, at minimum, $r = 2$ for arbitrary deletions. Lemma 4.11 generalizes the above for an arbitrary number of deletions $r < n$.

Lemma 4.11. *For $\mathbf{x}_{te} \in \mathcal{X}$, training set \mathcal{D} where $2^{\mathcal{D}}$ is its power set, $r \in [|\mathcal{D}| - 1]$, and $\xi \in \mathbb{R}$, denote a deterministic model trained on subset $D \subseteq \mathcal{D}$ as f_D . Given $\forall_{D' \in 2^{\mathcal{D}}} |D'| < r \implies f_{D \setminus D'}(\mathbf{x}_{te}) \leq \xi$, then for any $\tilde{\mathcal{D}} \subset \mathcal{D}$, if $f_{\tilde{\mathcal{D}}}(\mathbf{x}_{te}) > \xi$ then at least r instances from \mathcal{D} were deleted in $\tilde{\mathcal{D}}$.*

A strength of Lemma 4.11 is its flexibility; it can be adapted to any model class, including both classifiers and regressors. Its clear limitation is its computational complexity.

Computational Complexity: Certifying $r > 1$ requires training $\mathcal{O}(n^{(r-1)})$ models; this is a one-time, amortized cost.

Consider separately the cost to certify each prediction. During inference, the $\mathcal{O}(n^{(r-1)})$ models are checked. While this may be problematic in some cases, it should be contextualized. Recall that Sec. 4.4 explores IBLs like k NN, which have inference

complexity $\mathcal{O}(n)$. Therefore, our method to certify $r = 2$ has the same time complexity *during inference* as k NN.

4.7.3 More Submodels vs. Weighted Costs. Increasing submodel count L and using weighted costs are partially conflicting approaches to increase R . A natural question is which of the two approaches yields better certified robustness. Above, we explain why increasing L is a poor strategy for IBLs. For ensembles, increasing L generally means that each submodel is trained on fewer data.

As an intuition, consider when $\forall_l r_l = 2$. For a unit-cost ensemble to certify equivalent R , submodel count L must about double, and each submodel is trained on 50% fewer data, which can significantly degrade submodel performance. In contrast, weighted costs with $r = 2$ reduces submodel training set sizes by 1 (Lem. 4.11). By training weighted submodels on much more data, weighted submodels can outperform submodels from ensembles with larger L . This improved submodel performance can in turn improve certified robustness.

4.8 Evaluation

This section evaluates our five primary certified regressors: k NN-m certified regression (k NN-CR), partitioned certified regressors PCR & W-PCR as well as overlapping certified regressors OCR & W-OCR. Additional experimental results are in the supplement, including full k NN-CR certification plots (C.1.3). Additional results also appear in the supplemental materials of the original paper [HL23c, Sec. I].

To the extent of our knowledge, we propose the first pointwise certified regression methods that make no assumptions about the test distribution or model architecture. Without a clear baseline, we compare our five methods against each other. As a reference on the clean-data performance, we report each dataset’s “*uncertified*” (non-robust) accuracy.

4.8.1 Experimental Setup. For brevity, most evaluation setup details (e.g., hyperparameters) are deferred to suppl. Sec. D.1 with a brief summary below. For each experiment in this section, at least ten trials were performed. To improve readability, we only report the mean values below with variances in suppl. Sec. C.1.

Dataset Configuration Each (sub)model is trained on $\frac{1}{q}$ -th of the training data, where $q \in \mathbb{Z}_{>0}$. For k NN-CR, always $q = 1$. For our four ensemble-based methods (W-)PCR and (W-)OCR, q can significantly affect the ensemble’s accuracy and best-case certified robustness (R). As such, for each dataset, we report results with three different q values. For all ensembles, function h_{tr} partitions training set \mathcal{D} u.a.r.⁹

For our partitioned regressors (W-)PCR, \mathcal{D} is split into q blocks, with $L = q$. For our overlapping regressors (W-)OCR, we followed Wang et al.’s [WLF22a] overlapping certified classifier evaluation. Specifically, \mathcal{D} is partitioned into qd blocks u.a.r. All blocks have fixed spread degree $d > 1$ (see Tab. 1), and h_f assigns blocks to submodels at random. Hence, each overlapping ensemble necessarily has $L = qd$ submodels.

Submodel Architectures To demonstrate their generality, our ensemble methods use two different submodel architectures, namely ridge regression and XGBoost [CG16] gradient-boosted forests. Model determinism is enforced via a fixed random seed. Below, we report whichever submodel architecture performed the best on a held-out validation set.

Evaluation Metric For each test instance $(\mathbf{x}_{te}, y_{te})$, our goal is to determine the largest pointwise certified robustness R that guarantees $\xi_l \leq f(\mathbf{x}_{te}) \leq \xi_u$. Throughout

⁹Each dataset’s largest q value maximized the ensembles’ certified robustness (R). For each dataset, we also report small and medium q values. In practice, q should be as small as possible while guaranteeing sufficient robustness given each application’s maximum anticipated poisoning rate.

this evaluation, $\xi_l := y_{te} - \xi$ and $\xi_u := y_{te} + \xi$. These bounds are w.r.t. each test example’s *true target value* y_{te} , not predicted value $f(\mathbf{x}_{te})$. Therefore, a large certified robustness R means that the *prediction is both accurate and stable*. Here, *error threshold* ξ may be a specific fraction (e.g., 15%) of each instance’s target value y_{te} or a fixed value for the entire dataset (see Table 1). In practice, the appropriate ξ value is application specific.

Our evaluation metric is *certified accuracy*, which is the fraction of instances with robustness $R \geq \psi$ for $\psi \in \mathbb{N}$. In each trial, we calculated the certified robustness (R) for at least 100 random test instances and report the mean certified accuracy across all trials. See suppl. Sec. C.1 for the certified accuracy variance. Note that existing certified classifiers were previously evaluated using certified accuracy [Jia+22a; LF21; WLF22a] with $\xi = 0$, i.e., the predicted label must match true label y_{te} .

Datasets Our certified regressors are evaluated on six datasets: five regression and one binary classification. Like previous work [BHL23], the datasets are preprocessed where all categorical features are transformed into one-hot encodings. Table 1 summarizes each dataset’s key attributes, including its size, error threshold (ξ), ensemble submodel architecture, etc. A brief description of each dataset is below.

Ames [Coc11] and Austin [Pie21] estimate home prices in two American cities. Diamonds [Wic16] predicts a diamond’s price based on attributes such as cut, color, carat, etc. Weather [Mal+21] estimates ground temperature (in degrees Celsius) using date, time, and longitude/latitude information. For computational efficiency, Weather’s size was downsampled by $10\times$ u.a.r. Life [Raj21] estimates life expectancy (in years) using epidemiological and other national statistics. Spambase [Hop+17] is a binary classification dataset where emails are labeled as either spam or ham. Spambase’s positive training prior is 39%.

Table 1. **Evaluation Dataset Summary:** Training set size (n), data dimension, overlapping spread degree (d), error threshold (ξ), and submodel architecture for the six datasets. Error thresholds that are a percentage of each instance’s true target value are denoted $X\% \cdot y$. Alternate ξ values are evaluated in the original paper [HL23c, Fig. 9].

Dataset	Size (n)	Dim.	Deg. (d)	Error (ξ)	Submodel
Ames	2.6k	253	17	15% $\cdot y$	XGBoost
Austin	12k	315	13	15% $\cdot y$	XGBoost
Diamonds	48k	26	9	15% $\cdot y$	Ridge
Weather	308k	140	5	3° C	Ridge
Life	2.6k	204	13	3 years	XGBoost
Spambase	4.1k	57	17	0	Ridge

Certifying $r > 1$ For our two weighted methods, W-PCR and W-OCR, our evaluation attempts to certify each submodel’s robustness against deletions up to $r = 2$.

4.8.2 Analyzing the Certified Accuracy. Figure 7 visualizes our methods’ mean certified accuracy for the six datasets. For brevity, the corresponding numerical values, including variance, are deferred to Sec. C.1. Below, we briefly summarize the experiments’ primary takeaways.

Takeaway #1: *Both our ensemble and IBL regressors certify non-trivial fractions of the training set.* For the Ames and Life datasets, W-OCR certifies 50% of test predictions up to 1% training set corruption. Similarly, k NN-CR certifies 30% of predictions on Ames up to 4% corruption. These certified guarantees are without explicit assumptions about the data distribution or, in the case of ensembles, the submodel’s architecture. For other datasets, we certify predictions up to hundreds or thousands of training set modifications.

Takeaway #2: *Ensemble regressors have better peak performance.* Across all six datasets, the ensemble-based methods all had better peak certified accuracy than

k NN-CR. The performance gap was as large as $3.5\times$ and is not primarily due to feature dimension as k NN-CR performed worst on Diamonds, which has the smallest dimension by far.

Takeaway #3: W-OCR achieves the largest certified robustness (R) amongst the ensemble methods. This is observed using each dataset’s largest q value. For all six datasets, there is a (significant) gap between W-OCR (–) and our second-best ensemble method, W-PCR (–).

Takeaway #4: k NN-CR achieves the largest certified robustness. Although k NN-CR certifies (far) fewer instances than the ensembles, for instances that it can certify, its maximum robustness R is generally far larger than that of W-OCR. For example with Weather, k NN-CR’s maximum R is $5\times$ larger than W-OCR’s. Suppl. Sec. C.1.3 best visualizes this trend in its plots of k NN-CR’s full certified accuracy.

Takeaway #5: W-OCR achieves state-of-the-art certified accuracy for binary classification. While regression is this work’s primary focus, recall that binary classification can be solved by a regressor. For binary classification, k NN-CR’s R is identical to Jia et al.’s [Jia+22a] k NN classifier; PCR certifies equivalent robustness as DPA, and OCR very closely approximates Wang et al.’s [WLF22a] overlapping method. Observe that W-OCR outperforms the unweighted ensembles and k NN-CR on Spambase’s [Hop+17] two largest q values. Note that Spambase’s maximum q value cannot be increased further without severely degrading submodel performance. This provides empirical evidence for Sec. 4.7.3’s claim that a weighted strategy can outperform increasing submodel count L .

Takeaway #6: q can significantly affect certified accuracy. Previous certified classifier evaluations [Jia+22a; WLF22a; LF21] under-explore q ’s role. Those works primarily evaluate vision datasets where the training data is supplemented by (1) using

a pre-trained model to extract much better features [JCG21; Jia+22a] or (2) using vision data augmentation [LF21; WLF22a]. For the tabular datasets evaluated here, such options are not as available.

Without such augmentation, increasing q can significantly degrade an ensemble’s peak certified accuracy. As an example, the ensembles’ peak certified accuracy can decline by up to 28% between training a model on all of \mathcal{D} versus a dataset’s maximum q value (compare to uncertified accuracy \cdots in Fig. 7). Therefore, when thinking about certified classifiers and regressors, always consider the potential benefits of external (clean) data augmentation. For instance, in our experiments, XGBoost certified ensembles’ accuracy improved by multiple percent when using *mixup* data augmentation [Zha+18].

4.9 Conclusions

This chapter describes a novel reduction from certified regression to certified classification based on median perturbation. Applying this reduction, we propose six new certified regressors that require no assumptions about the data distribution or model architecture. As improved voting-based, certified classifiers are proposed in the future, our reduction can be applied to those methods too. This enables certified regression to keep pace with the rapid advances in certified classification.

While this work focuses on certified defenses against poisoning attacks, some certified *evasion* defenses also rely on voting-based guarantees [LF20b; Jia+22a]. Our reduction from certified regression to certified classification applies to those certified evasion defenses as well.

Lastly, our empirical results show that improved certified guarantees are possible when the unit-cost assumption is replaced by our tighter weighted analysis. These certification gains apply to both classification and regression, but Sec. 4.7.2’s approach

is computationally expensive. We advocate for better methods that efficiently certify beyond $r = 1$.

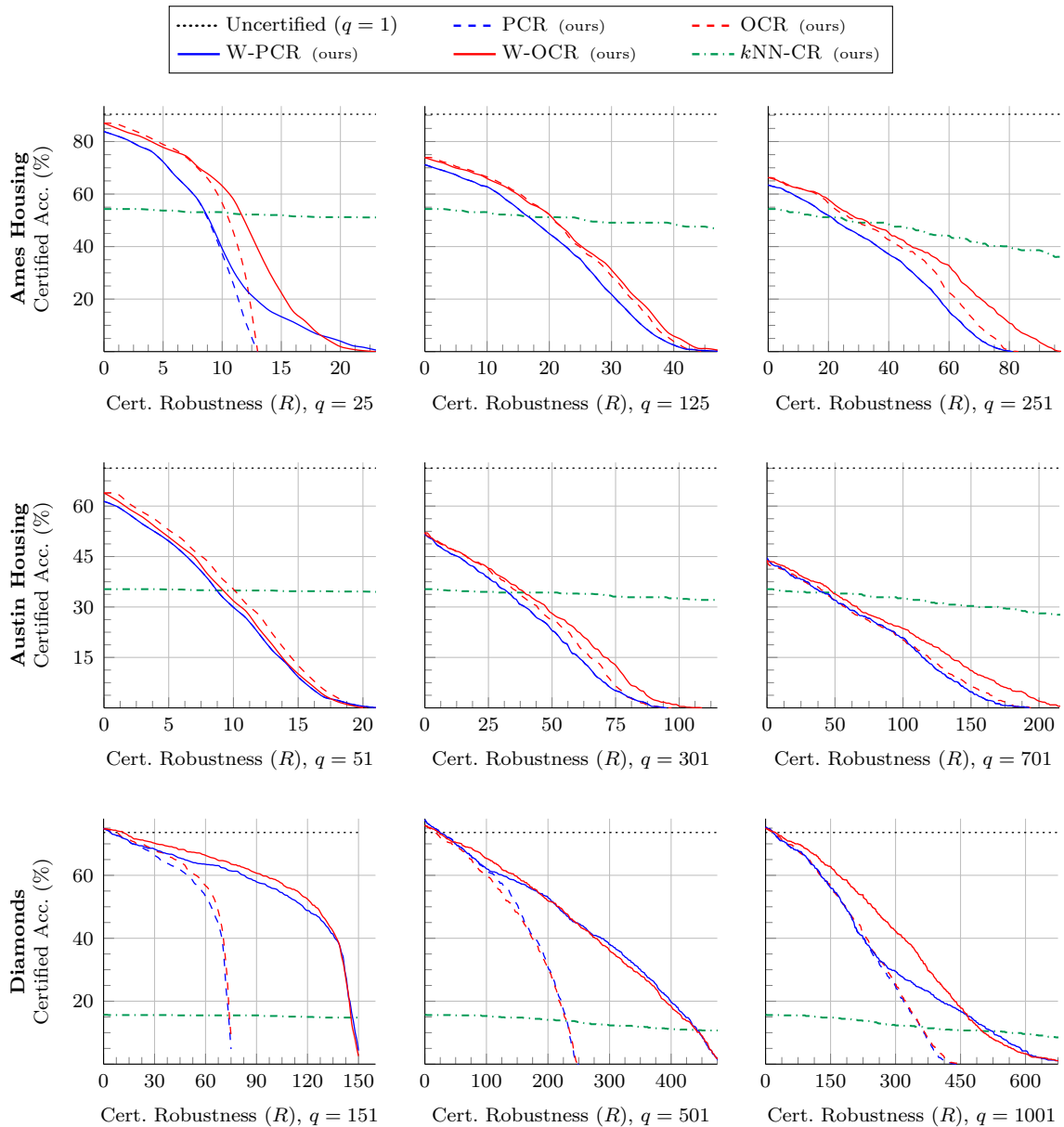


Figure 7. Certified Accuracy: Mean certified accuracy (larger is better) for our five primary certified regressors. k NN-CR is always trained on all of training set \mathcal{D} (i.e., $q = 1$). Ensemble submodels are trained on $\frac{1}{q}$ -th of \mathcal{D} , with three q values tested per dataset. The x-axis is clipped to enhance readability; see suppl. Sec. C.1.3 for k NN-CR’s full results. The best performing method depends on the target certified robustness R . For smaller R values, W-OCR achieves the best certified accuracy. For larger R values, k NN-CR outperforms the ensemble methods. This result aligns with previous findings on certified classification [Jia+22a]. Sec. 4.8.2 summarizes these experiments’ primary takeaways. *Figure continued on the next page.*

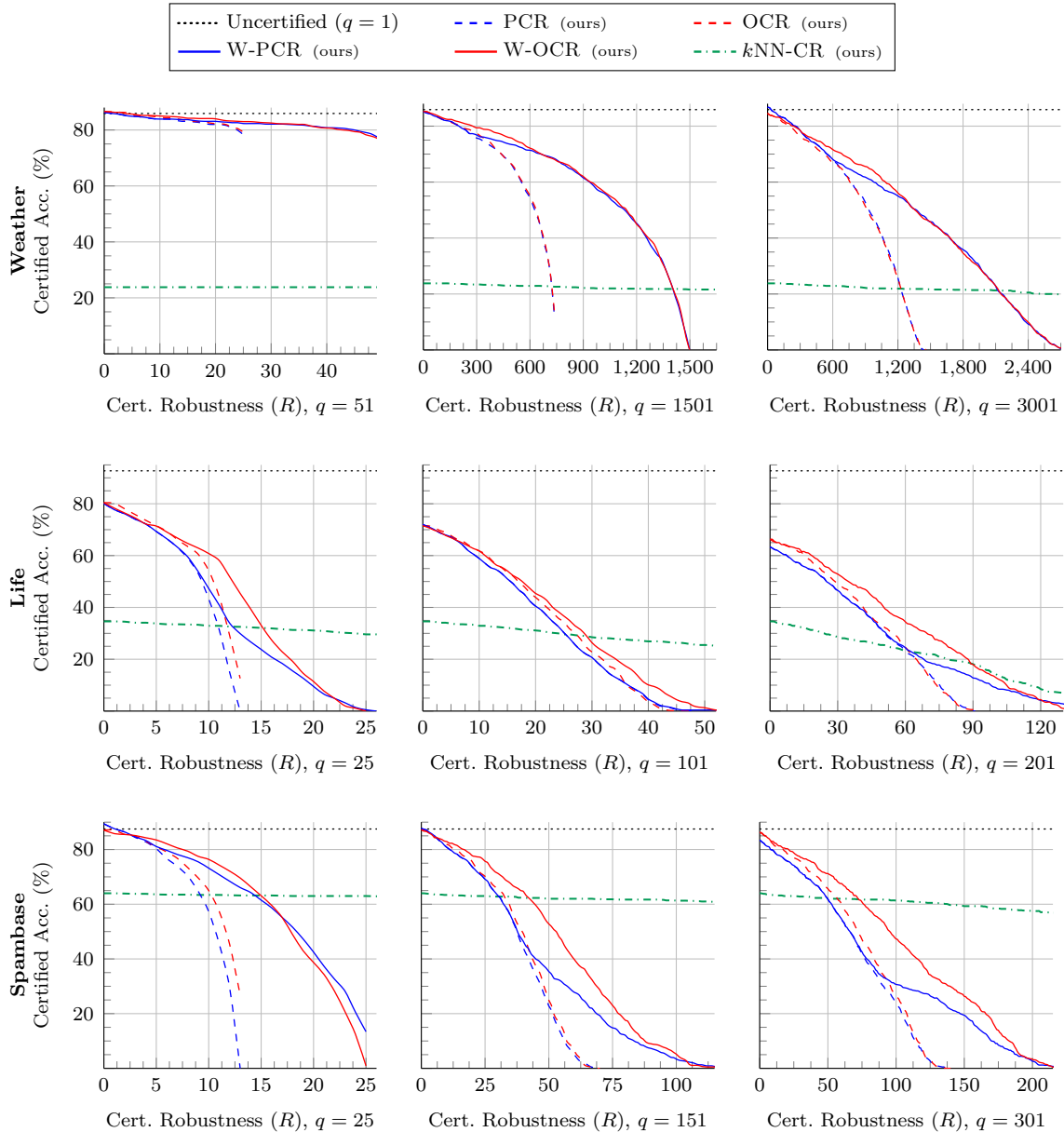


Figure 7. Certified Accuracy (cont.): Mean certified accuracy (larger is better) for our five primary certified regressors. k NN-CR is always trained on all of training set \mathcal{D} (i.e., $q = 1$). Ensemble submodels are trained on $\frac{1}{q}$ -th of \mathcal{D} , with three q values tested per dataset. The x-axis is clipped to enhance readability; see suppl. Sec. C.1.3 for k NN-CR’s full results. The best performing method depends on the target certified robustness R . For smaller R values, W-OCR achieves the best certified accuracy. For larger R values, k NN-CR outperforms the ensemble methods. This result aligns with previous findings on certified classification [Jia+22a]. Sec. 4.8.2 summarizes these experiments’ primary takeaways. See Sec. C.1 for the numerical results, including variance.

CHAPTER 5

CERTIFIED DEFENSE AGAINST A UNION OF ℓ_0 ATTACKS

This chapter contains previously published, coauthored material [HL23a]. Hammoudeh developed the primary method, developed all code, conducted all experiments, and wrote the manuscript. Lowd provided supervision, editorial suggestions, and input on experiments.

Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023. URL: <https://arxiv.org/abs/2302.11628>

This chapter focuses on ℓ_0 or *sparse* attacks,¹ where an adversary controls an unknown subset of the features. By certifying robustness w.r.t. the number of perturbed features, ℓ_0 analysis is particularly well-suited to heterogeneous (tabular) data where the features have different types (e.g., numerical, categorical) or scales. Moreover, ℓ_0 defenses provide provable robustness against real-world patch attacks [LF20a]. Several certified ℓ_0 defenses have been proposed [Lee+19; LF20b; Cal+21; Jia+22b], but these methods apply to evasion only, which can be limiting. For example, consider a distributed sensor network where each (tabular) feature is independently measured by a different sensor. Under this type of *vertical partitioning* where features are sourced from multiple parties, an attacker that controls a single feature (i.e., sensor) can partially perturb every instance – training and test – up to 100% poisoning rate [LDD21; Wei+22]. Existing ℓ_0 evasion defenses do not certify robustness over any training perturbation rendering them moot under such an attack. Moreover,

¹The ℓ_0 norm of vector $\mathbf{x} \in \mathbb{R}^d$ equals $\|\mathbf{x}\|_0 := |\{j : x_j \neq 0\}|$, where $x_j \in \mathbb{R}$ denotes the j -th dimension of \mathbf{x} . Put simply, \mathbf{x} ’s ℓ_0 -norm equals the number of non-zero dimensions in \mathbf{x} . Intuitively, ℓ_0 attacks bound the number of dimensions of \mathbf{x} whose perturbation value can be non-zero.

existing ℓ_0 defenses could not be combined with instance-wise poisoning defenses here since typically, the latter are only provably robust under small poisoning rates, e.g., $\leq 1\%$ [WLF22b; Rez+23].

To address these limitations, we propose *feature partition aggregation* (FPA) – a certified sparse defense jointly robust against both training and test feature perturbations. FPA uses a model ensemble approach, where each submodel is trained on a disjoint feature set, meaning any adversarially perturbed feature – training or test – affects at most one submodel prediction. Hence, FPA guarantees robustness over the *union* of ℓ_0 evasion, backdoor, and poisoning attacks – a strictly stronger guarantee than existing ℓ_0 methods [LF20b]. In our empirical evaluation, FPA’s certified median guarantees are up to $4\times$ larger than state-of-the-art ℓ_0 defenses [Jia+22b] with little to no decrease in classification accuracy; FPA is also up to $3,000\times$ faster. In other words, FPA provided additional dimensions of ℓ_0 robustness essentially for free. Our primary contributions are summarized below; additional theoretical analysis and all proofs are in the supplement.

- We define a new robustness paradigm we term *certified feature robustness* that generalizes ℓ_0 (sparse) robustness to encompass training set feature perturbations.
- We propose feature partition aggregation, a certified feature defense that uses an ensemble of submodels trained on disjoint feature sets. We detail two certification schemes – a simple one based on plurality voting and the other based on multi-round elections.
- We empirically evaluate FPA on two classification and two regression datasets. FPA provided simultaneously larger and stronger median guarantees than the

state-of-the-art certified ℓ_0 defenses while also being 2 to 3 orders of magnitude faster.

5.1 Preliminaries

Notation ℓ_0 norm $\|\mathbf{w}\|_0$ is the number of non-zero elements in vector \mathbf{w} . Given some matrix \mathbf{A} , denote its j -th column as \mathbf{A}_j . In a slight abuse of notation, let $\mathbf{A} \ominus \mathbf{A}' := \{j : \mathbf{A}_j \neq \mathbf{A}'_j\}$ denote the set of column *indices* over which equal-size matrices \mathbf{A} and \mathbf{A}' differ. Similarly, let $\mathbf{v} \ominus \mathbf{v}' \subseteq [|\mathbf{v}|]$ denote the set of *dimensions* where vectors \mathbf{v} and \mathbf{v}' differ.

Denote the training set’s *feature matrix* as $\mathbf{X} := [\mathbf{x}_1 \cdots \mathbf{x}_n]^\top$ where $\mathbf{X} \in \mathbb{R}^{n \times d}$, and denote the label vector $\mathbf{y} := [y_1, \dots, y_n]$. *feature partition aggregation* (FPA) is an ensemble of L *submodels* (see Figure 8). A *decision function* aggregates the L submodel predictions to form f ’s overall prediction. The model architecture and decision function combined dictate how a prediction’s *certified robustness* is calculated. For instance (\mathbf{x}, y) , let $g_l(\mathbf{x}, y)$ be the l -th submodel’s *logit* value for label y , where $g_l : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. Let $f_l(\mathbf{x})$ denote the l -th submodel’s predicted *label* for \mathbf{x} , where $f_l : \mathcal{X} \rightarrow \mathcal{Y}$ and $f_l(\mathbf{x}) := \arg \max_{y \in \mathcal{Y}} g_l(\mathbf{x}, y)$. Throughout this work, all ties are broken by selecting the label with the smallest index.

Feature set $[d]$ is *partitioned* across FPA’s L submodels. Let $\mathcal{S}_l \subset [d]$ be the features used by the l -th submodel where $\bigsqcup_{l=1}^L \mathcal{S}_l = [d]$. In other words, each FPA submodel considers a fixed, disjoint subset of the features for all training and test instances. The l -th submodel’s training set, D_l , consists of: label vector \mathbf{y} and the \mathcal{S}_l columns in \mathbf{X} . FPA submodels are *deterministic*, meaning fixing D_l , \mathcal{S}_l , and \mathbf{x} , in turn, fixes label $f_l(\mathbf{x})$ and logits $\forall_y g_l(\mathbf{x}, y)$.

Given \mathbf{x} and y , the pointwise *submodel vote count* is $\dot{c}_y(\mathbf{x}) := \sum_{l=1}^L \mathbb{1}[f_l(\mathbf{x}) = y]$. The *plurality* and *runner-up* labels receive the most and second-most votes (resp.),

i.e., $y_{\text{pl}} = \arg \max_{y \in \mathcal{Y}} \dot{c}_y(\mathbf{x})$ and $y_{\text{ru}} = \arg \max_{y \in \mathcal{Y} \setminus y_{\text{pl}}} \dot{c}_y(\mathbf{x})$. The pointwise *submodel vote gap* between labels $y, y' \in \mathcal{Y}$ is

$$\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) := \dot{c}_y(\mathbf{x}) - \dot{c}_{y'}(\mathbf{x}) - \mathbb{1}[y' < y], \quad (5.1)$$

with the indicator function used to break ties. Let $\ddot{c}_y(\mathbf{x}; y') := \sum_{l=1}^L \mathbb{1}[g_l(\mathbf{x}, y) > g_l(\mathbf{x}, y')]$ be y 's *logit vote count* w.r.t. $y' \in \mathcal{Y}$. The pointwise *logit vote gap* for y w.r.t. y' is

$$\text{GAP}_{\text{logit}}(y, y'; \mathbf{x}) := \ddot{c}_y(\mathbf{x}; y') - \ddot{c}_{y'}(\mathbf{x}; y) - \mathbb{1}[y' < y]. \quad (5.2)$$

Below, \mathbf{x} is dropped from GAP_{vote} and $\text{GAP}_{\text{logit}}$ when the feature vector of interest is clear from context.

Threat Model Given arbitrary (\mathbf{x}, y) , the attacker's objective is to ensure that $y \neq f(\mathbf{x})$. The adversary achieves this objective via two methods: (1) modify training features \mathbf{X} or (2) modify test instance \mathbf{x} 's features. An adversary may use either method individually or both methods jointly. An attacker can *perturb up to 100% of the training instances*.

Our Objective For arbitrary (\mathbf{x}, y) , determine the *certified feature robustness*, R (defined below). Note that *pointwise* guarantees certify the robustness of each instance (\mathbf{x}, y) individually.

Def. 5.1. Certified Feature Robustness *Given training set (\mathbf{X}, \mathbf{y}) , model f' trained on $(\mathbf{X}', \mathbf{y})$, and arbitrary feature vector $\mathbf{x}' \in \mathcal{X}$, certified feature robustness $R \in \mathbb{N}$ is a pointwise, deterministic guarantee w.r.t. instance (\mathbf{x}, y) where $|\mathbf{X} \ominus \mathbf{X}' \cup \mathbf{x} \ominus \mathbf{x}'| \leq R \implies y = f'(\mathbf{x}')$.*

Certified robustness R is not w.r.t. individual feature values. Rather, certified feature robustness provides a stronger guarantee allowing all values of a feature – training and test – to be perturbed.

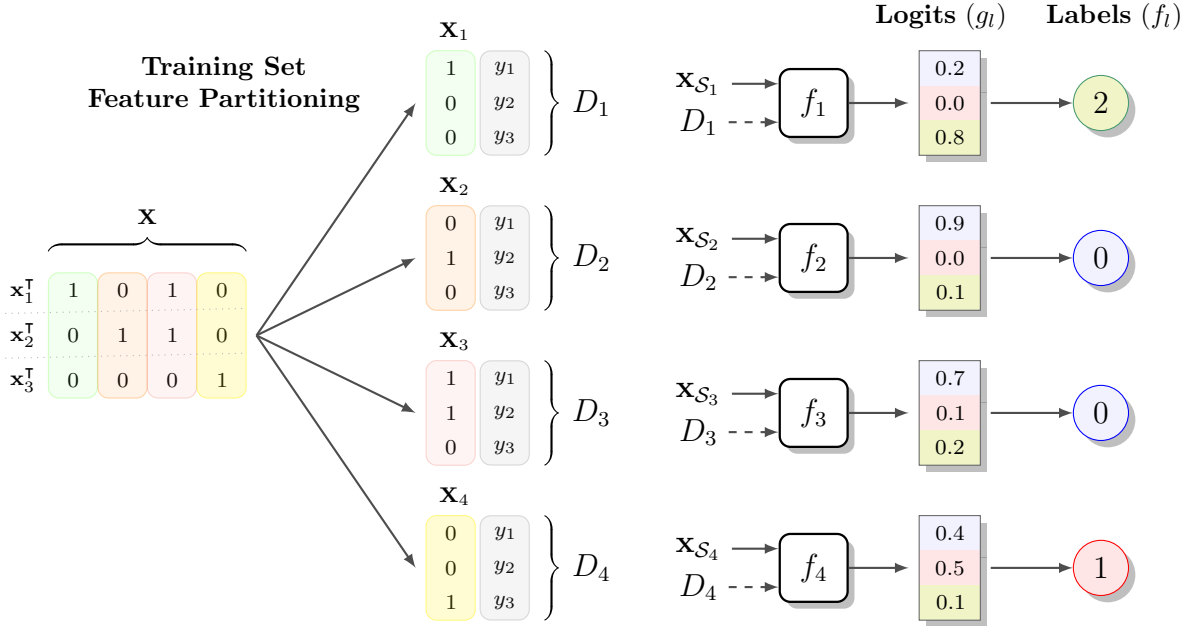


Figure 8. **Feature partition aggregation example** prediction for: test instance $\mathbf{x} \in \mathcal{X}$, $n = 3$, $d = 4$, and $|\mathcal{Y}| = 3$. Feature partitioning across $L = 4$ submodels, where the l -th submodel uses only feature dimensions $\mathcal{S}_l = \{l\} \subset [4]$ and training set D_l , i.e., the tuple containing the l -th column of feature matrix \mathbf{X} (denoted \mathbf{X}_l) and label vector $\mathbf{y} := [y_1, y_2, y_3]$. \mathbf{x}_{S_l} denotes the subvector of \mathbf{x} restricted to the feature dimensions in \mathcal{S}_l . Plurality label $y_{\text{pl}} = 0$; runner-up label $y_{\text{ru}} = 1$; and run-off label $y_{\text{ro}} = 0$. Under the plurality voting decision function (Sec. 5.3.1), $f(\mathbf{x})$ has certified feature robustness $R_{\text{pl}} = 0$. With run-off (Sec. 5.3.2), $f(\mathbf{x})$'s certified feature robustness is $R_{\text{ro}} = 1$.

5.2 Related Work

FPA marries ideas from two classes of certified adversarial defenses, which are discussed below. A more detailed discussion of related work in the full paper [HL23b; HL23a].

ℓ_0 -Norm Certified Evasion Defenses Representing the work most closely related to ours, these methods certify ℓ_0 -norm robustness (also known as “sparse robustness”), which we formalize below.

Def. 5.2. ℓ_0 -Norm Certified Robustness Given model f , $\alpha \in (0, 1)$, and arbitrary feature vector $\mathbf{x}' \in \mathcal{X}$, ℓ_0 -norm certified robustness $\rho \in \mathbb{N}$ is a pointwise guarantee w.r.t. instance (\mathbf{x}, y) where if $\|\mathbf{x} - \mathbf{x}'\|_0 \leq \rho$, then $y = f(\mathbf{x}')$ with probability at least $1 - \alpha$.

There are two main differences between certified ℓ_0 -norm robustness (Def. 5.2) and our certified feature robustness (Def. 5.1). (1) ℓ_0 -norm methods are not certifiably robust against any adversarial training perturbations (e.g., poisoning and backdoors). (2) ℓ_0 -norm robustness guarantees are *probabilistic*, while our feature guarantees are deterministic. Put simply, our certified feature guarantees are *strictly stronger* than ℓ_0 -norm guarantees.

Randomized ablation (RA) is the state-of-the-art certified ℓ_0 -norm defense [LF20b; Jia+22b]. RA adapts ideas from *randomized smoothing* [CRK19] to ℓ_0 evasion attacks [LF20b]. Specifically, RA creates a *smoothed classifier* by repeatedly evaluating different *ablated inputs*, each of which *keeps* a random subset of the features unchanged and masks out (*ablates*) all other features. RA’s *ablated training* generally permits only stochastically-trained, parametric model architectures. At inference, certifying a single prediction with RA requires evaluating up to 100k ablated inputs [Lee+19; Jia+22b]. Jia et al. [Jia+22b] improve RA’s guarantees via new certification analysis that is tight for top-1 predictions, meaning Jia et al.’s version of RA always performs at least as well as the original. Jia et al. [Jia+22b] also extend RA to certify ℓ_0 -norm robustness for top- k predictions.

Certified patch robustness is a restricted form of ℓ_0 -norm robustness where the perturbed test features are constrained to a specific, contiguous shape, e.g., square [MY21]. Existing patch defenses include (de)randomized smoothing (DRS) [LF20a] – a specialized version of randomized ablation for patch attacks. Like RA, DRS performs ablated training and inference. By assuming a single patch shape, the number of

possible attacks becomes linear in d , allowing DRS to only evaluate $\mathcal{O}(d)$ ablations during inference; this derandomizes the ablation set, making DRS’s patch guarantees deterministic.² More recently, Metzen and Yatsura [MY21] propose, BAGCERT – a certified patch defense that is less sensitive to patch shape than DRS. Note any certified feature or ℓ_0 -norm defense (e.g., FPA, RA) is also a certified patch defense, given the former’s stronger guarantees.

Instance-wise Certified Poisoning Defenses The second class of related defenses certify robustness under the arbitrary insertion or deletion of entire *instances* in the training set [Che+22; WF23] – generally a small poisoning rate (e.g., $\leq 1\%$). Like FPA, most instance-wise poisoning defenses are voting-based [JCG21; Jia+22a; WLF22a]. For example, *deep partition aggregation* (DPA) randomly partitions the training *instances* across an ensemble of L submodels [LF21]. More recently, Rezaei et al. [Rez+23] propose *run-off elections*, a novel decision function for DPA that can improve DPA’s certified robustness by several percentage points. While certified instance-wise poisoning defenses show promise, they are still vulnerable to test perturbations – even of a single feature.

5.3 Certifying Feature Robustness

Our certified defense, feature partition aggregation (FPA), can be viewed as the *transpose* of Levine and Feizi’s [LF21] deep partition aggregation (DPA). Both defenses are (1) ensembles, (2) rely on voting-based decision functions, and (3) partition the training set; the key difference is in the partitioning operation. DPA horizontally partitions the set of training *instances* (rows of feature matrix \mathbf{X}), enabling DPA to certify *instance-wise* robustness. In contrast, FPA vertically partitions along an

²(De)randomized smoothing’s deterministic guarantees do not scale to RA which considers $\mathcal{O}(2^d)$ attacks.

orthogonal dimension – the feature set (columns of \mathbf{X}) – enabling FPA to certify *feature-wise* robustness. Intuitively, *partitioning along orthogonal dimensions means that DPA and FPA certify orthogonal types of robustness*. Training FPA submodels on disjoint feature subsets (e.g., Figure 8) entails that a perturbed feature affects, at most, one submodel prediction. FPA leverages this property to certify feature robustness R . Below we describe two FPA *decision functions*: (1) a simpler scheme using plurality voting and (2) an enhanced multi-round voting procedure specialized for multiclass classification. The decision function combined with FPA’s architecture dictates how our robustness guarantee is calculated.

5.3.1 Feature Robustness Under Plurality Voting. For $\mathbf{x} \in \mathcal{X}$, the *plurality voting* decision function defines the model prediction as $f(\mathbf{x}) := y_{\text{pl}}$, i.e., the label that receives the most submodel votes. A successful attack requires perturbing enough submodels to change y_{pl} . Specifically, each submodel perturbation decreases the submodel vote gap (GAP_{vote}) between y_{pl} and the adversary’s selected label by two. Hence, the minimum number of submodel perturbations equals half the vote gap between y_{pl} and runner-up label y_{ru} . Thm. 5.3 formalizes this idea as a deterministic feature robustness guarantee. Eq. (5.3)’s decomposed form is similar to other voting-based certified defenses, including DPA [LF21; Jia+22a; HL23c].

Theorem 5.3. Certified Feature Robustness with Plurality Voting *For feature partition $\mathcal{S}_1, \dots, \mathcal{S}_L$, let f be an ensemble of L submodels using the plurality-voting decision function, where the l -th submodel uses the features in \mathcal{S}_l . For instance (\mathbf{x}, y) , the pointwise certified feature robustness is*

$$R_{\text{pl}} := \left\lfloor \frac{\text{GAP}_{\text{vote}}(y_{\text{pl}}, y_{\text{ru}})}{2} \right\rfloor. \quad (5.3)$$

Understanding Thm. 5.3 More Intuitively Let $\mathcal{A}_{\text{tr}} \subseteq [d]$ be the set of features (i.e., dimensions) an attacker modified in the training set, and let $\mathcal{A}_{\mathbf{x}} \subseteq [d]$ be the set of features the attacker modified in instance \mathbf{x} . As long as $|\mathcal{A}_{\text{tr}} \cup \mathcal{A}_{\mathbf{x}}| \leq R$, the adversarial perturbations did not change the model prediction. The union over the perturbed feature sets entails that a feature perturbed in both training and test counts only once against guarantee R . Put simply, there is no double counting of a perturbed feature. Thm. 5.3’s certified guarantees are implicitly agnostic to the ℓ_0 attack type. Certified feature robustness R applies equally to an ℓ_0 evasion attack ($\mathcal{A}_{\mathbf{x}}$ only) as it does to ℓ_0 poisoning (\mathcal{A}_{tr} only). Thm. 5.3’s guarantees also encompass more complex ℓ_0 backdoor attacks ($\mathcal{A}_{\text{tr}} \cup \mathcal{A}_{\mathbf{x}}$).

5.3.2 Feature Robustness Under Run-Off Elections. Under plurality voting, only submodels that predict either y_{pl} or y_{ru} are considered when determining the certified feature robustness (Eq. (5.3)). In other words, submodels predicting other labels essentially contribute nothing to plurality voting’s pointwise guarantees. Decision functions that leverage these “wasted” submodels may certify larger guarantees (see Figure 8). For instance, Rezaei et al. [Rez+23] propose *run-off elections*, an enhanced two-round DPA decision function for multiclass classification.³ Since FPA and DPA share the same basic architecture (excluding the partitioning dimension), run-off can be directly combined with FPA to improve our certified robustness.

We now describe run-off. Our presentation is similar to Rezaei et al.’s [Rez+23] except we standardize the formulation to align with previous work and to correct an error in Rezaei et al.’s preprint version. Formally, run-off’s decision function procedure is:

³Run-off only changes the decision function; no training or model architecture changes are required.

Round #1: Determine plurality and runner-up labels y_{pl} and y_{ru} (resp.) as above.

Round #2: Set run-off prediction y_{RO} to either label y_{pl} or y_{ru} based on the logit vote gap where

$$f(\mathbf{x}) = y_{\text{RO}} := \begin{cases} y_{\text{pl}} & \text{GAP}_{\text{logit}}(y_{\text{pl}}, y_{\text{ru}}) \geq 0 \\ y_{\text{ru}} & \text{Otherwise} \end{cases}. \quad (5.4)$$

Under run-off, ensemble prediction y_{RO} can only be perturbed in two ways: (1) overtake y_{RO} in round #2 or (2) eject y_{RO} from round #1's top-two labels. Run-off's certified (feature) robustness is lower bounded by whichever case takes fewer submodel perturbations. We discuss these two cases separately below; Thm. 5.4 combines these analyses to define run-off's overall feature robustness.

Case #1: Overtake y_{RO} in Round #2 Let $\tilde{y}_{\text{RO}} := \{y_{\text{pl}}, y_{\text{ru}}\} \setminus y_{\text{RO}}$ denote the label not selected in round #2. For a label y to overtake y_{RO} in round #2, y must simultaneously satisfy two requirements: (a) be in round #1's top-two labels (in turn ejecting \tilde{y}_{RO} from the top two) and (b) receive more logit votes than y_{RO} in round #2. Hence, the certified robustness for this case is bounded by whichever of these requirements requires more feature perturbations. Therefore, an attacker may control up to

$$R_{\text{RO}}^{\text{Case1}} := \min_{y \in \mathcal{Y} \setminus y_{\text{RO}}} \max \left\{ \left\lfloor \frac{\text{GAP}_{\text{vote}}(\tilde{y}_{\text{RO}}, y)}{2} \right\rfloor, \left\lfloor \frac{\text{GAP}_{\text{logit}}(y_{\text{RO}}, y)}{2} \right\rfloor \right\} \quad (5.5)$$

features, without y_{RO} being overtaken in round #2 (Lemma B.3).

Case #2: Eject y_{RO} from Round #1's Top-Two Labels In round #1, a label y is preferred over a different label y' iff $\text{GAP}_{\text{vote}}(y, y') \geq 0$ (Lemma B.2). Therefore, ejecting y_{RO} from round #1's top-two labels requires perturbing sufficient submodels

such that two labels have negative submodel vote gaps w.r.t. y_{RO} . Let dp be a function that takes two submodel vote gaps (e.g., $i, j \in \mathbb{N}$) and returns y_{RO} 's round #1 certified feature robustness. Recall that perturbing a submodel vote from y_{RO} to a different y decreases $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y)$ by 2; observe that this same submodel perturbation also decreases $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y')$ by 1 for all $y' \in \mathcal{Y} \setminus \{y_{\text{RO}}, y\}$. Combining these interactions, dp can be defined recursively as

$$\text{dp}[i, j] = \begin{cases} 0 & \max\{i, j\} \leq 1 \text{ and } (i, j) \neq (1, 1) \\ 1 + \min\{\text{dp}[i - 2, j - 1], \text{dp}[i - 1, j - 2]\} & \text{Otherwise} \end{cases}, \quad (5.6)$$

where the base case ensures at least one submodel vote gap is non-negative. Therefore, case #2's total certified robustness is

$$R_{\text{RO}}^{\text{Case2}} := \min_{y, y' \in \mathcal{Y} \setminus y_{\text{RO}}} \text{dp}[\text{gap}_y, \text{gap}_{y'}] \quad (5.7)$$

where $\text{gap}_{y^*} = \max\{0, \text{GAP}_{\text{vote}}(y_{\text{RO}}, y^*)\}$ (Lemma B.4). Recursive formulations like Eq. (5.6) are solvable using classic dynamic programming. $\mathcal{O}(L^2)$ -space matrix dp is prepopulated once, meaning the incremental lookup cost is only $\mathcal{O}(1)$ and $R_{\text{RO}}^{\text{Case2}}$'s total time complexity $\mathcal{O}(|\mathcal{Y}|^2)$.

Combining Cases #1 and #2 to Certify Feature Robustness Thm. 5.4 provides the certified feature robustness for an FPA prediction using the run-off decision function. Intuitively, an optimal attacker selects whichever of the two cases above requires fewer feature perturbations; hence, Eq. (5.8) below takes the minimum of $R_{\text{RO}}^{\text{Case1}}$ and $R_{\text{RO}}^{\text{Case2}}$.

Theorem 5.4. Certified Feature Robustness with Run-off *For feature partition $\mathcal{S}_1, \dots, \mathcal{S}_L$, let f be an ensemble of L submodels using the run-off decision function, where the l -th submodel uses only the features in \mathcal{S}_l . Then, for instance (\mathbf{x}, y) , the*

pointwise certified feature robustness is

$$R_{\text{RO}} = \min\{R_{\text{RO}}^{\text{Case1}}, R_{\text{RO}}^{\text{Case2}}\}. \quad (5.8)$$

5.3.3 Advantages of Feature Partition Aggregation. Below, we summarize FPA’s advantages over state-of-the-art certified ℓ_0 -norm defense randomized ablation (RA). These advantages apply irrespective of whether FPA uses plurality voting or run-off.

- (1) **Stronger Guarantees** FPA’s certified feature robustness guarantee (Def. 5.1) is strictly stronger than RA’s ℓ_0 -norm guarantee (Def. 5.2). First, FPA’s guarantees apply equally to ℓ_0 evasion, poisoning, and backdoor attacks while RA only applies to evasion. Second, FPA’s guarantees are deterministic while RA’s guarantees are only probabilistic.
- (2) **Faster** RA requires up to 100k forward passes to certify one prediction. FPA requires only L forward passes – one for each submodel – where $L < 200$ in general. FPA certification is, therefore, orders of magnitude faster than RA.
- (3) **Model Architecture Agnostic** RA’s feature ablation is specialized for parametric models like neural networks and generally prevents the use of tree-based models like gradient-boosted decision trees (GBDTs). By contrast, FPA supports any submodel architecture.

5.4 Feature Partitioning Strategies

The certification analysis above holds irrespective of the feature partitioning strategy. However, how the features are partitioned can have a *major* impact on the size of FPA’s certified guarantees. Below, we very briefly describe two insights into the properties of good feature partitions.

Insight #1 *Ensure sufficient feature information is available to each submodel.*

Each incorrect submodel or logit vote cancels out a correct one, meaning the goal should be to simultaneously maximize the number of correct submodel predictions and minimize incorrect ones. In other words, robustness is maximized when all submodels perform well, and feature information is divided equally.

Insight #2 *Limit information loss due to feature partitioning.* Feature partitioning is lossy from an information theoretic perspective. Fixing L , some partitions are more lossy than others, and good partitions limit the information lost.

5.4.1 Feature Partitioning Paradigms. Applying the above insights, we propose two general feature partitioning paradigms. In practice, the partitioning strategy is essentially a hyperparameter tunable on validation data. The validation set need not be clean so long as the perturbations are representative of the test distribution.

Balanced Random Partitioning Given no domain-specific knowledge, each feature’s expected information content is equal. *Balanced random partitioning* assigns each submodel a disjoint feature subset sampled uniformly at random, with subsets differing in size by at most one. Random partitioning has two primary benefits. First, each submodel has the same a priori expected information content. Second, random partitioning can be applied to any dataset. FPA with random partitioning is usually a good initial strategy and empirically performs quite well.

Deterministic Partitioning One may have application-related insights into quality feature partitions. For example, consider feature partitioning of images. Features (i.e., pixels) in an image are ordered, and that structure can be leveraged to

design better feature partitions. Often the most salient features are clustered in an image’s center. To ensure all submodels are high-quality, each submodel should be assigned as many highly salient features as possible. Moreover, adjacent pixels can be highly correlated, i.e., contain mostly the same information. Given a fixed set of pixels to analyze, the information contained in those limited features should be maximized, so a good strategy can be to select a subset of pixels spread uniformly across the image. Put simply, for images, random partitioning can have larger information loss than deterministic strategies. The supplement of the original paper empirically compares random and deterministic partitioning [HL23b; HL23a]. In short, a simple strided strategy that distributes features regularly across an image tends to work well for vision. Formally, given d pixels and L submodels, the l -th submodel’s feature set under *strided partitioning* is $\mathcal{S}_l = \{j \in [d] : j \bmod L = l - 1\}$.

5.5 Evaluation

Our empirical evaluation is modeled after Levine and Feizi’s [LF20b] evaluation of randomized ablation. For clarity, additional results are deferred to the supplement including the base (non-robust) accuracy for each dataset (C.2.1) and the full numerical results (C.2.2 & C.2.3). Additional results appear in the original paper including: hyperparameter sensitivity analysis, plurality voting vs. run-off comparison, random vs. deterministic partitioning comparison, and model training times [HL23b; HL23a].

5.5.1 Experimental Setup. Due to space, most evaluation setup details are deferred to suppl. Sec. D.2 with a brief summary below. We evaluate FPA with both the plurality-voting (Sec. 5.3.1) and run-off (Sec. 5.3.2) decision functions.

Baselines Randomized ablation (RA) is FPA’s most closely related work and the primary baseline below. We report the performance of both Levine and Feizi’s [LF20b] original version of RA as well as Jia et al.’s [Jia+22b] improved version, where

the certification analysis is tight for top-1 predictions. RA performs feature ablation during training and inference. Each ablated input keeps e randomly selected features unchanged and masks out the remaining $(d - e)$ features; RA evaluates up to 100,000 ablated inputs to certify each prediction. Recall that RA’s ℓ_0 -norm robustness only applies to evasion attacks (Def. 5.2), while FPA provides strictly stronger feature guarantees that cover manipulation of both training and test data (Def. 5.1).

We also compare FPA to three certified patch defenses: *(de)randomized smoothing* [LF20a], *patch interval bound propagation* (IBP) [Chi+20], and BAGCERT [MY21].

Performance Metrics Certified defenses generally trade-off robustness and (clean) accuracy. Hence, following Levine and Feizi’s [LF20b] evaluation of randomized ablation, performance is measured using two complementary metrics: (1) *median certified robustness*, the median value of the certified robustness across a dataset’s entire test set with misclassified instances assigned robustness $-\infty$ and (2) *classification accuracy*, the fraction of test predictions classified correctly. Below, R_{med} and ρ_{med} denote the median certified feature robustness (Def. 5.1) and ℓ_0 -norm robustness (Def. 5.2) resp. *Mean certification time* measures the time to certify a single prediction. Performance is also quantified using *certified accuracy*, i.e., the fraction of correctly-classified test instances that satisfy some specific robustness criterion; this criterion can be patch robustness or certified robustness of at least $\psi \in \mathbb{N}$.

Datasets We compare the methods on standard datasets used in data poisoning evaluation. First, following Levine and Feizi’s [LF20b] evaluation of baseline RA,

we consider MNIST and CIFAR10⁴ where each feature corresponds to one (RGB) pixel. Second, Chapter 4 proves that certified regression *reduces* to certified *binary* classification when median is used as the regressor’s decision function. We apply their reduction to both FPA and RA where for instance (\mathbf{x}, y) and hyperparameter $\xi \in \mathbb{R}_{\geq 0}$, the goal is to certify that $y - \xi \leq f(\mathbf{x}) \leq y + \xi$. We consider two tabular regression datasets evaluated in Chapter 4. (1) Weather [Mal+21] predicts the temperature using features such as date, longitude, and latitude ($\xi = 3^\circ\text{C}$). (2) Ames [Coc11] predicts housing prices using features such as square footage ($\xi = 15\%y$). These two regression datasets serve as a stand-in for vertically partitioned data, which are commonly tabular and, as mentioned at the beginning of this chapter, are particularly vulnerable to our union of ℓ_0 attacks threat model. Note run-off and plurality voting are identical under binary classification so we only report FPA’s plurality voting regression results.

Model Architectures For vision datasets MNIST and CIFAR10, all methods used convolutional neural networks. Gradient-boosted decision trees (GBDTs) generally work exceptionally well on tabular data [BHL23] so for regression datasets Weather and Ames, FPA used LightGBM GBDTs [Ke+17]. In contrast, RA’s feature ablation prevents the use of tree-based models like GBDTs, so RA instead used linear models for these two datasets (Hammoudeh and Lowd [HL23c] also used linear models for Weather). Even when restricted to linear submodels, FPA still had better median robustness and classification accuracy than RA; see suppl. Tables C.23 and C.24.

Feature Partitioning Strategy For CIFAR10 and MNIST, FPA used strided feature partitioning; each submodel considered the full image dimensions with any

⁴Existing certified poisoning defenses do not evaluate on full ImageNet due to the high training cost [Web+23; LF21; Jia+22a; WLF22a; WLF22b; Rez+23].

Table 2. **Median certified robustness.** Each dataset’s best performing method is in **bold**. Our median robustness was 20–30% larger for classification and 3 to 4× larger for regression while simultaneously providing stronger guarantees. For detailed results, see Sec. C.2.2.

Dataset	FPA (ours)		RA	
	Plural	Run-Off	[LF20b]	[Jia+22b]
CIFAR10	11	13	7	10
MNIST	9	12	8	10
Weather	4	–	0	1
Ames	3	–	1	1

Table 3. **Classification accuracy** (% – larger is better). We report FPA’s accuracy at both RA’s (middle, **bold**) and FPA’s (blue) best median robustness levels. At RA’s best median robustness, FPA had better classification accuracy for all four datasets. For full results, see Sec. C.2.2.

Dataset	FPA (ours)				RA [Jia+22b]	
	R_{med}	Acc.	R_{med}	Acc.	ρ_{med}	Acc.
CIFAR10	13	62.4	10	75.0	10	64.7
MNIST	12	87.2	10	96.1	10	93.1
Weather	4	76.1	1	85.3	1	75.2
Ames	3	65.5	1	84.6	1	67.2

pixels not in \mathcal{S}_l set to 0. For Weather and Ames, FPA used balanced random partitioning as the tabular features are unordered.

Hyperparameters Hyperparameters L (FPA’s submodel count) and e (RA’s kept feature count) control the corresponding method’s robustness vs. accuracy tradeoff. When optimizing patch and median robustness, hyperparameters L and e were tuned on validation data.⁵

Patch Robustness We consider two CIFAR10 patch attacks: (1) a 5×5 pixel square [LF20a] and (2) all 24-pixel rectangles (e.g., 1×24 pixels, 24×1 , 2×12 , etc.), reporting each method’s minimum and maximum certified accuracies across the eight valid shapes [MY21].

5.5.2 Main Results. Table 2 summarizes the median certified robustness for FPA and baseline RA. Tables 3 and 5 compare the methods and RA’s classification accuracy and mean certification time (resp.); note that, for clarity, these two tables

⁵Secs. C.2.2 & C.2.3 compare each method’s certified accuracy across a range of hyperparameter settings.

only report the results for Jia et al.’s [Jia+22b] better performing version of baseline RA. Table 4 analyzes FPA as a patch defense. We briefly summarize the experiments’ takeaways below. See suppl. Secs. C.2.2 and C.2.3 for the full numerical results, including comparing the methods at additional robustness levels.

Takeaway #1 *FPA simultaneously provided larger and stronger median robustness guarantees than RA.* As Table 2 details, FPA’s median certified robustness was 20–30% larger than RA for classification and 3 to 4× larger for regression. Importantly, FPA’s certified feature guarantees apply to evasion, poisoning, and backdoor attacks, while baseline RA only covers evasion attacks.

Takeaway #2 *FPA’s median robustness gains come at little cost in classification accuracy.* Table 3 reports FPA’s classification accuracy at two robustness levels: (1) FPA’s best median robustness (**blue**) and (2) RA’s best median robustness (**bold**). Table 3 also reports RA’s classification accuracy at its own best median robustness (last column). For CIFAR10 at median robustness of 10 pixels, FPA’s classification accuracy was 10.2 percentage points (pp) better than RA (75.0% vs. 64.7%). At $R_{\text{med}} = 13$, FPA’s CIFAR10 classification accuracy was 62.4%, only 2.3pp lower than RA’s classification accuracy at $\rho_{\text{med}} = 10$. For Weather at median robustness 1, FPA’s classification accuracy was 10.1pp better than RA (85.3% vs. 75.2%); even at $R_{\text{med}} = 4$, FPA’s classification accuracy was 76.1%, 0.9pp better than RA at $\rho_{\text{med}} = 1$. For MNIST at median robustness 10, FPA’s classification accuracy was 3pp better than RA (96.1% vs. 93.1%). At $R_{\text{med}} = 12$, FPA’s MNIST classification accuracy was 5.9pp lower than RA’s classification accuracy at $\rho_{\text{med}} = 10$ (87.2% vs. 93.1%).

Table 4. **CIFAR10 certified patch accuracy** (% – larger is better) for FPA, RA, and three dedicated patch defenses. FPA is competitive despite making fewer assumptions and providing stronger guarantees than patch defenses.

Method	24 Pixel Rect.		Square
	Min.	Max.	5×5
	FPA Plurality ($L = 180$, ours)	← 38.53 →	
FPA Run-Off ($L = 180$, ours)	← 41.60 →		40.95
Rand. Ablation [LF20b]	← 28.95 →		28.21
Rand. Ablation [Jia+22b]	← 37.31 →		36.43
(De)Random. Smoothing	0.0	72.68	57.69
BAGCERT	43.11	60.17	59.95
Patch IBP	—	—	30.30

Table 5. **Mean certification time** in seconds for FPA and Jia et al.’s [Jia+22b] randomized ablation (RA). FPA is 2 to 3 orders of magnitude faster than baseline RA.

Dataset	RA [Jia+22b]		FPA (ours)		Speedup
	e	Time	L	Time	
	CIFAR10	15	5.4E+0	115	
MNIST	25	6.8E−1	60	2.9E−3	235×
Weather	45	3.1E−1	21	1.0E−4	3,134×
Ames	60	3.8E−1	21	3.5E−4	1,082×

Takeaway #3 *FPA certifies predictions 2 to 3 orders of magnitude faster than RA.*

Table 5 compares the mean certification times using the hyperparameter settings with the best median robustness. To certify one prediction, Jia et al.’s [Jia+22b] improved RA evaluates 100k ablated inputs. In contrast, FPA requires exactly L forward passes per prediction (one per submodel).

Takeaway #4 *FPA provides strong patch robustness without any assumptions about patch shape or the number of patches.* As Table 4 details, FPA certifies 41.6% of CIFAR10 predictions at $R = 24$ perturbed pixels (2.3% of d) – regardless of patch shape or the number of patches. In contrast, (de)randomized smoothing’s [LF20a] (BS, $s = 12$) 24-pixel certified accuracy varies between 0% to 72.7% based on patch shape alone. BAGCERT’s certified accuracy drops as low as 43.1% for 24-pixel column and row patches – only 1.5pp better than FPA. Unlike FPA, patch defenses’ certified accuracy guarantees decline further or even evaporate under (1) multiple patches, (2) training data perturbations, and (3) amorphous shapes. While less effective in some settings than dedicated patch defenses that make stronger assumptions and

weaker guarantees, FPA is still competitive, providing patch guarantees essentially for free.

Takeaway #5 *FPA is the first integrated defense to provide significant pointwise robustness guarantees over the union of evasion, backdoor, and poisoning attacks – ℓ_0 or otherwise.* Consider CIFAR10 ($n = 50,000$) where FPA feature robustness $R \geq 25$ (Table 4) certifies 41.0% of predictions’ robustness against 1.25M arbitrarily perturbed pixels. In contrast, the only other certified defense robust over the union of evasion, backdoor, and poisoning attacks [Web+23] certifies the equivalent of 3 or fewer arbitrarily perturbed CIFAR10 pixels (i.e., a total training and test ℓ_2 perturbation distance of ≤ 3). Moreover, FPA certifies $R \geq 7$ for 35.1% of Weather predictions ($n > 3M$ – Table C.27) – a pointwise guaranteed robustness of up to 21M arbitrarily perturbed feature values.

5.6 Conclusions

This paper proposes *feature partition aggregation* – a certified defense against the union of ℓ_0 evasion, poisoning, and backdoor attacks. FPA provided stronger and larger robustness guarantees than the state-of-the-art ℓ_0 evasion defense, randomized ablation. FPA’s certified feature guarantees are particularly important for *vertically partitioned* data where a single compromised data source allows an attacker to arbitrarily modify a limited number of features for all instances – training and test. To our knowledge, FPA is the first integrated defense providing non-trivial pointwise robustness guarantees against the union of evasion, poisoning, and backdoor attacks – ℓ_0 or otherwise [Web+23]. Future work remains to develop other ℓ_p certified defenses over this union of attack types.

CHAPTER 6

IDENTIFYING POISONING AND BACKDOOR ATTACK TARGETS WHILE MITIGATING THE ATTACK

This chapter contains previously published and unpublished coauthored material [HL21; HL22a; HL22b; BHL23]. Hammoudeh developed the primary method, developed all code, conducted all experiments, and wrote the manuscript. Lowd provided supervision, editorial suggestions, and proposed some supplemental experiments.

Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security. CCS’22*. Los Angeles, CA: Association for Computing Machinery, 2022. URL: <https://arxiv.org/abs/2201.10055>

Zayd Hammoudeh and Daniel Lowd. “Training Data Influence Analysis and Estimation: A Survey”. In: *arXiv* (2022). arXiv: [2212.04612](https://arxiv.org/abs/2212.04612) [cs.LG]

Zayd Hammoudeh and Daniel Lowd. “Simple, Attack-Agnostic Defense Against Targeted Training Set Attacks Using Cosine Similarity”. In: *Proceedings of the 3rd ICML Workshop on Uncertainty and Robustness in Deep Learning. UDL’21*. 2021

Jonathan Brophy et al. “Adapting and Evaluating Influence-Estimation Methods for Gradient-Boosted Decision Trees”. In: *Journal of Machine Learning Research* 24 (2023), pp. 1–48. URL: <http://jmlr.org/papers/v24/22-0449.html>

Targeted training set attacks manipulate an ML system’s prediction on one or more *target* test instances by maliciously modifying the training data [Muñ+17; Sha+18; Agh+20; Sal+20; Hua+20; Gei+21; Wal+21]. Targeted attacks require very few corrupted instances [Wal+21], and their effect on the test error is quite small, making them harder to detect [Che+17].

Existing poisoning and backdoor defenses (Sec. 3.2) change the training procedure to mitigate the impact of an attack but provide little insight into an attacker’s goals, methods, or identity. As Sec. 3.3 explains, knowledge about an attacker is essential to anticipating attacks, designing targeted defenses, and building defenses outside the ML system.

This chapter’s defense against training set attacks focuses on the related goals of learning more about an attacker and stopping their attacks. We achieve this through a pair of related tasks:

1. *target identification*: identifying the target of a training set attack, which may provide insight into the attacker’s goals and how to defend against them; and
2. *adversarial-instance identification*: identifying the malicious instances that constitute the training set attack.

We are not aware of any work that studies poisoning and backdoor attack target identification beyond very simple settings.

Our *key insight* is the synergistic interplay between the two tasks above. If attackers can add only a limited number of training instances (as is often the case) [Che+17; Wal+21], then these malicious instances must be highly influential to change target predictions. Thus, targets are those test instances with an unusual number of highly influential training examples. In contrast, non-targets tend to have many weak

influences and few very strong ones. Thus, if we can (1) determine which training instances influence which predictions and (2) detect anomalies in this distribution, then we can jointly solve both tasks.

Unfortunately, determining which training instances are responsible for which model behaviors is provably hard for black-box models like neural networks [BR92]. *Influence estimators* [KL17; Yeh+18; Pru+20; FZ20; Che+21; BHL23; HL22b] quantify how much each training example contributes to a particular prediction. However, current influence analysis methods often perform poorly [BPF21; HL22b; ZZ22].

This chapter identifies a weakness common to gradient-based influence estimators [KL17; Yeh+18; Pru+20; Che+21; HL22b]: they induce a *low-loss penalty* that implicitly ranks confidently-predicted training instances as uninfluential. As a result, existing influence estimators can systematically overlook (groups of) highly influential, low-loss instances. This chapter provides a simple fix – *renormalization* that removes the low-loss penalty. Our new *renormalized influence estimators* consistently outperform the original estimators in both adversarial and non-adversarial settings. The most effective of these renormalized estimators, *gradient aggregated similarity* (GAS), often detects 100% of malicious training instances with no clean-data false positives.

Our framework for identifying targets of training set attacks, FIT, compares the distribution of influence values across test instances checking for anomalies. More concretely, FIT marks as potential targets those test instances with an unusual number of highly influential instances as explained above. Next, FIT mitigates the attack’s effect by removing exceptionally influential training instances associated with the target(s). Since mitigation considers only targets, training instance outliers that are “helpful” to non-targets are unaffected. This *target-driven mitigation* has a positive or

neutral effect on clean data yet is highly effective on adversarial data where finding even a single target suffices to disable the attack on almost all other targets.

By relying on influence estimation and not the properties of a particular attack or domain, GAS and FIT are *attack agnostic*; GAS and FIT can apply equally well to different attack types, including poisoning and backdoor attacks. Our approach works across data domains from CNN image classifiers to speech recognition to even text transformers.

In addition to learning more about the attack and attacker, *target identification enables targeted mitigation*. Defenses against poisoning and backdoor attacks implement countermeasures that affect predictions on *all* instances – not just the very few targets. Certified defenses can substantially degrade performance, in some cases causing up to $10\times$ more errors on clean data [Fow+21; HL23c; LF21].

Our chapter’s contributions are enumerated below.

1. We identify a weakness common to all gradient-based influence estimators and provide a simple renormalization correction that addresses this weakness.
2. Inspired by influence estimation, we propose GAS – a renormalized influence estimator that is highly adept at identifying influential groups of training instances.
3. Leveraging techniques from anomaly detection and *robust statistics*, we extend GAS into a general framework for identifying targets of training set attacks, FIT.
4. We use GAS in a target-driven data sanitizer that mitigates attacks while removing very few clean instances.

5. We demonstrate the effectiveness and attack agnosticism of GAS and FIT on a diverse set of attacks and data modalities, including speech recognition, vision, and text – even against an adaptive attacker that attempts to evade our method.

Below, we first specify this chapter’s threat model and defender objective. Section 6.2 reviews existing gradient-based influence estimators [HL22b]. Then in Section 6.3, we show how existing gradient-based influence estimators are inadequate for identifying (groups of) highly influential training instances. We also introduce our renormalization fix for influence estimation and describe four renormalized influence estimators. Section 6.4 builds on these improved influence estimates to define a framework for identifying the targets of an attack and mitigating the attack’s effect. Section 6.5 demonstrates the effectiveness of our methods.

6.1 Preliminaries

Notation Let $\hat{z}_{te} := (\mathbf{x}_{te}, \hat{y}_{te})$ be any *a priori* unknown test instance, where \hat{y}_{te} is the final model’s predicted label for \mathbf{x}_{te} . Observe that \hat{y}_{te} may not be \mathbf{x}_{te} ’s *true label*. Notation $\hat{\cdot}$ (e.g., \hat{z} , \hat{g}) denotes that the final predicted label $\hat{y} = f(\mathbf{x}; \theta_T)$ is used in place of \mathbf{x} ’s true label y .

Threat Model. The attacker crafts an *adversarial set* of perturbed instances, $\mathcal{D}_{adv} \subset \mathcal{D}$. Denote the *clean training set* $\mathcal{D}_{cl} := \mathcal{D} \setminus \mathcal{D}_{adv}$. We only consider successful attacks, as defined below.

Attacker Objective and Knowledge Let $\mathcal{X}_{targ} := \{\mathbf{x}_j\}_{j=1}^m$ be a set of target feature vectors with shared true label $y_{targ} \in \mathcal{Y}$. The attacker crafts \mathcal{D}_{adv} to induce the model to mislabel all of \mathcal{X}_{targ} as *adversarial label* y_{adv} . $\hat{\mathcal{Z}}_{targ} := \{(\mathbf{x}_j, y_{adv}) : \mathbf{x}_j \in \mathcal{X}_{targ}\}$ denotes the *target set* and $\hat{z}_{targ} := (\mathbf{x}_{targ}, y_{adv})$ an arbitrary target instance. To avoid detection, the attacked model’s clean-data performance should be (essentially)

unchanged. *Data poisoning* attacks only perturb adversarial set \mathcal{D}_{adv} . Target feature vectors are unperturbed/benign [BNL12; Muñ+17; Wal+21; Jag+21]. *Clean-label poisoning* leaves labels unchanged when crafting \mathcal{D}_{adv} from seed instances [Zhu+19; YHL23a; YHL23b]. *Backdoor* attacks perturb the features of both \mathcal{D}_{adv} and $\mathcal{X}_{\text{target}}$ – often with the same *adversarial trigger* (e.g., change a specific pixel to maximum value). Generally, these triggers can be inserted into any test example targeted by the adversary, making most backdoor attacks *multi-target* ($|\widehat{\mathcal{Z}}_{\text{target}}| > 1$) [TLM18; Gu+19; Lin+20; Web+23; YHL23a; YHL23b]. \mathcal{D}_{adv} ’s labels may also be changed.

To ensure the strongest adversary, the attacker knows any pre-trained initial parameters. Where applicable, the attacker also knows the training hyperparameters and clean dataset \mathcal{D}_{cl} . Like previous work [Zhu+19; Web+23; Wal+21], the attacker does not know the training procedure’s random seed, meaning the attack must be robust to randomness in batch ordering or parameter initialization.

Defender Objective and Knowledge Let $\widehat{\mathcal{Z}}_{\text{te}}$ denote the set of test instances the defender is concerned enough about to analyze as potential targets.¹ Our goals are to (1) *identify* any attack targets in $\widehat{\mathcal{Z}}_{\text{te}}$, and (2) mitigate the attack by removing the adversarial instances \mathcal{D}_{adv} associated with those target(s). No assumptions are made about the modality/domain (e.g., text, vision) or adversarial perturbation. We do not assume access to clean validation data.

6.2 Review of Influence Analysis and Estimation

As Sec. 3.2.1 discusses, most existing defenses against poisoning and backdoor attacks assume highly restricted threat models. In contrast, this section seeks a method that is *attack agnostic*, to make it harder for an adversary to adapt against. We mitigate

¹Generally, there are far fewer potential targets ($\widehat{\mathcal{Z}}_{\text{te}}$) than possible test examples.



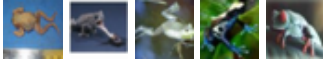
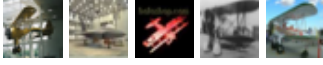
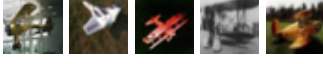




	<u>Method</u>	<u>AUPRC</u>	<u>Top-5 Highest Ranked</u>
Test Example  \hat{z}_{te}	Representer Point	0.030 ± 0.009	
	Influence Functions	0.029 ± 0.018	
	TracIn	0.140 ± 0.098	
	TracInCP	0.309 ± 0.260	
	Representer Point Renormalized (ours)	0.778 ± 0.144	
	Influence Functions Renormalized (ours)	0.215 ± 0.191	
	TracIn Renormalized (ours)	0.617 ± 0.115	
	GAS (ours) (TracInCP Renormalized)	0.977 ± 0.001	

Figure 9. Renormalized Influence: CIFAR10 & MNIST joint, binary classification for [frog] vs. [airplane & MNIST 0] with $|\mathcal{D}_{cl}| = 10,000$ & $|\mathcal{D}_{adv}| = 150$. Existing influence estimators (upper half) consistently failed to rank \mathcal{D}_{adv} 's MNIST training instances as highly influential on MNIST test instances. In contrast, all of our renormalized influence estimators (Section 6.3.3) outperformed their unnormalized version – with AUPRC improving up to 25 \times . Results averaged across 30 trials.

training set attacks by building upon influence estimation to identify the target(s) and adversarial set. We achieve agnosticism by building upon existing methods that are general and makes no assumption about the underlying attack. Specifically, we leverage influence analysis, which we formalize below. For a more complete discussion of influence analysis and estimation, see our detailed survey [HL22b].

Intuitively, in every successful attack, the inserted training instances change a model's prediction for specific input(s). If the attacker can only add a limited number of instances (e.g., 1% of \mathcal{D}), these inserted instances must be highly influential to achieve the attacker's objective.

Influence analysis's goal is to determine which training instances are most responsible for a model's prediction for a particular input [HL22b]. Influence is often viewed as a counterfactual: which instance (or group of instances) induces the biggest change when removed from the training data? While there are multiple definitions of influence, as detailed below, influence analysis methods can be broadly viewed as quantifying the (relative) responsibility of each training instance $z_i \in \mathcal{D}$ on some test prediction $f(\mathbf{x}_{\text{te}}; \theta_T)$.

Static influence estimators consider only the final model parameters θ_T . For example, Koh and Liang's [KL17] seminal work defines influence, $\mathcal{I}_{\text{IF}}(z_i, \hat{z}_{\text{te}})$, as the change in risk $\mathcal{L}(\hat{z}_{\text{te}}; \theta_T)$ if $z_i \notin \mathcal{D}$, i.e., the leave-one-out (LOO) change in test loss [CW82]. By assuming strict convexity and stationarity, Koh and Liang's *influence functions* estimator approximates the LOO influence as

$$\mathcal{I}_{\text{IF}}(z_i, \hat{z}_{\text{te}}) \approx \frac{1}{n} \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_T)^{\top} H_{\theta}^{-1} \nabla_{\theta} \mathcal{L}(z_i; \theta_T), \quad (6.1)$$

with H_{θ}^{-1} the inverse of risk Hessian $H_{\theta} := \frac{1}{n} \sum_{z_i \in \mathcal{D}} \nabla_{\theta}^2 \mathcal{L}(z_i; \theta_T)$.

Yeh et al. [Yeh+18]'s *representer point* static influence estimator exclusively considers the model's final, linear classification layer. All other model parameters are treated as a fixed feature extractor. Given final parameters θ_T , let \mathbf{f}_i denote \mathbf{x}_i 's penultimate feature representation (i.e., the *input* to the linear classification layer). Then the representer point influence of $z_i \in \mathcal{D}$ on \hat{z}_{te} is

$$\mathcal{I}_{\text{RP}}(z_i, \hat{z}_{\text{te}}) := -\frac{1}{2\lambda n} \left(\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}} \right) \langle \mathbf{f}_i, \mathbf{f}_{\text{te}} \rangle, \quad (6.2)$$

where $\lambda > 0$ is the weight decay (L_2) regularizer and $\langle \cdot, \cdot \rangle$ denotes vector dot product. Recall that a is the *output* of the model's linear classification layer, specifically here $a = f(\mathbf{x}_i; \theta_T)$. Scalar $\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}}$ is then the partial derivative of risk \mathcal{L} w.r.t. a 's y_i -th dimension.

Algorithm 1 TracIn, TracInCP, & GAS training phase

Input: Training set \mathcal{D} , iteration subset \mathcal{T} , iteration count T , learning rates η_1, \dots, η_T , and initial parameters θ_0

Output: Training parameters \mathcal{P}

- 1: $\mathcal{P} \leftarrow \emptyset$
 - 2: **for** $t \leftarrow 1$ **to** T **do**
 - 3: **if** $t \in \mathcal{T}$ **then**
 - 4: $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\eta_t, \theta_{t-1})\}$
 - 5: $\mathcal{B}_t \sim \mathcal{D}$
 - 6: $\theta_t \leftarrow \text{UPDATE}(\eta_t, \theta_{t-1}, \mathcal{B}_t)$
 - 7: **return** \mathcal{P}
-

Dynamic influence estimators measure influence based on how losses change during training. More formally, influence is quantified according to how batches $\mathcal{B}_1, \dots, \mathcal{B}_T$ affect model parameters $\theta_0, \dots, \theta_T$ and by consequence risks $\mathcal{L}(\cdot; \theta_0), \dots, \mathcal{L}(\cdot; \theta_T)$. For example, Pruthi et al.’s [Pru+20] *TracIn* estimates influence by “tracing” gradient descent – aggregating changes in \hat{z}_{te} ’s test loss each time training instance z_i ’s gradient updates parameters θ_t . For stochastic gradient descent (batch size $b = 1$), z_i ’s TracIn influence on \hat{z}_{te} is

$$\mathcal{I}_{\text{TracIn}}(z_i, \hat{z}_{\text{te}}) := \sum_{t=1}^T \mathbb{1}[z_i \in \mathcal{B}_t] \left(\mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1}) - \mathcal{L}(\hat{z}_{\text{te}}; \theta_t) \right), \quad (6.3)$$

where $\mathbb{1}[u]$ is the indicator function s.t. $\mathbb{1}[u] = 1$ if predicate u is true and 0 otherwise. Pruthi et al. approximate Eq. (6.3) as,

$$\mathcal{I}_{\text{TracIn}}(z_i, \hat{z}_{\text{te}}) \approx \sum_{z_i \in \mathcal{B}_t} \frac{\eta_t}{b} \left\langle \nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1}), \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1}) \right\rangle, \quad (6.4)$$

where η_t is iteration t ’s learning rate.

Alg. 1 details the minimal changes made to model training to support TracIn where $\mathcal{T} \subset \{1, \dots, T\}$ is a preselected *training iteration subset* and $\mathcal{P} := \{(\eta_t, \theta_{t-1}) : t \in \mathcal{T}\}$ contains the *serialized training parameters*. Alg. 2 outlines TracIn’s influence estimation procedure for *a priori* unknown test instance \hat{z}_{te} . Influence vector \mathbf{v} ($|\mathbf{v}| = n$) contains

Algorithm 2 TracIn influence estimation

Input: Training parameters \mathcal{P} , iteration subset \mathcal{T} , iteration count T , batches $\mathcal{B}_1, \dots, \mathcal{B}_T$, batch size b , and test example \hat{z}_{te}

Output: Influence vector \mathbf{v}

```
1:  $\mathbf{v} \leftarrow \vec{0}$  ▷ Initialize
2: for  $t \leftarrow 1$  to  $T$  do
3:   if  $t \in \mathcal{T}$  then
4:      $(\eta, \theta) \leftarrow \mathcal{P}[t]$  ▷ Equiv. to  $(\eta_t, \theta_{t-1})$ 
5:      $\hat{g}_{\text{te}} \leftarrow \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta)$ 
6:     for each  $z_i \in \mathcal{B}_t$  do ▷ Batch examples
7:        $g_i \leftarrow \nabla_{\theta} \mathcal{L}(z_i; \theta)$ 
8:        $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\eta}{b} \langle g_i, \hat{g}_{\text{te}} \rangle$  ▷ Unnormalized
9: return  $\mathbf{v}$ 
```

the TracIn influence estimates for each $z_i \in \mathcal{D}$. In practice, $|\mathcal{T}| \ll T$, and \mathcal{T} is evenly-spaced in $\{1, \dots, T\}$, meaning TracIn effectively treats multiple batches like a single model update.

Pruthi et al. also propose *TracIn Checkpoint* (TracInCP) – a more heuristic version of TracIn that considers *all* training examples at each checkpoint in \mathcal{T} – not just those instances in the intervening batches (see Alg. 3).² Formally,

$$\mathcal{I}_{\text{TracInCP}}(z_i, \hat{z}_{\text{te}}) := \sum_{t \in \mathcal{T}} \frac{\eta_t}{b} \left\langle \nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1}), \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1}) \right\rangle. \quad (6.5)$$

TracInCP is more computationally expensive than TracIn – with the slowdown linear w.r.t. the number of checkpoints per epoch.

A major advantage of TracIn and TracInCP over other estimators (e.g., influence functions) is that their only hyperparameter is iteration set \mathcal{T} , which we tuned based only on compute availability.

²Algorithm 3 combines two different methods TracInCP as well as GAS – our renormalized version of TracInCP discussed in Sec. 6.3.3.

6.3 Why Influence Estimation Often Fails and How to Fix It

Before addressing target identification, we first consider the related task of adversarial-instance identification. In the simplest case, if the attack’s target is known, then the malicious instances should be among the most influential instances for that target instance. In other words, *adversarial-instance identification reduces to influence estimation*. However, Sec. 6.2’s influence estimators share a common weakness that makes them poorly suited for this task: they all consistently rank confidently-predicted training instances as uninfluential. We illustrate this behavior below using a toy experiment. We then explain this weakness’s cause and propose a simple fix that addresses this limitation on adversarial and non-adversarial data, for all preceding estimators. Our fix is needed to successfully identify adversarial set \mathcal{D}_{adv} and, as detailed in Sec. 6.4, attack targets.

6.3.1 A Simple Experiment. Consider binary classification where clean set \mathcal{D}_{cl} is all `frog` and `airplane` training images in CIFAR10 ($|\mathcal{D}_{\text{cl}}| = 10,000$). To simulate a naive backdoor attack, adversarial set \mathcal{D}_{adv} is 150 randomly selected MNIST 0 images labeled as `airplane`. Clean data’s overall influence can be estimated indirectly by training only on \mathcal{D}_{cl} and observing the target set’s misclassification rate [FZ20]. This experiment used class pair `frog` and `airplane` because amongst the $\binom{10}{2}$ CIFAR10 class pairs, `frog` vs. `airplane`’s average MNIST *test* misclassification rate was closest to random (47.5% vs. 50% ideal). In contrast, when training on $\mathcal{D} := \mathcal{D}_{\text{adv}} \sqcup \mathcal{D}_{\text{cl}}$, MNIST 0 test instances were always classified as `airplane`, meaning \mathcal{D}_{adv} is overwhelmingly influential on MNIST predictions. MNIST is used instead of other CIFAR10 classes because the large (and simple [Sha+20]) difference between the data distributions leads to a strong signal that can be consistently learned from relatively few examples – much like backdoor or poisoning attacks [Yu+21].

We use this simple setup to evaluate different influence estimation methods. We trained 30 randomly-initialized state-of-the-art ResNet9 networks, and on each network, we performed influence estimation for a random MNIST 0 test instance to determine how well each estimator identified adversarial set \mathcal{D}_{adv} provided a known target.³ Given the large imbalance between the amount of clean and “adversarial” data, i.e., $|\mathcal{D}_{\text{adv}}| \ll |\mathcal{D}_{\text{cl}}|$, performance is measured using area under the precision-recall curve (AUPRC), which quantifies how well \mathcal{D}_{adv} ’s influence ranks relative to \mathcal{D}_{cl} . Precision-recall curves are preferred for highly-skewed classification tasks since they provide more insight into the false-positive rate [DG06].

Figure 9’s upper half shows how well each influence estimator in Section 6.2 identifies \mathcal{D}_{adv} , both quantitatively and qualitatively. Dynamic estimators significantly outperformed their static counterparts, with TracInCP the overall top performer. However, no influence estimator consistently ranked MNIST instances (i.e., \mathcal{D}_{adv}) in the top-5 most influential, with influence functions marking instances from the other class (**frog**) as most influential. Influence estimation’s poor performance here is particularly noteworthy as the task was designed to be unrealistically easy.

6.3.2 Why Influence Estimation Performs Poorly. Intra-training dynamics illuminate the primary cause of influence estimation’s poor performance in our toy experiment. Fig. 10 visualizes the median training loss of \mathcal{D}_{adv} and \mathcal{D}_{cl} at each training checkpoint. Also shown is the *gradient norm ratio*, which compares the median gradient magnitude of the adversarial and clean sets at each iteration, or formally

$$\text{GNR}_t := \frac{\text{med}\{\|\mathcal{L}(z; \theta_t)\| : z \in \mathcal{D}_{\text{adv}}\}}{\text{med}\{\|\mathcal{L}(z; \theta_t)\| : z \in \mathcal{D}_{\text{cl}}\}}. \quad (6.6)$$

³See supplemental Section D.3 for the complete experimental setup details.

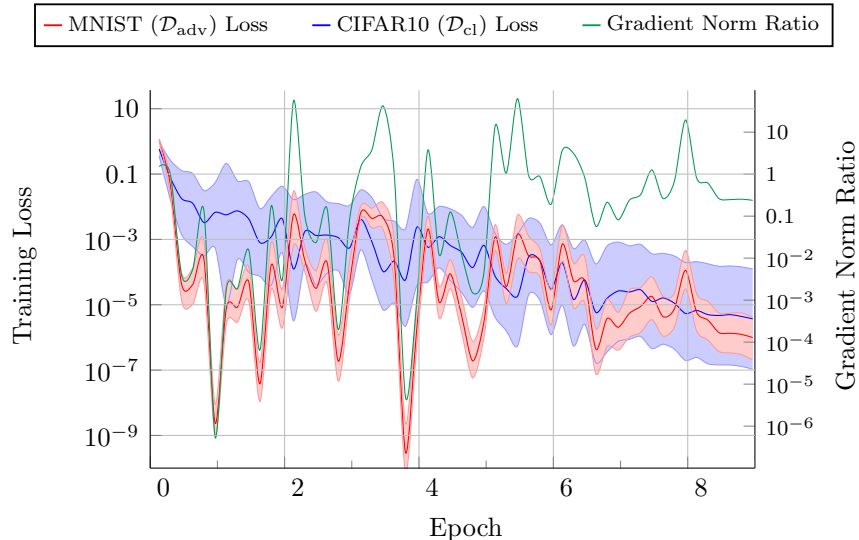


Figure 10. **CIFAR10 and MNIST Intra-training Loss Tracking:** \mathcal{D}_{adv} 's (-) and \mathcal{D}_{cl} 's (-) median cross-entropy losses (\mathcal{L}) at each training checkpoint for binary classification – frog vs. airplane & MNIST 0. The shaded regions correspond to each training set loss's interquartile range. MNIST's training losses are generally several orders of magnitude smaller than CIFAR10's losses. Gradient norm ratio (-) shows the tight coupling of loss and training gradient magnitude.

The gradient norm ratio closely tracks both training sets' loss values. Both during and at the end of training, \mathcal{D}_{adv} 's median loss is significantly smaller than many instances in \mathcal{D}_{cl} – often by several orders of magnitude.

The Low-Loss Penalty Observe that all influence methods in Sec. 6.2's scale their influence estimates by $\frac{\partial \mathcal{L}(a, y)}{\partial a}$ either directly (representer point (6.2)) or indirectly via the *chain rule* (influence functions (6.1), TracIn (6.4), and TracInCP (6.5)) as

$$\nabla_{\theta} \mathcal{L}(z; \theta) := \frac{\partial \mathcal{L}(f(\mathbf{x}), y)}{\partial \theta} = \frac{\partial \mathcal{L}(a, y)}{\partial a} \cdot \frac{\partial a}{\partial \theta}. \quad (6.7)$$

Therefore, gradient-based influence estimators *implicitly penalizes all training instances t with low training loss*, including \mathcal{D}_{adv} (MNIST 0) in our toy experiment above.

Theorem 6.1 summarizes this relationship when there is a single output activation ($|a| = 1$), e.g., binary classification and univariate regression. In short, when Theorem 6.1’s conditions are met, loss induces a perfect ordering on the corresponding norm.

Theorem 6.1. *Let loss function $\tilde{\mathcal{L}} : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ be twice-differentiable and strictly convex as well as either even⁴ or monotonically decreasing. Then, it holds that*

$$\tilde{\mathcal{L}}(a) < \tilde{\mathcal{L}}(a') \implies \left\| \nabla_a \tilde{\mathcal{L}}(a) \right\|_2 < \left\| \nabla_a \tilde{\mathcal{L}}(a') \right\|_2. \quad (6.8)$$

Loss functions satisfying Theorem 6.1’s conditions include binary cross-entropy (i.e., logistic) and quadratic losses. Theorem 6.1 generally applies to multiclass losses, but there are cases where the ordering is not perfect. Although Theorem 6.1 primarily relates to training instance gradients and losses, the theorem applies to test examples as well since dynamic estimators also apply a low-loss penalty to any iteration where test instance \hat{z}_{te} has low loss.

The preceding should *not* be interpreted to imply that large gradient magnitudes are unimportant. Quite the opposite, large gradients have large influences on the model. However, the approximations necessary to make influence estimation tractable go too far by often focusing almost exclusively on training loss – and by extension gradient magnitude – leading these estimators to systematically overlook training instances with smaller gradients. This overemphasis of instances with large losses and gradient magnitudes can also be viewed as a bias towards instances that are globally influential — affecting many examples’ predictions — over those that are locally influential – mainly affecting a small number of targets [BBD20].

⁴“Even” denotes that the function satisfies $\forall_a \tilde{\mathcal{L}}(a) = \tilde{\mathcal{L}}(-a)$.

Static Influence and the Low Loss Penalty Fig. 9’s static estimators (representer point and influence functions) significantly underperformed dynamic estimators (TracIn and TracInCP) by up to an order of magnitude. Static estimators only consider final model parameters θ_T , meaning they may only see the low-loss case. In contrast, dynamic estimators consider all of training, in particular iterations where \mathcal{D}_{adv} ’s loss exceeds that of \mathcal{D}_{cl} . This allows dynamic estimators to outperform static methods, albeit still poorly.

Training Randomness and the Low-Loss Penalty TracInCP significantly outperformed TracIn in Fig. 9 despite the TracIn being more theoretically sound. As intuition why, imagine the training set contains two identical copies of some instance. In expectation, these duplicates have equivalent influence on any test instance. However, TracIn assigns identical training examples different influence estimates based on their batch assignments; this difference can potentially be very large depending on training dynamics.

Fig. 10 exhibits this behavior where training loss fluctuates considerably intra-epoch. For example, \mathcal{D}_{adv} ’s median loss varies by seven orders of magnitude across the third epoch. TracIn’s low-loss penalty attributes much more influence to \mathcal{D}_{adv} instances early in that epoch compared to those later despite all MNIST instances having similar influence. By considering all examples at each checkpoint, TracInCP removes batch randomization’s direct effect on influence estimation,⁵ meaning TracInCP simulates *influence expectation* without needing to train and analyze multiple models.

⁵Batch randomization still indirectly affects TracInCP and GAS (Sec. 6.3.3) through the model parameters. This effect could be mitigated by training multiple models and averaging the (renormalized) influence, but that is beyond the scope of this work.

6.3.3 Renormalizing Influence Estimation. Our CIFAR10 and MNIST joint classification experiment above demonstrates that a training example having low loss does *not* imply that it and related instances are uninfluential. Most importantly in the context of adversarial attacks, highly-related groups of (adversarial) training instances may collectively cause those group members’ to have very low training losses – so-called *group effects*. Generally, targeted attacks succeed by leveraging the group effect of adversarial set \mathcal{D}_{adv} on the target(s). We address these group effects via *renormalization*, which is defined below.

Def. 6.2. For influence estimator \mathcal{I} , the renormalized influence, $\tilde{\mathcal{I}}$, replaces each gradient g in \mathcal{I} by its corresponding unit vector $\frac{g}{\|g\|}$.

We refer to this computation as renormalization since rescaling gradients removes the low-loss penalty. Renormalization places all training instances on equal footing and ensures that gradient and/or feature similarity is prioritized – not loss.

Renormalization is related to the relative influence (RelatIF) method introduced by Barshan et al. [BBD20], since both methods use a function of the gradient to downweight training instances with high losses. However, RelatIF only applies to influence functions and requires computing expensive Hessian-vector products, while renormalization is more efficient and can be applied to many influence estimators, as we show below. See the supplement of the original paper [HL22a] for additional discussion of alternative renormalization schemes.

Renormalized versions of Section 6.2’s static influence estimators are below. *Renormalized influence functions* in Eq. (6.9) does not include target gradient norm $\|\hat{g}_{\text{te}}\|$ since it is a constant factor. For simplicity, Eq. (6.10)’s *renormalized representer point* uses signum function $\text{sgn}(\cdot)$ since for any scalar $u \neq 0$, $\text{sgn}(u) = \frac{u}{|u|}$, i.e., signum

is equivalent to normalizing by magnitude.

$$\tilde{\mathcal{I}}_{\text{IF}}(z_i, \hat{z}_{\text{te}}) := \frac{1}{n} \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_T)^\top H_{\theta}^{-1} \left(\frac{\nabla_{\theta} \mathcal{L}(z_i; \theta_T)}{\|\nabla_{\theta} \mathcal{L}(z_i; \theta_T)\|} \right) \quad (6.9)$$

$$\tilde{\mathcal{I}}_{\text{RP}}(z_i, \hat{z}_{\text{te}}) := -\frac{1}{2\lambda n} \operatorname{sgn} \left(\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}} \right) \langle \mathbf{f}_i, \mathbf{f}_{\text{te}} \rangle \quad (6.10)$$

Renormalized versions of Section 6.2’s dynamic influence estimators appear below. Going forward, we refer to renormalized TracInCP (Eq. (6.12)) as *gradient aggregated similarity*, GAS, since it is essentially the weighted, gradient cosine similarity averaged across all of training. GAS’s procedure is detailed in Algorithm 3.⁶

$$\tilde{\mathcal{I}}_{\text{TracIn}}(z_i, \hat{z}_{\text{te}}) := \sum_{z_i \in \mathcal{B}_t} \frac{\eta_t}{b} \frac{\langle \nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1}), \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1}) \rangle}{\|\nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1})\| \|\nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1})\|} \quad (6.11)$$

$$\begin{aligned} \tilde{\mathcal{I}}_{\text{TracInCP}}(z_i, \hat{z}_{\text{te}}) &:= \sum_{t \in \mathcal{T}} \frac{\eta_t}{b} \frac{\langle \nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1}), \nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1}) \rangle}{\|\nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1})\| \|\nabla_{\theta} \mathcal{L}(\hat{z}_{\text{te}}; \theta_{t-1})\|} \\ &=: \text{GAS}(z_i, \hat{z}_{\text{te}}) \end{aligned} \quad (6.12)$$

Unlike static estimators, rescaling dynamic influence by target gradient norm $\|\hat{g}_{\text{te}}\|$ is quite important as mentioned earlier. Intuitively, $\|\hat{z}_{\text{te}}\|$ tends to be largest in two cases: (1) early in training due to initial parameter randomness and (2) when iteration t ’s predicted label conflicts with final label \hat{y}_{te} . Both cases are consistent with the features most responsible for predicting \hat{y}_{te} not yet dominating. Therefore, rescaling dynamic influence by $\|\hat{g}_{\text{te}}\|$ implicitly upweights iterations where \hat{y}_{te} is predicted confidently. It also inhibits any single checkpoint dominating the estimate.

Applying Renormalization to CIFAR10 and MNIST Joint Classification

Figure 9’s lower half demonstrates renormalization’s significant performance advantage over standard influence estimation – with the improvement in AUPRC as large as 25×.

⁶As shown in Algorithm 3, TracInCP’s procedure (Line 7) is identical to GAS (Line 9) other than influence renormalization.

Algorithm 3 GAS vs. TracInCP

Input: Training parameters \mathcal{P} , training set \mathcal{D} , batch size b , & test example \hat{z}_{te}

Output: (Renormalized) influence vector \mathbf{v}

```
1:  $\mathbf{v} \leftarrow \vec{0}$  ▷ Initialize
2: for each  $(\eta_t, \theta_{t-1}) \in \mathcal{P}$  do
3:    $\hat{g}_{te} \leftarrow \nabla_{\theta} \mathcal{L}(\hat{z}_{te}; \theta_{t-1})$ 
4:   for each  $z_i \in \mathcal{D}$  do ▷ All examples
5:      $g_i \leftarrow \nabla_{\theta} \mathcal{L}(z_i; \theta_{t-1})$ 
6:     if calculating TracInCP then
7:        $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\eta_t}{b} \langle g_i, \hat{g}_{te} \rangle$  ▷ Unnormalized (Sec. 6.2)
8:     else if calculating GAS then
9:        $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\eta_t}{b} \left\langle \frac{g_i}{\|g_i\|}, \frac{\hat{g}_{te}}{\|\hat{g}_{te}\|} \right\rangle$  ▷ Renormalized (Sec. 6.3.3)
10: return  $\mathbf{v}$ 
```

In particular, our renormalized estimators’ top-5 highest-ranked instances were all consistently from MNIST, unlike any of the standard influence estimators. Overall, GAS (renormalized TracInCP) was the top performer – even outperforming our other renormalized estimators by a wide margin.

6.3.4 Renormalization and More Advanced Attacks. Section 6.3.2 illustrates why influence performs poorly under a naive backdoor-style attack where the adversary does not optimize the adversarial set. Those concepts also generalize to more sophisticated attacks. For example, recent work shows that deep networks often predict the adversarial set with especially high confidence (i.e., low loss) due to *shortcut learning* – even on advanced attacks [Yu+21; Gei+20]. Those findings reinforce the need for renormalization. This can be viewed through the lens of *simplicity bias* where neural networks tend to confidently learn simple features (shortcuts) – regardless of whether those features actually generalize [Sha+20].

Dynamic estimators – both TracIn and GAS – outperform static ones for Sec. 6.3.1’s naive attack. The same can be expected for sophisticated attacks including ones that track adversarial-set gradients through simulated training [Hua+20; Wal+21].

For those attacks, adversaries can craft \mathcal{D}_{adv} to exhibit particular gradient signatures at the end of training to avoid static detection. Moreover, models learn adversarial data faster than clean data meaning training loss often drops abruptly and significantly early in training [Li+21]. For an attack to succeed, adversarial instances must align with the target at some point during training, meaning dynamic methods can detect them.

Lastly, our threat model specifies that attackers never know the random batch sequence nor any randomly initialized parameters. Therefore, attackers can only craft \mathcal{D}_{adv} to be *influential in expectation* over that randomness. Influence is stochastic, varying significantly across random seeds. However, estimating the true expected influence is computationally expensive. GAS and TracInCP, which simulate expectation, better align with how the adversary actually crafts the adversarial set, resulting in better \mathcal{D}_{adv} identification.

Below we detail how renormalization can be specialized further for better adversarial-set identification.

Extending Renormalization Layerwise: In practice, gradient magnitudes are often unevenly distributed across a neural network’s layers. For example, Figure 11 tracks an attack target’s average intra-training gradient magnitude for two different backdoor adversarial triggers on CIFAR10 binary classification ($y_{\text{targ}} = \text{airplane}$ and $y_{\text{adv}} = \text{bird}$).⁷ Specifically, target gradient norm, $\|\nabla_{\theta}\mathcal{L}(\hat{z}_{\text{targ}}; \theta_t)\|$, is decomposed into just the contributions of the network’s first convolutional layer (Conv1) and the final linear layer. Despite being only 0.04% of the model parameters, these two layers combined constitute >50% of the gradient norm. Therefore, the first and last layers’

⁷See supplemental Section D.3 for the complete experimental setup. The class pair and adversarial triggers were proposed by Weber et al. [Web+23].

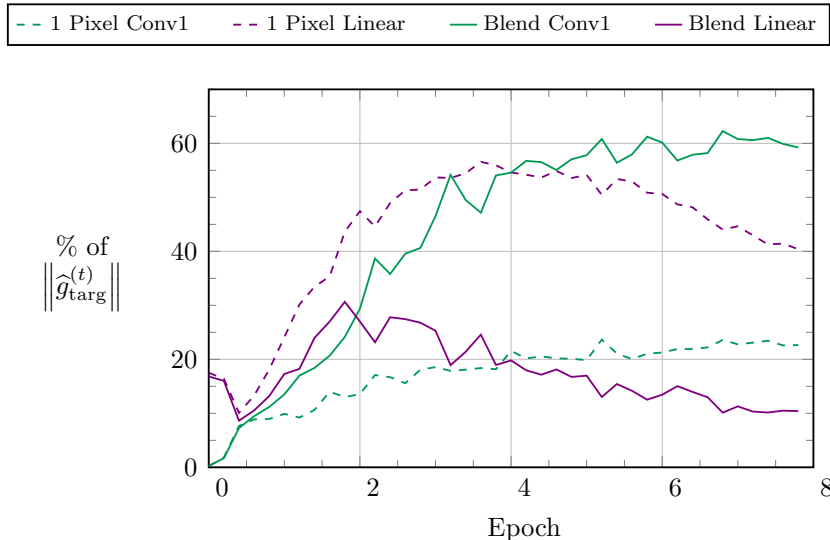


Figure 11. **Layerwise Decomposition of an Attack Target’s Intra-Training Gradient Magnitude:** One-pixel and blend backdoor adversarial triggers (dashed and solid lines respectively) trained separately on CIFAR10 binary classification ($y_{\text{target}} = \text{airplane}$ and $y_{\text{adv}} = \text{bird}$) using ResNet9. The network’s first convolutional (Conv1) and final linear layers are a small fraction of the parameters (0.03% and 0.01% resp.) but constitute most of the target’s gradient magnitude ($\|\hat{g}_{\text{target}}^{(t)}\|$) with the dominant layer attack dependent. Results are averaged over 20 trials.

parameters are, on average, weighted $>2,000\times$ more than other layers’ parameters. With simple renormalization, important parameters in those other layers may go undetected.

As an alternative to simply renormalizing by $\|\nabla_{\theta}\mathcal{L}(z; \theta_t)\|$, partition gradient vector g by layer into L disjoint vectors (where L is model f ’s layer count) and then independently renormalize each subvector separately. This *layerwise renormalization* can be applied to any estimator that uses training gradient g_i or test gradient \hat{g}_{te} , including influence functions, TracIn, and TracInCP. Layerwise renormalization still corrects for the low-loss penalty and does not change the asymptotic complexity. To switch GAS to layerwise, the only modification to Algorithm 3 is on Line 10 where

each dimension is divided by its corresponding layer’s norm instead of the full gradient norm.

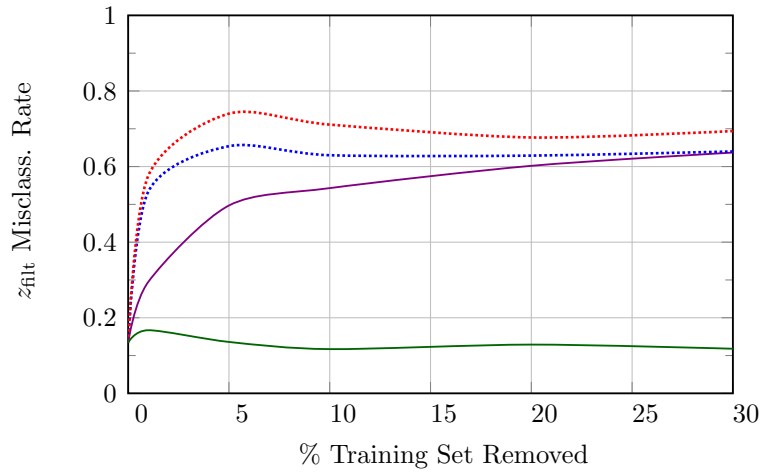
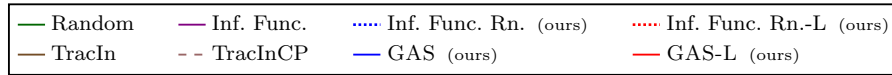
Notation: “-L” denotes layerwise renormalization, e.g., *layerwise* GAS is GAS-L. Suffix “(-L)”, e.g., GAS(-L), signifies that a statement applies irrespective of whether the renormalization is layerwise.

6.3.5 Renormalization and Non-Adversarial Data. Renormalization not only improves performance identifying an inserted adversarial set; it also improves performance in *non-adversarial* settings. Sec. 6.2 defines influence w.r.t. a single training example. Just as one instance may be more influential on a prediction than another, a group of training instances may be more influential than a different group. Renormalization improves identification of influential *groups* of examples, even on non-adversarial data.

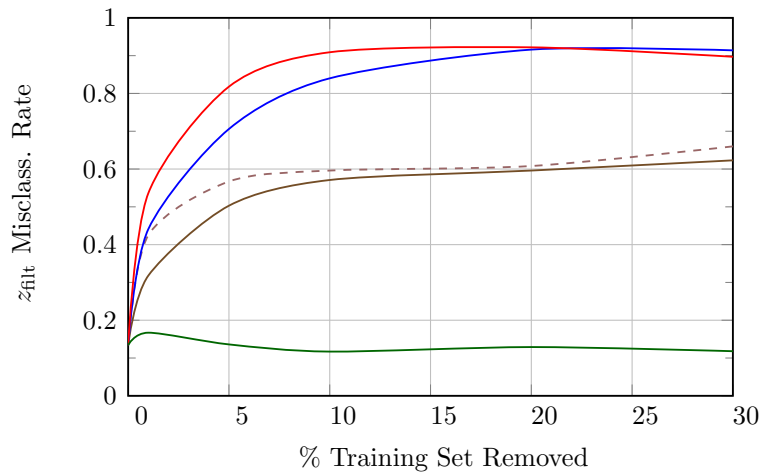
To empirically demonstrate this, consider CIFAR10 binary classification again. In each trial, ResNet9 was pre-trained on eight ($= 10 - 2$) held-out CIFAR10 classes. From the other two classes, test example z_{filt} was selected u.a.r. from those test instances with a moderate misclassification rate (10-20%) across multiple retrainings (i.e., fine-tunings) of the pre-trained network.⁸ (Renormalized) influence was then calculated for z_{filt} , with each estimator yielding a training set ranking. Each estimator’s top $p\%$ ranked instances were removed from the training set and 20 models trained from the pre-trained parameters using these reduced training sets. Performance is measured using z_{filt} ’s misclassification rate across those 20 models where a larger error rate entails a better overall ranking.

Figure 12 compares influence estimation’s filtering performance, with and without renormalization, against a random baseline averaged across five CIFAR10 class pairs,

⁸Using examples with a moderate misclassification rate ensures that dataset filtering’s effects are measurable even with a small fraction of the training data removed.



(a) Influence functions-based methods



(b) TracIn-based methods

Figure 12. Effect of Removing Influential, Non-Adversarial Training Data: Test example z_{fit} 's misclassification rate (larger is better) when filtering the training set using influence rankings based on influence functions (top) and TracIn (bottom). Renormalization (Rn.) always improved mean performance across all training set filtering percentages. Results are averaged across five CIFAR10 class pairs with 30 trials per class pair and 20 models trained per method per trial. Results are separated by the reference influence estimator.

namely the two pairs specified by Weber et al. [Web+23] and three additional random pairs. Influence, irrespective of renormalization, significantly outperformed random removal, meaning all of these estimators found influential subsets, albeit of varying quality.⁹ In all cases, renormalized influence had better or equivalent performance to the original estimator across all filtering fractions. This demonstrates that renormalization generalizes across estimators even beyond adversarial settings.

Overall, layerwise renormalization was the top performer across all setups except for large filtering percentages where GAS surpassed it slightly. Renormalized(-L) Influence functions and GAS(-L) performed similarly when filtering a small fraction (e.g., $\leq 5\%$) of the training data. However, the performance of renormalized influence functions plateaued for larger filtering fractions ($\geq 10\%$) while GAS(-L)'s performance continued to improve. In addition, renormalization's performance advantage over vanilla influence functions narrowed at larger filtering fractions. In contrast, GAS(-L)'s advantage over TracIn and TracInCP remained consistent. Recall that dynamic methods (e.g., GAS(-L) and TracIn) use significantly more gradient information than static methods (e.g., influence functions). This experiment again demonstrates that loss-based renormalization's benefits increase as more gradient information is used.

6.4 Identifying Attack Targets

Recall that non-targets have primarily weak influences and few very strong ones. Target instances are anomalous in that they have an unusual number of highly-influential training instances (Figure 13). This idea is the core of our *framework for identifying targets* of training set attacks, FIT. Alg. 4 formalizes FIT as an end-to-end

⁹Representer point (Eq. (6.2)) is excluded as it underperformed random filtering.

procedure to identify any targets in test example analysis set $\widehat{\mathcal{Z}}_{\text{te}}$.¹⁰ Overall, FIT has three sub-steps, described chronologically:

1. INF: Calculates (renormalized) influence vector \mathbf{v} for each test instance in analysis set $\widehat{\mathcal{Z}}_{\text{te}}$.
2. ANOMSCORE: Targets have an unusual number of highly-influential instances. Leveraging ideas from anomaly detection, this step analyzes each test instance’s influence vector \mathbf{v} and ranks those instances based on how anomalous their influence values are.
3. MITIGATE: Target-driven mitigation sanitizes model parameters θ_T and training set \mathcal{D} to remove the attack’s influence on the most likely target $\widehat{z}_{\text{targ}}$ (e.g., the most anomalous misclassified instance).

FIT is referred to as a “framework” since these subroutines are general and their underlying algorithms can change as new versions are developed. The next three subsections describe our implementation of each of these methods. For reference, suppl. Alg. 5 specializes Alg. 4 to more closely align with the implementation details below. See the original paper [HL22a] for a complete discussion of FIT’s end-to-end computational complexity.

6.4.1 Measuring (Renormalized) Influence. Algorithm 4 is agnostic of the specific (renormalized) influence estimator used to calculate \mathbf{v} , provided that method is sufficiently adept at identifying adversarial set \mathcal{D}_{adv} . We use GAS(-L) for the reasons explained in Section 6.3 as well as its simplicity, computational efficiency, and strong, consistent empirical performance.

¹⁰For simplicity, Alg. 4 considers a single identified target. If there are multiple identified targets, MITIGATE is invoked on each target serially with parameters $\widetilde{\theta}_T$ and $\widetilde{\mathcal{D}}_{\text{tr}}$.

Algorithm 4 FIT target identification & mitigation

Input: Training set \mathcal{D} , test example set $\widehat{\mathcal{Z}}_{\text{te}}$, and final params. θ_T

Output: Sanitized model parameters $\widetilde{\theta}_T$ & training set $\widetilde{\mathcal{D}}_{\text{tr}}$

- 1: $\mathcal{V} \leftarrow \{\text{INF}(\widehat{z}; \mathcal{D}) : \widehat{z} \in \widehat{\mathcal{Z}}_{\text{te}}\}$ \triangleright (Renorm.) Inf. (Alg. 3)
 - 2: $\Sigma \leftarrow \{\text{ANOMSCORE}(\mathbf{v}, \mathcal{V}) : \mathbf{v} \in \mathcal{V}\}$ \triangleright Anomaly score (Sec. 6.4.2)
 - 3: Rank $\widehat{\mathcal{Z}}_{\text{te}}$ by anomaly scores Σ
 - 4: $\widehat{z}_{\text{targ}} \leftarrow$ Most anomalous test example in $\widehat{\mathcal{Z}}_{\text{te}}$
 - 5: $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \text{MITIGATE}(\widehat{z}_{\text{targ}}, \theta_T, \mathcal{D})$ \triangleright Sec. 6.4.3
 - 6: **return** $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}}$
-

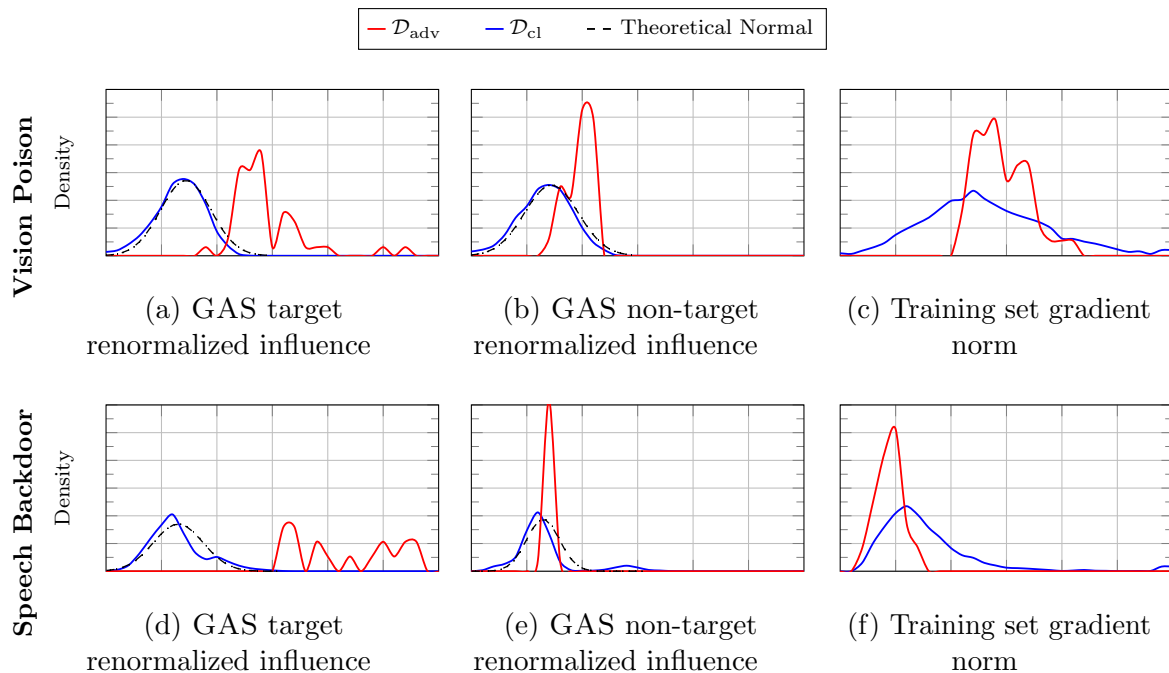


Figure 13. GAS renormalized influence, \mathbf{v} , density distributions for two training set attacks: CIFAR10 vision poisoning [Zhu+19] ($y_{\text{targ}} = \text{dog}$ and $y_{\text{adv}} = \text{bird}$) and speech-recognition backdoor [Liu+18] ($y_{\text{targ}} = 4$ and $y_{\text{adv}} = 5$). Theoretical normal ($\cdot\cdot$) is w.r.t. $\mathcal{D} := \mathcal{D}_{\text{adv}} \cup \mathcal{D}$. Observe that target examples (Figs. 13a and 13d) have significant \mathcal{D}_{adv} mass ($-$) well to the right of \mathcal{D}_{cl} 's mass ($-$). This upper-mass phenomenon is absent in non-targets (Figs. 13b and 13e). Training example gradient norms (Fig. 13c and 13f) are poorly correlated with whether the training example is adversarial. For example, speech recognition has \mathcal{D}_{cl} mass well to the right of even the right-most \mathcal{D}_{adv} mass, necessitating renormalization. See Sections 6.5.1 and D.3 for more details on these attacks.

Time and Space Complexity: Computing a gradient requires $\mathcal{O}(p)$ time and space. For fixed T and p , TracInCP, GAS, and GAS-L require $\mathcal{O}(n)$ time and space to calculate each test instance’s influence vector \mathbf{v} . The next section explains that FIT analyzes each test instance’s influence vector \mathbf{v} meaning GAS(-L) can be significantly sped-up by amortizing training gradient ($g_i^{(t)}$) computation across multiple test examples – either on a single node or across multiple nodes (e.g., using all-reduce).

6.4.2 Identifying Anomalous Influence. To change a prediction, adversarial set \mathcal{D}_{adv} must be highly influential on the target. When visualizing \hat{z}_{te} ’s influence vector \mathbf{v} as a density distribution, an exceptionally influential \mathcal{D}_{adv} manifests as a distinct density mass at the distribution’s positive extreme.

Figures 13a and 13d each plot an attack target’s GAS influence as a density for two different training set attacks – the first poisoning on vision [Zhu+19] and the other a backdoor attack on speech recognition [Liu+18]. For both attacks, adversarial set \mathcal{D}_{adv} ’s influence significantly exceeds that of \mathcal{D}_{cl} . When compared to theoretical normal (calculated¹¹ w.r.t. complete training set \mathcal{D}), \mathcal{D}_{adv} ’s target influence is highly anomalous. In Figures 13b and 13e, which plot the GAS influence of non-targets for the same two attacks, no extremely high influence instances are present.

Going forward, influence vectors \mathbf{v} with exceptionally high influence instances are referred to as having a *heavy upper tail*. Then, *target identification simplifies to identifying influence vectors whose values have anomalously heavy upper tails*. The preceding insight is relative and is w.r.t. to other test instances’ influence value distributions. Non-target baseline anomaly quantities vary with model, dataset, and

¹¹The plotted theoretical normal used robust statistics median and Q in place of mean and standard deviation.

Algorithm 5 FIT target identification implementation

Input: Training set \mathcal{D} , training set size n , test example set $\widehat{\mathcal{Z}}_{\text{te}}$, and upper-tail count κ

Output: Sanitized model parameters $\widetilde{\theta}_T$ & training set $\widetilde{\mathcal{D}}_{\text{tr}}$

- 1: $\mathcal{V} \leftarrow \left\{ \text{GAS}(\widehat{z}_j; \mathcal{D}) : \widehat{z}_j \in \widehat{\mathcal{Z}}_{\text{te}} \right\}$ ▷ Renorm. Inf. (Alg. 3)
 - 2: $\Sigma \leftarrow \left\{ \frac{\mathbf{v}^{(j)} - \mu^{(j)}}{Q^{(j)}} : \mathbf{v}^{(j)} \in \mathcal{V} \right\}$ ▷ Anomaly score (Sec. 6.4.2)
 - 3: $\mathcal{H} \leftarrow \left\{ \boldsymbol{\sigma}_{(n-\kappa)} : \boldsymbol{\sigma} \in \Sigma \right\}$ ▷ Upper-tail heaviness (Sec. 6.4.2)
 - 4: Rank $\widehat{\mathcal{Z}}_{\text{te}}$ by heaviness \mathcal{H}
 - 5: $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \theta_T, \mathcal{D}$
 - 6: **for each** target $\widehat{z}_{\text{targ}}$ identified using \mathcal{H} **do**
 - 7: $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \text{MITIGATE}(\widehat{z}_{\text{targ}}, \widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}})$ ▷ Sec. 6.4.3
 - 8: **return** $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}}$
-

hyperparameters. That is why suppl. Algorithm 5 ranks candidates in $\widehat{\mathcal{Z}}_{\text{te}}$ based on their upper-tail heaviness.

Quantifying Tail Heaviness: Determining whether \widehat{z}_{te} 's influence vector \mathbf{v} is abnormal simplifies to univariate anomaly detection for which significant previous work exists [BL78; RL87; HA04; RH17]. Observe in Figures 13b and 13e that \mathcal{D}_{cl} 's GAS influence vector \mathbf{v} tends to be normally distributed (see the close alignment to the dashed line). We, therefore, use the traditional *anomaly score*, $\boldsymbol{\sigma} := \frac{\mathbf{v} - \mu}{s}$, where μ and s are each \mathbf{v} 's center and dispersion statistics, resp.¹² Mean and standard deviation, the traditional center and dispersion statistics, resp., are not robust to outliers. Both have an asymptotic *breakdown point* of 0 (one anomaly can shift the estimator arbitrarily). Since \mathcal{D}_{adv} instances are inherently outliers, robust statistics are required.

Median serves as our center statistic μ given its optimal breakdown (50%). Although median absolute deviation (MAD) is the best-known robust dispersion statistic, we use Rousseeuw and Croux's [RC93] Q estimator, which retains

¹²In suppl. Algorithm 5, statistics $\mu^{(j)}$ and $Q^{(j)}$ are calculated separately for each test instance \widehat{z}_j 's influence vector $\mathbf{v}^{(j)}$.

MAD’s benefits while addressing its weaknesses. Specifically, both MAD and Q have optimal breakdowns, but Q has better Gaussian data *efficiency* (82% vs. 37%). Critically for our setting with one-sided anomalies, Q does not assume data symmetry – unlike MAD. Formally,

$$Q := c\{|\mathbf{v}_i - \mathbf{v}_l| : 1 \leq i < l \leq n\}_{(r)}, \quad (6.13)$$

where $\{\cdot\}_{(r)}$ denotes the set’s r -th *order statistic* with $r = \left(\lfloor \frac{n}{2} \rfloor + 1\right)$ and c is a distribution consistency constant which for Gaussian data, $c \approx 2.2219$ [RH17]. Eq. (6.13) requires only $\mathcal{O}(n)$ space and $\mathcal{O}(n \lg n)$ time as proven by Croux and Rousseeuw [CR92]. Provided anomaly score vector $\boldsymbol{\sigma}$, upper-tail heaviness is simply $\boldsymbol{\sigma}_{(n-\kappa)}$, which is $\boldsymbol{\sigma}$ ’s $(n - \kappa)$ th order statistic, i.e., \widehat{z}_{te} ’s κ th largest anomaly score value. The value of κ implicitly affects the size of the smallest detectable attack, where any attack with $|\mathcal{D}_{\text{adv}}| < \kappa$ is much harder to detect.

Multiclass vs. Binary Classification Different classes are implicitly generated from different data distributions. Each class’s data distribution may have different influence tails – in particular in multiclass settings. Target identification performance generally improves (1) when μ and Q are calculated w.r.t. only training instances labeled \widehat{y}_{te} and (2) \widehat{z}_{te} ’s upper-tail heaviness is ranked w.r.t. other test instances labeled \widehat{y}_{te} .

Faster FIT The execution time of TracInCP and by extension GAS(-L), depends on parameter count $|\theta|$. For very large models, target identification can be significantly sped up via a two-phase strategy. In phase 1, GAS(-L) uses a very small iteration subset (e.g., $\mathcal{T} = \{T\}$) to coarsely rank analysis set $\widehat{\mathcal{Z}}_{\text{te}}$. Phase 2 then uses the complete \mathcal{T} but only on a small fraction (e.g., 10%) of $\widehat{\mathcal{Z}}_{\text{te}}$ with the heaviest phase 1

tails. Section 6.5.3 applies this approach to natural-language data poisoning on RoBERTa_{BASE} [Liu+20b].

Computing each test instance’s ($\hat{z}_{te} \in \hat{\mathcal{Z}}_{te}$) influence vector \mathbf{v} is independent. Each dimension \mathbf{v}_i is also independent and can be separately computed. Hence, GAS(-L) is embarrassingly parallel allowing linear speed-up of target identification via parallelization.

6.4.3 Target Driven Attack Mitigation. A primary benefit of target identification is that attack mitigation becomes straightforward. Algorithm 6 mitigates attacks by sanitizing training set \mathcal{D} of adversarial set \mathcal{D}_{adv} . Most importantly, target identification solves data sanitization’s common pitfall (Sec. 3.2.1) of determining how much data to remove. *Sanitization stops when the target’s misprediction is eliminated.* Therefore, successfully identifying a target means sanitization is *guaranteed to succeed tautologically* (i.e., attack success rate on any analyzed targets is 0).

More concretely, Alg. 6 iteratively filters \mathcal{D} by thresholding anomaly score vector, $\boldsymbol{\sigma}$.¹³ Since adversarial instances are abnormally influential on targets, Alg. 6 filters \mathcal{D}_{adv} instances first. After each iteration, influence is remeasured to account for estimation stochasticity and because training dynamics may change with different training sets. Data removal cutoff ζ is tuned based on computational constraints – larger ζ results in less clean data removed but may take more iterations. Slowly annealing ζ also results in less clean-data removal.

Given forensic or human analysis of the identified target(s), simpler mitigation than Algorithm 6 is possible, e.g., a naive, rule-based, corrective lookup table that entails no clean data removal at all.

¹³Alg. 6 considers the more general case of a single identified target but can be extended to consider multiple targets. For instance, provided there is a single attack, average \mathbf{v} across all targets, and stop sanitizing once all targets are classified correctly.

Algorithm 6 Target-driven mitigation & sanitization

Input: Target $\widehat{z}_{\text{targ}} := (\mathbf{x}_{\text{targ}}, y_{\text{adv}})$, anomaly cutoff ζ , model f , initial params. θ_0 , final params. θ_T , and training set \mathcal{D}

Output: Clean model parameters $\widetilde{\theta}_T$ & sanitized training set $\widetilde{\mathcal{D}}_{\text{tr}}$

```
1: function MITIGATE( $\widehat{z}_{\text{targ}}, \theta_T, \mathcal{D}$ )
2:    $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \theta_T, \mathcal{D}$ 
3:   while  $\arg \max (f(\mathbf{x}_{\text{targ}}; \widetilde{\theta}_T)) = y_{\text{adv}}$  do
4:      $\mathbf{v} \leftarrow \text{INF}(\widehat{z}_{\text{targ}}; \mathcal{D})$  ▷ Renorm. Influence (Alg. 3)
5:      $\boldsymbol{\sigma} \leftarrow \frac{\mathbf{v} - \boldsymbol{\mu}}{Q}$  ▷ Anomaly score (Sec. 6.4.2)
6:      $\widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \widetilde{\mathcal{D}}_{\text{tr}} \setminus \{z_i : \sigma_i \geq \zeta \wedge z_i \in \widetilde{\mathcal{D}}_{\text{tr}}\}$  ▷ Sanitize
7:      $\widetilde{\theta}_T \leftarrow \text{RETRAIN}(\theta_0, \widetilde{\mathcal{D}}_{\text{tr}})$ 
8:     Optionally anneal  $\zeta$ 
9:   return  $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}}$ 
```

For learning environments where *certified training data deletion* is possible [Guo+20; MRA22], retraining (Alg. 6 Line 7) may not even be required — making our method even more efficient.

Enhancing Mitigation’s Robustness An adversary could attack FIT by injecting adversarial instances into \mathcal{D} to specifically trigger excessive, unnecessary sanitization. To mitigate such a risk, Alg. 6 could be tweaked to include a maximum sanitization threshold¹⁴ that would trigger additional (e.g., human, forensic) analysis. This threshold could be set empirically or using domain-specific knowledge (e.g., maximum possible poisoning rate). See the supplement of the original paper [HL22a] for further discussion.

6.5 Evaluation

We empirically demonstrate our method’s generality by evaluating training set attacks on different data modalities, including text, vision, and speech recognition. We

¹⁴This threshold could be w.r.t. the number of examples removed or the change in held-out loss. These quantities can be measured cumulatively or for targets individually.

consider both poisoning and backdoor attacks on pre-trained and randomly-initialized, state-of-the-art models in binary and multiclass settings. For brevity, most evaluation setup details (e.g., hyperparameters) are deferred to suppl. Section D.3. Additional experimental results also appear in the original paper [HL22a].

6.5.1 Training-Set Attacks Evaluated. We evaluated our method on four published training set attacks – two *single-target* data poisoning and two multi-target backdoor. Below are brief details regarding how each attack crafts adversarial set \mathcal{D}_{adv} , with the full details in suppl. Sec. D.3.2.4. Representative clean and adversarial training instances for each attack appear in the original paper [HL22a]. Table 6 lists each attack’s mean *success rate* aggregated across all related setups. Full granular results are in Section C.3.

Below, $y_{\text{targ}} \rightarrow y_{\text{adv}}$ denotes the target’s true and adversarial labels, respectively. When an attack considers multiple class pairs or setups, each is evaluated separately.

(1) *Speech Backdoor*: Liu et al.’s [Liu+18] speech recognition dataset contains spectrograms of human speech pronouncing in English digits 0 to 9 (10 classes, $|\mathcal{D}_{\text{cl}}| = 3,000 - 1\%$ backdoors). Liu et al. also provide 300 backdoored training instances evenly split between the 10 classes. Each class’s adversarial trigger – a short burst of white noise at the recording’s beginning – induces the spoken digit to be misclassified as the next largest digit (e.g., $0 \rightarrow 1$, $1 \rightarrow 2$, etc.). This small input-space signal induces a large feature-space perturbation – too large for many certified methods. Following Liu et al., our evaluation used a speech recognition CNN trained from scratch.

(2) *Vision Backdoor*: Weber et al. [Web+23] consider three different backdoor adversarial trigger patterns on CIFAR10 binary classification. Specifically, Weber et al.’s “pixel” attack patterns increase the pixel value of either one or four central

pixel(s) by a specified maximum ℓ_2 perturbation distance while their “blend” trigger pattern adds fixed $\mathcal{N}(\mathbf{0}, I)$ Gaussian noise across all perturbed images. We considered the same class pairs as Weber et al. (`auto` \rightarrow `dog` and `plane` \rightarrow `bird`) on the state-of-the-art ResNet9 [Pag20] CNN trained from scratch with $|\mathcal{D}_{\text{adv}}| = 150$ and $|\mathcal{D}| = 10,000$ (1.5% backdoors).

(3) *Natural Language Poison*: Wallace et al. [Wal+21] construct text-based poison by simulating bilevel optimization via second-order gradients. \mathcal{D}_{adv} ’s instances are crafted via iterative word-level substitution given a target phrase. We follow Wallace et al.’s [Wal+21] experimental setup of poisoning the Stanford Sentiment Treebank v2 (SST-2) sentiment analysis dataset [Soc+13] ($|\mathcal{D}_{\text{cl}}| = 67,349$ & $|\mathcal{D}_{\text{adv}}| = 50 - 0.07\%$ poison) on the RoBERTa_{BASE} transformer architecture (125M parameters) [Liu+20b].

(4) *Vision Poison*: Zhu et al.’s [Zhu+19] targeted, clean-label attack crafts poisons by forming a convex polytope around a single target’s feature representation. Following Zhu et al., the pre-train then fine-tune paradigm was used. In each trial, ResNet9 was pre-trained using half the classes (none were y_{targ} or y_{adv}). Targets were selected uniformly at random (u.a.r.) from test examples labeled y_{targ} , and 50 poison instances (0.2% of \mathcal{D}) were then crafted from seed examples labeled y_{adv} . The pre-trained network was fine-tuned using \mathcal{D}_{adv} and the five held-out classes’ training data ($|\mathcal{D}| = 25,000$). Like previous work [Sha+18; Hua+20], CIFAR10 class pairs `dog` vs. `bird` and `deer` vs. `frog` were evaluated, where each class in a pair serves alternately as y_{targ} and y_{adv} .

While it is not feasible to evaluate our approach on every attack (as new attacks are developed & published so frequently) we believe this diverse set of attacks is representative of training set attacks in general and demonstrates our approach’s

broad applicability. In particular, our method is not tailored to these attacks and could be used against future attacks as well, as long as the attack includes highly-influential training examples that attack specific targets.

6.5.2 Identifying Adversarial Set \mathcal{D}_{adv} . To identify the target (Alg. 4) or mitigate the attack (Alg. 6), we must be able to identify the likely adversarial instances \mathcal{D}_{adv} associated with a possible target \hat{z}_{target} . Our approach is to use influence-estimation methods, which should rank an actual adversarial attack \mathcal{D}_{adv} as more influential than clean instances \mathcal{D}_{cl} on the target. In this section, we evaluate how well different influence-estimation methods succeed at performing this ranking for a given target.

We compare the performance of our renormalized estimators, GAS and GAS-L, against Section 6.2’s four influence estimators: TracInCP, TracIn, influence functions, and representer point. As an even stronger baseline, where applicable, we also compare against Peri et al.’s [Per+20] Deep k -NN empirical training set defense specifically designed for Zhu et al.’s [Zhu+19]’s vision, clean-label poisoning attack; described briefly, Deep k -NN sanitizes the training set of instances whose nearest feature-space neighbors have a different label. Like Section 6.3’s CIFAR10 & MNIST joint classification experiment, class sizes are imbalanced ($|\mathcal{D}_{\text{adv}}| \ll |\mathcal{D}_{\text{cl}}|$) so performance is again measured using AUPRC.

For targets selected u.a.r., Figure 14 details each method’s averaged adversarial-set identification AUPRC for Section 6.5.1’s four attacks. In summary, GAS and GAS-L were each the top performer for one attack and had comparable performance for the other two.

GAS and GAS-L identified the adversarial instances nearly perfectly for Liu et al.’s speech backdoor and Wallace et al.’s text poisoning attacks. Standard influence

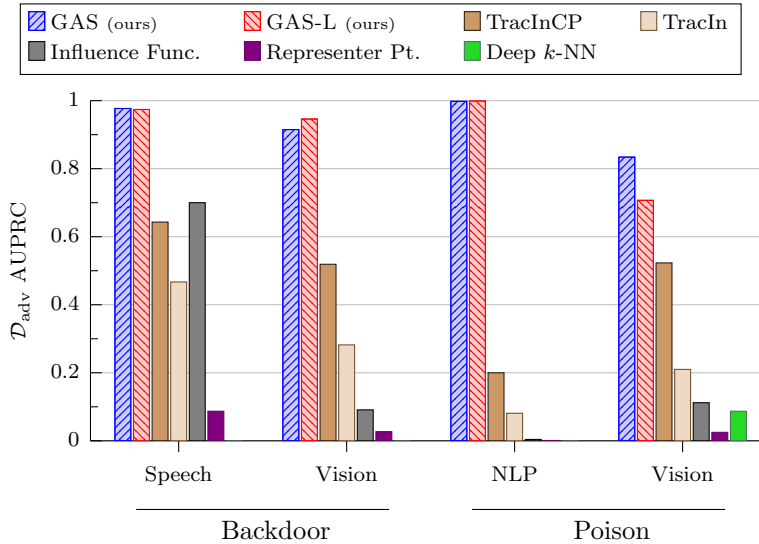


Figure 14. **Adversarial-Set Identification:** Mean AUPRC identifying adversarial set \mathcal{D}_{adv} using a randomly selected target for Sec. 6.5.1’s four attacks. Results averaged across related setups with ≥ 10 trials per setup. See supplemental Section C.3 for the full granular results.

estimation performed poorly on the text poisoning attack (in particular the static estimators) due to the large model, RoBERTa_{BASE}, that Wallace et al.’s attack considers. For the vision backdoor and poisoning attacks, our renormalized estimators successfully identified most of \mathcal{D}_{adv} – again, much better than the four original estimators. While Peri et al.’s [Per+20] Deep k -NN defense can be effective at stopping clean-label vision poisoning, it does so by removing a comparatively large fraction of clean data (up to 4.3% on average) resulting in poor AUPRC.

For completeness, Figure 15 provides adversarial-set identification results for our renormalized, static influence estimators. In all cases, renormalization improved the estimator’s performance, generally by an order of magnitude with a maximum improvement of $600\times$. These experiments highlight layerwise renormalization’s benefits. Influence functions’ Hessian-vector product algorithm [Pea94] can assign a large magnitude to some layers, and these layers then dominate the influence and GAS

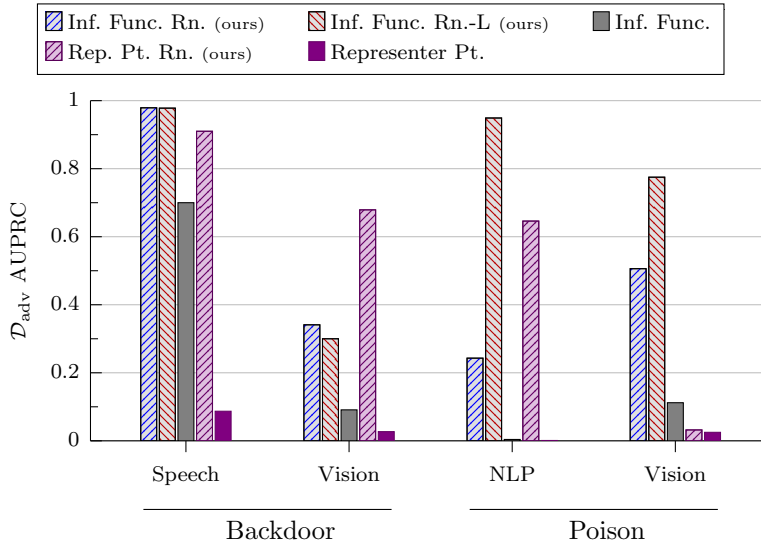


Figure 15. Static Influence Adversarial-Set Identification: Comparing the mean adversarial-set identification AUPRC of the static influence estimators and their corresponding renormalized (Rn.) versions. For all attacks, renormalization improved the static estimators’ mean performance by up to a factor of $>600\times$. These experiments also highlight layerwise renormalization’s performance gains, e.g., influence functions on natural-language poison. Results are averaged across related experimental setups with ≥ 10 trials per setup.

estimates. Layerwise renormalization addresses this, improving renormalized influence function’s adversarial-set identification AUPRC by up to $3.5\times$.

6.5.3 Identifying Attack Targets. The previous experiments demonstrate that knowledge of a target enables identification of the adversarial set when using renormalization. This section demonstrates that the distribution of renormalized influence values actually enables us to identify target(s) in the first place, through the interplay foundational to our target identification framework, FIT. Since target identification is a new task, we propose four target identification baselines. First, inspired by Peri et al.’s [Per+20] Deep k -NN empirical defense, *maximum k -NN distance* computes the distance from each test instance to its κ^{th} nearest neighbor in the training data, as measured by the L_2 distance between their penultimate

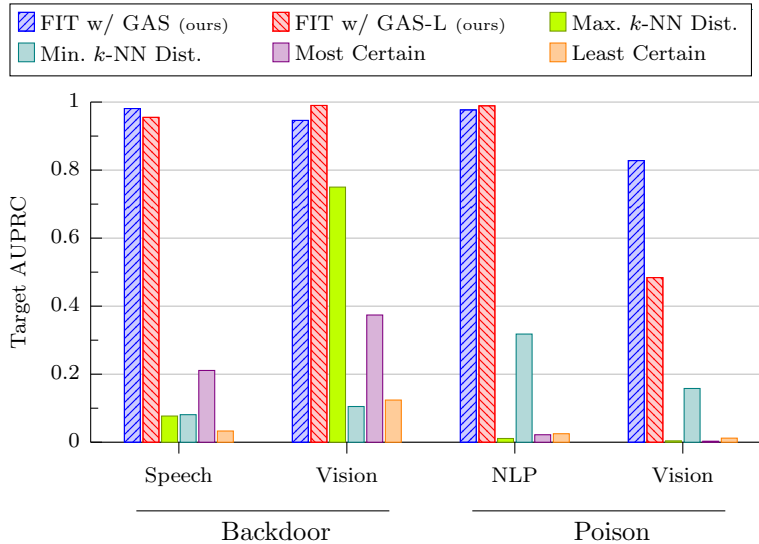


Figure 16. **Target Identification:** Mean target identification AUPRC for Sec. 6.5.1’s four attacks. “FIT w/ GAS” denotes GAS was FIT’s influence estimator with matching notation for GAS-L. Results averaged across setups with ≥ 10 trials per setup. See Sec. C.3 for the full granular results.

feature representations (\mathbf{f}). It orders them by this distance, starting with the largest distance to the κ^{th} neighbor, thus prioritizing outliers and instances in sparse regions of the learned representation space. *Minimum k -NN distance* is the reverse ordering, prioritizing instances in dense regions. The other two baselines are *most certain*, which ranks test examples in ascending order by loss while *least certain* ranks by descending loss.

There are far fewer targets than possible test examples so performance is again measured using AUPRC. See suppl. Table D.58 for the number of targets and non-targets analyzed for each attack. For single-target attacks (vision and natural language poisoning), target identification AUPRC is equivalent to the target’s inverse rank, causing AUPRC to decline geometrically.

Figure 16 shows that FIT – using either GAS or GAS-L as the influence estimator – achieves near-perfect target identification for both backdoor attacks and natural

language poisoning. Overall, FIT with GAS was the top performer on two attacks, and FIT with GAS-L was the best for the other two. Recall that the vision poisoning attack is single target. Hence, GAS-based FIT’s mean AUPRC of >0.8 equates to an average target rank better than 1.25 ($1/0.8$), i.e., three out of four times on average \hat{z}_{targ} was the top-ranked – also very strong target detection. FIT’s performance degradation on vision poisoning is due to GAS and GAS-L identifying this attack’s \mathcal{D}_{adv} slightly worse (Fig. 14). Only maximum k -NN approached FIT’s performance – specifically for Weber et al.’s vision backdoor attack. Note also that no baseline consistently outperformed the others. Hence, these attacks affect network behavior differently, further supporting that FIT is attack agnostic.

FIT’s performance is stable across a wide range of upper-tail cutoff thresholds κ (see supplement of the original paper [HL22a]). For example, FIT’s natural language target identification AUPRC varied only 0.2% and 2.1% when using GAS-L and GAS respectively for $\kappa \in [1, 25]$.

6.5.4 Target-Driven Mitigation. Section 6.4.3 explains that successfully identifying the target(s) enables *guaranteed attack mitigation* on those instances. Here, we evaluate GAS and GAS-L’s effectiveness in targeted data sanitization. Table 6 details our defense’s effectiveness against Sec. 6.5.1’s four attacks. As above, results are averaged across each attack’s class pairs/setup. Section 14’s baselines all have large false-positive rates when identifying \mathcal{D}_{adv} (Fig. 14), which caused them to remove a large fraction of \mathcal{D}_{cl} and are not reported in these results.

For three of four attacks, clean test accuracy after sanitization either improved or stayed the same. In the case of Weber et al.’s vision backdoor attack, the performance degradation was very small – 0.1%. Similarly, owing to renormalized influence’s effectiveness identifying \mathcal{D}_{adv} (Fig. 16), our defense removes very little clean data when

mitigating the attack – generally $<0.2\%$ of the clean training set. For comparison, Peri et al. [Per+20] report that their Deep k -NN clean-label, poisoning defense removes on average 4.3% of \mathcal{D}_{cl} on Zhu et al.’s [Zhu+19] vision poisoning attack. This is despite Peri et al.’s method being specifically tuned for Zhu et al.’s attack and their evaluation setup being both easier and less realistic by pre-training their model using a large known-clean set that is identically distributed to their \mathcal{D}_{cl} . In contrast, target-driven mitigation removed at most 0.03% of clean data on this attack – better than Peri et al. by two orders of magnitude.

Following Algorithm 6, Table 6’s experiments used only a single, randomly-selected target when performing sanitization. No steps were taken to account for additional potential targets, e.g., over-filtering the training set. Nonetheless, target-driven mitigation still significantly degraded multi-target attacks’ performance on other targets not considered when sanitizing. For example, despite considering one target, speech backdoor’s overall attack success rate (ASR) across all targets decreased from 100% to 4.7% and 6.5% for GAS and GAS-L, respectively – a $20\times$ reduction. For Weber et al.’s vision backdoor attack, ASR dropped from 90.5% to 11.9% and 6.7% with GAS and GAS-L, respectively. The *key takeaway* is that identifying a single target almost entirely mitigates the attack everywhere.

6.6 Adaptive Attacks

We now consider how an attacker who knows about our defense could evade it or otherwise exploit it. Our method relies on multiple attack instances having unusually high influence on the target instance, as measured by model gradients during training. As such, it may fail to detect an attack if (1) there are too few attack instances (relative to upper-tail count κ); (2) the attack is a large fraction of the data (e.g., 10%), in which case the instances are too common to be considered outliers; or (3) the attack

Table 6. **Target Driven Attack Mitigation:** Alg. 6’s target-driven, iterative data sanitization applied to Sec. 6.5.1’s four attacks for randomly selected targets. The attacks were neutralized with few clean instances removed and little change in test accuracy. Attack success rate (ASR) is w.r.t. the analyzed target. Results are averaged across related setups with ≥ 10 trials per setup. Detailed results appear in Sec. C.3.1–C.3.4.

	Dataset	Method	% Removed		ASR %		Test Acc. %	
			\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
Backdoor	Speech	GAS	98.4	0.07	99.8	0	97.7	0.0
		GAS-L	98.1	0.17		0		0.0
	Vision	GAS	87.6	0.50	90.5	0	96.2	-0.1
		GAS-L	92.3	0.73		0		-0.1
Poison	NLP	GAS	99.6	0.02	97.9	0	94.2	+0.2
		GAS-L	99.9	0.03		0		+0.1
	Vision	GAS	65.1	0.02	77.9	0	87.1	0.0
		GAS-L	58.6	0.03		0		0.0

instances appear no more influential on the target than clean instances. The first case is only a risk when the target instance is “easy” enough to influence that the attack can be carried out with very few instances. The second case requires a very powerful attacker, one who would be hard to stop without additional constraints or assumptions. The third case represents a possible weakness: if attackers can craft an attack that successfully changes the target label without appearing unusual, then our defense will fail. Whether or not the attacker can succeed is an empirical question, which will depend on the dataset, the model, the choice of target instance, and the attack method. Below, we provide evidence that FIT (and GAS in particular) remain effective against an attacker who is trying to evade our defense.

Seed-Instance Optimization: Some training set attacks rely on a *fixed*, predefined adversarial perturbation which is applied to clean seed instances [Liu+18; Web+23]. An attacker who is aware of our defense could choose seed instances that organically appear uninfluential on some target, as estimated by GAS(-L). We apply this idea

to Weber et al.’s [Web+23] CIFAR10 backdoor attack and find that our method continues to perform well against this simple, adaptive attacker: GAS(-L) achieve 0.93 AUPRC for adversarial-set identification, a 7% decline versus the baseline, even when the attacker is given information beyond our threat model, e.g., knowledge of the random, initial parameters. Overall, the attacker’s gains from choosing different seed instances are limited. See the supplement of the original paper for additional details [HL22a].

Perturbation Optimization: A stronger adaptive adversary actively optimizes the adversarial perturbation to be both highly effective *and* have a low (GAS) influence estimate. For attacks that find perturbations through gradient-based optimization [Wal+21; Zhu+19], the most natural way to incorporate knowledge of our method would be to add some estimate of GAS to the loss being optimized. Since GAS – like poison – relies on the entire training trajectory of the model, which in turn relies on the perturbations being crafted, computing GAS’s exact gradient is intractable [BR92]. However, the attacker can still use a surrogate that approximates GAS, such as by using fixed model checkpoints in the computation of GAS.

To evaluate the robustness of our methods to adaptive perturbations, we apply this joint optimization idea to Zhu et al.’s [Zhu+19] vision poison attack. We focused on Zhu et al.’s attack because (1) it is the attack on which our method performed the worst and (2) the other optimized attack we consider [Wal+21] is restricted to only discrete token replacements, which reduces the attacker’s flexibility.

Zhu et al. iteratively optimize a set of poison examples to minimize the adversarial loss. To increase the likelihood of successfully changing the target’s label, Zhu et al. compute this loss over multiple surrogate models. The perturbations of the poison examples are constrained to an ℓ_∞ ball so that they appear relatively natural to

humans. Our jointly optimized, adaptive attack adds a second term to Zhu et al.’s adversarial loss. This new term estimates the GAS influence using the same surrogate models. Hyperparameter β balances the two objectives. See suppl. Section C.4 for the full details of this jointly-optimized, adaptive attack, including tuning of β .

Where possible, these adaptive experiments followed the same vision poison evaluation setup detailed in Section 6.5.1. However, simultaneously optimizing surrogate GAS has a much higher GPU memory cost, so we were forced to adjust the poison size and number of surrogate checkpoints to 40 and four, respectively, which degrades the attacker’s success rate from 77.9% in Table 6 to 64.3%.

Fig. 17 summarizes our adversarial-set identification performance on Zhu et al.’s vision poisoning attack with and without the jointly optimized surrogate GAS loss term.¹⁵ Observe that the attack degraded the performance of GAS(-L) and TracInCP, albeit slightly. After accounting for all other factors, this joint optimization decreased GAS’s mean adversarial-set identification from a baseline of 0.73 AUPRC to 0.68 (a 7% drop). Fig. 18 visualizes joint attack optimization’s effect on target identification. Overall, joint optimization reduced FIT with GAS’s mean target identification AUPRC from a baseline of 0.86 to 0.78 (9% drop). Since Zhu et al.’s attack is single-target, this translates to the target’s average rank declining from 1.16 to 1.28 — still high performance.

Table 7 details target-driven mitigation’s effectiveness under this jointly-optimized attack. In summary, the attack results in very little clean data removal (at most 0.05% of \mathcal{D}_{cl} on average). Also, the average test accuracy after mitigation either improved or stayed the same in all but one case where it decreased by only 0.1%.

¹⁵Both versions of the attack (i.e., with and without joint optimization) used the same evaluation setup, including the reduced surrogate model count.

Table 7. **Attack Mitigation for the Adaptive Vision Poison Attack:** Algorithm 6’s target-driven data sanitization where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing the GAS influence. The results below consider exclusively the jointly-optimized attack with $\beta = 10^{-2}$. Clean-data removal remains low, and test accuracy either improved or stayed the same for in but one setup. The performance is comparable to the results with Zhu et al.’s [Zhu+19]’s standard vision poisoning attack (see Table C.40). Bold denotes the best mean performance with ≥ 10 trials per class pair.

Classes		Method	% Removed		ASR %		Test Acc. %	
y_{targ}	y_{adv}		\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
Bird	Dog	GAS	36.0	0.02	76.2	0	87.0	+0.1
		GAS-L	30.3	0.00		0		+0.1
Dog	Bird	GAS	21.6	0.00	57.1	0	87.1	+0.1
		GAS-L	21.9	0.00		0		-0.1
Frog	Deer	GAS	17.5	0.00	38.1	0	87.1	0.0
		GAS-L	19.4	0.00		0		0.0
Deer	Frog	GAS	85.0	0.18	81.0	0	87.1	0.0
		GAS-L	82.3	0.13		0		+0.1

In summary, even when the adversary specifically optimized for our defense, we still effectively identify both the adversarial set and the target and then mitigate the adaptive attack.

6.7 Discussion and Conclusions

This chapter explores two related tasks. First, we propose training set attack *target identification*, which plays an important part in the protection of critical ML systems but has thus far received relatively little attention. For example, it is impossible to conduct a truly informed cost-benefit analysis of risk without knowing the attacker’s target and by extension their objective. Knowledge of the target also enables forensic and security analysts to reason about an attacker’s identity – a key step to permanently stopping attacks by disabling the attacker. An open question is whether target identification can be combined with certified guarantees, either building on our FIT framework or creating an alternative to it.

FIT relies on identifying (groups of) highly influential training instances. To that end, we propose *renormalized influence*. By addressing influence’s low-loss penalty, renormalization significantly improves influence estimation in both adversarial and non-adversarial settings – often by an order of magnitude or more [HL22b].

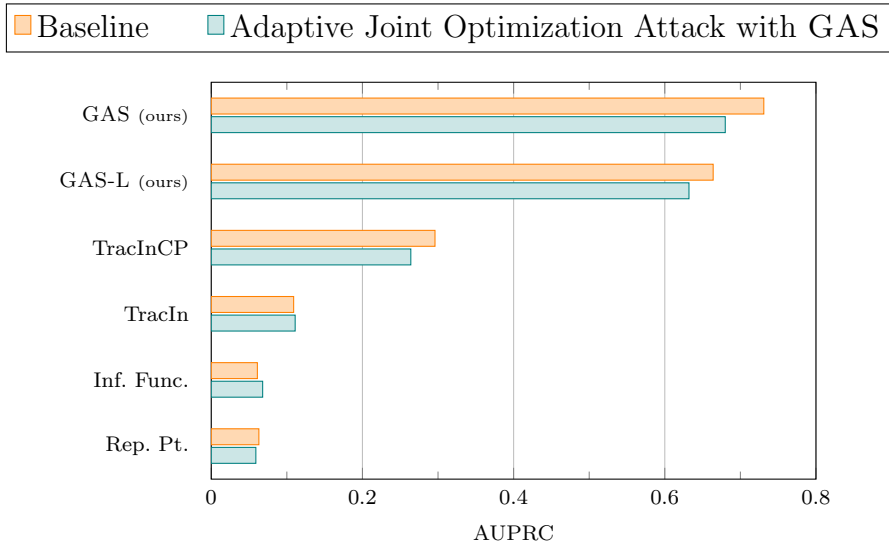


Figure 17. Adversarial-Set Identification for the Adaptive Vision Poison Attack: Mean AUPRC identifying the adversarial set where Zhu et al.’s vision poison attack is adapted to jointly minimize the adversarial loss and the GAS influence. The baseline results (orange) used Zhu et al.’s standard attack. Our jointly-optimized attack reduced the GAS similarity by 7% at the cost of a 19% decrease in ASR w.r.t. Table 6. See suppl. Sec. C.4 for the granular results.

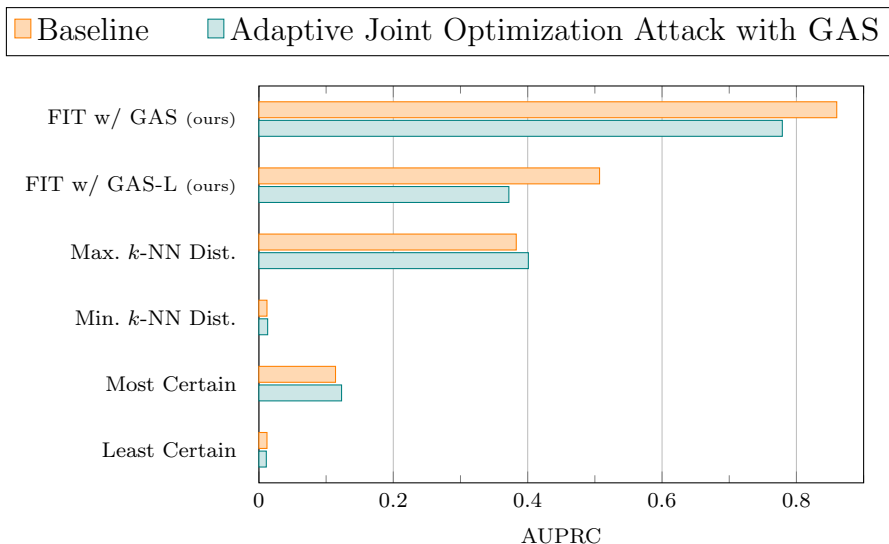


Figure 18. Target Identification for the Adaptive Vision Poison Attack: Mean target identification AUPRC where Zhu et al.’s vision poison attack is jointly optimized with minimizing GAS. FIT with GAS’s mean target identification AUPRC declined only 9% versus the baseline – an average change in target rank of 1.16 to 1.28 – still strong performance. Results are averaged across related setups with ≥ 10 trials per setup. See suppl. Sec. C.4 for the full results.

CHAPTER 7

CONCLUSIONS AND FUTURE DIRECTIONS

With machine learning systems increasingly deployed in settings critical to human well-being, the need for robust models similarly increases. Deploying models that have been adversarially manipulated can harm human health, increase unfairness, and degrade public trust in institutions. It is unlikely that the regulatory landscape will continue to allow empirical performance to be prioritized over all else. While this dissertation looks at model robustness through the lens of adversarial robustness, our contributions should not be viewed as solely applying to malicious attacks.

“Honest poison” instances can be unintentionally and unknowingly inserted into a model’s training data. For example, consider generative models trained on open-source code. Countless instances of buggy or insecure code are posted to public repositories like GitHub. Any model trained on this “dangerous” but non-malicious code is vulnerable [Pea+22; Pea+23]. Methods like those proposed in this dissertation not only provide robustness against adversarial poison but such honest poison as well. Adversarial robustness simply provides a vehicle to study worst-case training set perturbations.

In addition, while we propose and analyze GAS in the context of adversarial robustness, understanding the causal relationship between training data and model predictions generalizes to any setting where predictions must be interpretable. A better understanding of why models misbehave should enable the development of better models in the future. With algorithmic decision making increasingly common, black-box decisions will no longer be tolerated by society or regulators [GF17]. Insights into why poisoning attacks succeed may generalize to understanding why models learn spurious correlations in non-malicious settings as well.

Future Directions

Despite decades of study [Coo77; CW82], existing methods to make models (provably) robust to training set perturbations and spurious correlations remain rudimentary. This dissertation takes a step towards improved model robustness, yet significant work remains. Below, we briefly outline ideas for future research directions to improve model training’s robustness.

Lowering Certification’s Performance Penalty As Chapters 4 and 5 demonstrate, certified robustness against poisoning and backdoor attacks is not free. Non-trivial certified robustness against poisoning and backdoor attacks currently entails a (significant) performance penalty [FZ20; Zha+19]. This weakness is common to all existing certified methods that provide non-trivial robustness guarantees [LF21; WLF22a; Rez+23]. Expecting certified robustness to be achieved without any decrease in model performance is unrealistic. There are no free lunches. However, certified methods will not be a realistic option in practice until their deleterious side effects are substantially reduced. Without practical robust methods, society will bear the cost when today’s brittle models inevitably fail in real-world, safety-critical applications.

Unifying Robustness to Training and Test Perturbations Conventional wisdom treats training and test robustness as separate tasks with proposed methods targeting one task or the other. Chapter 5’s feature partition aggregation (FPA) is rather unique in that a single method simultaneously provides certified training and test robustness. An obvious limitation of FPA is its restriction to a single perturbation model, i.e., the ℓ_0 ball. It remains an open question whether other effective defenses over the union of training and test attacks exist. As Chapter 5 discusses, Weber et al. [Web+23] propose a defense that provides robustness of ℓ_2 training and test

attacks, but their robustness guarantees are very minimal. The more threat vectors a (certified) defense effectively covers, the more likely that defense will be deployed in practice. This broad statement includes defining defenses that are simultaneously robust to both training and test attacks.

Efficient Influence Analysis and Estimation Put simply, GAS is slow [HL22a, Sec. E.5]. Estimating the training set’s pointwise influence on a single prediction can take hours, even for relatively simple models [HL22b]. For influence analysis to be a practical tool, influence estimation must be at least one to two orders of magnitude faster [HL22b]. Recently, more efficient influence analysis methods have been proposed [Par+23], but significant work remains.

APPENDIX A

NOMENCLATURE REFERENCE

Table A.8. **General Nomenclature Reference:** This table contains symbols that are relevant to one or more chapters in this dissertation. Related symbols are grouped together with groups separated by dotted lines.

LightGBM	Gradient-boosted decision tree model architecture by Ke et al. [Ke+17]
XGBoost	Gradient-boosted decision tree model architecture by Chen and Guestrin [CG16]
DPA	Deep partition aggregation certified poisoning defense proposed by Levine and Feizi [LF21] (Sec. 3.2.2)
FA	(Deterministic) finite aggregation certified poisoning defense proposed by Wang et al. [WLF22a] (Sec. 3.2.2)
$[k]$	Integer set $\{1, \dots, k\}$
$2^{[k]}$	Power set of integer set $[k]$
$\mathbb{1}[q]$	Indicator function where $\mathbb{1}[q] = 1$ if q is true and 0 otherwise
med A	Median of (multi)set A
$H(k)$	k -th harmonic number where $H(k) = \sum_{i=1}^k \frac{1}{i}$
pp	Percentage points
\mathcal{X}	Feature domain where $\mathcal{X} \subseteq \mathbb{R}^d$
\mathbf{x}	Feature vector where $\forall_{\mathbf{x}} \mathbf{x} \in \mathcal{X}$
d	Feature dimension where $\forall_{\mathbf{x}} \mathbf{x} = d$
$[d]$	Complete feature set
\mathcal{Y}	Label set where $\mathcal{Y} \subseteq \mathbb{R}$
y	Instance label where $\forall_y y \in \mathcal{Y}$
\mathcal{Z}	Instance space where $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$
z	Feature vector, label tuple where $z := (\mathbf{x}, y) \in \mathcal{Z}$
z_{te}	Arbitrary test instance $z_{\text{te}} := (\mathbf{x}_{\text{te}}, y_{\text{te}}) \in \mathcal{Z}$ with true label $y_{\text{te}} \in \mathcal{Y}$
n	Number of training instances
z_i	i -training index where $z_i := (\mathbf{x}_i, y_i) \in \mathcal{D}$
\mathcal{D}	Training set where $\mathcal{D} := \{z_i\}_{i=1}^n$
f	A model (ensemble, instance-based learner, neural network, etc.) where $f : \mathcal{X} \rightarrow \mathcal{Y}$
$f(\mathbf{x}_{\text{te}})$	Model f 's prediction for test feature vector \mathbf{x}_{te}
L	Number of submodels in ensemble f
f_l	l -th submodel in ensemble f where $f_l : \mathcal{X} \rightarrow \mathcal{Y}$ and $l \in [L]$
θ	Model parameters where $\theta \in \mathbb{R}^p$
\mathcal{L}	Loss function where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$
$\mathcal{L}(z; \theta_t)$	Empirical risk of $z = (\mathbf{x}, y)$ w.r.t. model parameters θ_t where $\mathcal{L}(z; \theta_t) = \mathcal{L}(f(\mathbf{x}; \theta), y)$

Table A.9. **Chapter 4 Nomenclature Reference:** Notation specific to Chapter 4 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

k NN	Vanilla k -nearest neighbors (Sec. 4.4.1)
k NN-m	k -nearest neighbors with median as the decision function (Sec. 4.4.1)
k NN-CR	Our k NN-based certified regressor (Sec. 4.4.1)
r NN	Radius nearest neighbors (Sec. 4.4.2)
PCR	Our <u>p</u> artitioned <u>c</u> ertified <u>r</u> egressor (Sec. 4.5.1)
W-PCR	Our <u>w</u> eighted-cost <u>p</u> artitioned <u>c</u> ertified <u>r</u> egressor (Sec. 4.5.2)
PCR	Our <u>o</u> verlapping <u>c</u> ertified <u>r</u> egressor (Sec. 4.6.1)
W-PCR	Our <u>w</u> eighted-cost <u>o</u> verlapping <u>c</u> ertified <u>r</u> egressor (Sec. 4.6.2)
ξ	One-sided upper bound for regression robustness certification where $f(\mathbf{x}_{te}) \leq \xi$
ξ_l	Two-sided lower bound for robustness certification where $\xi_l \leq f(\mathbf{x}_{te}) \leq \xi_u$
ξ_u	Two-sided upper bound for robustness certification where $\xi_l \leq f(\mathbf{x}_{te}) \leq \xi_u$
m	Number of training set blocks
h_{tr}	Training set partitioning function where $h_{tr} : \mathcal{Z} \rightarrow [m]$
h_f	Overlapping training set block assignment function where $h_f : [m] \rightarrow 2^{[L]}$
j	Training set block index where $j \in [m]$
$D^{(j)}$	j -th training set block where $D^{(j)} := \{z \in \mathcal{D} : h_{tr}(z) = j\}$ and $\forall_{j' \neq j} D^{(j)} \cap D^{(j')} = \emptyset$
$\mathcal{N}(\mathbf{x}_{te})$	Nearest-neighbors neighborhood (multi)set for test feature vector \mathbf{x}_{te}
r_l	Number of training set modifications required to violate invariant $\xi_l \leq f_l(\mathbf{x}_{te}) \leq \xi_u$. Note that r_l is one larger than f_l 's certified robustness
r_{\max}	Maximum submodel modification requirement where $r_{\max} := \max_l r_l$
\mathcal{D}_l	Submodel f_l 's training set where for overlapping regression $\mathcal{D}_l = \bigcup_{\substack{j \in [m] \\ l \in h_f(j)}} D^{(j)}$
q	Inverse of the fraction of the training set used to train each submodel, where $\forall_l q = \frac{n}{ \mathcal{D}_l }$
$d^{(j)}$	Spread degree of training set block $D^{(j)}$ where $d^{(j)} := h_f(D^{(j)}) $
d_{\max}	Maximum spread degree where $d_{\max} := \max_j d^{(j)}$
\mathcal{T}_1	Set of ensemble submodels predicting $f_l(\mathbf{x}_{te}) \leq \xi$ where $\mathcal{T}_1 \subseteq [L]$
\mathcal{V}	Real-valued (multi)set, e.g., k NN neighborhood or ensemble submodel predictions, where $L = \mathcal{V} $
\mathcal{V}_l	Lower thresholded real-valued multiset where $\mathcal{V}_l := \{\nu_l \in \mathcal{V} : \nu_l \leq \xi\}$
\mathcal{V}_u	Upper thresholded real-valued multiset where $\mathcal{V}_u := \{\nu_l \in \mathcal{V} : \nu_l > \xi\}$
$\mathcal{V}_{\pm 1}$	Binarized multiset where $\mathcal{V}_{\pm 1} := \{\text{sgn}(\nu_l) : \nu_l \in \mathcal{V}\}$
$\tilde{\mathcal{V}}$	Adversarially perturbed real-valued (multi)set formed from (multi)set \mathcal{V}
\mathcal{R}	Weight set where $\mathcal{R} := \{r_l : l \in [L]\}$
\mathcal{R}_1	Weight set corresponding to values set \mathcal{V}_1 where $\mathcal{R}_1 := \{r_l \in \mathcal{R} : \nu_l \in \mathcal{V}_1\}$

(Continued ...)

Table A.9. **Chapter 4 Nomenclature Reference (Continued)**: Notation specific to Chapter 4 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

Δ	Midpoint distance where $\Delta := \mathcal{V} - \lceil \frac{L}{2} \rceil$
$\tilde{\mathcal{R}}_1$	Δ smallest values in \mathcal{R}_1
ILP	Integer linear program
$\omega^{(j)}$	ILP integral variable representing the number of instance modifications made to training set block $D^{(j)}$
δ_l	ILP binary variable which equals 1 if submodel f_l has been perturbed such that $f_l(\mathbf{x}_{te}) > \xi$ and 0 otherwise
σ	ILP binary variable which equals 1 if in the case of weighted analysis and 0 otherwise
PSMC	Partial set multicover
G	Upper-bound on certified robustness R returned by a greedy algorithm

Table A.10. **Chapter 5 Nomenclature Reference:** Notation specific to Chapter 5 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

FPA	Our certified defense, feature partition aggregation, against sparse poisoning, backdoor, evasion, and patch attacks
RA	Randomized ablation. Certified ℓ_0 -norm evasion defense. Proposed by Levine and Feizi [LF20b] and subsequently improved by Jia et al. [Jia+22b]
DRS	(De)randomized smoothing certified patch defense proposed by Levine and Feizi [LF20a]. Based on randomized ablation
Patch IBP	Certified patch defense based on interval bound propagation proposed by Chiang et al. [Chi+20]
BAGCERT	Certified patch defense proposed by Metzen and Yatsura [MY21]
RAB	Robustness against backdoors certified defense proposed by Weber et al. [Web+23]
R	Pointwise certified feature robustness – feature partition aggregation’s certification objective (Def. 5.1)
R_{med}	Median certified feature robustness w.r.t. a dataset’s test set
ρ	Pointwise ℓ_0 -norm certified evasion-only robustness (Def. 5.2). A weaker guarantee than certified feature robustness.
ρ_{med}	Median ℓ_0 -norm certified evasion-only robustness w.r.t. a dataset’s test set
$\ \mathbf{w}\ _0$	ℓ_0 norm for vector \mathbf{w} , i.e., the number of non-zero elements in \mathbf{w}
\mathbf{X}_j	j -th column of matrix \mathbf{X} where $j \in [d]$ and $\mathbf{X}_j \in \mathbb{R}^n$
$\mathbf{X} \ominus \mathbf{X}'$	Set of column indices over which equal-size matrices \mathbf{X} and \mathbf{X}' differ, where $\mathbf{X} \ominus \mathbf{X}' = \{j \in [d] : \mathbf{X}_j \neq \mathbf{X}'_j\}$
x_j	j -th dimension of vector \mathbf{x} where $j \in [d]$ and $x_j \in \mathbb{R}$
$\mathbf{x} \ominus \mathbf{x}'$	Set of dimensions over which vectors \mathbf{x} and \mathbf{x}' differ where $\mathbf{x} \ominus \mathbf{x}' = \{j \in [d] : x_j \neq x'_j\}$
$d_{\text{sym}}(\mathcal{D}, \mathcal{D}')$	Symmetric difference between sets \mathcal{D} and \mathcal{D}'
f	Voting-based, ensemble classifier trained over partitioned feature sets where $f : \mathcal{X} \rightarrow \mathcal{Y}$
\mathcal{S}_l	Feature subset considered by the l -th submodel during training and test where $\mathcal{S}_l \subset [d]$ and $\bigsqcup_{l=1}^L \mathcal{S}_l = [d]$
$\mathbf{x}_{\mathcal{S}_l}$	Subvector of $\mathbf{x} \in \mathcal{X}$ restricted to feature subset $\mathcal{S}_l \subset [d]$
D_l	Training set for the l -th submodel
$\dot{c}_y(\mathbf{x})$	Submodel vote count for label y and feature vector \mathbf{x} where $\dot{c}_y(\mathbf{x}) := \sum_{l=1}^L \mathbb{1}[f_l(\mathbf{x}) = y]$
$\text{GAP}_{\text{vote}}(y, y'; \mathbf{x})$	Submodel vote gap for instance $\mathbf{x} \in \mathcal{X}$ and labels $y, y' \in \mathcal{Y}$ where $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) := \dot{c}_y(\mathbf{x}) - \dot{c}_{y'}(\mathbf{x}) - \mathbb{1}[y' < y]$
y_{pl}	Submodel plurality label where $y_{\text{pl}} := \arg \max_{y \in \mathcal{Y}} \dot{c}_y(\mathbf{x})$ and ties broken by preferring the smaller label. FPA ensemble prediction under the plurality label decision function (Sec. 5.3.1)

(Continued ...)

Table A.10. **Chapter 5 Nomenclature Reference (Continued)**: Notation specific to Chapter 5 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

y_{ru}	Label with the second-most submodel votes (i.e., the “runner up”) where $y_{\text{ru}} := \arg \max_{y' \in \mathcal{Y} \setminus y_{\text{pl}}} \dot{c}_{y'}(\mathbf{x})$
$g_l(\mathbf{x}, y)$	Logit value predicted by the l -th submodel for instance $\mathbf{x} \in \mathcal{X}$ and label $y \in \mathcal{Y}$ where $g_l(\mathbf{x}, y) \in [0, 1]$
y_{RO}	FPA ensemble prediction under the run-off decision function (Sec. 5.3.2).
\tilde{y}_{RO}	Label in the run-off decision function’s second round that is not selected as the run-off prediction where $\tilde{y}_{\text{RO}} := \{y_{\text{pl}}, y_{\text{ru}}\} \setminus y_{\text{RO}}$
$\ddot{c}_{\mathbf{x}}(y; y')$	Pairwise logit count for instance \mathbf{x} and label $y \in \mathcal{Y}$ w.r.t. label $y' \in \mathcal{Y}$ where $\ddot{c}_{\mathbf{x}}(\mathbf{x}; y') := \sum_{l=1}^L \mathbb{1}[g_l(\mathbf{x}, y) > g_l(\mathbf{x}, y')]$
$\text{GAP}_{\text{logit}}(y, y'; \mathbf{x})$	Submodel logit vote gap for labels $y, y' \in \mathcal{Y}$ where $\text{GAP}_{\text{logit}}(y, y'; \mathbf{x}) := \ddot{c}_{\mathbf{x}}(\mathbf{x}; y') - \ddot{c}_{\mathbf{x}}(\mathbf{x}; y) - \mathbb{1}[y' < y]$
e	Randomized ablation hyperparameter – number of kept features with the other $(d - e)$ ablated where $e \in \mathbb{N}$.
BS	Blocking smoothing ablation paradigm used by (de)randomized smoothing [LF20a]

Table A.11. **Chapter 6 Nomenclature Reference:** Notation specific to Chapter 6 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

GAS	Gradient aggregated similarity – renormalized TracInCP (Sec. 6.3.3)
GAS-L	Layerwise gradient aggregated similarity (Sec. 6.3.3)
FIT	Our Framework for Identifying Targets of a poisoning and backdoor attack (Sec. 6.4)
TracIn	Tracing gradient descent influence estimator by Pruthi et al. [Pru+20]
TracInCP	TracIn Checkpoint – a heuristic version of TracIn by Pruthi et al. [Pru+20]
LOO	Leave-one-out influence [CW82]
IF	Influence functions estimator by Koh and Liang [KL17]
Rep. Pt.	Representer point influence estimator by Yeh et al. [Yeh+18]
Rn.	Renormalization for influence estimation (Sec. 6.3.3)
$\langle \cdot, \cdot \rangle$	Dot product
$\text{sgn}(\cdot)$	Signum function
$\mathcal{I}(z_i, \hat{z}_{te})$	Pointwise Influence of training instance $z_i \in \mathcal{D}$ on test instance $\hat{z}_{te} \in \mathcal{Z}$
$\tilde{\mathcal{I}}(z_i, \hat{z}_{te})$	Renormalized Influence of training instance $z_i \in \mathcal{D}$ on test instance $\hat{z}_{te} \in \mathcal{Z}$
y_{targ}	Target (i.e., source) label the attacker seeks to have mislabeled
y_{adv}	Attacker’s adversarial (i.e., destination) label
z_{te}	Arbitrary test instance $z_{te} := (\mathbf{x}_{te}, y_{te}) \in \mathcal{Z}$ with true label $y_{te} \in \mathcal{Y}$
\hat{z}_{te}	Arbitrary test instance where $\hat{z}_{te} := (\mathbf{x}_{te}, \hat{y}_{te}) \in \mathcal{Z}$ with predicted label $\hat{y}_{te} := f(\mathbf{x}_{te})$
z_i	i -training index where $z_i := (\mathbf{x}_i, y_i) \in \mathcal{D}$
\mathcal{D}	Training set where $\mathcal{D} := \{z_i\}_{i=1}^n$
\mathcal{D}_{adv}	Set of adversarially perturbed training instances where $\mathcal{D}_{\text{adv}} \subseteq \mathcal{D}$
\mathcal{D}_{cl}	Set of clean (unperturbed) training instances where $\mathcal{D}_{\text{cl}} := \mathcal{D} \setminus \mathcal{D}_{\text{adv}}$
$f(\mathbf{x}_{te})$	Model f ’s prediction for test feature vector \mathbf{x}_{te}
\hat{y}_{te}	Compact notation for predicted label for \mathbf{x}_{te} given parametric model f
T	Number of iterations performed by the training algorithm
t	Iteration number s.t. $t \in \{1, \dots, T\}$
\mathcal{T}	Subset of the training iterations considered by GAS and TracInCP where $\mathcal{T} \subseteq [T]$
η_t	Learning rate for iteration $t \in [T]$ where $\eta_t > 0$
λ	Weight decay hyperparameter
\mathcal{B}_t	Iteration t ’s training batch where $\mathcal{B}_t \subseteq \mathcal{D}$
b	Batch size where $\forall_{t \in [T]} b = \mathcal{B}_t $
θ	Model parameters where $\theta \in \mathbb{R}^P$
θ_0	Initial model parameters at the start of training
θ_t	Model parameters for iteration $t \in [T]$
θ_T	Model parameters at the end of training
\mathcal{P}	Serialized training parameters where $\mathcal{P} := \{(\eta_t, \theta_{t-1})\}_{t=1}^T$

(Continued ...)

Table A.11. **Chapter 6 Nomenclature Reference (Continued)**: Notation specific to Chapter 6 with related symbols grouped together. Groups are separated by dotted lines. Note that this table spans multiple pages.

\mathbf{f}_i	Penultimate feature representation for training feature vector \mathbf{x}_i
\mathbf{f}_{te}	Penultimate feature representation for test feature vector \mathbf{x}_{te}
$g_i^{(t)}$	Gradient for $z_i \in \mathcal{D}$ w.r.t. parameters θ_t where $g_i^{(t)} := \nabla_{\theta} \mathcal{L}(z_i; \theta_t)$
$\hat{g}_{te}^{(t)}$	Gradient for $\hat{z}_{te} \in \mathcal{Z}$ w.r.t. parameters θ_t where $\hat{g}_{te}^{(t)} := \nabla_{\theta} \mathcal{L}(\hat{z}_{te}; \theta_t)$
H_{θ}	Training set empirical risk Hessian where $H_{\theta} := \frac{1}{n} \sum_{z_i \in \mathcal{D}} \nabla_{\theta}^2 \mathcal{L}(z_i; \theta_T)$
H_{θ}^{-1}	Inverse of risk Hessian H_{θ}
\hat{z}_{targ}	Target test instance where $\hat{z}_{targ} := (\mathbf{x}_{targ}, \hat{y}_{targ}) \in \mathcal{Z}$
$\hat{\mathcal{Z}}_{te}$	Set of test instances to be analyzed for attack targets
\mathbf{v}	Influence vector where $\mathbf{v} \in \mathbb{R}^n$
μ	Median influence score (Sec. 6.4.2)
Q	Q -estimator Rousseeuw and Croux [RC93] for influence vector \mathbf{v} (Sec. 6.4.2)
MITIGATE	Target-driven mitigation function (Sec. 6.4.3)
κ	Heavy-tail influence cutoff count
σ	Influence anomaly score
ζ	Target driven sanitization cutoff score where $\zeta \in \mathbb{R}$
RETRAIN	Model retrain method
β	Adaptive attack tradeoff hyperparameter based on Zhu et al.'s [Zhu+19] vision poison attack

APPENDIX B

PROOFS

This chapter contains previously published, coauthored material [HL22a; HL23c; HL23a]. Hammoudeh designed and wrote all proofs in this section. Lowd provided supervision and editorial suggestions.

Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security. CCS’22*. Los Angeles, CA: Association for Computing Machinery, 2022. URL: <https://arxiv.org/abs/2201.10055>

Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning. SaTML’23*. 2023. URL: <https://arxiv.org/abs/2208.13904>

Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning. AdvML-Frontiers’23*. 2023. URL: <https://arxiv.org/abs/2302.11628>

This section provides proofs for all theoretical contributions in this dissertation.

B.1 Proofs for Chapter 4

This section contains the proofs for the theoretical contributions that are either in or relevant to Chapter 4.

Lemma B.1. *For real multiset \mathcal{V} of cardinality $L > 1$, if an arbitrarily-large value is inserted into \mathcal{V} or the smallest value in \mathcal{V} is deleted, the resulting sets’ medians are equivalent.*

Proof. For simplicity and w.l.o.g., assume \mathcal{V} is ordered where $\nu_1 \leq \dots \leq \nu_L$. Let $\alpha \in [1, L]$ denote the median's index. If L is odd, $\alpha = \lceil \frac{L}{2} \rceil$; otherwise, when L is even, $\alpha = \frac{L}{2} + \frac{1}{2}$, i.e., the midpoint between the $\frac{L}{2}$ -th and $(\frac{L}{2} + 1)$ -th largest values in \mathcal{V} .

Consider first an arbitrarily-large insertion when L odd. Each insertion increases the set's cardinality by 1. When \mathcal{V} 's cardinality is odd, then the cardinality of this new set after the first insertion is even. Therefore, this new set's median has index

$$\alpha' := \frac{L+1}{2} + \frac{1}{2} \quad \triangleright \text{Median's index for new set of even size } T+1 \quad (\text{B.1})$$

$$= \left\lceil \frac{L}{2} \right\rceil + \frac{1}{2} \quad (\text{B.2})$$

$$= \alpha + \frac{1}{2}. \quad (\text{B.3})$$

Since $L \geq \alpha'$ and the inserted element is larger than all values in \mathcal{V} , the value corresponding to index $(\lceil \frac{L}{2} \rceil - \frac{1}{2})$ is equivalent for both original set \mathcal{V} and the new set after the insertion.

Next, consider the deletion case for odd L . Similar to above, the cardinality of the modified set after one deletion is even; therefore, this modified set's median has index

$$\alpha'' := \frac{L-1}{2} + \frac{1}{2} \quad \triangleright \text{Median's index for new set of even size } T-1 \quad (\text{B.4})$$

$$= \left\lfloor \frac{L}{2} \right\rfloor - \frac{1}{2} \quad (\text{B.5})$$

$$= \alpha - \frac{1}{2}. \quad (\text{B.6})$$

This new set's cardinality is one smaller than the original set with the smallest element removed. Hence, the value corresponding to index $(\lfloor \frac{L}{2} \rfloor - \frac{1}{2})$ in this shrunken set equals the value at index $(\lceil \frac{L}{2} \rceil + \frac{1}{2})$ in original set \mathcal{V} .

Since indices α' and α'' correspond to the same value in \mathcal{V} , the resulting sets' medians are equivalent. \square

The primary takeaway from Lemma B.1 is that under the insertion/deletion paradigm, worst-case insertions and deletions are interchangeable. Note that for our purposes, there is an edge case where worst-case insertions and deletions exhibit divergent behavior. Specifically, after L deletions (i.e., all elements in \mathcal{V} are removed), the median of an empty set is not generally defined. In contrast, the median after L arbitrarily-large insertions is itself arbitrarily large. For consistency, we define the empty set's median as ∞ to match the insertion case.

Proof of Lemma 4.2

Proof. Let \mathcal{V}_1 be all elements in \mathcal{V} that do not exceed ξ .

Under the swap paradigm, the optimal strategy to maximally increase a set's median is to iteratively replace the set's smallest value with ∞ . Apply this optimal strategy to \mathcal{V} . After one swap, the resulting set contains $|\mathcal{V}_1| - 1$ elements that are less than or equal to ξ . After two swaps, there are $|\mathcal{V}_1| - 2$ such elements with each subsequent swap's effects proceeding inductively. Once the modified set contains exactly $\lceil \frac{L}{2} \rceil$ elements less than or equal to ξ , no additional swaps are possible without causing the resulting set's median to exceed ξ .

Therefore, by induction, the maximum number of swaps that can be performed on \mathcal{V} and it remains guaranteed that the resulting set's median does not exceed ξ is

$$R = |\mathcal{V}_1| - \left\lceil \frac{L}{2} \right\rceil. \tag{B.7}$$

□

Proof of Lemma 4.3

Proof. For simplicity and w.l.o.g., assume \mathcal{V} is ordered where $\nu_1 \leq \dots \leq \nu_L$. An attacker's optimal insertion strategy is to insert arbitrarily-large values into \mathcal{V} while

the optimal deletion strategy is to always delete \mathcal{V} 's smallest value. Lemma B.1 proves that these worst-case operations perturb the median identically so we only consider insertions below.

Let α denote the median's index. If L is odd, $\alpha = \lceil \frac{L}{2} \rceil$; otherwise, when L is even, $\alpha = \frac{L}{2} + \frac{1}{2}$, i.e., the midpoint between the $\frac{L}{2}$ -th and $(\frac{L}{2} + 1)$ -th largest values in \mathcal{V} .

Each insertion increases the set's size by 1. When \mathcal{V} 's size is odd, then the size of this new set after the first insertion is even. Therefore, this new set's median has index

$$\alpha' := \frac{L+1}{2} + \frac{1}{2} \quad \text{Median's index for new even size } T+1 \quad (\text{B.8})$$

$$= \left\lceil \frac{L}{2} \right\rceil + \frac{1}{2} \quad (\text{B.9})$$

$$= \alpha + \frac{1}{2}. \quad (\text{B.10})$$

The analysis is essentially identical when L is even and is excluded for brevity. Note that each insertion always increases the median's index α by $\frac{1}{2}$.

As long as $\alpha \leq |\mathcal{V}_1|$, it is guaranteed that $\text{med } \mathcal{V} \leq \xi$. Since each insertion changes α by $\frac{1}{2}$, then $2(|\mathcal{V}_1| - \alpha)$ arbitrary insertions can be made in \mathcal{V} with it remaining guaranteed that the modified set's median does not exceed ξ . Regardless of whether L is odd or even,¹ it holds that

$$R \geq 2(|\mathcal{V}_1| - \alpha) = 2|\mathcal{V}_1| - 2\alpha = 2|\mathcal{V}_1| - L - 1. \quad (\text{B.11})$$

□

Proof of Lemma 4.4

Proof. The first portion of this proof follows the same argument as the proof of Lemma 4.2, with one primary difference. There, the optimal strategy to perturb

¹When L is odd, $2\alpha = 2\lceil \frac{L}{2} \rceil = L + 1$ while in the even case $2\alpha = 2(\frac{L}{2} + \frac{1}{2}) = L + 1$.

\mathcal{V} 's median swapped out the smallest values in \mathcal{V} first. For the weighted version, the optimal (greedy) strategy swaps out whichever value in $\mathcal{V}_1 \subseteq \mathcal{V}$ has the smallest weight.

To perturb \mathcal{V} 's median above ξ , it is sufficient to swap any $|\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$ values in \mathcal{V}_1 with an arbitrarily large replacement. For simplicity and without loss of generality, let $\tilde{r}_1, \dots, \tilde{r}_{|\mathcal{V}_1|}$ be the weights of the elements in \mathcal{V}_1 arranged in ascending order. Define $\Delta := |\mathcal{V}_1| - \lceil \frac{L}{2} \rceil$. Applying Lemma 4.2, up to Δ values in \mathcal{V} can be replaced without perturbing the median. This entails a minimum cost of

$$R \geq \sum_{l=1}^{\Delta} \tilde{r}_l. \quad (\text{B.12})$$

Denote the $(\Delta + 1)$ -th largest weight in \mathcal{R}_1 as $\tilde{r}_{\Delta+1}$. Observe that adding $\tilde{r}_{\Delta+1} - 1$ to Eq. (B.12) is insufficient to swap out any remaining elements in \mathcal{V}_1 since all elements with weight less than $\tilde{r}_{\Delta+1}$ are already replaced and all remaining elements have weight at least $\tilde{r}_{\Delta+1}$. Therefore, the certified robustness is

$$R = (\tilde{r}_{\Delta+1} - 1) + \sum_{l=1}^{\Delta} \tilde{r}_l \quad (\text{B.13})$$

$$= \sum_{l=1}^{\Delta+1} \tilde{r}_l - 1 \quad (\text{B.14})$$

$$= \sum_{l=1}^{|\mathcal{V}_1| - \lceil \frac{L}{2} \rceil + 1} \tilde{r}_l - 1 \quad (\text{B.15})$$

$$= \sum_{r \in \tilde{\mathcal{R}}_1} r - 1. \quad (\text{B.16})$$

Moreover, increasing Eq. (B.16) by one would allow for the $(\Delta + 1)$ -th largest value in \mathcal{V} to be swapped, which would in turn perturb the set's median above ξ . Therefore, Eq. (B.16)'s bound is tight. \square

Proof of Lemma 4.5

Proof. The median perturbation paradigms formalized in Lemmas 4.2, 4.3, and 4.4 calculate their certified robustness using three values, namely: L , $|\mathcal{V}_1|$, and $\lceil \frac{L}{2} \rceil$. If these three values are equivalent for \mathcal{V} and $\mathcal{V}_{\pm 1}$, then their associated certified robustness (R) must also be equal.

Since $|\mathcal{V}| = |\mathcal{V}_{\pm 1}|$, they have equivalent L and $\lceil \frac{L}{2} \rceil$. By definition, the binarization of \mathcal{V} to $\mathcal{V}_{\pm 1}$ does not change the value of $|\mathcal{V}_1|$ either. Therefore, for all three median perturbation paradigms, binary multiset $\mathcal{V}_{\pm 1}$ and real multiset \mathcal{V} have equivalent certified robustness R . \square

Proof of Theorem 4.6

Proof. For fixed-population IBLs, certifying that $f(\mathbf{x}_{te}) \leq \xi$ simplifies to median perturbation under Sec. 4.2.1's unweighted swap paradigm since all necessary criteria are met, namely that

1. f 's decision function is a median operation over a set of values, i.e., $f(\mathbf{x}_{te}) := \text{med } \mathcal{N}(\mathbf{x}_{te})$.
2. Neighborhood $\mathcal{N}(\mathbf{x}_{te})$ has fixed cardinality L , and L is odd.
3. A worst-case modification to training set \mathcal{D} causes an element in $\mathcal{N}(\mathbf{x}_{te})$ to be replaced with a different one.

Lemma 4.2, therefore, provides a (lower) bound on the number of training set modifications that can be made without the resulting model violating the requirement that $f(\mathbf{x}_{te}) \leq \xi$. That is why certified robustness R in Eqs. (4.7) and (4.2) (Thm. 4.6 & Lem. 4.2, resp.) are equivalent. \square

Proof of Theorem 4.7

Proof. This proof follows a very similar structure as Theorem 4.6’s proof above. The primary distinction is that a different median perturbation paradigm from Sec. 4.2 is needed here.

For region-based IBLs, certifying that $f(\mathbf{x}_{te}) \leq \xi$ simplifies to median perturbation under Sec. 4.2.2’s insertion/deletion paradigm since the three necessary criteria are met:

1. f ’s decision function is a median operation over a set of values, i.e., $f(\mathbf{x}_{te}) := \text{med } \mathcal{N}(\mathbf{x}_{te})$.
2. Neighborhood cardinality L is not fixed but can increase and/or decrease.
3. Each modification of $\mathcal{N}(\mathbf{x}_{te})$ takes the form of either an insertion or deletion, i.e., not swaps.

Therefore, Lemma 4.3 bounds the total number of training set insertions/deletions that can be performed without violating the requirement that $f(\mathbf{x}_{te}) \leq \xi$. That is why R ’s definition in Eq. (4.8) is identical to Eq. (4.3). \square

Proof of Theorem 4.8

Proof. Certifying here that $f(\mathbf{x}_{te}) \leq \xi$ simplifies to median perturbation under the unweighted swap paradigm since all necessary criteria are satisfied, specifically that

1. f ’s decision function is a median operation over a set of fixed, deterministic values, i.e., $f(\mathbf{x}_{te}) := \text{med } \{f_l(\mathbf{x}_{te}; 1), \dots, f_l(\mathbf{x}_{te}; L)\}$.
2. Since the submodels are trained on disjoint data/feature regions, a change to one submodel (i.e., value $f_l(\mathbf{x}_{te})$) has no effect on any other submodel (value).

3. Each submodel perturbation causes an existing value in the set to be replaced by a new value.
4. L is fixed and odd-valued.
5. The cost to change any submodel (i.e., value) is one, i.e., $\forall_l r_l = 1$.

Lemma 4.2 provides a (lower) bound on the number of training set modifications that can be performed without violating the requirement that $f(\mathbf{x}_{te}) \leq \xi$. Certified robustness R in Eq. (4.9) is then identical to Lemma 4.2's Eq. (4.2). \square

Proof of Theorem 4.9

Proof. Here, we extend the argument in Theorem 4.8's proof to the weighted case. Four of the five criteria in Thm. 4.8's proof still hold, specifically that

1. f 's decision function is a median over a set of values.
2. Each submodel is independent and deterministic.
3. Modifications to the set of values take the form of swaps.
4. L is fixed and odd-valued.

The only difference is that the perturbations are weighted where each value $f_l(\mathbf{x}_{te})$ now has an associated cost $r_l \geq 0$. Therefore, Sec. 4.2.3's weighted swap paradigm applies. Certified robustness R in Eq. (4.10) follows directly from and is identical to R in Lemma 4.4's Eq. (4.5). \square

Proof of Lemma 4.10

Proof. To prove a problem is NP-hard, it suffices to show that there exists a polynomial time reduction from a known NP-hard problem to it. As explained in Sec. 4.6.1, partial set cover is NP-hard [Sla97b; Sla97a]. Below we map partial set cover to overlapping certified regression.

Let $U := [L']$ be a ground set of L' elements, and let $\mathbf{Q} := \{\mathcal{Q}_1, \dots, \mathcal{Q}_m\}$ be a collection of sets where each $\mathcal{Q}_j \subseteq U$ and

$$\bigcup_{\mathcal{Q} \in \mathbf{Q}} \mathcal{Q} = U.$$

The goal is to find the subcover $\mathcal{F} \subseteq \mathbf{Q}$ of minimum cardinality s.t.

$$\Delta \leq \left| \bigcup_{\mathcal{Q} \in \mathcal{F}} \mathcal{Q} \right|,$$

where $\Delta \in [L']$.

It is straightforward to map the above to overlapping certified regression. Let the ensemble have $(2L' + 1)$ submodels. Function h_{tr} partitions the training set into m blocks with the blocks denoted $D^{(1)}, \dots, D^{(m)}$. Define the block mapping function as:

$$h_f(j) := \begin{cases} \mathcal{Q}_j, & j \leq L' \\ \emptyset, & \text{Otherwise} \end{cases}. \quad (\text{B.17})$$

Intuitively, each of the first L' submodels is trained on one of the subsets in \mathbf{Q} , while the remaining models are not trained on any data.

Let all submodels be constant a function. Define the submodel function as

$$f_l(\mathbf{x}_{\text{te}}) := \begin{cases} -\infty, & l \leq \Delta + \lceil \frac{L'}{2} \rceil \\ \infty, & \text{Otherwise} \end{cases}. \quad (\text{B.18})$$

For any finite ξ , $|\mathcal{V}_l| = \Delta + \lceil \frac{L'}{2} \rceil$. Applying Theorem 4.8, the number of submodels overlapping certified regression perturbs is $|\mathcal{V}_l| - \lceil \frac{L'}{2} \rceil = \Delta$.

Overlapping certified regression’s robustness R is the solution to the original partial set cover problem because

1. Only models with index $l \leq L'$ will be perturbed since all other submodels have no training data.
2. The training set of each of these L' submodels maps directly to a subset in \mathbf{Q} .
3. Overlapping certified regression seeks to find the minimum number of dataset blocks that must be modified to perturb the median prediction. In this formulation, the number of blocks to be modified is Δ – same as in the original partial-set cover problem.

If overlapping certified regression were solvable in polynomial-time, then partial set cover would also be solvable in polynomial time. However, partial set cover is NP-hard, meaning overlapping certified regression must also be NP-hard. \square

Proof of Corollary 4.10.1

Proof. From Lemma 4.10 above, (unweighted) overlapping certified regression is NP-hard. The unweighted case trivially maps to the weighted one where $\forall_l r_l = 1$. Therefore, weighted OCR must be at least as hard as the unweighted case meaning W-OCR is also NP-hard. \square

Proof of Lemma 4.11

Proof. By construction

Given a deterministic training algorithm, a model’s prediction can be certified against the deletion of any subset $D \subset \mathcal{D}$ by training a model on just dataset $\mathcal{D} \setminus D$ and verifying the prediction does not violate the associated *invariant*, i.e., $f_{\mathcal{D} \setminus D}(\mathbf{x}_{te}) \leq \xi$.

Consider training a separate model on each subset of \mathcal{D} of size at least $n - r + 1$. If *all* of those models also satisfy the invariant, then by construction, $r - 1$ deletions or fewer are insufficient to violate the invariant. If $r - 1$ deletions are not enough, then at least r deletions are required. \square

Lemma 4.11’s proof above only applies if model prediction and training is deterministic, i.e., repeating training and then the prediction always yields the same predicted value. Otherwise, proof by construction would require verifying all random seeds for each subset of \mathcal{D} .

B.2 Proofs for Chapter 5

This section contains the proofs for the theoretical contributions that are either in or relevant to Chapter 5.

Theorem 5.3

Proof. Let

$$\Delta := \dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x}) \leq \forall_{y' \in \mathcal{Y} \setminus \{y_{\text{pl}}, y_{\text{ru}}\}} \dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y'}(\mathbf{x}). \quad (\text{B.19})$$

In words, vote-count difference Δ between plurality label y_{pl} and runner-up label y_{ru} is at least as small as the gap between y_{pl} and any other label.

In the worst case, a single feature perturbation changes a single submodel’s vote from plurality label y_{pl} to a label of the adversary’s choosing. Each perturbed submodel prediction reduces the gap between the plurality label and the adversary’s chosen label by two. By Eq. (B.19), it takes the fewest number of vote changes for y_{ru} to overtake plurality label y_{pl} with the proof following by induction. Δ then lower bounds the certified robustness. When determining R , Δ may be even or odd. We separately consider both cases below.

Case #1: Δ is odd.

Since Δ is odd, there can never be a tie between labels y_{pl} and y_{ru} , simplifying the analysis. Then, the maximum number of submodel predictions that can change without changing the plurality label is any $R \in \mathbb{N}$ satisfying

$$\begin{aligned}
\dot{c}_{y_{\text{ru}}}(\mathbf{x}) + 2R &< \dot{c}_{y_{\text{pl}}}(\mathbf{x}) \\
R &< \frac{\dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x})}{2} \\
R &= \left\lfloor \frac{\dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x})}{2} \right\rfloor &> R \text{ must be a whole number} \\
&= \left\lfloor \frac{\dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x}) - \mathbb{1}[y_{\text{ru}} < y_{\text{pl}}]}{2} \right\rfloor &> \text{Subtract 1 no effect for odd } \Delta \\
&= \left\lfloor \frac{\text{GAP}_{\text{vote}}(y_{\text{pl}}, y_{\text{ru}}; \mathbf{x})}{2} \right\rfloor &> \text{Eq. (5.1)}.
\end{aligned}$$

Case #2: Δ is even.

For even-valued Δ , ties can occur. If $y_{\text{ru}} < y_{\text{pl}}$, the tie between y_{pl} and y_{ru} is broken in favor of y_{ru} . Then, the number of submodel predictions that can change without changing the plurality label is any $R \in \mathbb{N}$ satisfying

$$\begin{aligned}
\dot{c}_{y_{\text{ru}}}(\mathbf{x}) + \mathbb{1}[y_{\text{ru}} < y_{\text{pl}}] + 2R &< \dot{c}_{y_{\text{pl}}}(\mathbf{x}) \\
R &\leq \frac{\dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x}) - \mathbb{1}[y_{\text{ru}} < y_{\text{pl}}]}{2} \\
R &= \left\lfloor \frac{\dot{c}_{y_{\text{pl}}}(\mathbf{x}) - \dot{c}_{y_{\text{ru}}}(\mathbf{x}) - \mathbb{1}[y_{\text{ru}} < y_{\text{pl}}]}{2} \right\rfloor &> R \text{ is a whole number} \\
&= \left\lfloor \frac{\text{GAP}_{\text{vote}}(y_{\text{pl}}, y_{\text{ru}}; \mathbf{x})}{2} \right\rfloor &> \text{Eq. (5.1)}.
\end{aligned}$$

□

Theorem 5.3's definition of R follows the same basic structure as that of *deep partition aggregation* [LF21, Eq. (10)].

Claims Related to Theorem 5.4

Lemma B.2. *Let f_1, \dots, f_L be a set of L models where $\forall_{l \in [L]} f_l : \mathcal{X} \rightarrow \mathcal{Y}$. Under submodel voting, label $y \in \mathcal{Y}$ is preferred over label $y' \in \mathcal{Y} \setminus y$ w.r.t. instance $\mathbf{x} \in \mathcal{X}$ if and only if $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) \geq 0$.*

Proof. Label y is preferred over label y' in only two cases:

1. y receives more (sub)model votes than y' , i.e., $\dot{c}_y(\mathbf{x}) > \dot{c}_{y'}(\mathbf{x})$.
2. y and y' receive the same number of votes and $y < y'$.

In the first case,

$$\begin{aligned} \text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) &:= \dot{c}_y(\mathbf{x}) - \dot{c}_{y'}(\mathbf{x}) - \mathbb{1}[y' < y] \\ &\geq 1 - \mathbb{1}[y' < y] \\ &\geq 1 - 1 = 0. \end{aligned}$$

In the second case,

$$\begin{aligned} \text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) &:= \dot{c}_y(\mathbf{x}) - \dot{c}_{y'}(\mathbf{x}) - \mathbb{1}[y' < y] \\ &= 0 - \mathbb{1}[y' < y] \\ &= 0 - 0 = 0. \end{aligned}$$

The reverse direction where $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) \geq 0 \implies y$ is preferred over y' can be proven by contradiction using similar logic as above. If y' receives more votes than y , then $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) < 0$, a contradiction. Similarly, if $\dot{c}_y(\mathbf{x}) = \dot{c}_{y'}(\mathbf{x})$ then necessarily $y' < y$. This also leads to a contradiction as $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x})$ would be negative. \square

Lemma B.3. Runoff Elections Case #1 Certified Feature Robustness *Given submodel feature partition $\mathcal{S}_1, \dots, \mathcal{S}_L$, let f be a voting-based ensemble of L submodels, where the l -th submodel uses only the features in \mathcal{S}_l . For instance $\mathbf{x} \in \mathcal{X}$, let y_{RO} be the label selected by the run-off decision function. The certified feature robustness of y_{RO} getting overtaken in round #2 of the run-off election is*

$$R_{\text{RO}}^{\text{Case1}} := \min_{y \in \mathcal{Y} \setminus y_{\text{RO}}} \max \left\{ \left\lfloor \frac{\text{GAP}_{\text{vote}}(\tilde{y}_{\text{RO}}, y)}{2} \right\rfloor, \left\lfloor \frac{\text{GAP}_{\text{logit}}(y_{\text{RO}}, y)}{2} \right\rfloor \right\}$$

Proof. For a label $y \in \mathcal{Y} \setminus y_{\text{RO}}$ to overtake y_{RO} , two requirements must be simultaneously met:

- y and y_{RO} must be round #1’s top-two labels, and
- y must be preferred over y_{RO} in round #2.

Let $\tilde{y}_{\text{RO}} \in \mathcal{Y} \setminus y_{\text{pl}}$ denote the other top-two label in round #1. Note that \tilde{y}_{RO} may or may not be the same as y . The robustness of \tilde{y}_{RO} to being overtaken by y in round #1 follows directly from Theorem 5.3 and equals

$$R' = \left\lfloor \frac{\text{GAP}_{\text{vote}}(\tilde{y}_{\text{RO}}, y; \mathbf{x})}{2} \right\rfloor. \quad (\text{B.20})$$

Concerning the second requirement, y_{RO} is preferred over y in round #2 so long as $\text{GAP}_{\text{logit}}(y_{\text{RO}}, y; \mathbf{x}) \geq 0$. Following similar logic as above, y_{RO} ’s certified feature robustness in round #2 is

$$R'' = \left\lfloor \frac{\text{GAP}_{\text{logit}}(y_{\text{RO}}, y; \mathbf{x})}{2} \right\rfloor. \quad (\text{B.21})$$

Since both requirements must hold, the certified feature robustness is lower bounded by both (i.e., the maximum) of Eqs. (B.20) and (B.21). Moreover, the optimal label $y \in \mathcal{Y} \setminus y_{\text{RO}}$ is not determined a priori meaning all labels need to be checked. \square

Lemma B.4. Runoff Elections Case #2 Certified Feature Robustness *Given submodel feature partition $\mathcal{S}_1, \dots, \mathcal{S}_L$, let f be a voting-based ensemble of L submodels, where the l -th submodel uses only the features in \mathcal{S}_l . For instance $\mathbf{x} \in \mathcal{X}$, let y_{RO} be the label selected by the run-off decision function. Define recursive function dp as*

$$\text{dp}[i, j] = \begin{cases} 0 & \min\{i, j\} \leq 1 \text{ and } (i, j) \neq (1, 1) \\ 1 + \min\{\text{dp}[i - 2, j - 1], \text{dp}[i - 1, j - 2]\} & \text{Otherwise} \end{cases} \quad (\text{B.22})$$

Then y_{RO} 's certified feature robustness of remaining in the top-two round #1 labels predicted by the submodels is

$$R_{\text{RO}}^{\text{Case2}} := \min_{y, y' \in \mathcal{Y} \setminus y_{\text{RO}}} \text{dp}[\text{gap}_y, \text{gap}_{y'}]$$

where $\text{gap}_{y^*} = \max\{0, \text{GAP}_{\text{vote}}(y_{\text{RO}}, y^*)\}$.

Proof. Lemma B.2 proves that a label y is preferred over another label y' iff $\text{GAP}_{\text{vote}}(y, y'; \mathbf{x}) \geq 0$. For label y_{RO} to be in round #1's top two, no pair of labels can have negative submodel vote gaps w.r.t. y_{RO} . Determining y_{RO} 's round #1 certified feature robustness reduces to determining the maximum number of submodel votes that can be perturbed with it remaining guaranteed that both labels do not have negative submodel vote gaps.

In the best case for an attacker, perturbing a single submodel changes the submodel's predicted label from y_{RO} to a label of the attacker's choosing, e.g., $y \neq y_{\text{RO}}$; this perturbation decreases $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y; \mathbf{x})$ by 2. For all other $y' \in \mathcal{Y} \setminus \{y_{\text{RO}}, y\}$, this perturbation also decreases $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y'; \mathbf{x})$ by 1.

By definition, y_{RO} is in the top-two round #1 labels, meaning $R_{\text{RO}}^{\text{Case2}} \geq 0$. Consider first when $\max\{\text{GAP}_{\text{vote}}(y_{\text{RO}}, y), \text{GAP}_{\text{vote}}(y_{\text{RO}}, y')\} \leq 1$ and $(i, j) \neq (1, 1)$. The attacker perturbs whichever label y, y' has the larger submodel vote gap. Since at most one of these two labels has a positive gap, an additional submodel perturbation could make *both* $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y)$ and $\text{GAP}_{\text{vote}}(y_{\text{RO}}, y')$ negative meaning no further feature perturbations are possible. In the special case of $i = j = 1$, perturbing a submodel predicting either label y or y' never causes the other label's submodel vote gap to be negative meaning one additional submodel feature perturbation is possible. When $\max\{\text{GAP}_{\text{vote}}(y_{\text{RO}}, y), \text{GAP}_{\text{vote}}(y_{\text{RO}}, y')\} > 1$, the proof follows by induction where recursive function dp returns the fewest number of submodel perturbations required given $y, y' \in \mathcal{Y}$.

Since the attacker’s optimal pair of labels y, y' is not determined a priori, Eq. (5.7)’s feature guarantee considers all pairs of labels and returns the robustness of the pair most advantageous to the attacker. \square

Theorem 5.4

Proof. For a given $\mathbf{x} \in \mathcal{X}$, there are only two possible ways that run-off prediction $y_{\text{RO}} \in \mathcal{Y}$ can be perturbed, namely:

1. y_{RO} loses in run-off’s second round.
2. y_{RO} fails to qualify for the second round by not being in the top two labels in round #1.

These two cases align directly with Lemmas B.3 and B.4, respectively. An optimal attacker targets whichever of the two cases requires fewer feature perturbations. Therefore, run-off’s certified feature robustness is the minimum of Eqs. (5.5) and (5.7). \square

B.3 Proof for Chapter 6

This section contains the proofs for the theorem in Chapter 6.

Theorem. *Let loss function $\tilde{\mathcal{L}} : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ be twice-differentiable and strictly convex as well as either even² or monotonically decreasing. Then, it holds that*

$$\tilde{\mathcal{L}}(a) < \tilde{\mathcal{L}}(a') \implies \left\| \nabla_a \tilde{\mathcal{L}}(a) \right\|_2 < \left\| \nabla_a \tilde{\mathcal{L}}(a') \right\|_2. \tag{B.23}$$

Proof.

Theorem 6.1 specifies that property,

$$\tilde{\mathcal{L}}(a) < \tilde{\mathcal{L}}(a') \implies \left\| \nabla_a \tilde{\mathcal{L}}(a) \right\|_2 < \left\| \nabla_a \tilde{\mathcal{L}}(a') \right\|_2,$$

²“Even” denotes that the function satisfies $\forall_a \tilde{\mathcal{L}}(a) = \tilde{\mathcal{L}}(-a)$.

holds when loss function $\widetilde{\mathcal{L}}$ is strictly convex (i.e., $\forall_{a \in \mathbb{R}} \nabla_a^2 \widetilde{\mathcal{L}}(a) > 0$) and either monotonically decreasing or even. We prove the claim separately for these two disjoint cases.

Case #1: Monotonically Decreasing For any monotonically decreasing $\widetilde{\mathcal{L}}$, by definition

$$\widetilde{\mathcal{L}}(a) < \widetilde{\mathcal{L}}(a') \implies a > a'.$$

Then, given $\forall_{a \in \mathbb{R}} \nabla_a^2 \widetilde{\mathcal{L}}(a) > 0$, it holds that

$$\nabla_a \widetilde{\mathcal{L}}(a) > \nabla_a \widetilde{\mathcal{L}}(a'). \quad (\text{B.24})$$

For any scalar, monotonically decreasing function $\widetilde{\mathcal{L}}$, it holds that $\nabla_a \widetilde{\mathcal{L}}(a') \leq 0$ meaning Eq. (B.24)'s inequality flips w.r.t. L_2 norms, i.e.,

$$\left\| \nabla_a \widetilde{\mathcal{L}}(a) \right\|_2 < \left\| \nabla_a \widetilde{\mathcal{L}}(a') \right\|_2, \quad (\text{B.25})$$

as for any $x, x' \in \mathbb{R}_{\leq 0}$ it holds that $x > x' \implies \|x\|_2 < \|x'\|_2$.

Case #2: Even Formally, a function $\widetilde{\mathcal{L}}$ is even if

$$\forall_a \widetilde{\mathcal{L}}(a) = \widetilde{\mathcal{L}}(-a). \quad (\text{B.26})$$

For even $\widetilde{\mathcal{L}}$, it holds that $\nabla_a \widetilde{\mathcal{L}}(0) = 0$ provided twice differentiability. Given $\forall_{a \in \mathbb{R}} \nabla_a^2 \widetilde{\mathcal{L}}(a) > 0$, then $\forall_{a < 0} \nabla_a \widetilde{\mathcal{L}}(a) < 0$. Hence over restricted domain $\mathbb{R}_{\leq 0}$, $\widetilde{\mathcal{L}}$ is monotonically decreasing. Above it was shown that Eq. (6.8) holds for monotonically decreasing functions so

$$\widetilde{\mathcal{L}}(-|a|) < \widetilde{\mathcal{L}}(-|a'|) \implies \left\| \nabla_a \widetilde{\mathcal{L}}(-|a|) \right\|_2 < \left\| \nabla_a \widetilde{\mathcal{L}}(-|a'|) \right\|_2. \quad (\text{B.27})$$

Evenness induces function symmetry about the origin so

$$\forall_a \left| \nabla_a \widetilde{\mathcal{L}}(a) \right| = \left| \nabla_a \widetilde{\mathcal{L}}(-a) \right|, \quad (\text{B.28})$$

and by extension

$$\forall_a \left\| \nabla_a \widetilde{\mathcal{L}}(a) \right\|_2 = \left\| \nabla_a \widetilde{\mathcal{L}}(-a) \right\|_2. \quad (\text{B.29})$$

Eqs. (B.26) and (B.29) allow Eq. (B.27)'s absolute values and negations to be dropped completing the proof. \square

APPENDIX C

DETAILED EMPIRICAL RESULTS

This chapter contains previously published, coauthored material [HL21; HL22a; HL23c; HL23a]. Hammoudeh wrote this complete section, coded all experiments, designed the experiments, and analyzed all the results. Lowd provided supervision, editorial suggestions, and input on experiment design.

Zayd Hammoudeh and Daniel Lowd. “Simple, Attack-Agnostic Defense Against Targeted Training Set Attacks Using Cosine Similarity”. In: *Proceedings of the 3rd ICML Workshop on Uncertainty and Robustness in Deep Learning*. UDL’21. 2021

Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security*. CCS’22. Los Angeles, CA: Association for Computing Machinery, 2022. URL: <https://arxiv.org/abs/2201.10055>

Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning*. SaTML’23. 2023. URL: <https://arxiv.org/abs/2208.13904>

Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023. URL: <https://arxiv.org/abs/2302.11628>

This chapter provides detailed empirical results that were excluded from the main body of the dissertation to improve readability and brevity. We break these detailed results by the corresponding chapter in the main paper.

C.1 Chapter 4 Detailed Results

C.1.1 Baseline Accuracy. Table C.12 shows the baseline accuracy when a model is trained on all of training set \mathcal{D} (i.e., $q = 1$). For each dataset, the model architecture (either ridge regression or XGBoost) aligns with those used for Sec. 4.8’s ensembles. See Table 1.

Table C.12. **Baseline Accuracy:** Summary of the baseline (i.e., uncertified) accuracy mean and standard deviation for Sec. 4.8’s six datasets. Submodels were trained on all of training set \mathcal{D} (i.e., $q = 1$). Beside each dataset’s name is the submodel architecture used by the ensemble. Threshold ξ matches values in Table 1.

Dataset	Submodel	Base Acc. (%)
Ames	XGBoost	90.4 ± 2.4
Austin	XGBoost	71.3 ± 4.1
Diamonds	Ridge	73.6 ± 4.0
Weather	Ridge	85.9 ± 3.4
Life	XGBoost	92.7 ± 3.1
Spambase	Ridge	87.5 ± 2.9

C.1.2 Numerical Results. Fig. 7 visualizes our certified regressors’ certified robustness on six datasets – five regression and one binary classification. This section provides the certified accuracy in numerical form, including the associated variance.

Table C.13. **Ames Housing Full Results:** Certified accuracy mean and standard deviation for the Ames Housing [Coc11] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	90.4 ± 2.4	90.4 ± 2.4	90.4 ± 2.4	90.4 ± 2.4	54.3 ± 3.8
	1	82.3 ± 3.1	86.8 ± 2.8	82.3 ± 3.1	85.2 ± 2.9	54.3 ± 3.8
	4	76.3 ± 3.7	81.2 ± 2.9	76.3 ± 3.7	80.0 ± 2.7	54.0 ± 3.6
25	8	57.4 ± 3.8	69.6 ± 2.7	57.5 ± 3.7	70.1 ± 3.2	53.1 ± 3.5
	12	11.0 ± 3.7	28.2 ± 3.8	23.7 ± 5.1	48.8 ± 4.4	52.2 ± 3.9
	16	0.0 ± 0.0	0.0 ± 0.0	11.3 ± 3.8	15.3 ± 3.2	51.5 ± 3.8
125	1	70.4 ± 3.7	73.9 ± 1.7	70.4 ± 3.7	73.1 ± 1.6	54.3 ± 3.8
	10	62.8 ± 3.4	66.5 ± 2.0	62.8 ± 3.4	65.9 ± 1.9	53.1 ± 3.5
	20	44.9 ± 4.2	52.2 ± 2.9	44.9 ± 4.2	52.2 ± 3.0	51.2 ± 3.6
	30	21.8 ± 4.2	28.7 ± 3.8	21.8 ± 4.2	31.1 ± 3.3	49.1 ± 3.7
	40	2.5 ± 1.3	3.9 ± 1.7	2.5 ± 1.3	5.9 ± 2.3	48.5 ± 4.0
251	1	63.1 ± 3.4	66.3 ± 3.8	63.1 ± 3.4	66.0 ± 3.7	54.3 ± 3.8
	20	51.8 ± 3.2	56.4 ± 2.9	51.8 ± 3.2	57.9 ± 2.9	51.2 ± 3.6
	40	37.1 ± 3.2	42.5 ± 3.8	37.1 ± 3.2	45.3 ± 2.8	48.5 ± 4.0
	60	15.3 ± 3.8	22.5 ± 3.4	15.3 ± 3.8	32.8 ± 3.7	44.1 ± 4.3
	80	0.2 ± 0.4	0.6 ± 0.5	0.2 ± 0.4	10.9 ± 2.5	40.1 ± 3.9

Table C.14. **Austin Housing Full Results:** Certified accuracy mean and standard deviation for the Austin Housing [Pie21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	71.3 \pm 4.1	71.3 \pm 4.1	71.3 \pm 4.1	71.3 \pm 4.1	35.3 \pm 4.8
51	1	59.9 \pm 4.5	63.7 \pm 4.6	59.9 \pm 4.5	61.7 \pm 4.6	35.3 \pm 4.8
	5	49.6 \pm 5.1	52.9 \pm 3.4	49.6 \pm 5.1	50.8 \pm 4.2	35.2 \pm 4.8
	10	29.9 \pm 3.7	35.3 \pm 2.8	29.9 \pm 3.7	31.8 \pm 3.2	34.9 \pm 4.9
	15	9.2 \pm 2.0	12.6 \pm 2.8	9.2 \pm 2.0	10.1 \pm 2.7	34.7 \pm 4.9
	20	0.5 \pm 0.5	0.3 \pm 0.7	0.5 \pm 0.5	0.0 \pm 0.0	34.6 \pm 4.6
301	1	51.0 \pm 3.9	52.0 \pm 4.1	51.0 \pm 3.9	51.1 \pm 4.1	35.3 \pm 4.8
	20	41.4 \pm 3.6	43.3 \pm 5.9	41.4 \pm 3.6	43.3 \pm 5.9	34.6 \pm 4.6
	40	29.7 \pm 4.1	32.2 \pm 5.2	29.7 \pm 4.1	33.7 \pm 5.7	34.3 \pm 4.7
	60	15.4 \pm 3.0	19.1 \pm 4.1	15.4 \pm 3.0	22.7 \pm 4.9	34.0 \pm 4.5
	80	3.2 \pm 1.8	3.1 \pm 1.2	3.2 \pm 1.8	7.7 \pm 3.4	32.9 \pm 4.5
701	1	43.9 \pm 5.0	42.7 \pm 5.5	43.9 \pm 5.0	43.6 \pm 5.7	35.3 \pm 4.8
	40	34.5 \pm 6.0	35.0 \pm 6.2	34.5 \pm 6.0	36.9 \pm 5.9	34.3 \pm 4.7
	80	25.3 \pm 4.8	24.7 \pm 6.1	25.3 \pm 4.8	27.0 \pm 6.1	32.9 \pm 4.5
	120	13.1 \pm 2.6	14.6 \pm 5.0	13.1 \pm 2.6	18.9 \pm 4.4	31.6 \pm 4.9
	160	2.7 \pm 0.9	4.8 \pm 3.3	2.7 \pm 0.9	9.1 \pm 2.9	30.0 \pm 4.7

Table C.15. **Diamonds Full Results:** Certified accuracy mean and standard deviation for the Diamonds [Wic16] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	73.6 \pm 4.0	73.6 \pm 4.0	73.6 \pm 4.0	73.6 \pm 4.0	15.7 \pm 3.5
151	1	74.6 \pm 3.8	74.8 \pm 4.5	74.6 \pm 3.8	74.7 \pm 4.4	15.7 \pm 3.5
	35	64.4 \pm 4.6	67.1 \pm 4.8	67.2 \pm 4.6	69.7 \pm 4.1	15.6 \pm 3.4
	70	38.6 \pm 5.7	42.2 \pm 4.0	62.4 \pm 4.3	64.7 \pm 5.2	15.5 \pm 3.4
	105	0.0 \pm 0.0	0.0 \pm 0.0	54.5 \pm 5.9	57.4 \pm 5.2	15.2 \pm 3.3
	140	0.0 \pm 0.0	0.0 \pm 0.0	35.4 \pm 5.8	34.3 \pm 7.1	14.8 \pm 3.4
501	1	77.3 \pm 4.2	75.8 \pm 3.9	77.3 \pm 4.2	75.7 \pm 4.1	15.7 \pm 3.5
	75	66.2 \pm 4.0	65.0 \pm 4.4	66.3 \pm 4.0	68.7 \pm 4.4	15.5 \pm 3.4
	150	50.7 \pm 4.9	48.2 \pm 4.5	57.8 \pm 4.7	59.6 \pm 4.9	14.8 \pm 3.4
	300	0.0 \pm 0.0	0.0 \pm 0.0	38.0 \pm 6.1	36.2 \pm 3.2	12.3 \pm 3.4
	450	0.0 \pm 0.0	0.0 \pm 0.0	8.8 \pm 3.3	9.0 \pm 2.1	10.7 \pm 2.9
1001	1	75.2 \pm 4.1	74.9 \pm 5.5	75.2 \pm 4.1	74.9 \pm 5.5	15.7 \pm 3.5
	150	56.0 \pm 4.9	56.3 \pm 5.8	56.0 \pm 4.9	62.8 \pm 6.1	14.8 \pm 3.4
	300	24.7 \pm 4.4	25.3 \pm 4.8	29.5 \pm 4.1	42.3 \pm 6.4	12.3 \pm 3.4
	450	0.0 \pm 0.0	0.0 \pm 0.0	16.9 \pm 4.0	17.9 \pm 5.1	10.7 \pm 2.9
	600	0.0 \pm 0.0	0.0 \pm 0.0	4.2 \pm 1.5	3.3 \pm 3.1	9.6 \pm 3.1

Table C.16. **Weather Full Results:** Certified accuracy mean and standard deviation for the Weather [Mal+21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	85.9 ± 3.4	85.9 ± 3.4	85.9 ± 3.4	85.9 ± 3.4	23.8 ± 4.5
51	1	86.0 ± 3.1	86.5 ± 3.8	86.0 ± 3.1	86.5 ± 3.8	23.8 ± 4.5
	10	83.9 ± 3.5	84.6 ± 3.8	83.9 ± 3.5	85.0 ± 3.8	23.8 ± 4.5
	20	82.0 ± 3.5	82.3 ± 4.2	83.1 ± 3.4	84.0 ± 3.8	23.8 ± 4.5
	35	0.0 ± 0.0	0.0 ± 0.0	81.8 ± 3.7	82.0 ± 4.4	23.8 ± 4.5
	50	0.0 ± 0.0	0.0 ± 0.0	76.8 ± 5.1	75.8 ± 4.9	23.8 ± 4.5
1501	1	85.2 ± 3.9	85.2 ± 4.2	85.2 ± 3.9	85.2 ± 4.2	23.8 ± 4.5
	300	75.8 ± 4.3	77.6 ± 4.2	76.8 ± 4.2	79.4 ± 4.2	23.4 ± 4.6
	600	54.3 ± 4.7	55.1 ± 5.4	71.4 ± 3.7	72.2 ± 5.1	22.9 ± 4.3
	1000	0.0 ± 0.0	0.0 ± 0.0	56.5 ± 5.1	57.4 ± 4.6	22.0 ± 4.6
	1400	0.0 ± 0.0	0.0 ± 0.0	22.9 ± 2.6	22.5 ± 3.2	21.8 ± 4.8
3001	1	86.7 ± 2.7	84.6 ± 2.9	86.7 ± 2.7	84.6 ± 2.9	23.8 ± 4.5
	600	67.7 ± 2.7	66.9 ± 4.0	68.1 ± 2.9	71.5 ± 4.0	22.9 ± 4.3
	1200	25.7 ± 5.8	25.8 ± 4.9	55.0 ± 4.2	56.2 ± 3.8	21.9 ± 4.7
	1800	0.0 ± 0.0	0.0 ± 0.0	35.8 ± 4.8	34.7 ± 4.0	21.5 ± 4.7
	2400	0.0 ± 0.0	0.0 ± 0.0	9.3 ± 3.0	9.9 ± 2.5	20.5 ± 4.9

Table C.17. **Life Full Results:** Certified accuracy mean and standard deviation for the Life [Raj21] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	92.7 ± 3.1	92.7 ± 3.1	92.7 ± 3.1	92.7 ± 3.1	34.6 ± 3.1
25	1	77.7 ± 4.4	80.2 ± 4.0	77.7 ± 4.4	78.3 ± 5.2	34.6 ± 3.1
	5	69.3 ± 4.9	71.5 ± 4.7	69.3 ± 4.9	71.4 ± 4.7	33.8 ± 2.8
	10	43.3 ± 5.6	54.2 ± 6.0	47.3 ± 5.9	60.8 ± 4.9	32.9 ± 2.8
	15	0.0 ± 0.0	0.0 ± 0.0	23.8 ± 4.0	33.1 ± 3.2	32.1 ± 2.9
	20	0.0 ± 0.0	0.0 ± 0.0	9.5 ± 2.9	11.4 ± 3.2	31.1 ± 2.3
101	1	71.1 ± 3.8	71.5 ± 4.3	71.1 ± 3.8	70.8 ± 4.1	34.6 ± 3.1
	10	58.9 ± 4.3	61.8 ± 5.1	58.9 ± 4.3	61.8 ± 5.1	32.9 ± 2.8
	20	40.5 ± 5.8	43.9 ± 4.3	40.5 ± 5.8	45.4 ± 4.7	31.1 ± 2.3
	30	20.7 ± 3.8	22.8 ± 4.4	20.7 ± 3.8	26.4 ± 3.9	28.5 ± 2.5
	40	4.4 ± 2.4	3.5 ± 1.4	4.6 ± 2.5	10.1 ± 2.7	26.9 ± 2.4
201	1	62.9 ± 4.1	66.3 ± 3.0	62.9 ± 4.1	65.7 ± 2.4	34.6 ± 3.1
	30	46.6 ± 3.8	49.0 ± 2.8	46.6 ± 3.8	52.9 ± 3.0	28.5 ± 2.5
	60	23.3 ± 2.7	24.6 ± 4.1	24.4 ± 2.6	34.4 ± 3.9	23.4 ± 2.3
	90	0.1 ± 0.3	0.6 ± 0.5	12.8 ± 2.9	18.0 ± 3.6	18.1 ± 2.1
	120	0.0 ± 0.0	0.0 ± 0.0	4.1 ± 1.6	4.4 ± 2.2	8.5 ± 1.4

Table C.18. **Spambase Full Results:** Certified accuracy mean and standard deviation for the Spambase [Hop+17] dataset. Each ensemble submodel was trained on $\frac{1}{q}$ -th of the training set with three q values tested per dataset, while k NN-CR was always trained on the whole training set (i.e., $q = 1$). The certified accuracy results of five robustness values (R) are reported per q value. Also reported as a baseline is the uncertified accuracy ($R = 0$) when training a single model on all of training set \mathcal{D} ($q = 1$). Results are averaged across 10 trials per method, with each R 's best mean certified accuracy in **bold**.

q	R	PCR	OCR	W-PCR	W-OCR	k NN-CR
1	0	87.5 \pm 2.9	87.5 \pm 2.9	87.5 \pm 2.9	87.5 \pm 2.9	64.0 \pm 4.3
25	1	87.6 \pm 3.5	87.1 \pm 3.8	87.6 \pm 3.5	85.8 \pm 3.6	64.0 \pm 4.3
	5	80.3 \pm 3.8	81.0 \pm 3.4	81.1 \pm 3.6	83.5 \pm 3.7	63.6 \pm 4.1
	10	57.3 \pm 4.5	65.0 \pm 3.1	73.2 \pm 3.8	76.4 \pm 3.8	63.4 \pm 4.3
	15	0.0 \pm 0.0	0.0 \pm 0.0	61.5 \pm 4.8	63.1 \pm 2.4	63.2 \pm 4.4
	20	0.0 \pm 0.0	0.0 \pm 0.0	42.5 \pm 4.4	38.7 \pm 4.2	63.0 \pm 4.4
151	1	87.4 \pm 2.9	87.2 \pm 2.2	87.4 \pm 2.9	86.7 \pm 2.6	64.0 \pm 4.3
	25	69.1 \pm 4.3	70.2 \pm 5.5	69.1 \pm 4.3	75.7 \pm 4.9	63.0 \pm 4.4
	50	22.8 \pm 5.8	24.9 \pm 4.0	35.4 \pm 6.3	52.8 \pm 4.0	62.0 \pm 4.8
	75	0.0 \pm 0.0	0.0 \pm 0.0	14.8 \pm 3.0	23.0 \pm 3.9	61.8 \pm 4.7
	100	0.0 \pm 0.0	0.0 \pm 0.0	3.4 \pm 2.2	5.2 \pm 2.4	61.3 \pm 4.2
301	1	83.1 \pm 2.8	86.2 \pm 3.3	83.1 \pm 2.8	86.0 \pm 3.2	64.0 \pm 4.3
	45	65.1 \pm 4.7	68.6 \pm 3.9	65.1 \pm 4.7	72.1 \pm 3.9	62.3 \pm 4.3
	90	30.4 \pm 3.5	34.6 \pm 4.9	33.6 \pm 3.2	53.7 \pm 2.7	61.7 \pm 4.6
	135	0.5 \pm 0.7	0.1 \pm 0.3	23.7 \pm 3.5	31.1 \pm 4.6	60.2 \pm 4.2
	180	0.0 \pm 0.0	0.0 \pm 0.0	7.2 \pm 2.5	11.3 \pm 2.5	58.3 \pm 4.3

C.1.3 k NN-CR Full Certified Accuracy Plots. To improve readability, Fig. 7 does not show k NN-CR’s full certified accuracy trend. Instead, Fig. C.19 below plots k NN-CR’s full mean certified accuracy against that of W-OCR (using each dataset’s maximum q value) for each of Sec. 4.8’s six datasets. Fig. C.19 also visualizes the variance of each method by showing one standard deviation of the certified accuracy as a shaded region around the mean line. In summary, while W-OCR certifies more instances (i.e., has larger peak certified accuracy), its maximum certified robustness R is (significantly) smaller than that of k NN-CR.

Table C.19. **W-OCR q Values:** As detailed in Sec. 4.8.1, ensemble submodels were trained on $\frac{1}{q}$ -th of the training data where q varies by dataset. Below are the W-OCR q values used in Fig. C.19.

Dataset	Ames	Austin	Diamonds	Weather	Life	Spambase
q	251	701	1,001	3,001	201	301

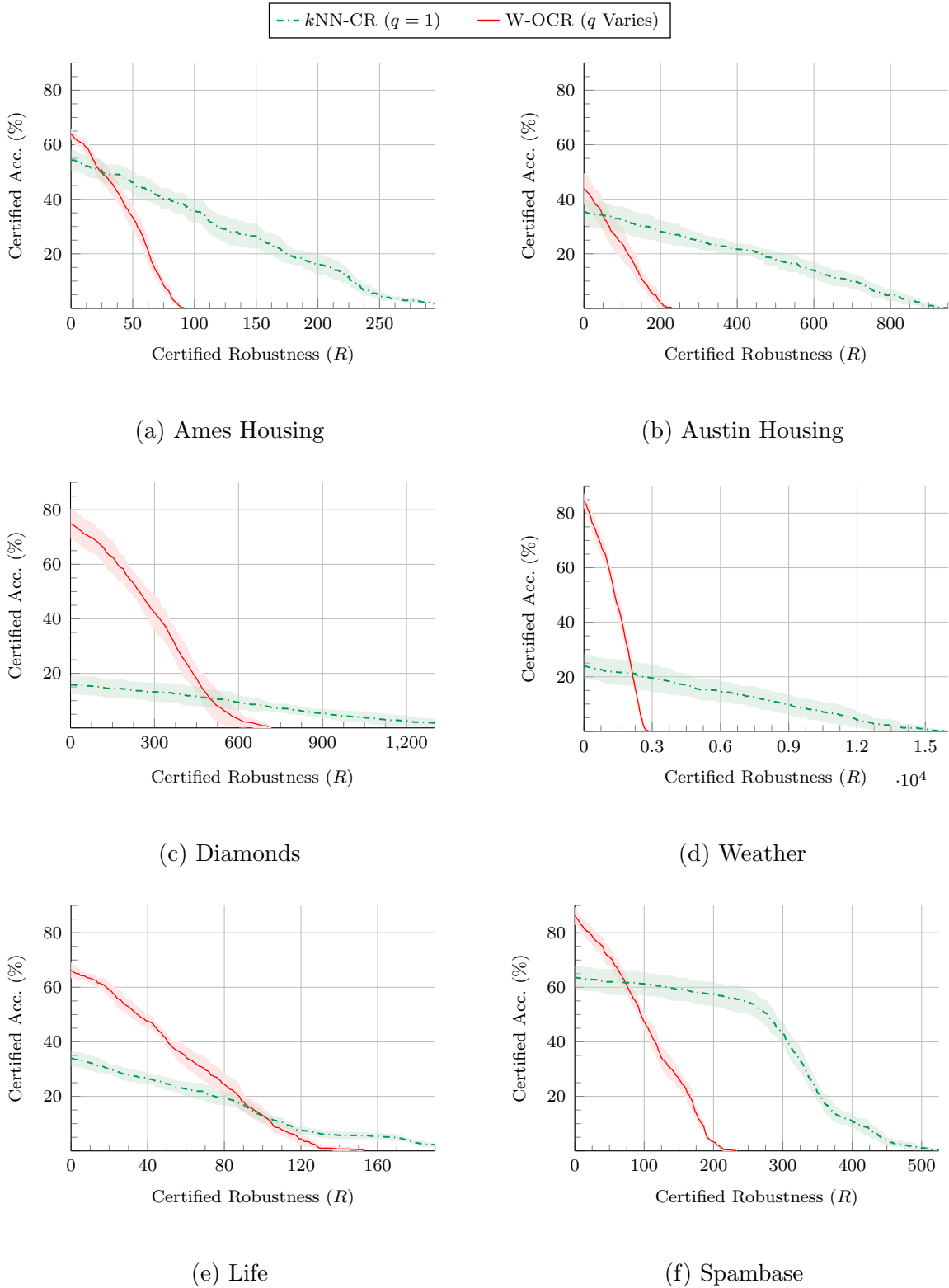


Figure C.19. $k\text{NN-CR}$ vs. W-OCR Certified Accuracy: Full plots of the mean certified accuracy for Sec. 4.8’s six datasets. The shaded regions visualize one standard deviation of the certified accuracy for each R value. W-OCR ’s q value for each dataset is in Table C.19.

C.2 Chapter 5 Detailed Results

Limited space prevents us from including all experimental results in the main paper. We provide additional results below.

C.2.1 Non-Robust Accuracy. Table C.20 provides the non-robust (i.e., uncertified) accuracy when training a single model ($L = 1$) on each of Sec. 5.5’s four datasets. The non-robust accuracy provides an upper-bound reference for the maximum achievable accuracy given the training set and the model architectures we used.

For regression, the “non-robust accuracy” denotes the single model’s prediction satisfies the error bounds, i.e., $\xi_l \leq f(\mathbf{x}) \leq \xi_u$. Given arbitrary instance (\mathbf{x}, y) , we follow Hammoudeh and Lowd [HL23c] and use for Weather $\xi_l = y - 3^\circ\text{C}$ and $\xi_u = y + 3^\circ\text{C}$ as well as for Ames $\xi_l = y - 15\%y$ and $\xi_u = y + 15\%y$.

Table C.20. **Non-Robust Accuracy:** Prediction accuracy when training a single model on all model features, i.e., $L = 1$. These values represent an upper bound on the potential accuracy of our method given the training set, model architecture, and hyperparameters.

Dataset	Accuracy
CIFAR10	95.40%
MNIST	99.57%
Weather	92.61%
Ames	88.05%

C.2.2 Detailed Median Certified Robustness Results. In Section 5.5.2 of the main paper, Tables 2 and 3 summarize the median certified robustness and classification accuracies of feature partition aggregation (FPA) and baseline randomized ablation [LF20b; Jia+22b]. In the tables, “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA, and “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved RA; “Plural” denotes FPA using plurality voting as the decision function (Sec. 5.3.1) while “Run-Off” denotes FPA with Sec. 5.3.2’s run-off elections.

Recall that FPA’s primary hyperparameter is L – the number of ensemble submodels. RA’s primary hyperparameter is e – the number of kept (unchanged) pixels in each ablated input. L and e control the corresponding method’s accuracy-robustness trade-off where smaller L and larger e entail better accuracy. As a rule of thumb, the fairest comparison across methods sets $L \approx \frac{d}{e}$, since this relationship entails that each FPA and RA prediction uses approximately the same number of features from instance \mathbf{x} .

This section explores the relationship between each method’s hyperparameter settings and the corresponding median robustness and classification accuracy. Each dataset’s results are split into separate tables similar to Levine and Feizi’s [LF20b, Tables 1 and 2] presentation in the original RA paper.

For CIFAR10 and MNIST, FPA uses deterministic partitioning. Specifically, we use a striding strategy as Section 5.4.1 details. Depending on the image dimensions, some stride lengths are substantially worse than others, leading to non-monotonic changes in median robustness as a function of L . Tables C.21 and C.22 do not report the particularly poor choices of L that severely degrade median robustness, e.g., when L is evenly divisible by the image width.

Below, any misclassified prediction is assigned robustness of $-\infty$, meaning the median certified robustness can in some cases be negative.

Table C.21. **CIFAR10 Detailed Results:** Classification accuracy (%) and median certified robustness (larger is better) for the CIFAR10 [KNH14] dataset ($d = 1024$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. Each certification method’s hyperparameter setting with the best median robustness is shown in **bold**. The best overall median robustness is shown in **blue**.

(a) Feature Partition Aggregation (Ours)					(b) Randomized Ablation (RA – Baseline)				
L	Plural		Run-Off		e	[LF20b]		[Jia+22b]	
	Acc. (%)	R_{med}	Acc. (%)	R_{med}		Acc. (%)	ρ_{med}	Acc. (%)	ρ_{med}
5	91.46	2	91.77	2	250	88.77	2	88.56	2
10	86.09	4	86.20	4	225	88.05	2	87.90	2
20	81.38	7	81.40	7	200	86.76	3	86.54	3
25	78.65	8	78.58	8	175	86.16	3	85.94	3
40	74.74	9	74.95	10	150	84.23	4	84.08	4
55	70.44	10	70.34	11	125	82.66	5	82.49	5
70	67.46	9	67.47	11	100	80.43	6	80.05	6
85	66.24	10	66.61	12	75	78.48	7	78.11	7
105	63.55	10	63.61	12	50	73.26	7	72.79	8
115	62.39	11	62.35	13	35	70.34	7	69.72	9
140	60.35	10	60.57	12	30	69.62	7	69.01	9
165	57.91	8	58.48	10	25	68.81	6	68.08	9
185	56.08	7	56.39	9	20	67.01	5	66.15	9
200	55.80	7	56.43	9	15	65.68	3	64.74	10
225	56.27	6	56.56	8	12	63.93	0	62.91	10
250	53.30	4	53.46	5	10	62.73	0	61.71	10
					8	60.24	0	59.12	9
					7	59.08	0	57.83	8
					5	53.20	0	51.84	3

Table C.22. **MNIST Detailed Results:** Classification accuracy (%) and median certified robustness (larger is better) for the MNIST [LeC+98] dataset ($d = 784$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. Each certification method’s hyperparameter setting with the best median robustness is shown in **bold**. The best overall median robustness is shown in **blue**.

(a) Feature Partition Aggregation (Ours)					(b) Randomized Ablation (RA – Baseline)				
L	Plural		Run-Off		e	[LF20b]		[Jia+22b]	
	Acc. (%)	R_{med}	Acc. (%)	R_{med}		Acc. (%)	ρ_{med}	Acc. (%)	ρ_{med}
5	99.50	2	99.51	2	100	98.78	4	98.75	4
10	98.64	4	98.67	4	95	98.75	5	98.72	5
15	96.82	7	97.02	7	90	98.62	5	98.56	5
20	96.36	8	96.53	8	85	98.60	5	98.52	5
25	95.77	9	96.06	10	80	98.46	6	98.40	6
35	91.70	9	93.05	11	75	98.35	6	98.27	6
40	89.37	9	91.32	11	70	98.14	6	98.07	6
50	84.54	8	88.46	11	65	98.04	7	97.98	7
60	83.54	9	87.22	12	60	97.85	7	97.78	7
70	79.71	8	85.87	11	55	97.58	7	97.39	8
80	71.29	6	79.05	9	50	97.26	7	97.07	8
90	69.94	6	79.25	9	45	96.88	8	96.68	8
105	62.53	4	74.45	8	40	96.42	8	96.13	9
120	63.03	3	74.09	7	35	95.69	8	95.32	9
130	57.48	2	69.93	7	30	94.87	7	94.47	9
150	52.51	0	67.30	5	25	93.55	6	93.09	10
					20	90.99	3	90.07	9
					15	86.71	0	85.24	8
					10	76.78	0	74.69	6
					5	35.54	$-\infty$	32.89	$-\infty$

Table C.23. **Weather Detailed Results:** Classification accuracy (%) and median certified robustness (larger is better) for the Weather [Mal+21] dataset ($d = 128$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. FPA considers only plurality voting-based certification (Sec. 5.3.1) since the reduction is from certified regression to certified *binary* classification. FPA results are reported using both GBDTs [Ke+17] and linear submodels. Median robustness “ $-\infty$ ” denotes that the classification accuracy was less than 50%. Each approach’s hyperparameter setting with the best median robustness is shown in **bold**. The best overall median robustness is shown in **blue**.

(a) Feature Partition Aggregation (Ours)					(b) Randomized Ablation (RA – Baseline)				
L	LightGBM		Linear		e	[LF20b]		[Jia+22b]	
	Acc. (%)	R_{med}	Acc. (%)	R_{med}		Acc. (%)	ρ_{med}	Acc. (%)	ρ_{med}
1	92.70	0	86.05	0	65	80.70	0	78.63	0
5	85.29	2	83.34	2	60	80.33	0	78.01	0
11	82.48	3	79.55	2	55	79.52	0	77.05	0
15	81.09	3	76.15	3	50	78.62	0	76.59	0
21	76.10	4	67.09	2	45	77.20	0	75.19	1
25	71.40	3	64.77	2	40	76.56	0	74.82	1
31	67.06	3	58.71	2	35	74.76	0	73.22	1
35	62.56	3	55.95	1	30	72.04	0	70.74	1
41	60.19	2	51.57	0	25	69.77	0	68.72	1
51	55.34	1	45.84	$-\infty$	20	66.94	0	65.87	1
75	42.20	$-\infty$	26.93	$-\infty$	16	63.89	0	63.10	1
101	28.67	$-\infty$	21.26	$-\infty$	12	58.59	0	57.74	1
					8	53.44	0	52.82	0
					6	47.94	$-\infty$	47.25	$-\infty$
					4	40.70	$-\infty$	39.91	$-\infty$

Table C.24. **Ames Detailed Results:** Classification accuracy (%) and median certified robustness (larger is better) for the Ames [Coc11] dataset ($d = 352$) for our certified sparse defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) across various hyperparameter settings. FPA considers only plurality voting-based certification (Sec. 5.3.1) since the reduction is from certified regression to certified *binary* classification. FPA results are reported using both GBDTs [Ke+17] and linear submodels. Median robustness “ $-\infty$ ” denotes that the classification accuracy was less than 50%. Each approach’s hyperparameter setting with the best median robustness is shown in **bold**. The best overall median robustness is shown in **blue**.

(a) Feature Partition Aggregation (Ours)					(b) Randomized Ablation (RA – Baseline)				
L	LightGBM		Linear		e	[LF20b]		[Jia+22b]	
	Acc. (%)	R_{med}	Acc. (%)	R_{med}		Acc. (%)	ρ_{med}	Acc. (%)	ρ_{med}
1	88.05	0	89.25	0	70	68.60	0	66.89	0
5	84.64	1	82.08	1	60	68.94	0	67.24	1
11	78.50	2	74.40	1	50	67.58	1	66.89	1
15	73.04	2	66.55	2	40	61.77	1	61.77	1
21	65.53	3	61.60	2	35	61.09	0	60.07	1
25	61.77	2	57.34	1	30	57.68	0	57.00	1
31	57.68	2	53.58	0	25	53.58	0	52.56	1
35	55.97	1	50.34	0	20	51.54	0	49.49	$-\infty$
41	52.90	1	46.42	$-\infty$	15	45.05	$-\infty$	44.37	$-\infty$
51	47.10	$-\infty$	40.10	$-\infty$	10	37.20	$-\infty$	37.54	$-\infty$
75	36.86	$-\infty$	35.15	$-\infty$	5	33.79	$-\infty$	33.79	$-\infty$

C.2.3 Feature Partition Aggregation vs. Randomized Ablation

Certified Accuracy Detailed Comparison. Levine and Feizi [LF20b] use median certified robustness and classification accuracy as the two primary metrics by which they compare RA against previous work. In this section, we present an alternative evaluation strategy comparing the methods’ certified accuracy across a range of robustness levels.

Specifically, we consider the same four datasets from Section 5.5, namely classification datasets CIFAR10 [KNH14] and MNIST [LeC+98] as well as regression datasets Weather [Mal+21] and Ames [Coc11]. Like in Section 5.5, we report FPA’s performance using both the plurality-voting and run-off decision functions for classification and only plurality voting for regression. For baseline randomized ablation (RA), we again report the performance of Levine and Feizi’s [LF20b] original version of RA as well as the improved version by Jia et al. [Jia+22b].

This section also compares FPA and RA against a *naive baseline* that is generally low accuracy but maximally robust. For classification, the naive baseline always predicts $f(\mathbf{x}) = 1$; for regression, the naive baseline always predicts the training set’s median target value.

Recall that hyperparameters L for FPA and e for baseline randomized ablation control the corresponding method’s accuracy versus robustness trade-off. Specifically, a smaller value of L and a larger value of e entails better accuracy. As a **rule of thumb**, the fairest comparison between FPA and RA is when $L \approx \frac{d}{e}$ as each FPA and RA prediction, in expectation, uses a comparable amount of information (i.e., number of features). For each dataset, we report each method’s certified accuracy across 10 hyperparameter settings, roughly following the rule of thumb above. Section C.2.3.1

presents the experimental results in tabular form, and Section C.2.3.2 visualizes the methods’ certified accuracy graphically.

C.2.3.1 Numerical Comparison of Feature Partition Aggregation and Randomized Ablation. *Certified accuracy* w.r.t. $\psi \in \mathbb{N}$ quantifies the fraction of correctly-classified test instances with certified robustness at least ψ .

Tables C.25, C.26, C.27, and C.28 numerically display the certified accuracies for our certified feature defense, feature partition aggregation (FPA), and baseline randomized ablation (RA) for CIFAR10, MNIST, Weather, and Ames, respectively. For each dataset, the corresponding table lists the certified accuracy at 11 equally spaced certified robustness levels.

Recall that RA’s ℓ_0 -norm robustness (Def. 5.2) is a strictly weaker guarantee than FPA’s certified feature robustness (Def. 5.1). Put simply, a true direct comparison is not possible here since FPA provides stronger certified guarantees than the baseline. Despite that, FPA can achieve larger certified accuracies than the baseline while simultaneously providing stronger guarantees.

Table C.25. CIFAR10 ($d = 1024$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “Plurality” denotes FPA with plurality voting as the decision function while “Run-Off” denotes FPA using run-off elections as the decision function. “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. We also consider an additional naive baseline that always predicts $f(\mathbf{x}) = 1$. For each certified robustness level, each method’s best performing hyperparameter setting is shown in **bold** with the overall best performing method shown in **blue**.

Method	Cert. Alg.	Hyper. Setting	Certified Robustness										
			0	13	26	39	52	65	78	91	104	117	130
Always $f(\mathbf{x}) = 1$		N/A	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
FPA (ours)	Plural	5	91.46	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		25	78.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		35	69.62	36.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		55	70.44	44.06	10.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		85	66.24	46.67	26.87	7.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		115	62.39	47.74	33.48	19.67	6.97	0.00	0.00	0.00	0.00	0.00	0.00
		160	60.94	42.27	27.77	16.95	9.00	3.89	0.52	0.00	0.00	0.00	0.00
		250	53.30	43.98	35.63	28.37	21.54	15.57	10.91	7.04	4.02	1.62	0.00
		500	43.79	38.75	33.63	28.86	24.65	20.86	17.56	14.32	11.56	9.38	7.66
	1024	33.01	29.70	26.95	24.14	21.68	19.33	17.24	15.41	13.92	12.29	11.05	
	Run-Off	5	91.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		25	78.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		35	69.92	37.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		55	70.34	46.71	11.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		85	66.61	49.26	30.25	8.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		115	62.35	50.04	36.76	22.64	8.21	0.00	0.00	0.00	0.00	0.00	0.00
		160	61.34	45.54	32.71	21.16	11.96	5.06	0.56	0.00	0.00	0.00	0.00
		250	53.46	45.48	38.40	31.70	25.24	19.02	13.48	8.94	4.99	1.88	0.00
		500	44.58	39.58	35.25	31.17	27.60	24.21	20.57	17.62	14.74	12.33	10.25
1024	35.50	32.01	28.80	25.89	23.22	20.74	18.63	16.85	15.20	13.80	12.57		
RA	[LF20b]	250	88.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		75	78.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		50	73.26	25.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		25	68.81	38.82	11.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		15	65.68	38.81	23.59	9.42	0.00	0.00	0.00	0.00	0.00	0.00	
		10	62.73	37.60	27.46	17.72	9.74	1.89	0.00	0.00	0.00	0.00	
		7	59.08	33.44	25.65	18.58	12.56	7.77	3.71	1.09	0.00	0.00	
	5	53.20	28.47	22.80	17.85	14.04	10.10	6.87	4.20	2.31	0.94	0.05	
	2	40.44	14.03	12.37	10.62	9.12	7.91	6.96	5.95	5.16	4.51	3.98	
	1	21.16	4.37	3.87	3.37	2.91	2.58	2.35	1.90	1.68	1.42	1.21	
	[Jia+22b]	250	88.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		75	78.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		50	72.79	26.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
25		68.08	43.10	12.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
15		64.74	46.17	28.17	11.25	0.00	0.00	0.00	0.00	0.00	0.00		
10		61.71	47.54	34.36	22.44	11.99	2.31	0.00	0.00	0.00	0.00		
7		57.83	46.43	35.75	26.23	17.70	10.79	4.96	1.33	0.00	0.00		
5	51.84	43.08	34.70	27.14	20.77	15.27	10.36	6.32	3.34	1.21	0.06		
2	38.70	33.84	29.15	25.01	21.22	17.95	14.90	12.49	10.33	8.54	7.03		
1	19.64	17.96	15.83	14.06	12.48	11.18	10.17	9.06	8.24	7.35	6.48		

Table C.26. MNIST ($d = 784$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “Plurality” denotes FPA with plurality voting as the decision function while “Run-Off” denotes FPA using run-off elections as the decision function. “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. We also consider an additional naive baseline that always predicts $f(\mathbf{x}) = 1$. For each certified robustness level, each method’s best performing hyperparameter setting is shown in **bold** with the overall best performing method shown in **blue**.

Method	Cert. Alg.	Hyper. Setting	Certified Robustness										
			0	4	8	12	16	20	24	28	32	36	40
Always $f(\mathbf{x}) = 1$		N/A	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35
FPA (ours)	Plural	5	99.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		10	98.64	87.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		25	95.77	86.48	66.42	20.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		35	91.70	79.49	59.53	35.95	13.18	0.00	0.00	0.00	0.00	0.00	0.00
		60	83.54	70.30	54.72	39.10	26.26	16.08	7.95	1.78	0.00	0.00	0.00
		75	74.99	61.44	47.75	34.97	25.34	17.90	12.43	8.11	3.89	0.42	0.00
		90	69.94	57.11	43.89	33.01	24.52	17.89	12.99	9.16	6.24	3.22	0.71
		105	62.53	50.33	39.10	29.27	22.13	16.52	13.04	10.51	8.42	6.61	4.63
		130	57.48	46.68	36.45	28.38	22.70	18.52	15.23	12.54	10.45	8.38	6.30
		240	28.13	24.67	21.81	19.57	17.63	16.33	15.16	14.40	13.79	13.00	12.30
	Run-Off	5	99.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		10	98.67	87.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		25	96.06	88.72	71.52	20.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		35	93.05	83.56	67.58	44.72	14.36	0.00	0.00	0.00	0.00	0.00	0.00
		60	87.22	76.59	63.67	50.52	37.10	23.91	12.14	2.97	0.00	0.00	0.00
		75	81.74	68.54	56.44	44.65	34.68	25.48	17.82	11.09	5.28	0.45	0.00
		90	79.25	66.38	53.93	43.35	33.92	26.20	20.14	14.71	9.98	6.02	2.34
		105	74.45	61.76	50.73	40.32	31.38	24.57	19.00	14.85	11.80	9.05	6.46
		130	69.93	58.88	48.44	38.73	31.04	25.06	20.82	17.47	14.69	12.00	9.85
		240	48.33	40.31	33.37	28.30	24.57	21.29	18.71	17.17	15.82	14.82	13.81
RA	[LF20b]	100	98.78	84.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		85	98.60	86.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		60	97.85	84.30	35.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		50	97.26	81.56	49.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		40	96.42	76.53	51.99	16.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		30	94.87	66.97	46.33	26.88	7.30	0.00	0.00	0.00	0.00	0.00	0.00
		20	90.99	48.11	34.38	23.77	15.23	7.50	0.96	0.00	0.00	0.00	0.00
		10	76.78	20.36	16.22	13.08	10.62	8.40	5.99	3.72	1.54	0.16	0.00
		5	35.54	10.85	10.31	9.75	9.17	8.69	7.86	6.90	5.73	4.42	3.23
		3	16.91	11.13	10.96	10.70	10.51	10.19	9.84	9.41	8.87	8.21	7.04
[Jia+22b]	100	98.75	86.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	85	98.52	88.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	60	97.78	88.45	39.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	50	97.07	87.28	57.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	40	96.13	85.69	62.37	21.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	30	94.47	82.47	62.32	36.45	11.20	0.00	0.00	0.00	0.00	0.00	0.00	
	20	90.07	76.29	58.26	39.39	24.36	12.98	2.70	0.00	0.00	0.00	0.00	
	10	74.69	59.11	44.55	32.87	23.94	17.91	13.49	10.38	7.33	3.73	0.80	
	5	32.89	26.17	21.19	19.56	15.76	14.46	13.43	12.52	11.51	10.77	10.05	
	3	15.91	14.97	13.90	13.10	12.46	12.01	11.71	11.50	11.40	11.30	11.30	

Table C.27. Weather [Mal+21] dataset ($d = 128$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. Hammoudeh and Lowd’s [HL23c] reduction is from certified regression to certified binary classification. Run-off is identical to plurality voting under binary classification, so we report only the plurality voting results below. We also consider an additional naive baseline that always predicts the median training set target value (i.e., $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$). For each certified robustness level, each method’s best performing hyperparameter setting is shown in **bold** with the overall best performing method shown in **blue**. These numerical results are visualized graphically as envelope plots in Figure C.21.

Method	Cert. Alg.	Hyper. Setting	Certified Robustness												
			0	1	2	3	4	5	6	7	8	9	10		
Always $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$		N/A	21.90	21.90	21.90	21.90	21.90	21.90	21.90	21.90	21.90	21.90	21.90		
FPA (ours)	Plural	5	85.29	77.38	62.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
		11	82.48	76.34	67.59	55.50	39.02	18.42	0.00	0.00	0.00	0.00	0.00		
		15	81.09	75.23	68.16	58.98	48.08	35.81	19.92	7.77	0.00	0.00	0.00		
		21	76.10	70.78	64.73	57.69	50.01	41.48	33.04	23.78	14.30	6.47	0.91		
		25	71.40	66.29	60.70	55.03	49.17	42.93	35.88	28.92	21.58	14.29	7.12		
		31	67.06	62.80	58.18	53.39	48.76	43.85	38.49	32.77	27.12	21.51	15.81		
		35	62.56	58.84	54.93	50.72	46.54	42.03	37.62	33.08	28.10	22.76	17.18		
		41	60.19	56.83	53.34	49.72	45.99	42.34	38.55	34.60	30.44	26.09	21.47		
		45	57.96	54.99	51.94	48.81	45.57	42.26	38.78	35.11	31.29	27.23	22.91		
		127	23.43	22.95	22.49	22.04	21.61	21.19	20.77	20.38	20.00	19.61	19.23		
		RA	[LF20b]	50	78.62	22.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				40	76.56	31.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				30	72.04	39.64	9.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				20	66.94	45.11	20.61	6.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				16	63.89	45.77	26.67	11.64	3.83	0.04	0.00	0.00	0.00	0.00	0.00
				12	58.59	45.19	31.87	18.36	9.67	4.37	1.06	0.00	0.00	0.00	0.00
				9	54.68	44.55	35.11	25.05	15.88	9.48	5.26	2.26	0.61	0.01	0.00
6	47.94			41.22	34.84	28.60	22.32	16.45	11.82	8.60	6.00	3.90	2.37		
3	36.88			33.32	30.57	27.90	25.63	23.08	20.58	18.16	15.97	13.91	11.87		
1	21.00			20.68	20.61	20.48	20.35	20.19	20.05	19.93	19.77	19.67	19.43		
[Jia+22b]	50			76.59	47.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	40	74.82	53.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			
	30	70.74	56.18	31.24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			
	20	65.87	56.66	44.24	26.06	3.94	0.00	0.00	0.00	0.00	0.00	0.00			
	16	63.10	55.29	46.24	34.49	19.75	5.20	0.00	0.00	0.00	0.00	0.00			
	12	57.74	51.96	45.73	38.47	29.53	19.26	10.88	0.00	0.00	0.00	0.00			
	9	53.97	49.95	45.97	41.18	35.62	29.11	21.44	14.51	9.10	2.63	0.00			
	6	47.25	44.86	41.94	39.16	36.21	33.00	29.54	25.82	21.18	16.82	13.31			
	3	36.01	34.97	33.59	32.19	31.02	29.72	28.46	27.33	26.28	25.21	23.99			
1	20.84	20.76	20.72	20.63	20.58	20.50	20.41	20.31	20.25	20.14	20.03				

Table C.28. Ames [Coc11] dataset ($d = 352$) certified accuracy for feature partition aggregation (FPA) and baseline randomized ablation (RA). “[LF20b]” denotes Levine and Feizi’s [LF20b] original version of RA while “[Jia+22b]” denotes Jia et al.’s [Jia+22b] improved version of RA. Hammoudeh and Lowd’s [HL23c] reduction is from certified regression to certified binary classification. Run-off is identical to plurality voting under binary classification, so we report only the plurality voting results below. We also consider an additional naive baseline that always predicts the median training set target value (i.e., $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$). For each certified robustness level, each method’s best performing hyperparameter setting is shown in **bold** with the overall best performing method shown in **blue**. These numerical results are visualized graphically as envelope plots in Figure C.21.

Method	Cert. Alg.	Hyper. Setting	Certified Robustness										
			0	1	2	3	4	5	6	7	8	9	10
Always $f(\mathbf{x}) = \text{med}\{y_i\}_{i=1}^n$		N/A	31.40	31.40	31.40	31.40	31.40	31.40	31.40	31.40	31.40	31.40	31.40
FPA (ours)	Plural	5	84.64	72.01	39.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		11	78.50	70.99	58.70	40.96	22.53	5.12	0.00	0.00	0.00	0.00	0.00
		21	65.53	60.41	54.95	50.17	41.64	32.42	22.87	12.63	5.46	1.37	0.00
		25	61.77	58.36	54.27	49.83	43.69	35.84	28.67	20.82	12.63	6.14	2.39
		31	57.68	54.95	51.54	48.12	42.66	37.20	32.08	26.28	20.82	15.02	10.24
		35	55.97	52.56	48.81	45.73	42.32	38.23	33.79	29.01	24.57	19.45	14.68
		41	52.90	50.51	47.10	43.34	40.96	37.20	34.47	31.06	27.65	24.23	20.82
		51	47.10	44.37	41.98	39.25	37.88	35.49	34.13	32.08	30.03	28.33	26.28
		65	41.64	39.25	37.88	37.20	36.01	34.47	33.45	32.42	31.40	30.38	29.69
		101	33.45	33.11	32.76	32.76	32.42	32.08	32.08	31.74	31.74	31.74	31.40
RA	[LF20b]	60	68.94	43.34	11.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		50	67.58	52.56	32.08	7.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		40	61.77	50.17	38.23	18.09	4.10	0.00	0.00	0.00	0.00	0.00	0.00
		35	61.09	49.49	39.93	20.48	10.24	1.71	0.00	0.00	0.00	0.00	0.00
		30	57.68	48.46	39.59	26.96	16.38	5.46	0.00	0.00	0.00	0.00	0.00
		25	53.58	47.78	38.91	27.65	20.82	15.02	4.10	0.34	0.00	0.00	0.00
		20	51.54	43.34	38.23	32.76	26.28	20.48	15.02	7.85	2.39	0.00	0.00
		15	45.05	39.25	36.18	34.81	29.69	27.99	23.21	19.45	13.99	9.90	5.80
		10	37.20	36.18	35.15	33.11	32.76	31.40	28.67	26.62	25.26	24.57	22.87
		5	33.79	33.11	32.76	32.08	32.08	32.08	31.74	31.40	31.06	30.38	30.38
RA	[Jia+22b]	60	67.24	59.73	46.76	13.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		50	66.89	59.73	48.81	31.40	7.17	0.00	0.00	0.00	0.00	0.00	0.00
		40	61.77	55.63	49.49	38.57	25.60	6.48	0.00	0.00	0.00	0.00	0.00
		35	60.07	52.90	48.12	38.91	31.06	16.38	2.39	0.00	0.00	0.00	0.00
		30	57.00	51.88	47.10	41.30	34.81	26.96	15.36	2.39	0.00	0.00	0.00
		25	52.56	50.17	45.39	40.27	35.84	31.06	24.91	17.06	6.48	0.34	0.00
		20	49.49	45.73	44.03	41.30	37.54	33.79	30.38	25.94	22.53	13.99	6.83
		15	44.37	42.32	40.96	39.93	35.84	35.49	32.76	30.72	27.65	24.91	22.18
		10	37.54	36.52	35.84	33.79	33.79	33.45	32.42	31.06	30.38	29.35	29.01
		5	33.79	33.45	33.45	33.11	33.11	33.11	32.76	32.76	32.42	32.08	32.08

C.2.3.2 Graphical Comparison of Feature Partition Aggregation and Randomized Ablation. Recall that hyperparameters L for FPA and e for baseline randomized ablation control the corresponding method’s accuracy-robustness trade-off. Specifically, a smaller value of L and a larger value of e entails better accuracy. This section emulates a defender that tunes FPA’s and randomized ablation’s hyperparameters to maximize the certified accuracy *at each individual robustness level individually*.

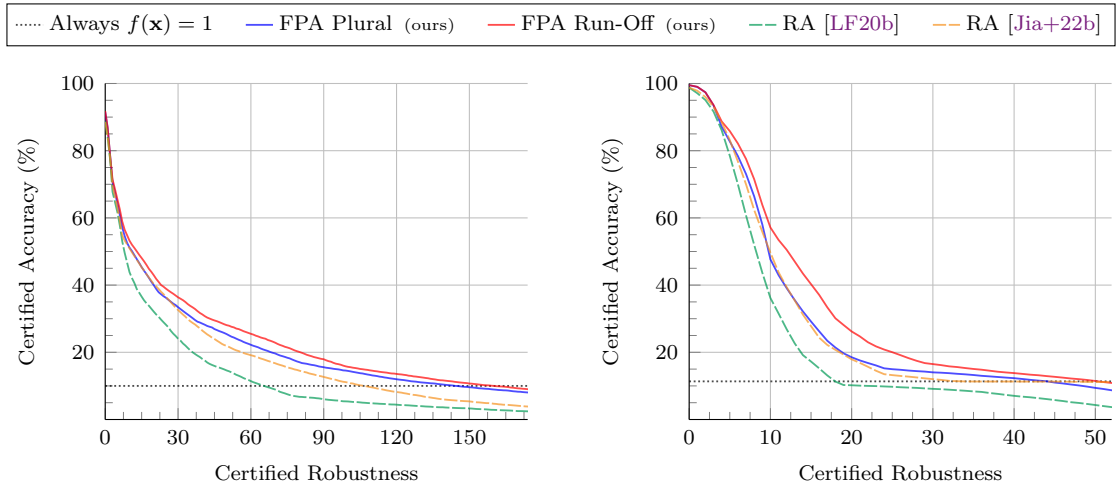
Tables C.25 through C.28 above report each method’s certified accuracy across 10 comparable hyperparameter settings. For a given method, each hyperparameter setting provides a certified accuracy versus certified robustness curve. This section considers each defense’s certified accuracy *envelope*. Specifically, an envelope in mathematics represents the supremum of a set of curves. Intuitively, taking the certified accuracy envelope emulates maximizing a method’s performance at each certified robustness level individually across the 10 hyperparameter settings.

Figures C.20 and C.21 visualize the certified accuracy envelopes in two ways. First, Figures C.20a, C.20b, C.21a, and C.21b visualize the envelope curves themselves. These figures also visualize the same naive baselines considered in Sec. C.2.3.1 above (e.g., always predict label 1 for classification and median $\text{med}\{y_i\}_{i=1}^n$ for regression). Second, Figures C.20c, C.20d, C.21c, and C.21d visualize the improvement in certified accuracy between FPA and the two versions of randomized ablation across the range of certified robustness levels. A positive value in these four subfigures entails that FPA outperformed the corresponding baseline (i.e., FPA had a larger certified accuracy), while a negative value entails the baseline outperformed FPA.

For CIFAR10 and MNIST, FPA with run-off’s envelope had larger certified accuracy than the envelope of both versions of baseline RA across the entire certified

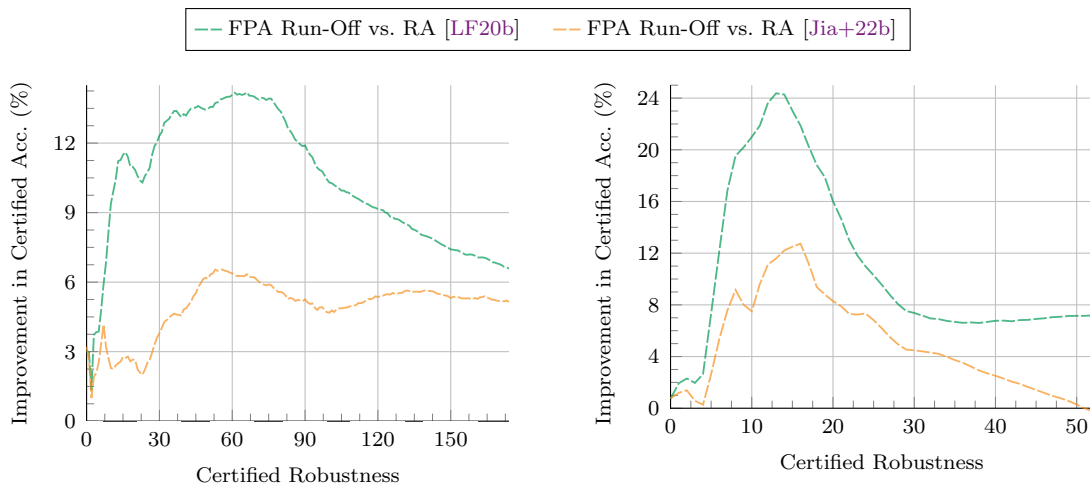
robustness range (x-axis). Specifically, for Levine and Feizi’s [LF20b] version of RA, FPA with run-off’s certified accuracy advantage was as large as 14.17 and 24.28 percentage points (pp) for CIFAR10 and MNIST, respectively. For Jia et al.’s [Jia+22b] version of RA, FPA with run-off’s certified accuracy advantage was as large as 6.54pp and 12.74pp for CIFAR10 and MNIST, respectively.

For regression datasets Weather and Ames, FPA’s envelope had larger certified accuracy than the envelope of both versions of baseline RA across most of the certified accuracy range. At the largest robustness values, [Jia+22b] marginally outperformed both FPA and the naive baseline by <2pp. At smaller certified robustness values, FPA outperformed Jia et al.’s [Jia+22b] version of RA by up to 21.9pp and 17.4pp for Weather and Ames, respectively.



(a) CIFAR10: Certified Accuracy Envelope

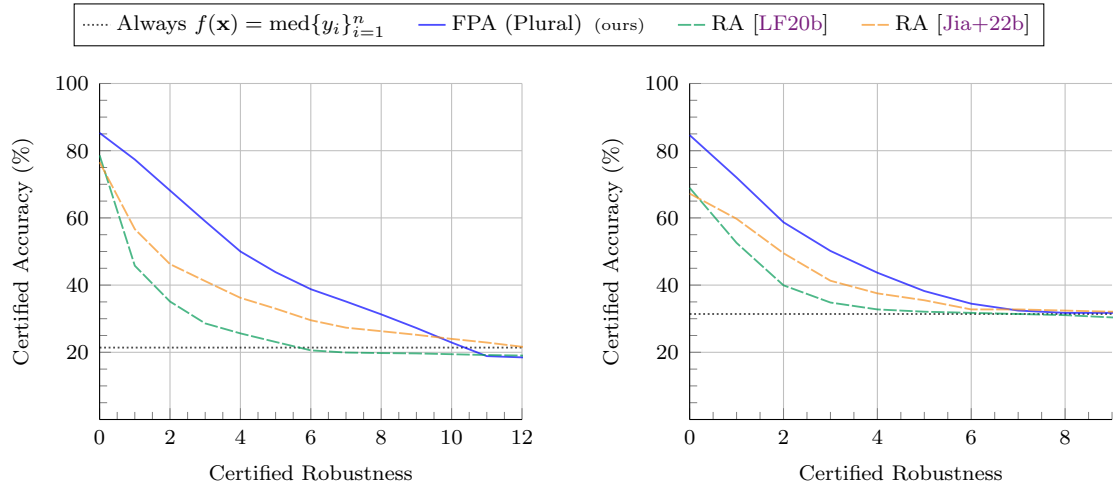
(b) MNIST: Certified Accuracy Envelope



(c) CIFAR10: FPA’s Certified Accuracy Improvement over RA

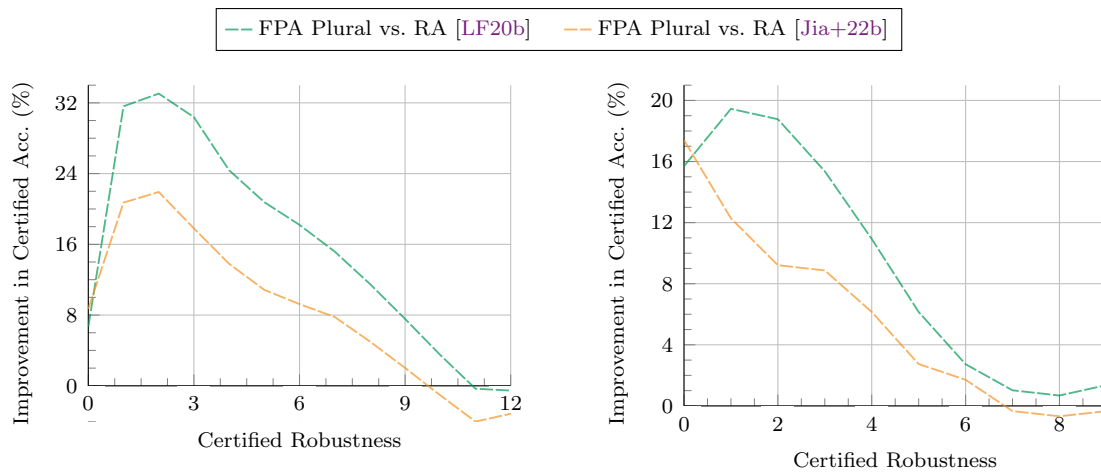
(d) MNIST: FPA’s Certified Accuracy Improvement over RA

Figure C.20. Classification certified accuracy envelope for datasets CIFAR10 ($d = 1024$) and MNIST ($d = 784$) for feature partition aggregation (FPA) and baseline randomized ablation (RA). Each method’s envelope considers the corresponding hyperparameters in Tables C.25 and C.26, emulating a certified defense where the hyperparameters are roughly tuned to maximize the certified accuracy at each robustness level. Subfigures C.20a and C.20b visualize each method’s certified accuracy envelope (larger is better); also shown in these subfigures is a naive baseline where the decision function always predicts label $f(\mathbf{x}) = 1$. Subfigures C.20c and C.20d visualize the improvement in certified accuracy when using FPA with the run-off decision function over the two randomized ablation baselines from Levine and Feizi [LF20b] and Jia et al. [Jia+22b]. The envelope plots’ underlying numerical values are provided in Table C.25 for CIFAR10 and Table C.26 for MNIST.



(a) Weather: Certified Accuracy Envelope

(b) Ames: Certified Accuracy Envelope



(c) Weather: FPA’s Certified Accuracy Improvement over RA

(d) Ames: FPA’s Certified Accuracy Improvement over RA

Figure C.21. Regression certified accuracy envelope for the Weather [Mal+21] ($d = 128$) and Ames [Coc11] ($d = 352$) datasets for feature partition aggregation (FPA) and baseline randomized ablation (RA). Each method’s envelope considers the corresponding hyperparameters in Tables C.27 and C.28, emulating a certified defense where the hyperparameters are tuned to maximize each robustness level’s certified accuracy. Subfigures C.21a and C.21b visualize each method’s certified accuracy envelope (larger is better); also shown in these subfigures is a naive baseline that always predicts the median training data target value. Subfigures C.21c and C.21d visualize the improvement in certified accuracy when using FPA (with plurality voting) as the decision function over the two randomized ablation baselines from Levine and Feizi [LF20b] and Jia et al. [Jia+22b]. FPA outperforms randomized ablation for smaller certified robustness values, while Jia et al.’s [Jia+22b] version of RA marginally outperformed both FPA and the naive baseline at larger robustness values. The envelope plots’ underlying numerical values are provided in Table C.27 for Weather and Table C.28 for Ames.

C.3 Chapter 6 Detailed Results

Section 6.5 provided averaged results for each related experimental setup. This section provides detailed results for each attack setup individually (including variance).

C.3.1 Speech Recognition Backdoor Full Results.

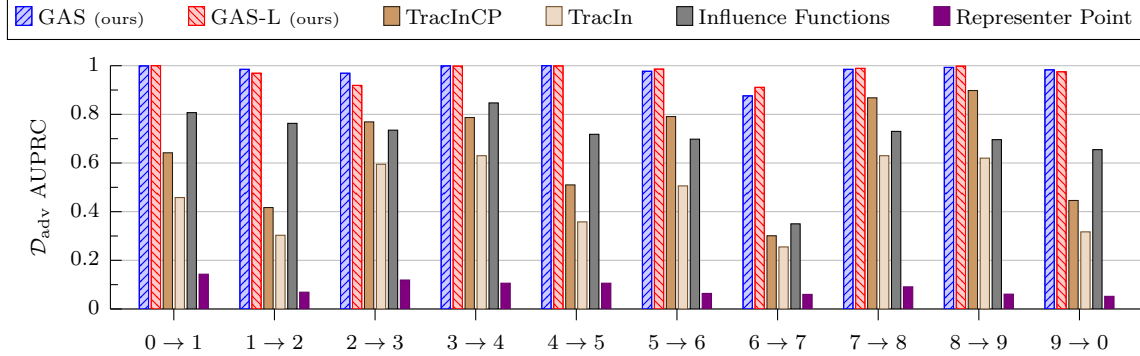


Figure C.22. **Speech Backdoor Adversarial Set Identification:** Mean backdoor set (\mathcal{D}_{adv}) identification AUPRC across 30 trials for all 10 class pairs with $21 \leq |\mathcal{D}_{adv}| \leq 28$ (varies by class pair, see Tab. D.57). GAS and GAS-L outperformed all baselines in all experiments, with GAS-L the overall top performer on 6/10 class pairs. See Table C.29 for the numerical results.

Table C.29. **Speech Backdoor Adversarial Set Identification:** Mean AUPRC across 30 trials for speech backdoor dataset [Liu+18] with $21 \leq |\mathcal{D}_{adv}| \leq 28$. GAS(-L) always outperformed the baselines. Bold denotes the best mean performance. Mean results are shown graphically in Figs. 14 and C.22. Variance results appear in the original paper [HL22a, Sec. F.1.1].

Digits		Ours		Baselines			
$y_{targ} \rightarrow y_{adv}$		GAS	GAS-L	TracInCP	TracIn	Inf. Func.	Rep. Pt.
0	1	0.999	1.000	0.642	0.458	0.807	0.143
1	2	0.985	0.969	0.417	0.303	0.763	0.069
2	3	0.969	0.919	0.769	0.595	0.735	0.119
3	4	0.999	0.998	0.787	0.630	0.847	0.106
4	5	1.000	0.999	0.510	0.358	0.718	0.106
5	6	0.977	0.986	0.791	0.506	0.698	0.064
6	7	0.876	0.911	0.301	0.255	0.350	0.060
7	8	0.985	0.989	0.868	0.630	0.730	0.091
8	9	0.993	0.998	0.898	0.620	0.696	0.061
9	0	0.983	0.975	0.446	0.317	0.655	0.052

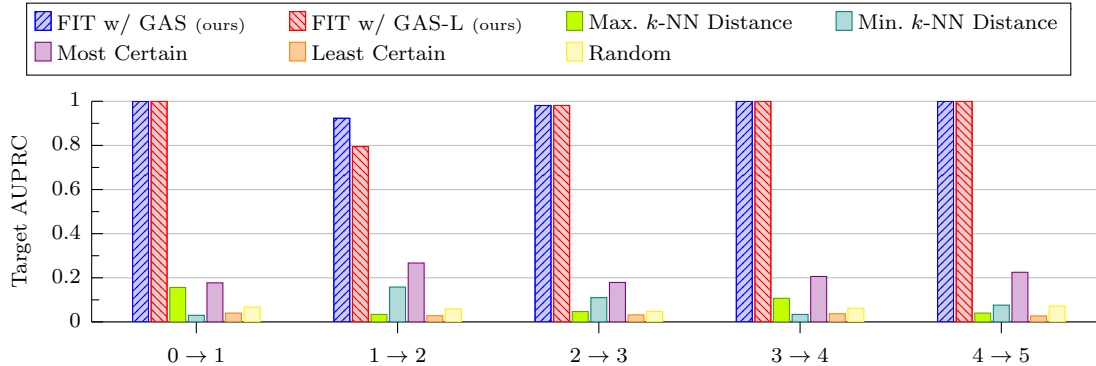


Figure C.23. **Speech Backdoor Target Identification:** See Table C.30 for numerical results.

Table C.30. **Speech Backdoor Target Identification:** Bold denotes the best mean performance. Mean results are shown graphically in Figures 16 and C.23. Variance results appear in the original paper [HL22a, Sec. F.1.1].

Digits		Ours		Baselines				
y_{targ}	y_{adv}	GAS	GAS-L	Max k -NN	Min k -NN	Most Certain	Least Certain	Random
0	1	1	1	0.156	0.030	0.177	0.040	0.067
1	2	0.923	0.795	0.034	0.158	0.267	0.028	0.059
2	3	0.981	0.981	0.047	0.110	0.179	0.032	0.047
3	4	1	1	0.107	0.034	0.206	0.037	0.062
4	5	1	1	0.040	0.076	0.225	0.027	0.072

Table C.31. **Speech Backdoor Attack Mitigation:** Bold denotes the best mean performance with 10 trials per class pair. Aggregated results are shown in Table 6.

Digits		Method	% Removed		ASR %		Test Acc. %	
y_{targ}	y_{adv}		\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
0	1	GAS	100	0.06	100	0	97.7	0.0
		GAS-L	100	0.03	100	0	97.7	0.0
1	2	GAS	100.0	0.02	100	0	97.7	0.0
		GAS-L	99.8	0.09	100	0	97.7	0.0
2	3	GAS	93.7	0.08	99.9	0	97.8	-0.1
		GAS-L	92.6	0.21	99.9	0	97.8	-0.1
3	4	GAS	98.7	0.10	99.4	0	97.7	-0.1
		GAS-L	99.3	0.35	99.4	0	97.7	0.0
4	5	GAS	99.1	0.01	100	0	97.8	0.0
		GAS-L	98.6	0.01	100	0	97.8	0.0

C.3.2 Vision Backdoor Full Results.

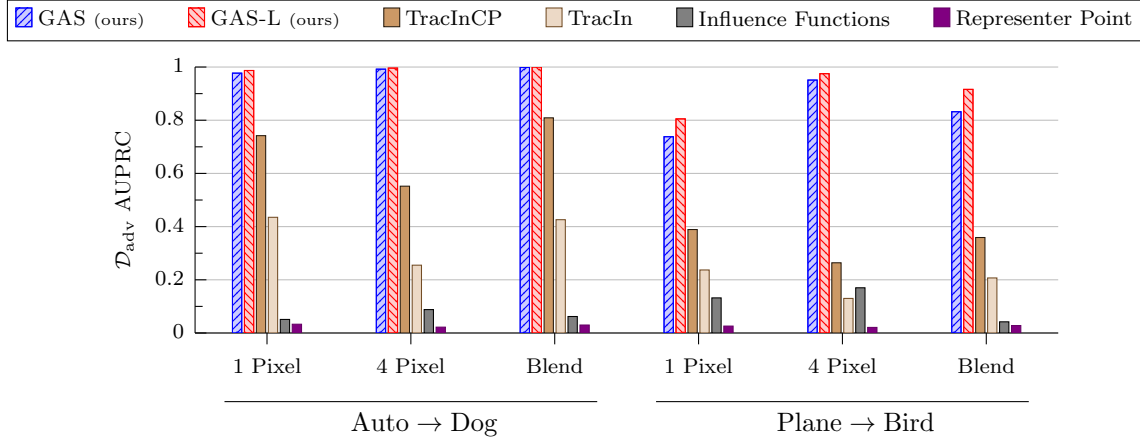


Figure C.24. **Vision Backdoor Adversarial-Set Identification:** Backdoor set, \mathcal{D}_{adv} , identification mean AUPRC across >30 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns with a randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $|\mathcal{D}_{\text{adv}}| = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. See Table C.32 for the numerical results.

Table C.32. **Vision Backdoor Adversarial-Set Identification:** Backdoor set, \mathcal{D}_{adv} , identification mean AUPRC across >30 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns with a randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $|\mathcal{D}_{\text{adv}}| = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. Bold denotes the best mean performance. Mean results are shown graphically in Figures 14 and C.24. Variance results appear in the original paper [HL22a, Sec. F.1.2].

Classes $y_{\text{targ}} \rightarrow y_{\text{adv}}$	Trigger Pattern	Ours		Baselines			
		GAS	GAS-L	TracInCP	TracIn	Inf. Func.	Rep. Pt.
Auto → Dog	1 Pixel	0.977	0.987	0.742	0.435	0.051	0.033
	4 Pixel	0.992	0.996	0.552	0.255	0.088	0.022
	Blend	0.999	1.000	0.809	0.426	0.062	0.030
Plane → Bird	1 Pixel	0.738	0.805	0.389	0.237	0.132	0.026
	4 Pixel	0.951	0.975	0.264	0.130	0.170	0.021
	Blend	0.832	0.916	0.359	0.207	0.042	0.028

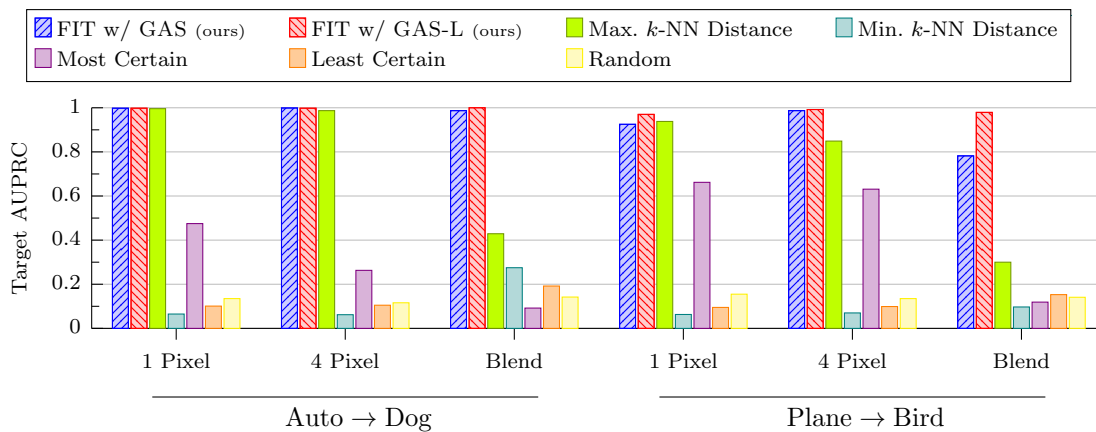


Figure C.25. Vision Backdoor Target Identification: Mean target identification AUPRC across 15 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns and randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. $|\mathcal{D}_{\text{adv}}| = 150$. Notation $y_{\text{targ}} \rightarrow y_{\text{adv}}$. See Table C.33 for the numerical results.

Table C.33. **Vision Backdoor Target Identification:** Target identification mean AUPRC across 15 trials for Weber et al.’s [Web+23] three CIFAR10 backdoor attack patterns and randomly selected reference \hat{z}_{targ} . All experiments performed binary classification on randomly-initialized ResNet9. Bold denotes the best mean performance. Mean results are shown graphically in Figures 16 and C.25. Variance results appear in the original paper [HL22a, Sec. F.1.2].

Classes $y_{\text{targ}} \rightarrow y_{\text{adv}}$	Trigger Pattern	Ours		Baselines				
		GAS	GAS-L	Max k -NN	Min k -NN	Most Certain	Least Certain	Random
Auto \rightarrow Dog	1 Pixel	0.998	0.998	0.996	0.065	0.475	0.101	0.135
	4 Pixel	0.999	0.998	0.987	0.062	0.263	0.105	0.116
	Blend	0.987	1	0.429	0.275	0.092	0.192	0.142
Plane \rightarrow Bird	1 Pixel	0.925	0.970	0.938	0.063	0.662	0.095	0.155
	4 Pixel	0.987	0.992	0.849	0.070	0.631	0.099	0.135
	Blend	0.782	0.979	0.300	0.097	0.119	0.153	0.141

Table C.34. **Vision Backdoor Attack Mitigation:** Bold denotes the best mean performance with 15 trials per setup. Aggregated results are shown in Table 6.

Classes $y_{\text{targ}} \rightarrow y_{\text{adv}}$	Attack	Method	% Removed		ASR %		Test Acc. %	
			\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
Auto \rightarrow Dog	1 Pixel	GAS	92.6	0.28	87.7	0	98.8	0.0
		GAS-L	94.4	0.08		0		0.0
	4 Pixel	GAS	92.8	0.26	95.0	0	98.9	0.0
		GAS-L	92.6	0.05		0		0.0
	Blend	GAS	99.9	0.67	98.6	0	99.0	-0.1
		GAS-L	100	0.41		0		-0.1
Plane \rightarrow Bird	1 Pixel	GAS	65.8	0.66	80.8	0	93.5	-0.1
		GAS-L	75.5	0.84		0		0.0
	4 Pixel	GAS	92.7	0.39	89.0	0	93.5	0.0
		GAS-L	95.2	0.80		0		0.0
	Blend	GAS	81.9	1.39	92.0	0	93.7	-0.4
		GAS-L	95.9	1.55		0		-0.5

C.3.3 Natural Language Poisoning Full Results.

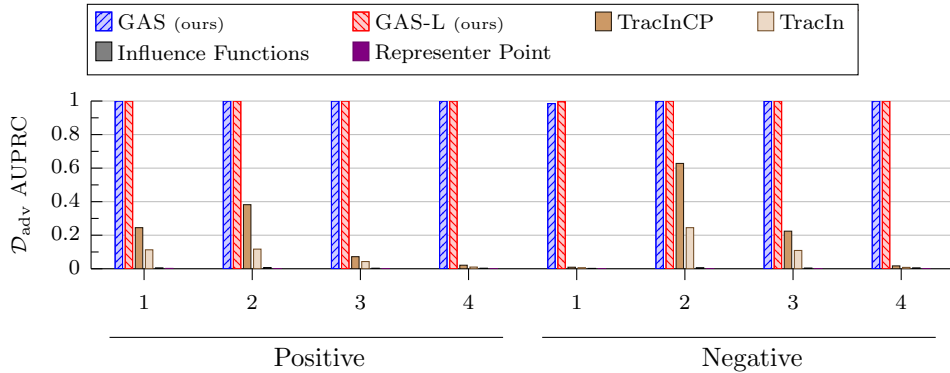


Figure C.26. Natural Language Poisoning Adversarial-Set Identification: See Table C.35 for the numerical results.

Table C.35. Natural Language Poisoning Adversarial-Set Identification: Poison identification mean AUPRC across 10 trials for 4 positive and 4 negative sentiment SST-2 movie reviews [Soc+13] with $|\mathcal{D}_{\text{adv}}| = 50$. GAS-L perfectly identified all poison in all but one trial. Bold denotes the best mean performance. Mean results are shown graphically in Figures 14 and C.26. Variance results appear in the original paper [HL22a, Sec. F.1.3].

Review		Ours		Baselines			
Sentiment	No.	GAS	GAS-L	TracInCP	TracIn	Inf. Func.	Rep. Pt.
↑ Positive	1	1	1	0.245	0.113	0.005	0.002
	2	1	1	0.382	0.117	0.007	0.001
	3	1	1	0.072	0.043	0.003	0.001
	4	1	1	0.021	0.010	0.003	0.001
↑ Negative	1	0.985	0.996	0.009	0.006	0.002	0.001
	2	1	1	0.628	0.245	0.006	0.001
	3	0.998	1	0.224	0.109	0.004	0.001
	4	1	1	0.017	0.008	0.005	0.001

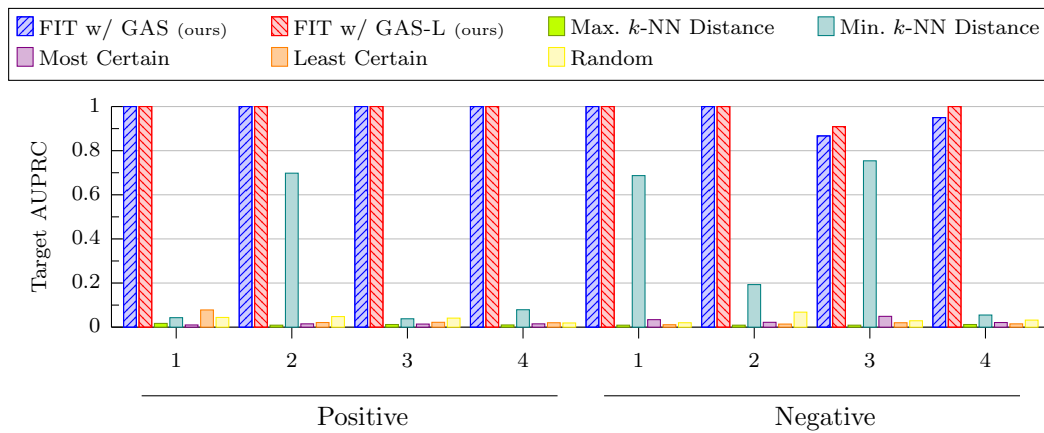


Figure C.27. **Natural Language Poisoning Target Identification:** See Table C.36 for the numerical results.

Table C.36. **Natural Language Poisoning Target Identification:** Bold denotes the best mean performance with 10 trials per review. Mean results are shown graphically in Figures 16 and C.27. Variance results appear in the original paper [HL22a, Sec. F.1.3].

Review		Ours		Baselines				
Sentiment	No.	GAS	GAS-L	Max k -NN	Min k -NN	Most Certain	Least Certain	Random
↑ Positive ↓	1	1	1	0.017	0.043	0.010	0.078	0.044
	2	1	1	0.009	0.698	0.015	0.021	0.048
	3	1	1	0.012	0.038	0.014	0.022	0.041
	4	1	1	0.010	0.079	0.015	0.020	0.019
↑ Negative ↓	1	1	1	0.009	0.687	0.034	0.011	0.020
	2	1	1	0.009	0.193	0.022	0.014	0.068
	3	0.867	0.909	0.009	0.754	0.049	0.020	0.029
	4	0.950	1	0.012	0.055	0.021	0.015	0.032

Table C.37. **Natural Language Poisoning Attack Mitigation:** Bold denotes the best mean performance with 10 trials per review. Aggregated results are shown in Table 6.

Review		Method	% Removed		ASR %		Test Acc. %	
Sentiment	No.		\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
↑ Positive ↓	1	GAS	100	0.01	100	0	94.1	0.0
		GAS-L	100	0.01				
	2	GAS	100	0.01	100	0	94.2	+0.1
		GAS-L	100	0.02				
	3	GAS	100	0.01	100	0	94.2	+0.1
		GAS-L	100	0.00				
	4	GAS	99.9	0	100	0	94.3	-0.1
		GAS-L	100	0.02				
↑ Negative ↓	1	GAS	97.1	0.01	100	0	94.3	+0.3
		GAS-L	99.5	0.02				
	2	GAS	100	0.17	100	0	94.3	+0.1
		GAS-L	100	0.04				
	3	GAS	99.5	0.05	90	0	94.1	+0.3
		GAS-L	100	0.01				
	4	GAS	100	0	100	0	94.2	+0.1
		GAS-L	100	0				

C.3.4 Vision Poisoning Full Results.

Section 6.5.2 considers Peri et al.’s [Per+20] dedicated, clean-label poison defense Deep k -NN as an additional baseline. By default, nearest neighbor algorithms yield a label, not a score. To be compatible with AUPRC, we modified Deep k -NN to rank each training example by the difference between the size of the neighborhood’s plurality class and the number of neighborhood instances that share the corresponding example’s label.

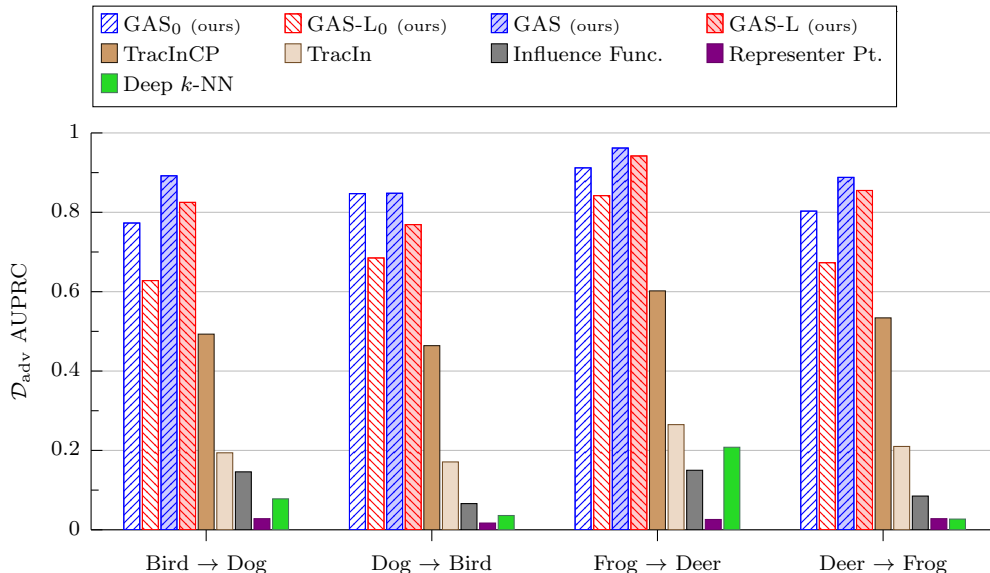


Figure C.28. **Vision Poisoning Adversarial-Set Identification:** Adversarial set (\mathcal{D}_{adv}) identification mean AUPRC across >15 trials for four CIFAR10 class pairs with $|\mathcal{D}_{\text{adv}}| = 50$. Our renormalized influence estimators, GAS and GAS-L, using just initial parameters θ_0 and with 5 subepoch checkpointing outperformed all baselines for all class pairs.

Table C.38. **Vision Poisoning Adversarial-Set Identification:** Adversarial set (\mathcal{D}_{adv}) identification mean AUPRC across >15 trials for four CIFAR10 class pairs with $|\mathcal{D}_{\text{adv}}| = 50$. Our renormalized influence estimators, GAS and GAS-L, using just initial parameters θ_0 and with 5 subepoch checkpointing outperformed all baselines for all class pairs. Bold denotes the best mean performance. Mean results are shown graphically in Figure 14 and C.28. Variance results appear in the original paper [HL22a, Sec. F.1.4].

Classes $y_{\text{targ}} \rightarrow y_{\text{adv}}$	Ours				Baselines				
	GAS ₀	GAS-L ₀	GAS	GAS-L	TracInCP	TracIn	Inf. Func.	Rep. Pt.	Deep k -NN
Bird \rightarrow Dog	0.773	0.628	0.892	0.825	0.493	0.194	0.146	0.028	0.078
Dog \rightarrow Bird	0.847	0.685	0.848	0.769	0.464	0.171	0.066	0.017	0.036
Frog \rightarrow Deer	0.912	0.842	0.962	0.942	0.602	0.265	0.150	0.026	0.208
Deer \rightarrow Frog	0.803	0.673	0.888	0.855	0.534	0.210	0.085	0.028	0.027

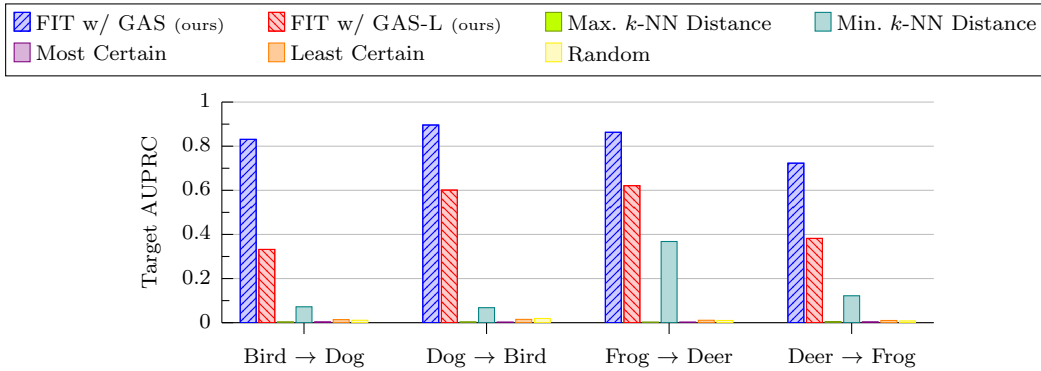


Figure C.29. **Vision Poisoning Target Identification:** See Table C.39 for the numerical results.

Table C.39. **Vision Poisoning Target Identification:** Bold denotes the best mean performance with ≥ 15 trials per class pair. Mean results are shown graphically in Figures 16 and C.29. Variance results appear in the original paper [HL22a, Sec. F.1.4].

Classes		Ours		Baselines				
y_{targ}	y_{adv}	GAS	GAS-L	Max k -NN	Min k -NN	Most Certain	Least Certain	Random
Bird	Dog	0.831	0.332	0.004	0.072	0.004	0.014	0.011
Dog	Bird	0.896	0.601	0.004	0.068	0.003	0.015	0.008
Frog	Deer	0.863	0.621	0.003	0.368	0.003	0.011	0.019
Deer	Frog	0.723	0.382	0.005	0.122	0.004	0.010	0.010

Table C.40. **Vision Poisoning Attack Mitigation:** Bold denotes the best mean performance with ≥ 15 trials per class pair. Aggregated results are shown in Table 6.

Classes		Method	% Removed		ASR %		Test Acc. %	
y_{targ}	y_{adv}		\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
Bird	Dog	GAS	72.1	0.04	91.4	0	87.0	0.0
		GAS-L	65.2	0.04				0
Dog	Bird	GAS	54.1	0.01	80.0	0	87.1	-0.1
		GAS-L	46.6	0.01				0
Frog	Deer	GAS	36.0	0.01	60.0	0	87.1	0.0
		GAS-L	30.2	0.03				0
Deer	Frog	GAS	88.9	0.03	80.0	0	87.0	+0.1
		GAS-L	83.5	0.03				0

C.4 Convex Polytope Poisoning and GAS Joint Optimization

Zhu et al. [Zhu+19] prove that under specific assumptions (e.g., a linear classifier), an adversarial set is guaranteed to alter a model’s prediction on a target if that target’s feature vector lies inside a convex polytope of the adversarial instances’ feature vectors.

Overview of Zhu et al.’s Attack Intuitively, Zhu et al.’s attack attempts to construct a convex hull of poison instances around a target – all within feature space. By design, deep models are non-linear and non-convex, so Zhu et al.’s underlying assumption does not directly apply. However, the convex-polytope attack will succeed if the trained model’s penultimate feature representation (i.e., the input into the final, linear classification layer) forms a convex hull around the target’s penultimate representation.

To that end, Zhu et al.’s iterative, bilevel poison optimization considers solely this feature representation. In the attacker’s ideal case, the adversarial set’s feature-space representation would be optimized w.r.t. the final trained model. However, attackers do not know training’s random seed. Moreover, any change to a training instance necessarily affects the final model parameters (and thus the penultimate feature representation as well), inducing a cyclic dependency that makes poison crafting non-trivial.

To increase the likelihood that the attack succeeds, Zhu et al. optimize the poison’s feature-space representation across a suite of m surrogate models. For each model $f^{(j)}$ ($j \in \{1, \dots, m\}$), denote the model’s penultimate-feature extraction function as $\phi^{(j)}(\cdot)$.

Zhu et al. specify a bilevel optimization to iteratively form these feature-space convex hulls, where adversarial set $\mathcal{D}_{\text{adv}} := \{(x_l, y_{\text{adv}})\}_{l=1}^K$ is crafted from a set of K *clean* seed instances, denoted $\{x_l^{\text{cl}}\}_{l=1}^K$. Zhu et al. restrict the adversarial perturbations to an ℓ_∞ ball of radius ε around those clean seed instances. The feature-space convex

hull requirement is enforced coefficients via $c_l^{(j)} \geq 0$. Zhu et al.’s bilevel optimization is reproduced in Eq. (C.1), with an additional term, $\beta \widehat{\text{GAS}}$, that is explained below. Note that in Zhu et al.’s formulation $\beta = 0$.

$$\begin{aligned}
& \min_{\{c_l^{(j)}\}, \mathcal{D}_{\text{adv}}} && \beta \widehat{\text{GAS}} + \frac{1}{2} \sum_{j=1}^m \frac{\left\| \phi^{(j)}(\mathbf{x}_{\text{targ}}) - \sum_{l=1}^K c_l^{(j)} \phi^{(j)}(x_l) \right\|^2}{\left\| \phi^{(j)}(\mathbf{x}_{\text{targ}}) \right\|^2} \\
& \text{s.t.} && \sum_{l=1}^K c_l^{(j)} = 1, \quad \forall j \\
& && c_l^{(j)} \geq 0, \quad \forall l, j \\
& && \|x_l - x_l^{\text{cl}}\|_{\infty} \leq \varepsilon, \quad \forall l.
\end{aligned} \tag{C.1}$$

Joint Optimization Formulation An attacker may attempt to evade our defense by optimizing adversarial set \mathcal{D}_{adv} to *appear* uninfluential on target $\widehat{z}_{\text{targ}}$.¹ Eq. (C.1) formalizes this idea by simultaneously optimizing for both poison effectiveness as well as for low GAS influence where hyperparameter $\beta > 0$ trades off between these two sub-objectives. Following Zhu et al.’s [Zhu+19] paradigm as described above, the attacker uses surrogate models to estimate the GAS influence.

Specifically, the adversary trains a *gray-box model*² using the same architecture, hyperparameters, clean training data (\mathcal{D}_{cl}), and pre-trained parameters as the target model. The surrogate set is then formed from m model checkpoints evenly spaced across this gray-box training. This quantity then estimates the GAS influence in the final trained model. Formally,

$$\widehat{\text{GAS}} := \sum_{j=1}^m \sum_{l=1}^K \frac{\langle g_l^{(j)}, \widehat{g}_{\text{targ}}^{(j)} \rangle}{\|g_l^{(j)}\| \|\widehat{g}_{\text{targ}}^{(j)}\|}. \tag{C.2}$$

¹ \mathcal{D}_{adv} must remain *actually* influential. Otherwise, the model’s prediction on $\widehat{z}_{\text{targ}}$ would not change.

²Gray-box attacks assume the attacker has access to detailed (but not complete) information about the target model like our case above.

It is important to note that optimizations like Eq. (C.1) create an implicit tension. Training-set attacks commonly attempt to make \mathcal{D}_{adv} and $\widehat{z}_{\text{targ}}$ have similar feature-space representations [Sha+18; Zhu+19; Hua+20; Wal+21]. Since each example’s features inform the model gradients (g), appearing less influential can affect the attack’s effectiveness.

Practical Challenges of Joint Optimization In modern neural networks, each parameter only directly affects or is directly affected by a subset of the other parameters – specifically those in adjacent layers. This limited interdependency makes back-propagation more tractable and efficient.

Recall that GAS normalizes by the gradient magnitude. When calculating influence in practice, this does not change the memory or computational complexity. However, when trying to optimize surrogate $\widehat{\text{GAS}}$ (Eq. (C.2)), normalizing by the gradient magnitude creates pairwise dependencies between all parameters, i.e., $\Theta(|\theta|^2)$ memory complexity for automatic differentiation systems. Therefore, renormalized estimators like GAS are significantly more memory intensive to optimize against in practice than the baseline influence estimators where this quadratic memory complexity is not induced.

Section 6.6’s experiments were affected by joint optimization’s increased memory complexity, where the GPU VRAM requirements increased by $\geq 12\times$. This created significant issues even for the comparatively small ResNet9 neural network [Pag20].

For example, when the adversarial set size was larger than 40, joint optimization exceeded the GPU VRAM memory capacity.³ In contrast, our original paper includes an ablation study tests more than 400 poison samples for Zhu et al.’s baseline attack

³Experiments were performed on Nvidia Tesla K80 GPUs with 11.5GB of VRAM.

using the same hardware [HL22a]. Furthermore, joint optimization’s larger memory footprint necessitated that only a small number of surrogate checkpoints could be used – specifically four checkpoints. This then increases the coarseness of $\widehat{\text{GAS}}$ ’s influence estimate.

Setting Joint Optimization Hyperparameter β As detailed above, hyperparameter β induces a trade-off between Zhu et al.’s convex-polytope loss and the surrogate GAS estimate. Section 6.6’s “baseline” results used $\beta = 0$. To ensure a strong adversary, Section 6.6’s “Adaptive Joint Optimization Attack with GAS” results used $\beta = 10^{-2}$ since that was the largest value of β that did not result in a significant drop in attacker success rate as detailed in Table C.41.

Table 6 in Section 6.5.4 reports that the vision poisoning’s attack success rate was 77.9%. Even when $\beta = 0$ (i.e., the surrogate GAS loss is ignored), there was still a substantial decrease in ASR to 64.3%. Recall that joint optimization’s memory complexity is $\Theta(|\theta|^2)$ which necessitated using fewer surrogate models (due to GPU VRAM capacity). This, in turn, degraded attack performance.⁴ Put simply, joint adversarial set optimization is *not necessarily a free lunch*. It may come at the cost of a worse attacker success rate.

Table C.41. Effect of joint-optimization hyperparameter β on the attacker’s success rate (ASR). Observe that even at $\beta = 0$, the attack success rate is significantly lower than the 77.9% ASR in Table 6 due to the fewer surrogate models that could be used during jointly-optimized poison crafting as explained above.

β	ASR (%)
0	64.3
10^{-2}	63.1
$2 \cdot 10^{-2}$	50.0
10^{-1}	4.8

⁴We separately verified that reducing the adversarial-set size from 50 to 40 did not meaningfully change the ASR.

Section 6.6 summarizes the adversarial-set and target identification results for this jointly-optimized attack. Sections C.4.1 and C.4.2 (resp.) provide more granular versions of those results.

Section C.4.3 provides additional results on target-driven attack mitigation’s effectiveness on this jointly optimized attack.

C.4.1 Adversarial-Set Identification of the Jointly Optimized Poisoning Attack.

Table C.42. **Adversarial-Set Identification for the Adaptive Vision Poison Attack:** Adversarial-set identification mean AUPRC with ≥ 10 trials per setup as described in Section C.4. Section 6.6’s **baseline** results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. Bold denotes the best mean performance. Mean results are shown graphically in Figures 17 and C.30. Variance results appear in the original paper [HL22a, Sec. F.2.1].

Param.	Classes	Ours				Baselines			
β	$y_{\text{targ}} \rightarrow y_{\text{adv}}$	GAS ₀	GAS-L ₀	GAS	GAS-L	TracInCP	TracIn	Inf. Func.	Rep. Pt.
0	Bird \rightarrow Dog	0.567	0.418	0.766	0.690	0.275	0.085	0.081	0.032
	Dog \rightarrow Bird	0.663	0.532	0.660	0.560	0.272	0.098	0.035	0.017
	Frog \rightarrow Deer	0.755	0.680	0.827	0.787	0.393	0.135	0.079	0.020
	Deer \rightarrow Frog	0.610	0.477	0.669	0.617	0.243	0.119	0.059	0.018
10^{-2}	Bird \rightarrow Dog	0.611	0.470	0.646	0.590	0.282	0.093	0.067	0.026
	Dog \rightarrow Bird	0.708	0.553	0.558	0.479	0.180	0.072	0.030	0.014
	Frog \rightarrow Deer	0.823	0.753	0.858	0.818	0.404	0.173	0.077	0.021
	Deer \rightarrow Frog	0.790	0.625	0.660	0.640	0.189	0.106	0.063	0.022

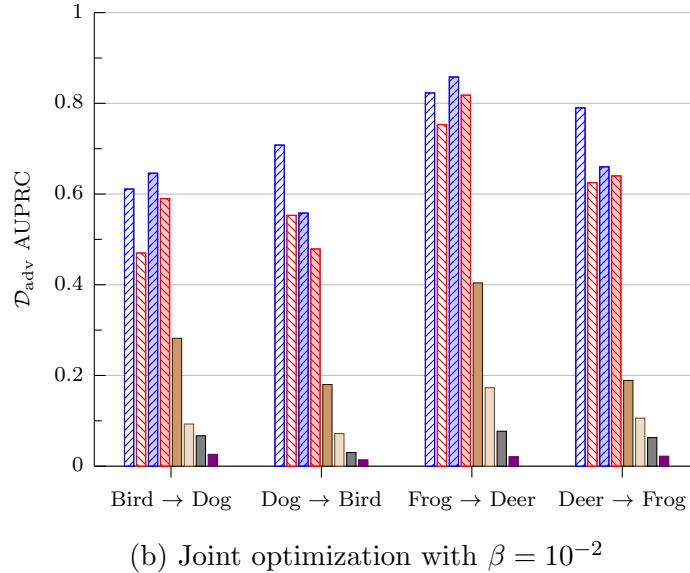
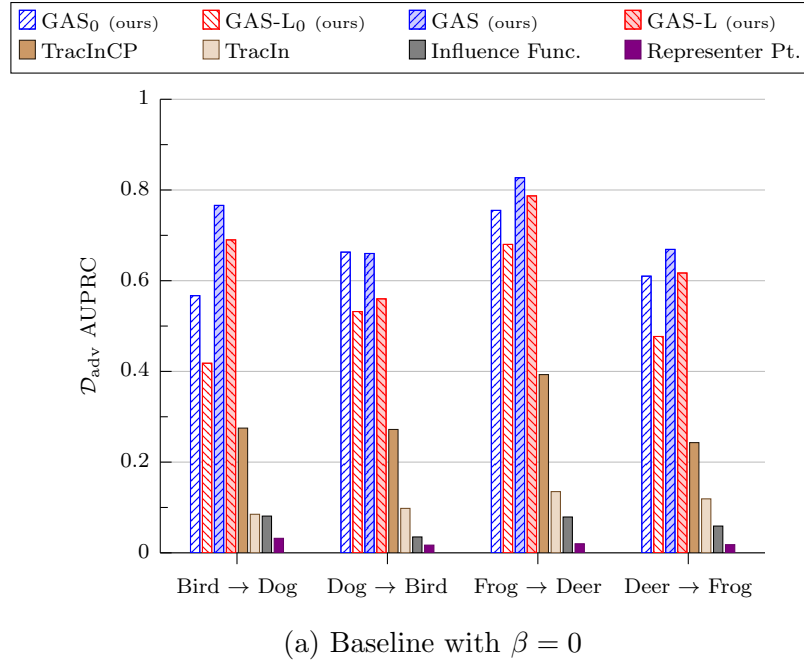


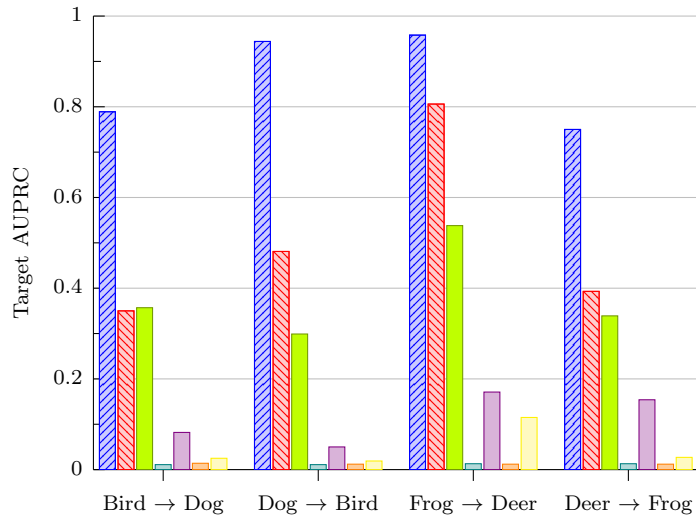
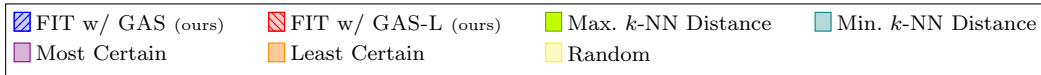
Figure C.30. Adversarial-Set Identification for the Adaptive Vision Poison Attack: Mean AUPRC identifying the adversarial set where Zhu et al.’s vision poison attack is jointly optimized with minimizing GAS with ≥ 10 trials per setup as described in Section C.4. Section 6.6’s baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. This joint optimization reduces the GAS similarity by 7% at the cost of a 19% decrease in ASR w.r.t. Table 6. See Table C.42 (below) for the numerical results.

C.4.2 Target Identification of the Jointly Optimized Poisoning

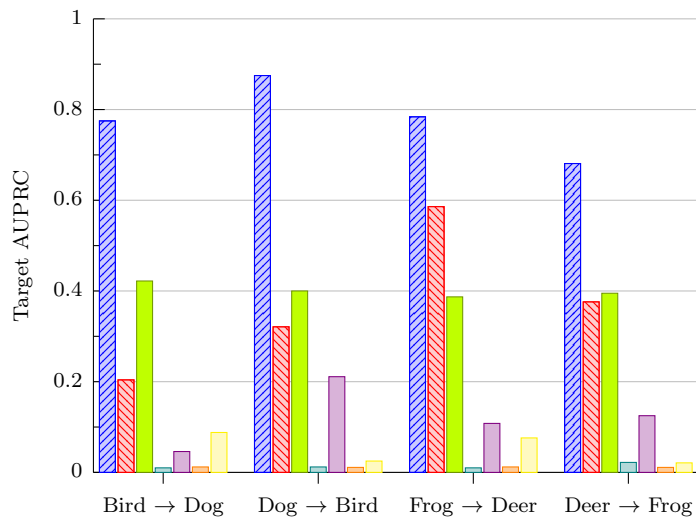
Attack.

Table C.43. **Target Identification for the Adaptive Vision Poison Attack:** Target identification mean AUPRC where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing GAS. Section 6.6’s **baseline** results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. Bold denotes the best mean performance with ≥ 10 trials per class pair. Mean results are shown graphically in Figures 18 and C.31. Variance results appear in the original paper [HL22a, Sec. F.2.2].

Param.	Classes		Ours		Baselines				
β	y_{targ}	y_{adv}	GAS	GAS-L	Max k -NN	Min k -NN	Most Certain	Least Certain	Random
0	Bird	Dog	0.789	0.350	0.357	0.011	0.082	0.014	0.025
	Dog	Bird	0.944	0.481	0.299	0.011	0.050	0.012	0.019
	Frog	Deer	0.958	0.806	0.538	0.013	0.171	0.012	0.115
	Deer	Frog	0.750	0.393	0.339	0.013	0.154	0.012	0.027
10^{-2}	Bird	Dog	0.775	0.204	0.422	0.010	0.046	0.012	0.088
	Dog	Bird	0.875	0.321	0.400	0.012	0.211	0.011	0.025
	Frog	Deer	0.784	0.586	0.387	0.010	0.108	0.012	0.076
	Deer	Frog	0.681	0.376	0.395	0.022	0.125	0.011	0.021



(a) Baseline with $\beta = 0$



(b) Joint optimization with $\beta = 10^{-2}$

Figure C.31. Target Identification for the Adaptive Vision Poison Attack: Mean target identification AUPRC where Zhu et al.'s [Zhu+19] vision poison attack is jointly optimized with minimizing GAS. Section 6.6's baseline results set trade-off hyperparameter $\beta = 0$, meaning the poison was not jointly optimized. The jointly optimized results used $\beta = 10^{-2}$ as explained in suppl. Section C.4. See Table C.43 (below) for the numerical results.

C.4.3 Target-Driven Attack Mitigation of the Jointly Optimized Poisoning Attack.

This section examines joint optimization’s effect on target-driven mitigation. Averaging across all class pairs, target-driven mitigation using GAS and GAS-L removed 0.05% and 0.03% (resp.) of the clean training data (\mathcal{D}_{cl}). For comparison, Zhu et al.’s [Zhu+19] baseline attack removed on average 0.02% and 0.03% of clean training data for GAS and GAS-L respectively (see Table 6). Moreover, after mitigating this jointly-optimized attack, average test accuracy either improved or stayed the same in all but one case.

Table C.44. **Target-Driven Attack Mitigation for the Adaptive Vision Poison Attack:** Algorithm 6’s target-driven data sanitization where Zhu et al.’s [Zhu+19] vision poison attack is jointly optimized with minimizing the GAS influence. The results below consider exclusively the jointly-optimized attack with $\beta = 10^{-2}$. Clean-data removal remains low, and test accuracy either improved or stayed the same for in but one setup. The performance is comparable to the results with Zhu et al.’s [Zhu+19]’s standard vision poisoning attack (see Table C.40). Bold denotes the best mean performance with ≥ 10 trials per class pair.

Classes		Method	% Removed		ASR %		Test Acc. %	
y_{targ}	y_{adv}		\mathcal{D}_{adv}	\mathcal{D}_{cl}	Orig.	Ours	Orig.	Chg.
Bird	Dog	GAS	36.0	0.02	76.2	0	87.0	+0.1
		GAS-L	30.3	0.00				+0.1
Dog	Bird	GAS	21.6	0.00	57.1	0	87.1	+0.1
		GAS-L	21.9	0.00				-0.1
Frog	Deer	GAS	17.5	0.00	38.1	0	87.1	0.0
		GAS-L	19.4	0.00				0.0
Deer	Frog	GAS	85.0	0.18	81.0	0	87.1	0.0
		GAS-L	82.3	0.13				+0.1

APPENDIX D

EVALUATION SETUPS

This chapter contains previously published, coauthored material [HL21; HL22a; HL23c; HL23a]. Hammoudeh wrote this complete section and designed the experiments. Lowd provided supervision, editorial suggestions, and input on experiment design.

Zayd Hammoudeh and Daniel Lowd. “Simple, Attack-Agnostic Defense Against Targeted Training Set Attacks Using Cosine Similarity”. In: *Proceedings of the 3rd ICML Workshop on Uncertainty and Robustness in Deep Learning*. UDL’21. 2021

Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security*. CCS’22. Los Angeles, CA: Association for Computing Machinery, 2022. URL: <https://arxiv.org/abs/2201.10055>

Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning*. SaTML’23. 2023. URL: <https://arxiv.org/abs/2208.13904>

Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023. URL: <https://arxiv.org/abs/2302.11628>

This chapter details the evaluation setup for the experiments Chapters 4, 5, and 6. Since each chapter has a different evaluation setup, we separate the setup of each chapter in a different section below.

D.1 Evaluation Setup for the Experiments in Chapter 4

This section details the evaluation setup used in Section 4.8’s experiments, including implementation details, dataset configuration, and hyperparameter settings. Chapter 4’s source code can be downloaded from <https://github.com/ZaydH/certified-regression>. All experiments were implemented and tested in Python 3.7.1. Experiments were performed using one core of a fourteen-core Intel E5-2690v4 CPU and 12GB of RAM. Ridge regression models were trained using Scikit-Learn [Ped+11], while the decision forests used the XGBoost library [CG16].

The overlapping regressor ILPs (Fig. 6) were optimized using Gurobi [Gur22] with a time limit of 1200s.

D.1.1 Dataset Configuration. Chapter 4’s source code automatically downloads all necessary datasets. Regarding dataset preprocessing, categorical features were transformed into one-hot-encoded features in line with previous work [BHL23]. Standardizing features by dataset mean/variance breaks submodel independence and so was not performed. Minimal manual feature engineering was performed to improve the housing datasets’ results, e.g., adding a home’s age, total square feet, total number of bathrooms, etc.; this feature engineering was done based on existing features in the dataset (e.g., total square feet equals the sum of the first and second-floor square footage). None of the engineered features affect submodel independence.

Most of the six datasets in Sec. 4.8.1 do not have a dedicated test set. In such cases, the data was split 90%/10% at random between training and test.

When training k NN-CR models, each feature dimension was normalized to the range $[0, 1]$. Without feature normalization, k NN-CR generally prioritizes whichever feature has the largest magnitude. This transformation implicitly restricts arbitrary insertions to the feature range in the original dataset. Such normalization is implicitly

done in certified classifier evaluation on image datasets where each pixel has a consistent, fixed range.

D.1.2 Dataset Target Value Statistics. Table D.45 summarizes the test set’s target (y) value distribution statistics for Sec. 4.8’s five regression datasets.

Recall from Table 1 that the Ames, Austin, and Diamonds datasets set error threshold ξ as a fixed percentage of y_{te} . This choice was made because these three datasets exhibit significant y variance. For example, for Diamonds, the largest y value (\$18.8k) is about two orders of magnitude larger than the smallest y value (\$339). Using a fixed ξ value on these three datasets would have made certifying instances with small y unrealistically easy while making certification of instances with large y unreasonably difficult. Making the error threshold a fraction of y_{te} allows the certification difficulty to be more consistent across the range of y values.

Datasets Weather and Life used fixed ξ values of 3 degrees (Celsius) and 3 years respectively. Both of these threshold values are less than one-third of each dataset’s y standard deviation.

In the original paper [HL23c], we evaluate the performance of our certified regressors on additional ξ values – both larger and smaller than the ξ values used in Sec. 4.8.

Table D.45. **Target Value Test Distribution Statistics:** Mean (\bar{y}), standard deviation (σ_y), minimum value (y_{\min}) and maximum value (y_{\max}) for the test instances’ target y value for Sec. 4.8’s five regression datasets.

Dataset	\bar{y}	σ_y	y_{\min}	y_{\max}
Ames	\$184k	\$83.4k	\$12.8k	\$585k
Austin	\$466k	\$266k	\$81.0k	\$2.6M
Diamonds	\$3.8k	\$3.9k	\$0.3k	\$18.8k
Weather	14.9°C	10.3°C	−44.0°C	54.0°C
Life	69.3 years	9.6 years	36.3 years	89.0 years

D.1.3 Hyperparameters. Following Jia et al.’s [Jia+22a] certified k NN classifier evaluation, k NN-CR’s neighborhood size, k , was set to the (larger) odd integer nearest to $\frac{n}{2}$. We use the Minkowski distance as the neighborhood’s distance metric.

For our ensemble regressors, hyperparameters were tuned using Bayesian optimization as implemented in the `scikit-optimize` library [Hea+21]. The partitioned and overlapping certified regressors (unweighted and weighted) used the same hyperparameter settings.

Ridge Regression Hyperparameters For three datasets – Diamonds [Wic16], Weather [Mal+21], and Spambase [Hop+17] – our four ensemble regressors used ridge regression as the submodel architecture. For each dataset and q value, we tuned three ridge regression hyperparameters. Below, we list those hyperparameters along with the set of values considered.

- *Weight Decay* (λ): L_2 regularization strength. We considered values between 10^{-8} and 10^4 .
- *Error Tolerance* (ε): Minimum validation error that defines when a model is considered converged. The tested values were $\{10^{-8}, 10^{-7}, \dots, 10^{-3}\}$.
- *Maximum Number of Iterations* (# Itr.): Defines the maximum number of optimizer iterations. If the error tolerance is achieved before the iteration count is met, the model is treated as converged, and optimization stops. The tested values were $\{10^2, 10^3, \dots, 10^8\}$.

Table D.46 lists the final hyperparameters for each experimental setup that used ridge regression as the submodel architecture.

XGBoost Hyperparameters For three datasets – Ames Housing [Coc11], Austin Housing [Pie21], and Life [Raj21] – our four ensemble regressors used XGBoost [CG16] as the submodel architecture. For each dataset and q value, we tuned seven XGBoost hyperparameters. Below, we list those hyperparameters along with the set of values considered.

- *Number of Trees* (τ): Number of trees in the ensemble. The tested values were $\{50, 100, 250, 500, 1000\}$.
- *Maximum Tree Depth* (h): Maximum depth of each tree in the ensemble. The tested values were $\{1, \dots, 4\}$.
- *Evaluation Metric* (\mathcal{L}): Applied to the validation set and is the metric being minimized. The tested values were root mean squared error (RMSE) and mean absolute error (MAE).
- *Weight Decay* (λ): L_2 regularization strength. We considered values between 10^{-3} and 10^5 .
- *Minimum Split Loss* (γ): Minimum reduction in loss required to split a node instead of making it a leaf. The values considered were $\{0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 1\}$.
- *Learning Rate* (η): Larger value makes the boosting more conservative. The tested values were $\{0.01, 0.1, 0.3, 1\}$.

Table D.47 lists the final hyperparameters for each experimental setup that used XGBoost as the submodel architecture. Mixup [Zha+18] data augmentation was used to improve XGBoost’s performance.¹

¹Mixup does not apply to convex models like ridge regression.

Table D.46. **Ridge Regression Hyperparameters:** Hyperparameter settings for the three datasets that used ridge regression as the ensemble submodel architecture. Hyperparameters are reported for the three q values used in Fig. 7 and Sec. C.1. We also report the hyperparameters for uncertified accuracy when $q = 1$.

Dataset	q	λ	ϵ	# Itr.
Diamonds	1	3.16E−3	1E−6	1E6
	151	6.01E−2	1E−7	1E8
	501	1.00E−8	1E−6	1E8
	1001	1.38E−8	1E−6	1E2
Weather	1	3.16E−3	1E−8	1E7
	51	1.00E+3	1E−5	1E6
	1501	3.16E+2	1E−6	1E2
	3001	3.16E+2	1E−6	1E3
Spambase	1	3.16E+2	1E−6	1E5
	25	3.16E−3	1E−6	1E6
	151	3.16E−6	1E−7	1E6
	301	3.16E−3	1E−6	1E6

D.2 Evaluation Setup for the Experiments in Chapter 5

This section details the evaluation setup used in Section 5.5’s experiments, including implementation details, dataset configuration, and hyperparameter settings.

Our source code can be downloaded from <https://github.com/ZaydH/feature-partition>. All experiments were implemented and tested in either Python 3.7.13 or 3.10.10. All neural networks were implemented in PyTorch version 1.12.0 [Pas+19]. LightGBM decision forests were trained using the official `lightgbm` Python module, version 3.3.3.99 [Ke+17].

D.2.1 Hardware Setup. Experiments were performed on a desktop system with a single AMD 5950X 16-core CPU, 64GB of 3200MHz DDR4 RAM, and a single NVIDIA 3090 GPU.

D.2.2 Baselines. To the extent of our knowledge, no existing method considers certified feature robustness guarantees (Def. 5.1). *Randomized ablation* – our most closely related method – considers ℓ_0 -norm certified robustness (Def. 5.2) [LF20b]. RA is a specialized form of randomized smoothing [CRK19; LXL23] targeted

Table D.47. **XGBoost Hyperparameters**: Hyperparameter settings for the three datasets that used XGBoost as the ensemble submodel architecture. Hyperparameters are reported for the three q values used in Fig. 7 and Sec. C.1. We also report the hyperparameters for uncertified accuracy when $q = 1$.

Dataset	q	τ	h	\mathcal{L}	λ	γ	η
Ames Housing	1	250	2	RMSE	1E-1	5E-3	0.3
	25	500	2	MAE	1E-3	5E-3	0.3
	125	500	3	RMSE	1E-2	5E-3	1.0
	251	250	1	RMSE	1E-1	5E-3	1.0
Austin Housing	1	500	4	MAE	1E+2	1E-2	0.3
	151	1000	1	RMSE	1E-2	5E-3	1.0
	301	250	1	MAE	1E+0	1E-2	1.0
	701	250	1	MAE	1E-2	1E-2	1.0
Life	1	500	5	RMSE	1E+1	1E-2	0.1
	25	250	4	RMSE	0E+0	5E-2	0.3
	101	250	3	MAE	1E+0	1E-2	1.0
	201	250	4	RMSE	0E+0	5E-3	0.3

towards sparse evasion attacks. In terms of the state of the art, Jia et al. [Jia+22b] provide the tightest certification analysis for randomized ablation.

Recall that feature partition aggregation (FPA) provides strictly stronger certified guarantees than baseline RA. Put simply, FPA is solving a harder task than baseline randomized ablation. Therefore, when FPA achieves the same certified accuracy as the baseline, FPA is performing provably better, given FPA’s stronger guarantees.

We also compare FPA to three certified patch defenses, namely: (de)randomized smoothing (DRS) [LF20a], patch interval bound propagation (IBP) [Chi+20], and BAGCERT [MY21]. Note that BAGCERT’s implementation is not open source, and Metzen and Yatsura [MY21] have indicated they do not plan to open source the code in the future.² As such, BAGCERT’s results in the main paper were provided by Metzen and Yatsura via personal correspondence. BAGCERT’s closed source code prohibited the collection of its certification time. Nonetheless, comparing FPA’s certification

²The author’s comments regarding open-sourcing their code can be found on BAGCERT’s OpenReview page.

time to that of BAGCERT provides only limited insight since FPA and BAGCERT certify very different types of guarantees.

D.2.3 Datasets. Our empirical evaluation considers four datasets. First, MNIST [LeC+98] and CIFAR10 [KNH14] are vision classification datasets with 10 classes each.

Although all certified sparse defenses considered in this work are exclusively proposed in the context of classification, Hammoudeh and Lowd [HL23c] prove that certified regression *reduces* to voting-based certified classification. Hence, it is straightforward to transform FPA and randomized ablation into certified regression defenses. We reuse this reduction and evaluate two tabular regression datasets, Weather [Mal+21] and Ames [Coc11].

For Weather, we follow Hammoudeh and Lowd’s [HL23c] empirical evaluation, where the objective is to predict ground temperature within $\pm 3^\circ\text{C}$ using features that include the date, time of day, longitude, and latitude. Similarly, we follow Hammoudeh and Lowd’s [HL23c]’s empirical evaluation for Ames, where the objective is to predict a property’s sale price within $\pm 15\%$ of the actual price. Since ablated training requires a custom feature encoding to differentiate ablated and non-ablated features, min-max scaling was applied to both datasets’ features for RA to normalize all feature values to the range $[0, 1]$.

We chose these two regression datasets as a stand-in for vertically partitioned data, which are commonly tabular and particularly vulnerable to sparse backdoor and evasion attacks.

Table D.48 provides basic information about the four datasets, including their sizes and feature dimension. Table D.49 provides summary statistics for the regression datasets’ test target-value (i.e., y) distribution.

Table D.48. Evaluation dataset information

Dataset	# Classes	# Feats	# Train	# Test
CIFAR10	10	1,024	50,000	10,000
MNIST	10	784	60,000	10,000
Weather	N/A	128	3,012,917	531,720
Ames	N/A	352	2,637	293

Table D.49. **Target Value Test Distribution Statistics:** Mean (\bar{y}), standard deviation (σ_y), minimum value (y_{\min}) and maximum value (y_{\max}) for the test instances’ target y value for regression datasets Weather and Ames.

	\bar{y}	σ_y	y_{\min}	y_{\max}
Weather	14.9°C	10.3°C	-44.0°C	54.0°C
Ames	\$184k	\$83.4k	\$12.8k	\$585k

Our source code automatically downloads all necessary dataset files.

D.2.4 Network Architectures. Table D.50 details the CIFAR10 neural network architecture. Specifically, we follow previous work on CIFAR10 data poisoning [HL22a] and use Page’s [Pag20] ResNet9 architecture. ResNet9 is ideal for our experiments since it is very fast to train, as ranked on DAWNbench [Col+17]. ResNet9’s fast training significantly reduces the overhead of training L submodels for FPA.

We directly adapt Page’s [Pag20] published implementation³ including the use of ghost batch normalization [SD20] and the CELU activation function with $\alpha = 0.075$ [Bar17].

Three forms of data augmentation were also used in line with Page’s [Pag20] implementation. First, a random crop with four pixels of padding was performed. Next, the image was flipped horizontally with a 50% probability. Finally, a random 8×8 pixel portion of the image was randomly erased. Note that these transformations were

³Source code: <https://github.com/davidcpage/cifar10-fast>.

Table D.50. ResNet9 neural network architecture

	Conv1	In=3	Out=64	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=64			
	CELU				
	Conv2	In=64	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	CELU				
	MaxPool2D	2 × 2			
↑ ResNet1	ConvA	In=128	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	CELU				
↓	ConvB	In=128	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	CELU				
	Conv3	In=128	Out=256	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=256			
	CELU				
	MaxPool2D	2 × 2			
	Conv4	In=256	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	CELU				
	MaxPool2D	2 × 2			
↑ ResNet2	ConvA	In=512	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	CELU				
↓	ConvB	In=512	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	CELU				
	MaxPool2D	4 × 4			
	Linear	Out=10			

performed *after* the pixels were disabled in the image, meaning these transformations do not result in a network seeing additional pixel information.

In a separate paper, Levine and Feizi [LF21] propose *deep partition aggregation* (DPA), a certified defense against poisoning attacks. Here, we follow Levine and Feizi’s [LF21] public implementation⁴ and use the Network-in-Network (NiN)

⁴Source code: <https://github.com/alevine0/DPA>.

Table D.51. Network-in-Network neural network architecture

Block 1	Conv1	In=3	Out=192	Kernel=5 × 5	Pad=2
	BatchNorm2D	Out=192			
	ReLU				
	Conv2	In=192	Out=160	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=160			
	ReLU				
	Conv3	In=160	Out=96	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=96			
	ReLU				
MaxPool2D	3 × 3				
Block 2	Conv1	In=96	Out=192	Kernel=5 × 5	Pad=2
	BatchNorm2D	Out=192			
	ReLU				
	Conv2	In=192	Out=192	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=192			
	ReLU				
	Conv3	In=192	Out=192	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=192			
	ReLU				
AvgPool2D	3 × 3				
Block 3	Conv1	In=192	Out=192	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=192			
	ReLU				
	Conv2	In=192	Out=192	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=192			
	ReLU				
	Conv3	In=192	Out=192	Kernel=1 × 1	Pad=1
	BatchNorm2D	Out=192			
	ReLU				
GlobalAvgPool2D	Out=192				
Linear	Out=10				

architecture [LCY14] when evaluating our method on MNIST. Table D.51 visualizes the MNIST NiN architecture.

D.2.5 Hyperparameters. For simplicity, FPA used the same hyperparameter settings for a given dataset irrespective of L . Therefore, FPA’s results could be further improved in practice by tuning the hyperparameter settings to optimize the ensemble’s performance for a specific submodel count.

Table D.52 details the CIFAR10 and MNIST hyperparameter settings for feature partition aggregation.

Table D.52. FPA’s neural network training hyperparameters

	CIFAR10	MNIST
Data Augmentation?	✓	
Validation Split	N/A	5%
Optimizer	SGD	AdamW
Batch Size	512	128
# Epochs	80	25
Learning Rate (Peak)	$1 \cdot 10^{-3}$	$3.16 \cdot 10^{-4}$
Learning Rate Scheduler	One cycle	Cosine
Weight Decay (L_2)	$1 \cdot 10^{-1}$	$1 \cdot 10^{-3}$

For CIFAR10 and MNIST, we directly used Levine and Feizi’s [LF20b] published randomized ablation training source code, which includes pre-specified hyperparameter settings for the learning rate, weight decay, and optimizer hyperparameters.

Recall from Sec. 5.5 that for the Weather and Ames datasets, FPA’s submodels are LightGBM [Ke+17] gradient-boosted decision tree (GBDT) regressors. Table D.53 details FPA’s LightGBM hyperparameter settings. For a more direct comparison with randomized ablation which cannot use a GBDT, we also evaluated FPA with linear submodels. FPA’s linear submodel hyperparameter settings for the regression datasets are in Table D.54.

Table D.53. Regression datasets LightGBM submodel training hyperparameters

	Weather	Ames
Boosting Type	GBDT	GBDT
# Estimators	500	1,000
Max. Depth	10	6
Min. Child Samples	20	5
Max. # Leaves	127	127
L_1 Regularizer	0	$1 \cdot 10^{-3}$
L_2 Regularizer	0	$1 \cdot 10^2$
Objective	Huber	MAE
Learning Rate	0.5	$1 \cdot 10^2$
Subsampling	0.9	0.9

Table D.54. Regression datasets linear submodel training hyperparameters

	Weather	Ames
L_1 Regularizer	$3.16 \cdot 10^{-3}$	$4.15 \cdot 10^{-5}$
Max. # Iterations	$1 \cdot 10^4$	$1 \cdot 10^6$
Tolerance	$1 \cdot 10^{-3}$	$1 \cdot 10^{-8}$

Levine and Feizi [LF20b] only evaluate classification datasets in their original paper. As such, there are no existing hyperparameter settings for randomized ablation on Weather and Ames. We manually tuned randomized ablation’s learning rate for the regression datasets considering all values in the set $\{10^{-2}, 10^{-3}, 10^{-4}\}$. We also tested numerous different settings for the number of training epochs. To ensure a strong baseline, we report the best performing randomized ablation hyperparameter settings.

Recall from Sec. 5.2 that randomized ablation only provides probabilistic guarantees. By contrast, feature partition aggregation provides deterministic guarantees. To facilitate a more direct comparison between certified feature and ℓ_0 -norm guarantees, $\alpha = 0.0001$ in all experiments.

D.3 Evaluation Setup for the Experiments in Chapter 6

This section details the evaluation setup used in Section 6.3 and 6.5’s experiments, including dataset specifics, hyperparameters, and the neural network architectures.

Our source code can be downloaded from https://github.com/ZaydH/target_identification. All experiments used the PyTorch automatic differentiation framework [Pas+19] and were tested with Python 3.6.5. Wallace et al.’s [Wal+21] sentiment analysis data poisoning source code will be published by its authors at <https://github.com/Eric-Wallace/data-poisoning>.

D.3.1 Dataset Configurations. This subsection provides details related to dataset configurations.

Section 6.3.1 performs binary classification of `frog` vs. `airplane` from CIFAR10. Added as a small adversarial set (\mathcal{D}_{adv}) is 150 MNIST 0 training instances selected at random. We considered this class pair specifically since among the $\binom{10}{2}$ possible CIFAR10 class pairs, the MNIST test misclassification rate was closest to uniformly at random (u.a.r.) for `frog` vs. `airplane` (47.5% actual vs. 50% u.a.r. – uniformly at random). Hence, on average, neither `frog` nor `airplane` is overly influential on MNIST. Note that no external constraints induced this near u.a.r. misclassification rate.

Section 6.3.5 compares the ability of influence estimators, with and without renormalization, to identify influential groups of training examples on non-adversarial, CIFAR10, binary classification with Figure 12’s results averaged across five class pairs. Two of the class pairs, `airplane` vs. `bird` and `automobile` vs. `dog`, were studied by Weber et al. [Web+23] in relation to certified defenses. The three other class pairs – `cat` vs. `ship`, `frog` vs. `horse`, and `frog` vs. `truck` – were selected at random.

Wallace et al.’s [Wal+21] poisoning method attacks the SST-2 dataset [Soc+13]. We consider detection on 8 short movie reviews – four positive and four negative – all selected at random by Wallace et al.’s implementation. The specific reviews considered appear in Table D.55.

Table D.55. SST-2 movie reviews selected by Wallace et al.’s [Wal+21] poisoning attack implementation.

Sentiment	No.	Text
↑ Positive ↓	1	<i>a delightful coming-of-age story .</i>
	2	<i>a smart , witty follow-up .</i>
	3	<i>ahhhh ... revenge is sweet !</i>
	4	<i>a giggle a minute .</i>
↑ Negative ↓	1	<i>oh come on .</i>
	2	<i>do not see this film .</i>
	3	<i>it 's a buggy drag .</i>
	4	<i>or emptying rat traps .</i>

The next section provides details regarding the adversarial datasets sizes.

D.3.1.1 Training Set Sizes. Table D.56 details the dataset sizes used to train all evaluated models in Section 6.5.

Table D.56. Chapter 6 target identification dataset sizes

Dataset	Attack	# Classes	# Train	# Test
CIFAR10 [KNH14]	Poison	5	25,000	5,000
SST-2 ⁵ [Soc+13]	Poison	2	67,349	N/A
Speech [Liu+18]	Backdoor	10	3,000 ⁶	1,184
CIFAR10 [KNH14]	Backdoor	2	10,000	2

Liu et al.’s [Liu+18] speech backdoor dataset includes training and test examples with their associated adversarial trigger already embedded. We used their adversarial dataset unchanged. Table D.57 details $|\mathcal{D}_{\text{adv}}|$ (i.e., adversarial training set size) for each speech digit pair after a fixed, random train-validation split.

Table D.57. Number of backdoor training examples for each speech backdoor digit pair. As detailed above, Liu et al.’s [Liu+18] dataset provides 30 backdoored instances for each digit pair. The remainder of the 30 instances for each digit pair are part of the fixed, validation set.

Digit Pair	0 → 1	1 → 2	2 → 3	3 → 4	4 → 5	5 → 6	6 → 7	7 → 8	8 → 9	8 → 9
$ \mathcal{D}_{\text{adv}} $	26	27	24	24	26	28	26	26	22	21

D.3.1.2 Target Set Sizes. Table D.58 details the sizes of the target and non-target sets considered in Section 6.5.3’s target identification experiments. Davis and Goadrich [DG06] explain that the class imbalance ratio between classes defines the unattainable regions in the precision-recall curve. By extension, this ratio also dictates the baseline AUPRC value if examples are labeled randomly.

⁵Stanford Sentiment Treebank dataset (SST-2) is used for sentiment analysis

⁶Clean only. Dataset also has 300 backdoored samples divided evenly among the 10 attack class pairs (e.g., 0 → 1, 1 → 2, etc.).

Table D.58. Target and non-target set sizes used in Section 6.5.3’s target identification experiments.

Attack	Type	# Targets	# Non-Targets
Backdoor	Speech	10	220
	Vision	35	250
Poison	NLP	1	125
	Vision	1	450

D.3.2 Hyperparameters. This section details three primary hyperparameter types, namely: hyperparameters used to create adversarial set \mathcal{D}_{adv} (if any), hyperparameter used when training model f , and influence estimator hyperparameters.

D.3.2.1 Model Training. Table D.59 enumerates the hyperparameters used when training the models analyzed in Section 6.3.

Table D.59. Renormalized influence model training hyperparameter settings

Hyperparameter	CIFAR10 & MNIST	Filtering
θ_0 Pretrained?		✓*
Data Augmentation?		✓
Validation Split	$\frac{1}{6}$	$\frac{1}{6}$
Optimizer	Adam	Adam
$ \mathcal{D}_{\text{adv}} $	150	N/A
Batch Size	64	64
# Epochs	10	10
# Subepochs (ω) ⁷	5	3
η (Peak)	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
η Scheduler	One cycle	One cycle
λ (Weight Decay)	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$

Table D.60 enumerates the hyperparameters used when training the adversarially-attacked models analyzed in Sections 6.5.

D.3.2.2 Upper-Tail Heaviness Hyperparameters. Section 6.4.2 defines the upper-tail heaviness of influence vector \mathbf{v} as the κ -th largest anomaly score in

⁷We use the term “ ω subepoch checkpointing” ($\omega \in \mathbb{Z}_+$) to denote that iteration subset \mathcal{T} is formed from ω evenly-spaced checkpoints within each epoch. ω was not tuned, and was selected based on overall execution time and compute availability.

⁸Varies by digit pair. See Table D.57.

Table D.60. Training-set attack model training hyperparameter settings

Hyperparameter	Poison		Backdoor	
	CIFAR10	SST-2	Speech	CIFAR10
θ_0 Pretrained?	✓	✓		
Existing Adv. Dataset			✓	
Data Augmentation?	✓			
Validation Split	$\frac{1}{6}$	Predefined	$\frac{1}{6}$	$\frac{1}{6}$
Optimizer	SGD	Adam	SGD	Adam
$ \mathcal{D}_{\text{adv}} $	50	50	21–28 ⁸	150
$ \mathcal{D}_{\text{cl}} $	24,950	67,349	3,000	9,850
Poisoning Rate $\left(\frac{ \mathcal{D}_{\text{adv}} }{ \mathcal{D} }\right)$	0.20%	0.07%	0.99%	1.50%
Batch Size	256	32	32	64
# Epochs	30	4	30	10
# Subepochs (ω)	5	3	3	5
η (Peak)	$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
η Scheduler	One cycle	Poly. decay	One cycle	One cycle
λ (Weight Decay)	$1 \cdot 10^{-1}$	$1 \cdot 10^{-1}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Dropout Rate	N/A	0.1	N/A	N/A

vector σ . Table D.61 defines the hyperparameter value κ used for each of Section 6.5.1’s four attacks.

Table D.61. Upper-tail heaviness cutoff count (κ)

Attack	Type	Tail Count (κ)
Backdoor	Speech	10
	Vision	10
Poison	NLP	10
	Vision	2

D.3.2.3 Target-Driven Mitigation Hyperparameters. Algorithm 6 details our target-driven attack mitigation algorithm, which uses filtering cutoff hyperparameter ζ to tune how much data to filter in each filtering iteration. Table D.62 details the hyperparameter settings used in Section 6.5.4’s attack mitigation experiments.

For each attack, multiple trials were performed with different target examples, class pairs, attack triggers, etc. For each such trial, we repeated the mitigation

experiment multiple times to ensure the most representative numbers with the number of repeats enumerated in Table D.62.

In addition, cutoff threshold ζ was set to an initial value. After a specified number of iterations l , ζ was decreased by a specified step-size. This process continued until the attack had been mitigated. To summarize, iteration l 's mitigation cutoff value ζ is

$$\zeta_l = \zeta_{\text{initial}} - \psi \left\lfloor \frac{l}{\text{StepCount}} \right\rfloor, \quad (\text{D.1})$$

with the corresponding value of each parameter in Table D.62.

Table D.62. Target-driven attack mitigation hyperparameters

Hyperparameter	Poison		Backdoor	
	CIFAR10	SST-2	Speech	CIFAR10
Repeats Per Trial	3	3	5	5
ζ_{initial} Initial Cutoff	3	4	3	2
Anneal Step Size (ψ)	0.25	0.5	0.25	0.25
Anneal Step Count	1	1	4	4

D.3.2.4 Adversarial Set \mathcal{D}_{adv} Crafting. Liu et al.'s [Liu+18] speech recognition dataset comes bundled with 300 backdoor training examples. The adversarial trigger takes the form of white noise inserted at the beginning of the speech recording. We used the dataset unchanged except for a fixed training/validation split used in all experiments. Only one backdoor digit pair (e.g, $0 \rightarrow 1$, $1 \rightarrow 2$, etc.) is considered at a time.

Weber et al. [Web+23] consider backdoor three different backdoor adversarial trigger types on CIFAR10 binary classification. The three attack patterns are:

1. *1 Pixel*: The image's center pixel is perturbed to the maximum value.
2. *4 Pixel*: Four specific pixels near the image's center had their pixel value increased a fixed amount.
3. *Blend*: A fixed isotropic Gaussian-noise pattern ($\mathcal{N}(0,I)$) across the entire image.

Table D.63 defines each attack pattern’s maximum L_2 perturbation distance. Any perturbation that exceeded the pixel minimum/maximum values was clipped to the valid range.

Table D.63. CIFAR10 vision backdoor adversarial trigger maximum ℓ_2 -norm perturbation distance

Pattern	Max. ℓ_2
1 Pixel	$\sqrt{3}$
4 Pixel	2
Blend	4

Wallace et al. [Wal+21] construct single-target natural language poison using the traditional poisoning bilevel optimization,

$$\arg \min_{\mathcal{D}_{\text{adv}}} \mathcal{L}_{\text{adv}}(\hat{z}_{\text{targ}}; \arg \min_{\theta} \sum_{z \in \mathcal{D}_{\text{cl}} \cup \mathcal{D}_{\text{adv}}} \mathcal{L}(z_i; \theta)), \quad (\text{D.2})$$

where L_{adv} uses the attacker’s *adversarial loss function*, $\mathcal{L}_{\text{adv}} : \mathcal{A} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$, in place of training loss function \mathcal{L} [BNL12; Muñ+17]. To make the computation tractable, Wallace et al. approximate inner minimizer, $\arg \min_{\theta} \sum_{z \in \mathcal{D}_{\text{cl}} \cup \mathcal{D}_{\text{adv}}} \mathcal{L}(z_i; \theta)$, using second-order gradients similar to [FAL17; Wan+18; Hua+20]. Wallace et al.’s method initializes each poison instance from a seed phrase, and tokens are iteratively replaced with alternates that align well with the poison example’s gradient.

Like Wallace et al., our experiments attacked sentiment analysis on the Stanford Sentiment Treebank v2 (SST-2) dataset [Soc+13]. We targeted 8 (4 positive & 4 negative – see Table D.55) reviews selected by Wallace et al.’s implementation and generated $|\mathcal{D}_{\text{adv}}| = 50$ new poison in each trial.

Zhu et al.’s [Zhu+19] targeted, clean-label attack crafts a set of poisons by forming a *convex polytope* around the target’s feature representation. Our experiments used the author’s open-source implementation when crafting the poison. Their

implementation is gray-box and assumes access to a known pre-trained network (excluding the randomly-initialized, linear classification layer).

Both Zhu et al.’s [Zhu+19] and Wallace et al.’s [Wal+21] poison crafting algorithms have their own dedicated hyperparameters, which are detailed in Tables D.64 and D.65 respectively. Note that Table D.65’s hyperparameters are taken unchanged from the original source code provided by Wallace et al.

Table D.64. Convex polytope poison crafting [Zhu+19] hyperparameter settings

Hyperparameter	Value
# Iterations	1,000
Learning Rate	$4 \cdot 10^{-2}$
Weight Decay	0
Max. Perturb. (ϵ)	0.1

Table D.65. SST-2 sentiment analysis poison crafting hyperparameter settings. These are identical to Wallace et al.’s [Wal+21] hyperparameter settings.

Hyperparameter	Value
Optimizer	Adam
Total Num. Updates	20,935
# Warmup Updates	1,256
Max. Sentence Len.	512
Max. Batch Size	7
Learning Rate	$1 \cdot 10^{-5}$
LR Scheduler	Polynomial Decay

D.3.2.5 Baselines.

Baselines for Identifying Adversarial Set \mathcal{D}_{adv} We exclusively considered influence-estimation methods applicable to neural models and excluded influence methods specific to alternate architectures [BHL23].

Koh and Liang’s [KL17] influence functions estimator uses Pearlmutter’s [Pea94] stochastic Hessian-vector product (HVP) estimation algorithm. Pearlmutter’s

algorithm requires 5 hyperparameters, and we follow Koh and Liang’s notation for these parameters below.

Influence functions’ five hyperparameters are required to ensure estimator quality and to prevent numerical instability/divergence. Table D.66 details the influence functions hyperparameters used for each of Section 6.5’s datasets. t and r were selected to make a single pass through the training set in accordance with the procedure specified by Koh and Liang.

As noted by Basu et al. [BPF21], influence functions can be fragile on deep networks. We tuned β and γ to prevent HVP divergence, which is common with influence functions.

Our influence functions implementation was adapted from the versions published by [Guo+21] and in the Python package `pytorch_influence_functions`.⁹

Table D.66. Influence functions hyperparameter settings

Hyperparameter	Renormalization		Poison		Backdoor	
	CIFAR10 & MNIST	Non-adv.	CIFAR10	SST-2	Speech	CIFAR10
Batch Size	1	1	1	1	1	1
Damp (β)	$1 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-2}$
Scale (γ)	$3 \cdot 10^7$	$1 \cdot 10^4$	$3 \cdot 10^7$	$1 \cdot 10^6$	$1 \cdot 10^4$	$3 \cdot 10^7$
Recursion Depth (t)	1,000	1,000	2,500	6,740	1,000	1,000
Repeats (r)	10	10	10	10	10	10

Second-order influence functions [BYF20] are more brittle and computationally expensive than the first-order version. Renormalization is intended as a first-order correction and addresses our two tasks without the costs/issues related to second-order methods.

Chen et al.’s [Che+21] HYDRA is an additional dynamic influence estimator. However, HYDRA’s $\mathcal{O}(np)$ memory complexity makes it impractical in most modern

⁹Package source code: https://github.com/nimarb/pytorch_influence_functions.

applications with large models and datasets. We focus on TracIn as its memory complexity is only $\mathcal{O}(n)$. HYDRA and TracIn were published contemporaneously and share the same core idea.¹⁰

Peri et al.’s [Per+20] Deep k -NN defense labels a training example as poison if its label does not match the plurality of its neighbors. For Deep k -NN to accurately identify poison, it must generally hold that $k > 2|\mathcal{D}_{\text{adv}}|$. Peri et al. propose selecting k using the *normalized k -ratio*, k/N , where N is the size of the largest class in \mathcal{D} .

Peri et al.’s ablation study showed that Deep k -NN generally performed best when the normalized k -ratio was in the range $[0.2, 2]$. To ensure a strong baseline, our experiments tested Deep k -NN with three normalized k -ratio values, $\{0.2, 1, 2\}$,¹¹ and we report the top-performing k ’s result.

Baselines Identifying the Attack Target(s) Target identification baselines maximum and minimum k -NN distance depend on k in order to generate target rankings. Given k ’s similarity to our tail cutoff count κ , we use the same hyperparameter settings for both with the values in Table D.61.

D.3.3 Network Architectures. Table D.67 details the CIFAR10 neural network architecture. Specifically, we used Page’s [Pag20] ResNet9 architecture, which is the state-of-the-art for fast, high-accuracy ($>94\%$) CIFAR10 classification on DAWNbench [Col+17] at the time of writing.

Following Wallace et al. [Wal+21], natural language poisoning attacked Liu et al.’s [Liu+20b] RoBERTa_{BASE} pre-trained parameters. All language model training used Facebook AI Research’s fairseq sequence-to-sequence toolkit [Ott+19] as specified

¹⁰Our influence renormalization – proposed in Section 6.3 – also applies to HYDRA.

¹¹This corresponds to $k \in \{833, 4167, 8333\}$ for 25,000 CIFAR10 training examples and a $\frac{1}{6}$ validation split ratio.

by Wallace et al. The text was encoded using Radford et al.’s [Rad+19] *byte-pair encoding* (BPE) scheme.

The speech classification convolutional neural network is identical to that used by Liu et al. [Liu+18] except for two minor changes. First, batch normalization [IS15] was used instead of dropout to expedite training convergence. In addition, each convolutional layer’s kernel count was halved to allow the model to be trained on a single NVIDIA Tesla K80 GPU.

Table D.67. Simplified ResNet9 neural network architecture used for Sec. 6.5’s CIFAR10 binary classification

	Conv1	In=3	Out=64	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=64			
	ReLU				
	Conv2	In=64	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	ReLU				
	MaxPool2D	2 × 2			
↑ ResNet1	ConvA	In=128	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	ReLU				
↓	ConvB	In=128	Out=128	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=128			
	ReLU				
	Conv3	In=128	Out=256	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=256			
	ReLU				
	MaxPool2D	2 × 2			
	Conv4	In=256	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	ReLU				
	MaxPool2D	2 × 2			
↑ ResNet2	ConvA	In=512	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	ReLU				
↓	ConvB	In=512	Out=512	Kernel=3 × 3	Pad=1
	BatchNorm2D	Out=512			
	ReLU				
	MaxPool2D	2 × 2			
	Linear	Out=10			

Table D.68. Speech recognition convolutional neural network

Conv1	In=3	Out=48	Kernel= 11×11	Pad=1
MaxPool2D	3×3			
BatchNorm2D	Out=48			
Conv2	In=48	Out=128	Kernel= 5×5	Pad=2
MaxPool2D	3×3			
BatchNorm2D	Out=128			
Conv3	In=128	Out=192	Kernel= 3×3	Pad=1
ReLU				
BatchNorm2D	Out=192			
Conv4	In=192	Out=192	Kernel= 3×3	Pad=1
ReLU				
BatchNorm2D	Out=192			
Conv5	In=192	Out=128	Kernel= 3×3	Pad=1
ReLU				
MaxPool2D	3×3			
BatchNorm2D	Out=128			
Linear	Out=10			

REFERENCES CITED

- [Aga+19] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. “Protecting World Leaders Against Deep Fakes”. In: *Proceedings of the CVPR Workshop on Media Forensics*. Long Beach, California, 2019.
- [Agh+20] Hojjat Aghakhani, Thorsten Eisenhofer, Lea Schönherr, Dorothea Kolossa, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. “VENOMAVE: Clean-Label Poisoning Against Speech Recognition”. In: (2020). arXiv: [2010.10682](https://arxiv.org/abs/2010.10682) [cs.SD].
- [AKA91] David W. Aha, Dennis Kibler, and Marc K. Albert. “Instance-Based Learning Algorithms”. In: *Machine Learning* 6.1 (1991), pp. 37–66.
- [ACW18] Anish Athalye, Nicholas Carlini, and David A. Wagner. “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: ICML’18 (2018). URL: <https://arxiv.org/abs/1802.00420>.
- [Awa+18] Edmond Awad, Sohan Dsouza, Richard Kim, Jonathan Schulz, Joseph Henrich, Azim Shariff, Jean-François Bonnefon, and Iyad Rahwan. “The Moral Machine Experiment”. In: *Nature* 563.7729 (2018), pp. 59–64.
- [Bai+21] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. “Recent Advances in Adversarial Training for Adversarial Robustness”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. IJCAI’21. 2021. DOI: [10.24963/ijcai.2021/591](https://doi.org/10.24963/ijcai.2021/591). URL: <https://arxiv.org/abs/2102.01356>.
- [BL78] Vic Barnett and Toby Lewis. *Outliers in Statistical Data*. 2nd edition. Hoboken, New Jersey, USA: John Wiley & Sons Ltd., 1978.
- [Bar17] Jonathan T. Barron. *Continuously Differentiable Exponential Linear Units*. 2017. arXiv: [1704.07483](https://arxiv.org/abs/1704.07483) [cs.LG].
- [BBD20] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. “RelatIF: Identifying Explanatory Training Samples via Relative Influence”. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. AISTATS’20. 2020.
- [BPF21] Samyadeep Basu, Phil Pope, and Soheil Feizi. “Influence Functions in Deep Learning Are Fragile”. In: *Proceedings of the 9th International Conference on Learning Representations*. ICLR’21. Virtual Only, 2021.

- [BYF20] Samyadeep Basu, Xuchen You, and Soheil Feizi. “On Second-Order Group Influence Functions for Black-Box Predictions”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. Virtual Only: PMLR, 2020.
- [BT74] Albert E. Beaton and John W. Tukey. “The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data”. In: *Technometrics* 16.2 (1974), pp. 147–185.
- [Ben75] Jon L. Bentley. *A Survey of Techniques for Fixed Radius Near Neighbor Searching*. Tech. rep. Stanford, CA, USA: Stanford University, 1975.
- [BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning Attacks against Support Vector Machines”. In: *Proceedings of the 29th International Conference on Machine Learning*. ICML’12. Edinburgh, Great Britain: PMLR, 2012. URL: <https://arxiv.org/abs/1206.6389>.
- [BR92] Avrim L. Blum and Ronald L. Rivest. “Training a 3-Node Neural Network is NP-Complete”. In: *Neural Networks* 5.1 (1992), pp. 117–127.
- [Blu+73] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. “Time Bounds for Selection”. In: *Journal of Computer and System Sciences* 7.4 (1973), pp. 448–461. ISSN: 0022-0000.
- [BHL23] Jonathan Brophy, Zayd Hammoudeh, and Daniel Lowd. “Adapting and Evaluating Influence-Estimation Methods for Gradient-Boosted Decision Trees”. In: *Journal of Machine Learning Research* 24 (2023), pp. 1–48. URL: <http://jmlr.org/papers/v24/22-0449.html>.
- [Buc+18] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. “Thermometer Encoding: One Hot Way To Resist Adversarial Examples”. In: *Proceedings of the 6th International Conference on Learning Representations*. ICLR’18. 2018. URL: <https://openreview.net/forum?id=S18Su--CW>.
- [Cal+21] Stefano Calzavara, Claudio Lucchese, Federico Marcuzzi, and Salvatore Orlando. “Feature Partitioning for Robust Tree Ensembles and their Certification in Adversarial Scenarios”. In: *EURASIP Journal on Information Security* (Dec. 2021), pp. 245–317.
- [Car19] Nicholas Carlini. *On Evaluating Adversarial Robustness*. 2019. URL: <https://youtu.be/-p2il-V-0fk?t=1574>.

- [Car+23] Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. *Poisoning Web-Scale Training Datasets is Practical*. arXiv. 2023. eprint: [2302.10149](https://arxiv.org/abs/2302.10149) (cs.CR). URL: <https://arxiv.org/abs/2302.10149>.
- [CW17] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *Proceedings of the 2017 IEEE Symposium on Security and Privacy*. SP’17. IEEE Computer Society, 2017. DOI: [10.1109/SP.2017.49](https://doi.ieeecomputersociety.org/10.1109/SP.2017.49). URL: <https://doi.ieeecomputersociety.org/10.1109/SP.2017.49>.
- [Che+19] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering”. In: *Proceedings of the AAAI Workshop on Artificial Intelligence Safety*. SafeAI’19. Honolulu, Hawaii, USA: Association for the Advancement of Artificial Intelligence, 2019.
- [Che+22] Ruoxin Chen, Zenan Li, Jie Li, Chentao Wu, and Junchi Yan. “On Collective Robustness of Bagging Against Data Poisoning”. In: *Proceedings of the 39th International Conference on Machine Learning*. ICML’22. PMLR, 2022.
- [CG16] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD’16. New York, NY, USA: Association for Computing Machinery, 2016.
- [Che+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. *Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning*. 2017. arXiv: [1712.05526](https://arxiv.org/abs/1712.05526) [cs.CR].
- [Che+21] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. “HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks”. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. AAAI’21. Virtual Only: Association for the Advancement of Artificial Intelligence, 2021.
- [CCM13] Yudong Chen, Constantine Caramanis, and Shie Mannor. *Robust High Dimensional Sparse Regression and Matching Pursuit*. arXiv. 2013. eprint: [1301.2725](https://arxiv.org/abs/1301.2725) (stat.ML).
- [Chi+20] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. “Certified Defenses for

- Adversarial Patches”. In: *Proceedings of the 8th International Conference on Learning Representations*. ICLR’20. Virtual Only, 2020. URL: <https://openreview.net/forum?id=HyeaSkRYPH>.
- [Chv79] Vasek Chvatal. “A Greedy Heuristic for the Set-Covering Problem”. In: *Mathematics of Operations Research* 4.3 (1979), pp. 233–235. ISSN: 0364-765X.
- [Coc11] Dean De Cock. “Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project”. In: *Journal of Statistics Education* 19.3 (2011).
- [CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. “Certified Adversarial Robustness via Randomized Smoothing”. In: *Proceedings of the 36th International Conference on Machine Learning*. ICML’19. PMLR, 2019. URL: <https://arxiv.org/abs/1902.02918>.
- [Col+17] Cody A. Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. “DAWNbench: An End-to-End Deep Learning Benchmark and Competition”. In: *Proceedings of the 2017 NeurIPS Workshop on Machine Learning Systems*. Long Beach, California, USA: Curran Associates, Inc., 2017.
- [Col22] Kevin Collier. *Former Twitter Employee Sentenced to More than Three Years in Prison for spying for Saudi Arabia*. Dec. 2022. URL: <https://www.nbcnews.com/tech/security/former-twitter-employee-sentenced-three-years-prison-spying-saudi-arab-rcna61384>.
- [Coo77] R. Dennis Cook. “Detection of Influential Observation in Linear Regression”. In: *Technometrics* 19.1 (1977), pp. 15–18.
- [CW82] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression*. New York: Chapman and Hall, 1982. ISBN: 041224280X.
- [CR92] Christophe Croux and Peter J. Rousseeuw. “A Class of High-Breakdown Scale Estimators Based on Subranges”. In: *Communications in Statistics - Theory and Methods* 21.7 (1992), pp. 1935–1951.
- [DAm+20] Alexander D’Amour, Katherine A. Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yi-An Ma, Cory Y. McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson,

- Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. *Underspecification Presents Challenges for Credibility in Modern Machine Learning*. 2020. arXiv: [2011.03395](https://arxiv.org/abs/2011.03395) [cs.LG].
- [DPV08] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2008. ISBN: 978-0-07-352340-8.
- [DG06] Jesse Davis and Mark Goadrich. “The Relationship Between Precision-Recall and ROC Curves”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML’06. Pittsburgh, Pennsylvania: PMLR, 2006.
- [Dhi+18] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossai, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. “Stochastic Activation Pruning for Robust Adversarial Defense”. In: *Proceedings of the 6th International Conference on Learning Representations*. ICLR’18. 2018. URL: <https://openreview.net/forum?id=H1uR4GZRZ>.
- [Ebr+18] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. “HotFlip: White-Box Adversarial Examples for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. ACL’18. 2018.
- [EK10] Tapio Elomaa and Jussi Kujala. “Covering Analysis of the Greedy Algorithm for Partial Cover”. In: *Algorithms and Applications: Essays Dedicated to Esko Ukkonen on the Occasion of His 60th Birthday*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 102–113. ISBN: 3642124755.
- [Fel20] Vitaly Feldman. “Does Learning Require Memorization? A Short Tale about a Long Tail”. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. STOC’20. 2020.
- [FZ20] Vitaly Feldman and Chiyuan Zhang. “What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation”. In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. NeurIPS’20. Virtual Only: Curran Associates, Inc., 2020.
- [FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. ICML’17. Sydney, Australia: PMLR, 2017.

- [FB81] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. ISSN: 0001-0782.
- [Fow+21] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojtek Czaja, and Tom Goldstein. “Adversarial Examples Make Strong Poisons”. In: *Proceedings of the 35th Conference on Neural Information Processing Systems*. NeurIPS’21. Virtual Only: Curran Associates, Inc., 2021.
- [Gao+19] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. “STRIP: A Defence against Trojan Attacks on Deep Neural Networks”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC’19. San Juan, Puerto Rico, USA: Association for Computing Machinery, 2019.
- [Gei+21] Jonas Geiping, Liam Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. “Witches’ Brew: Industrial Scale Data Poisoning via Gradient Matching”. In: *Proceedings of the 9th International Conference on Learning Representations*. ICLR’21. Virtual Only, 2021.
- [Gei+20] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. “Shortcut Learning in Deep Neural Networks”. In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.
- [GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. “Extremely Randomized Trees”. In: *Machine Learning* 63.1 (2006), pp. 3–42. ISSN: 1573-0565.
- [GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *Proceedings of the 3rd International Conference on Learning Representations*. ICLR’15. 2015. URL: <https://arxiv.org/abs/1412.6572>.
- [GF17] Bryce Goodman and Seth Flaxman. “European Union Regulations on Algorithmic Decision-Making and a ‘Right to Explanation’”. In: *AI Magazine* 38.3 (Oct. 2017), pp. 50–57.
- [Gow+19] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Arthur Mann, and Pushmeet Kohli. “Scalable Verified Training for Provably Robust Image Classification”. In: *Proceedings of the 2019 IEEE/CVF*

International Conference on Computer Vision. ICCV'19. 2019. DOI: 10.1109/ICCV.2019.00494.

- [Gu+19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* 7 (2019), pp. 47230–47244. URL: <https://ieeexplore.ieee.org/document/8685687>.
- [Guo+20] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. “Certified Data Removal from Machine Learning Models”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. ICML'20. 2020, pp. 3832–3842.
- [Guo+18] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. “Countering Adversarial Images using Input Transformations”. In: *Proceedings of the 6th International Conference on Learning Representations. ICLR'18. 2018*. URL: <https://openreview.net/forum?id=SyJ7ClWCb>.
- [Guo+21] Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. “FastIF: Scalable Influence Functions for Efficient Model Interpretation and Debugging”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. EMNLP'21. 2021*.
- [Gur22] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2022. URL: <https://www.gurobi.com>.
- [HL21] Zayd Hammoudeh and Daniel Lowd. “Simple, Attack-Agnostic Defense Against Targeted Training Set Attacks Using Cosine Similarity”. In: *Proceedings of the 3rd ICML Workshop on Uncertainty and Robustness in Deep Learning. UDL'21. 2021*.
- [HL22a] Zayd Hammoudeh and Daniel Lowd. “Identifying a Training-Set Attack’s Target Using Renormalized Influence Estimation”. In: *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security. CCS'22. Los Angeles, CA: Association for Computing Machinery, 2022*. URL: <https://arxiv.org/abs/2201.10055>.
- [HL22b] Zayd Hammoudeh and Daniel Lowd. “Training Data Influence Analysis and Estimation: A Survey”. In: *arXiv* (2022). arXiv: [2212.04612](https://arxiv.org/abs/2212.04612) [cs.LG].

- [HL23a] Zayd Hammoudeh and Daniel Lowd. “Feature Partition Aggregation: A Fast Certified Defense Against a Union of ℓ_0 Attacks”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023. URL: <https://arxiv.org/abs/2302.11628>.
- [HL23b] Zayd Hammoudeh and Daniel Lowd. *Feature Partition Aggregation: A Fast Certified Defense Against a Union of Sparse Adversarial Attacks*. 2023. arXiv: [2302.11628](https://arxiv.org/abs/2302.11628) [cs.LG].
- [HL23c] Zayd Hammoudeh and Daniel Lowd. “Reducing Certified Regression to Certified Classification for General Poisoning Attacks”. In: *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning*. SaTML’23. 2023. URL: <https://arxiv.org/abs/2208.13904>.
- [HNM19] Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. “Data Cleansing for Models Trained with SGD”. In: *Proceedings of the 33rd Conference on Neural Information Processing Systems*. NeurIPS’19. Vancouver, Canada: Curran Associates, Inc., 2019.
- [Hea+21] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. *scikit-optimize: Sequential model-based optimization in Python*. Version v0.9.0. 2021.
- [HA04] Victoria J. Hodge and Jim Austin. “A Survey of Outlier Detection Methodologies”. In: *Artificial Intelligence Review* 22.2 (Oct. 2004), pp. 85–126.
- [Hop+17] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. *UCI Machine Learning Repository: Spambase Dataset*. 2017. URL: <https://archive.ics.uci.edu/ml/datasets/spambase>.
- [Hua+20] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. “MetaPoison: Practical General-purpose Clean-label Data Poisoning”. In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. NeurIPS’20. Virtual Only: Curran Associates, Inc., 2020. URL: <https://arxiv.org/abs/2004.00225>.
- [Hub64] Peter J. Huber. “Robust Estimation of a Location Parameter”. In: *Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101.
- [Ily+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. “Adversarial Examples Are Not Bugs, They Are Features”. In: *Proceedings of the 33rd International*

Conference on Neural Information Processing Systems. NeurIPS'19. Red Hook, NY, USA: Curran Associates Inc., 2019.

- [IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. ICML'15. Lille, France: PMLR, 2015.
- [Jag+18] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. “Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning”. In: *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. SP'18. 2018.
- [Jag+21] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. “Subpopulation Data Poisoning Attacks”. In: *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security*. CCS'21. Virtual Only: Association for Computing Machinery, 2021.
- [JCG21] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. “Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks”. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. AAAI'21. 2021.
- [Jia+22a] Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. “Certified Robustness of Nearest Neighbors against Data Poisoning and Backdoor Attacks”. In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. AAAI'22. 2022. URL: <https://arxiv.org/abs/2012.03765>.
- [Jia+22b] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, Hongbin Liu, and Neil Zhenqiang Gong. “Almost Tight ℓ_0 -norm Certified Robustness of Top-k Predictions against Adversarial Perturbations”. In: *Proceedings of the 10th International Conference on Learning Representations*. ICLR'22. 2022. URL: <https://openreview.net/forum?id=gJLEy3ySpu>.
- [Jin+20] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. “Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment”. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI'20. 2020.
- [Joh74] David S Johnson. “Approximation Algorithms for Combinatorial Problems”. In: *Journal of Computer and System Sciences* 9.3 (1974), pp. 256–278.

- [JW78] John E. Dennis Jr. and Roy E. Welsch. “Techniques for nonlinear least squares and robust regression”. In: *Communications in Statistics - Simulation and Computation* 7.4 (1978), pp. 345–359.
- [Ke+17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NeurIPS’17. 2017.
- [KB15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *Proceedings of the 3rd International Conference on Learning Representations*. ICLR’15. 2015.
- [KKM18] Adam R. Klivans, Pravesh K. Kothari, and Raghu Meka. “Efficient Algorithms for Outlier-Robust Regression”. In: *Proceedings of the 31st Conference on Learning Theory*. COLT’18. PMLR, 2018.
- [KL17] Pang Wei Koh and Percy Liang. “Understanding Black-box Predictions via Influence Functions”. In: *Proceedings of the 34th International Conference on Machine Learning*. ICML’17. Sydney, Australia: PMLR, 2017.
- [Kol+19] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. “Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs”. In: *Proceedings of the 32nd Conference on Computer Vision and Pattern Recognition*. CVPR’19. Long Beach, California, USA, 2019.
- [KNH14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *The CIFAR-10 Dataset*. 2014.
- [Kum+20] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comisssoneru, Matt Swann, and Sharon Xia. “Adversarial Machine Learning – Industry Perspectives”. In: *Proceedings of the 2020 IEEE Security and Privacy Workshops*. SPW’20. 2020. URL: <https://arxiv.org/abs/2002.05646>.
- [KGB16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial Examples in the Physical World”. In: (2016). arXiv: 1607.02533. URL: <http://arxiv.org/abs/1607.02533>.
- [LSF21] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. “Perceptual Adversarial Robustness: Defense Against Unseen Threat Models”. In: *Proceedings of the 9th International Conference on Learning Representations*.

- ICLR'21. Virtual Only, 2021. URL: <https://arxiv.org/abs/2006.12655>.
- [Lec89] Yvan G. Leclerc. “Constructing Simple Stable Descriptions for Image Partitioning”. In: *International Journal of Computer Vision* 3.1 (1989), pp. 73–102.
- [LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE*. Vol. 86. 1998, pp. 2278–2324.
- [Léc+19] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. “Certified Robustness to Adversarial Examples with Differential Privacy”. In: *Proceedings of the 2019 IEEE Symposium on Security and Privacy*. SP'19. IEEE, 2019. URL: <https://arxiv.org/abs/1802.03471>.
- [Lee+19] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. “Tight Certificates of Adversarial Robustness for Randomly Smoothed Classifiers”. In: *Proceedings of the 33rd Conference on Neural Information Processing Systems*. NeurIPS'19. 2019. URL: <https://arxiv.org/abs/1906.04948>.
- [Lee16] Peter Lee. *Learning from Tay's Introduction*. Mar. 2016. URL: <https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/>.
- [LLP20] Sungyoon Lee, Jaewook Lee, and Saerom Park. “Lipschitz-Certifiable Training with a Tight Outer Bound”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NeurIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.
- [Lev+11] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas C. Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. “Click Trajectories: End-to-End Analysis of the Spam Value Chain”. In: *2011 IEEE Symposium on Security and Privacy*. SP'11 (2011), pp. 431–446.
- [LF20a] Alexander Levine and Soheil Feizi. “(De)Randomized Smoothing for Certifiable Defense against Patch Attacks”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NeurIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020. URL: <https://arxiv.org/abs/2002.10733>.

- [LF20b] Alexander Levine and Soheil Feizi. “Robustness Certificates for Sparse Adversarial Attacks by Randomized Ablation”. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 2020. URL: <https://arxiv.org/abs/1911.09272>.
- [LF21] Alexander Levine and Soheil Feizi. “Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks”. In: *Proceedings of the 9th International Conference on Learning Representations*. ICLR’21. Virtual Only, 2021. URL: <https://arxiv.org/abs/2006.14768>.
- [Li+19] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. “Certified Adversarial Robustness with Additive Noise”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. NeurIPS’19. Red Hook, NY, USA: Curran Associates Inc., 2019. URL: <https://arxiv.org/abs/1809.03113>.
- [LXL23] Linyi Li, Tao Xie, and Bo Li. “SoK: Certified Robustness for Deep Neural Networks”. In: *Proceedings of the 44th IEEE Symposium on Security and Privacy*. SP’23. IEEE, 2023. URL: <https://arxiv.org/abs/2009.04131>.
- [LDD21] Xiling Li, Rafael Dowsley, and Martine De Cock. “Privacy-Preserving Feature Selection with Secure Multiparty Computation”. In: *Proceedings of the 38th International Conference on Machine Learning*. ICML’21. 2021. URL: <https://arxiv.org/abs/2102.03517>.
- [Li+21] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. “Anti-Backdoor Learning: Training Clean Models on Poisoned Data”. In: *Proceedings of the 35th Conference on Neural Information Processing Systems*. NeurIPS’21. Virtual Only: Curran Associates, Inc., 2021.
- [Li+22] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. “Backdoor Learning: A Survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022). DOI: [10.1109/TNNLS.2022.3182979](https://doi.org/10.1109/TNNLS.2022.3182979). URL: <https://arxiv.org/abs/2007.08745>.
- [Lin+20] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. “Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS’20. Virtual Only: Association for Computing Machinery, 2020.

- [LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. “Network in Network”. In: *Proceedings of the 2nd International Conference on Learning Representations*. ICLR’14. 2014. URL: <https://arxiv.org/abs/1312.4400>.
- [Liu+17] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. “Robust Linear Regression Against Training Data Poisoning”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec’17. New York, NY, USA: Association for Computing Machinery, 2017.
- [LDG18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks”. In: *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*. RAID’18. Heraklion, Crete, Greece: Springer, 2018, pp. 273–294.
- [Liu+20a] Liu Liu, Yanyao Shen, Tianyang Li, and Constantine Caramanis. “High Dimensional Robust Sparse Regression”. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. AISTATS’20. 2020.
- [Liu+18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. “Trojaning Attack on Neural Networks”. In: *Proceedings of the 25th Annual Network and Distributed System Security Symposium*. NDSS’18. San Diego, California, USA, 2018.
- [Liu+20b] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *Proceedings of the 8th International Conference on Learning Representations*. ICLR’20. Virtual Only, 2020.
- [LXS17] Yuntao Liu, Yang Xie, and Ankur Srivastava. “Neural Trojans”. In: *Proceedings of the 2017 IEEE International Conference on Computer Design*. ICCD’17. 2017. DOI: [10.1109/ICCD.2017.16](https://doi.org/10.1109/ICCD.2017.16). URL: <https://ieeexplore.ieee.org/document/8119189>.
- [Lov75] László Lovász. “On the Ratio of Optimal Integral and Fractional Covers”. In: *Discrete Mathematics* 13.4 (1975), pp. 383–390.
- [Ma+18] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. “Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality”.

- In: *Proceedings of the 6th International Conference on Learning Representations*. ICLR'18. 2018. URL: <https://arxiv.org/abs/1801.02613>.
- [Mad+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *Proceedings of the 6th International Conference on Learning Representations*. ICLR'18. 2018. URL: <https://arxiv.org/abs/1706.06083>.
- [MWK20] Pratyush Maini, Eric Wong, and J. Zico Kolter. “Adversarial Robustness Against the Union of Multiple Perturbation Models”. In: *International Conference on Machine Learning*. ICML'20. 2020. URL: <https://arxiv.org/abs/1909.04068>.
- [Mal+21] Andrey Malinin, Neil Band, Yarin Gal, Mark Gales, Alexander Ganshin, German Chesnokov, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, Vyas Raina, Denis Roginskiy, Mariya Shmatova, Panagiotis Tigas, and Boris Yangel. “Shifts: A Dataset of Real Distributional Shift Across Multiple Large-Scale Tasks”. In: *Proceedings of the 35th Conference on Neural Information Processing Systems*. NeurIPS'21. Curran Associates, Inc., 2021.
- [MRA22] Neil G. Marchant, Benjamin I. P. Rubinstein, and Scott Alfeld. “Hard to Forget: Poisoning Attacks on Certified Machine Unlearning”. In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. AAAI'22. 2022.
- [MY21] Jan Hendrik Metzen and Maksym Yatsura. “Efficient Certified Defenses Against Patch Attacks on Image Classifiers”. In: *Proceedings of the 9th International Conference on Learning Representations*. ICLR'21. 2021. URL: <https://openreview.net/forum?id=hr-3PMvDpil>.
- [Muñ+17] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. “Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec'17. Dallas, Texas, USA: Association for Computing Machinery, 2017.
- [Mur16] Madhumita Murgia. “Microsoft’s Racist Bot Shows We Must Teach AI to Play Nice and Police Themselves”. In: *The Telegraph* (Mar. 2016). URL: <https://www.telegraph.co.uk/technology/2016/03/25/we-must-teach-ai-machines-to-play-nice-and-police-themselves/>.

- [NP33] Jerzy Neyman and Egon S. Pearson. “On the Problem of the Most Efficient Tests of Statistical Hypotheses”. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231 (1933), pp. 289–337. ISSN: 02643952. URL: <http://www.jstor.org/stable/91247>.
- [Ott+19] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.
- [Pag20] David Page. *How to Train Your ResNet*. May 2020. URL: <https://myrtle.ai/learn/how-to-train-your-resnet/>.
- [Par+23] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. “TRAK: Attributing Model Behavior at Scale”. In: *Proceedings of the 40th International Conference on Machine Learning*. ICML’23. 2023. URL: <https://arxiv.org/abs/2303.14186>.
- [Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Proceedings of the 33rd Conference on Neural Information Processing Systems*. NeurIPS’19. Vancouver, Canada: Curran Associates, Inc., 2019. URL: <https://arxiv.org/abs/1912.01703>.
- [Pea+22] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. “Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions”. In: *Proceedings of the 43rd IEEE Symposium on Security and Privacy*. SP’22. IEEE, 2022. URL: <https://arxiv.org/abs/2108.09293>.
- [Pea+23] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. “Examining Zero-Shot Vulnerability Repair with Large Language Models”. In: *Proceedings of the 44th IEEE Symposium on Security and Privacy*. SP’23. IEEE, 2023. URL: <https://arxiv.org/abs/2112.02125>.
- [Pea94] Barak A. Pearlmutter. “Fast Exact Multiplication by the Hessian”. In: *Neural Computation* 6 (1994), pp. 147–160.

- [Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [Per+20] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. “Deep k-NN Defense Against Clean-label Data Poisoning Attacks”. In: *Proceedings of the ECCV Workshop on Adversarial Robustness in the Real World*. AROW’20. Virtual Only, 2020. URL: <https://arxiv.org/abs/1909.13374>.
- [Pie21] Eric Pierce. *Austin, TX House Listings*. 2021. URL: <https://www.kaggle.com/datasets/ericpierce/austinhousingprices>.
- [Pit+09] James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. “Using Game Theory for Los Angeles Airport Security”. In: *AI Magazine* 30 (2009), pp. 43–57.
- [Pru+20] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. “Estimating Training Data Influence by Tracing Gradient Descent”. In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. NeurIPS’20. Virtual Only: Curran Associates, Inc., 2020.
- [Rad+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. 2019.
- [Raj21] Kumar Rajarshi. *Life Expectancy (WHO)*. 2021. URL: <https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who>.
- [Ran+21] Yingli Ran, Zhao Zhang, Shaojie Tang, and Ding-Zhu Du. “Breaking the r_{\max} Barrier: Enhanced Approximation Algorithms for Partial Set Multicover Problem”. In: *INFORMS Journal on Computing* 33.2 (2021), pp. 774–784.
- [Rez+23] Keivan Rezaei, Kiarash Banihashem, Atoosa Chegini, and Soheil Feizi. “Run-Off Election: Improved Provable Defense against Data Poisoning Attacks”. In: *Proceedings of the 40th International Conference on Machine Learning*. ICML’23. 2023. URL: <https://arxiv.org/abs/2302.02300>.

- [Ric22] Terry Richards. *Ex-Hospital Worker Arrested in SGMC Data Breach*. Jan. 2022. URL: https://www.valdostadailytimes.com/news/local_news/ex-hospital-worker-arrested-in-sgmc-data-breach/article_7ca92b22-a2e5-5541-b3b3-38472d3706b1.html.
- [Ros+20] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and J. Zico Kolter. “Certified Robustness to Label-Flipping Attacks via Randomized Smoothing”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. ICML’20. PMLR, 2020, pp. 8230–8241.
- [RC93] Peter Rousseeuw and Christophe Croux. “Alternatives to the Median Absolute Deviation”. In: *Journal of the American Statistical Association* (1993).
- [RH11] Peter J. Rousseeuw and Mia Hubert. “Robust statistics for outlier detection”. In: *WIREs Data Mining and Knowledge Discovery* 1.1 (2011), pp. 73–79.
- [RH17] Peter J. Rousseeuw and Mia Hubert. “Anomaly Detection by Robust Statistics”. In: *WIREs Data Mining and Knowledge Discovery* 8.2 (Nov. 2017).
- [RL87] Peter J. Rousseeuw and Annick. M. Leroy. *Robust Regression and Outlier Detection*. USA: John Wiley & Sons, Inc., 1987. ISBN: 0471852333.
- [Sal+20] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. “Dynamic Backdoor Attacks Against Machine Learning Models”. In: (2020). arXiv: [2003.03675](https://arxiv.org/abs/2003.03675) [cs.CR].
- [Sch+19] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. “Towards the first adversarially robust neural network model on MNIST”. In: *Proceedings of the 7th International Conference on Learning Representations*. ICLR’19. 2019. URL: <https://openreview.net/forum?id=S1EH0sC9tX>.
- [Sha+18] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suci, Christoph Studer, Tudor Dumitras, and Tom Goldstein. “Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. NeurIPS’18. Montreal, Canada: Curran Associates, Inc., 2018. URL: <https://arxiv.org/abs/1804.00792>.
- [Sha+20] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. “The Pitfalls of Simplicity Bias in Neural

- Networks”. In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. NeurIPS’20. 2020.
- [Shi+19] Yishuo Shi, Yingli Ran, Zhao Zhang, James Willson, Guangmo Tong, and Ding-Zhu Du. “Approximation algorithm for the partial set multi-cover problem”. In: *Journal of Global Optimization* 75.4 (2019), pp. 1133–1146.
- [Sla97a] Petr Slavík. “A Tight Analysis of the Greedy Algorithm for Set Cover”. In: *Journal of Algorithms* (1997), pp. 237–254.
- [Sla97b] Petr Slavík. “Improved Performance of the Greedy Algorithm for Partial Cover”. In: *Information Processing Letters* 64.5 (1997), pp. 251–254. ISSN: 0020-0190.
- [Soc+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 8th Conference on Empirical Methods in Natural Language Processing*. EMNLP’13. 2013.
- [Sor+20] Ezekiel O. Soremekun, Sakshi Udeshi, Sudipta Chattopadhyay, and Andreas Zeller. *Exposing Backdoors in Robust Machine Learning Models*. 2020. arXiv: 2003.00865 [cs.LG].
- [SGF22] Gaurang Sriramanan, Maharshi Gor, and Soheil Feizi. “Toward Efficient Robust Training against Union of ℓ_p Threat Models”. In: *Proceedings of the 36th Conference on Neural Information Processing Systems*. NeurIPS’22. 2022. URL: <https://openreview.net/forum?id=6qdUJblMHqy>.
- [SKL17] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. “Certified Defenses for Data Poisoning Attacks”. In: *Proceedings of the 31st Conference on Neural Information Processing Systems*. NeurIPS’17. Long Beach, California, USA: Curran Associates, Inc., 2017.
- [SD20] Cecilia Summers and Michael J. Dinneen. “Four Things Everyone Should Know to Improve Batch Normalization”. In: *Proceedings of the 8th International Conference on Learning Representations*. ICLR’20. Virtual Only, 2020.
- [TZ00] Philip H. S. Torr and Andrew Zisserman. “MLE SAC: A New Robust Estimator with Application to Estimating Image Geometry.” In: *Computer Vision and Image Understanding* 78.1 (2000), pp. 138–156.

- [TB19] Florian Tramer and Dan Boneh. “Adversarial Training and Robustness for Multiple Perturbations”. In: *Proceedings of the 33rd Conference on Neural Information Processing Systems*. NeurIPS’19. Curran Associates, Inc., 2019. URL: <https://arxiv.org/abs/1904.13000>.
- [Tra+20] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. “On Adaptive Attacks to Adversarial Example Defenses”. In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. NeurIPS’20. Curran Associates, Inc., 2020. URL: <https://arxiv.org/abs/2002.08347>.
- [TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. “Spectral Signatures in Backdoor Attacks”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. NeurIPS’18. Montreal, Canada: Curran Associates, Inc., 2018.
- [Tur20] Matt Turek. *Artificial Intelligence Exploration Opportunity DARPA-PA-19-03-09 Reverse Engineering of Deceptions (RED) Amendment #1*. United States Defense Advanced Research Projects Agency (DARPA). 2020.
- [Ude+19] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. *Model Agnostic Defence against Backdoor Attacks in Machine Learning*. 2019. arXiv: [1908.02203](https://arxiv.org/abs/1908.02203) [cs.LG].
- [Van14] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Boston, MA: Springer, 2014. ISBN: 978-3-030-39414-1.
- [VB20] Miguel Villarreal-Vasquez and Bharat K. Bhargava. *ConFoc: Content-Focus Protection Against Trojan Attacks on Neural Networks*. 2020. arXiv: [2007.00711](https://arxiv.org/abs/2007.00711) [cs.CV].
- [Wal+21] Eric Wallace, Tony Z. Zhao, Shi Feng, and Sameer Singh. “Concealed Data Poisoning Attacks on NLP Models”. In: *Proceedings of the North American Chapter of the Association for Computational Linguistics*. NAACL’21. 2021.
- [Wan+18] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. *Dataset Distillation*. 2018. arXiv: [1811.10959](https://arxiv.org/abs/1811.10959) [cs].
- [WF23] Wenxiao Wang and Soheil Feizi. *Temporal Robustness Against Data Poisoning*. 2023. arXiv: [2302.03684](https://arxiv.org/abs/2302.03684) [cs.LG]. URL: <https://arxiv.org/abs/2302.03684>.
- [WLF22a] Wenxiao Wang, Alexander Levine, and Soheil Feizi. “Improved Certified Defenses against Data Poisoning with (Deterministic) Finite

- Aggregation”. In: *Proceedings of the 39th International Conference on Machine Learning*. ICML’22. 2022. URL: <https://arxiv.org/abs/2202.02628>.
- [WLF22b] Wenxiao Wang, Alexander Levine, and Soheil Feizi. “Lethal Dose Conjecture on Data Poisoning”. In: *Proceedings of the 36th Conference on Neural Information Processing Systems*. NeurIPS’22. Curran Associates, Inc., 2022. URL: <https://arxiv.org/abs/2208.03309>.
- [Web+23] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. “RAB: Provable Robustness Against Backdoor Attacks”. In: *Proceedings of the 44th IEEE Symposium on Security and Privacy*. SP’23. IEEE, 2023. URL: <https://arxiv.org/abs/2003.08904>.
- [Wei+22] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, and Thilina Ranbaduge. *Vertical Federated Learning: Challenges, Methodologies and Experiments*. 2022. arXiv: [2202.04309](https://arxiv.org/abs/2202.04309) [cs.LG]. URL: <https://arxiv.org/abs/2202.04309>.
- [Wen+18] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. “Towards Fast Computation of Certified Robustness for ReLU Networks”. In: *Proceedings of the 35th International Conference on Machine Learning*. ICML’18. PMLR, 2018. URL: <https://arxiv.org/abs/1804.09699>.
- [Wic16] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. New York, NY: Springer-Verlag, 2016. ISBN: 978-3-319-24277-4.
- [XZZ20] Chang Xiao, Peilin Zhong, and Changxi Zheng. “Enhancing Adversarial Defense by k-Winners-Take-All”. In: *Proceedings of the 8th International Conference on Learning Representations*. ICLR’20. Virtual Only, 2020. URL: <https://arxiv.org/abs/1905.10510>.
- [Xia+15] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. “Is Feature Selection Secure against Training Data Poisoning?” In: *Proceedings of the 32nd International Conference on Machine Learning*. ICML’15. Lille, France: PMLR, 2015.
- [Yeh+18] Chih-Kuan Yeh, Joon Sik Kim, Ian E.H. Yen, and Pradeep Ravikumar. “Representer Point Selection for Explaining Deep Neural Networks”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. NeurIPS’18. Montreal, Canada: Curran Associates, Inc., 2018.
- [YHL23a] Wencong You, Zayd Hammoudeh, and Daniel Lowd. “Large Language Models Are Better Adversaries: Exploring Generative Clean-Label

- Backdoor Attacks Against Text Classifiers”. In: *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning*. AdvML-Frontiers’23. 2023.
- [YHL23b] Wencong You, Zayd Hammoudeh, and Daniel Lowd. “Large Language Models Are Better Adversaries: Exploring Generative Clean-Label Backdoor Attacks Against Text Classifiers”. In: *Findings of the Association for Computational Linguistics*. ELMNLP’23. 2023.
- [Yu+21] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. *Indiscriminate Poisoning Attacks are Shortcuts*. 2021. arXiv: [2111.00898](https://arxiv.org/abs/2111.00898) [cs.LG].
- [Zha+19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. “Theoretically Principled Trade-off between Robustness and Accuracy”. In: *Proceedings of the 36th International Conference on Machine Learning*. ICML’19. PMLR, 2019. URL: <https://arxiv.org/abs/1901.08573>.
- [Zha+18] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. “mixup: Beyond Empirical Risk Minimization”. In: *Proceedings of the 6th International Conference on Learning Representations*. ICLR’18. 2018.
- [ZZ22] Rui Zhang and Shihua Zhang. “Rethinking Influence Functions of Neural Networks in the Over-Parameterized Regime”. In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. AAAI’22. Vancouver, Canada: Association for the Advancement of Artificial Intelligence, 2022.
- [Zhu+19] Chen Zhu, W Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. “Transferable Clean-Label Poisoning Attacks on Deep Neural Nets”. In: *Proceedings of the 36th International Conference on Machine Learning*. ICML’19. Los Angeles, CA: PMLR, 2019.
- [Zhu+21] Liuwan Zhu, Rui Ning, Chunsheng Xin, Chonggang Wang, and Hongyi Wu. “CLEAR: Clean-up Sample-Targeted Backdoor in Neural Networks”. In: *Proceedings of the 18th International Conference on Computer Vision*. ICCV’21. 2021.