

# **AI-INFUSED SECURITY: ROBUST DEFENSE BY BRIDGING THEORY AND PRACTICE**

A Dissertation  
Presented to  
The Academic Faculty

By

Shang-Tse Chen

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Computer Science

Georgia Institute of Technology

December 2019

Copyright © Shang-Tse Chen 2019

# **AI-INFUSED SECURITY: ROBUST DEFENSE BY BRIDGING THEORY AND PRACTICE**

Approved by:

Dr. Duen Horng Chau, Advisor  
School of Computational Science and  
Engineering  
*Georgia Institute of Technology*

Dr. Maria-Florina Balcan, Co-advisor  
School of Computer Science  
*Carnegie Mellon University*

Dr. Wenke Lee  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Le Song  
School of Computational Science and  
Engineering  
*Georgia Institute of Technology*

Dr. Kevin A. Roundy  
*Symantec Research Labs*

Dr. Cory Cornelius  
*Intel Labs*

Date Approved: August 19, 2019

For my parents, brother, and sister.

## ACKNOWLEDGEMENTS

My Ph.D. journey at Georgia Tech has been an amazing experience thanks to the help from many people. Polo, you are my inspiration and role model. Your patience and cheerful attitude has saved me many times from self-doubt. Nina, thank you for your insights and advice on the theoretical side of my research. Having two advisors is an wonderful experience, and I am fortunate enough to be able to learn from the best of both worlds.

I would like to thank my mentors at Symantec and Intel. Kevin, I am impressed by your leadership and deep knowledge in security and machine learning. I really enjoyed brainstorming on our Virtual Product project with you and other Symantec colleagues. Cory, you changed my research topic in a very pleasant way. ShapeShifter has become one of my most well-known works, and it would not have been possible without your help.

My academic siblings in Polo Club, I treasure our friendship and collaboration: Acar Tamersoy, Robert Pienta, Chad Stolper, Elias Khalil, Minsuk Kahng, Fred Hohman, Nilaksh Das, Haekyu Park, Scott Freitas, Jay Wang, Austin Wright, Madhuri Shanbhogue, Peter Polack, Varun Bezzam, Prasenjeet Biswal, Paras Jain, Zhiyuan Lin, Yiqi Chen, Dezhi Fang, Siwei Li, Samuel Clarke, Joon Kim, and Alex Cabrera.

I thank my friends, collaborators, and colleagues for their friendship and help to my research: Wenke Lee, Le Song, Bistra Dilkina, Rich Vuduc, Santosh Vempala, Avrim Blum, Bogdan Carbunar, Yingyu Liang, Chris Berlind, Jason Martin, Michael E. Kounavis, Li Chen, Chris Gates, Yufei Han, Shou-De Lin, Hsuan-Tien Lin, and Chi-Jen Lu.

Special thanks to my roommates Chih-Wei Wu and Shin-Huei Chiou. You make my journey away from home less stressful. I will miss eating dinner, watching TV shows, playing video games, grocery shopping, and discussing crazy ideas with you guys.

Last but not least, my deepest gratitude to my parents, sister, and brother. You are always supportive no matter what I choose to do. I love you all.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	iv
<b>List of Tables</b> . . . . .	xii
<b>List of Figures</b> . . . . .	xvi
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Thesis Overview and Main Ideas . . . . .	1
1.1.1 Part I: Adversarial Attack and Defense of Deep Neural Networks . .	3
1.1.2 Part II: Theoretically-Principled Defense via Game Theory and ML	6
1.1.3 Part III: Applying AI to Protect Enterprise and Society . . . . .	7
1.2 Thesis Statement . . . . .	9
1.3 Research Contributions . . . . .	10
1.4 Impact . . . . .	11
<b>Chapter 2: Survey</b> . . . . .	13
2.1 Security of Machine Learning . . . . .	13
2.2 Applications of Machine Learning to Cybersecurity . . . . .	15
<b>I Adversarial Attack and Defense of Deep Neural Networks</b>	<b>17</b>
<b>Chapter 3: ShapeShifter: Robust Physical Adversarial Attack on Object Detector</b>	<b>19</b>

3.1	Introduction . . . . .	19
3.1.1	Our Contributions . . . . .	21
3.2	Background . . . . .	22
3.2.1	Adversarial Attacks . . . . .	22
3.2.2	Faster R-CNN . . . . .	22
3.3	Threat Model . . . . .	23
3.4	Attack Method . . . . .	24
3.4.1	Attacking an Image Classifier . . . . .	24
3.4.2	Extension to Attacking Faster R-CNN . . . . .	26
3.5	Evaluation . . . . .	27
3.5.1	Digitally Perturbed Stop Sign . . . . .	27
3.5.2	Physical Attack . . . . .	29
3.6	Discussion & Future Work . . . . .	36
3.7	Conclusion . . . . .	38
 <b>Chapter 4: SHIELD: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression . . . . .</b>		 39
4.1	Introduction . . . . .	39
4.1.1	Our Contributions and Impact . . . . .	41
4.2	Background: Adversarial Attacks . . . . .	42
4.3	Proposed Method: Compression as Defense . . . . .	43
4.3.1	Preprocessing Images using Compression . . . . .	44
4.3.2	Vaccinating Models with Compressed Images . . . . .	45
4.3.3	SHIELD: Multifaceted Defense Framework . . . . .	46

4.4	Evaluation . . . . .	47
4.4.1	Experiment Setup . . . . .	47
4.4.2	Defending Gray-Box Attacks with Image Preprocessing . . . . .	49
4.4.3	Black-Box Attack with Vaccination and Ensembling . . . . .	52
4.4.4	Transferability in Black-Box Setting . . . . .	54
4.4.5	NIPS 2017 Competition Results . . . . .	55
4.5	Significance and Impact . . . . .	55
4.5.1	Software and Hardware Integration Milestones . . . . .	56
4.5.2	New Computational Paradigm: Secure Deep Learning . . . . .	56
4.5.3	Scope and Limitations . . . . .	57
4.6	Related Work . . . . .	57
4.6.1	Image Preprocessing as Defense . . . . .	58
4.6.2	Attacks against Preprocessing Techniques . . . . .	58
4.7	Conclusion . . . . .	59

**Chapter 5: UnMask: Adversarial Detection and Defense in Deep Learning Through Building-Block Knowledge Extraction . . . . . 61**

5.1	Introduction . . . . .	61
5.1.1	Contributions . . . . .	63
5.2	UNMASK: Detection and Defense Framework . . . . .	64
5.2.1	Intuition: Protection via Building-Block Knowledge Extraction . . . . .	66
5.2.2	Overview of UNMASK . . . . .	66
5.2.3	Technical Walk-Through of UNMASK . . . . .	67
5.3	Evaluation . . . . .	68

5.3.1	Experiment Setup . . . . .	69
5.3.2	Evaluating UNMASK Defense and Detection . . . . .	72
5.4	Conclusion & Discussion . . . . .	74
<b>II</b>	<b>Theoretically-Principled Defense via Game Theory and ML</b>	<b>81</b>
<b>Chapter 6:</b>	<b>Diversified Strategies for Mitigating Adversarial Attacks in Multia-</b>	
	<b>gent Systems . . . . .</b>	<b>83</b>
6.1	Introduction . . . . .	83
6.1.1	Related Work . . . . .	86
6.2	Zero-Sum Games . . . . .	87
6.2.1	Multiplicative Weights and Diversified Strategies . . . . .	87
6.2.2	Diversifying Dynamics . . . . .	90
6.2.3	How Close Is $v_\epsilon$ to $v$ ? . . . .	93
6.3	General-Sum Games . . . . .	95
6.3.1	The Benefits of Diversification . . . . .	95
6.3.2	The Diversified Price of Anarchy . . . . .	97
6.3.3	The Cost of Diversification . . . . .	101
6.4	Distributed Setting . . . . .	103
6.5	Experiments . . . . .	105
6.6	Conclusion . . . . .	107
<b>Chapter 7:</b>	<b>Communication Efficient Distributed Agnostic Boosting . . . . .</b>	<b>108</b>
7.1	Introduction . . . . .	108
7.2	Problem Setup . . . . .	111

7.2.1	Statistical Learning Problem . . . . .	111
7.2.2	Extension to the Distributed Setting . . . . .	112
7.3	Distributed Agnostic Boosting . . . . .	112
7.3.1	Agnostic Boosting: Centralized Version . . . . .	113
7.3.2	Agnostic Boosting: Distributed Version . . . . .	115
7.4	Experiments . . . . .	120
7.4.1	Experiment Setup . . . . .	120
7.4.2	Synthetic Dataset . . . . .	120
7.4.3	Real-world Datasets . . . . .	121
7.5	Conclusions . . . . .	122

### **III Applying AI to Protect Enterprise and Society 124**

#### **Chapter 8: Predicting Cyber Threats with Virtual Security Products . . . . . 126**

8.1	Introduction . . . . .	126
8.2	Related Work . . . . .	130
8.3	Proposed Model: Virtual Product . . . . .	132
8.3.1	Semi-Supervised Non-negative Matrix Factorization (SSNMF) . . .	133
8.3.2	Optimization Algorithm . . . . .	135
8.4	Evaluation . . . . .	137
8.4.1	Data Collection . . . . .	137
8.4.2	Experiment Setup and Overview . . . . .	139
8.4.3	Evaluation on Reconstruction Accuracy . . . . .	140
8.4.4	Evaluation: Incident Detection and Categorization . . . . .	142

8.4.5	Evaluation of Computational Cost . . . . .	146
8.4.6	Improvement in Analyst Response Predictions . . . . .	147
8.5	Impact and Deployment . . . . .	148
8.5.1	Case Studies and Impact . . . . .	148
8.5.2	Deployment . . . . .	153
8.6	Conclusions and Discussion . . . . .	154
 <b>Chapter 9: Firebird: Predicting Fire Risk and Prioritizing Fire Inspections in Atlanta . . . . .</b>		
9.1	Introduction . . . . .	155
9.2	Related Work . . . . .	159
9.3	Data Description . . . . .	161
9.3.1	Data Sources . . . . .	161
9.3.2	Data Joining . . . . .	163
9.4	Identifying New Properties Needing Inspection . . . . .	164
9.5	Predictive Model of Fire Risk . . . . .	166
9.5.1	Data Cleaning . . . . .	166
9.5.2	Feature Selection . . . . .	166
9.5.3	Evaluation of the Models . . . . .	167
9.5.4	Further Discussion of the Models . . . . .	169
9.5.5	Assignment of Risk Scores . . . . .	170
9.6	Impact On AFRD and Atlanta . . . . .	172
9.6.1	Previous Inspection Process . . . . .	172
9.6.2	Technology Transfer to AFRD . . . . .	172

9.6.3	Impact on AFRD Processes . . . . .	174
9.7	Challenges . . . . .	176
9.8	Conclusions and Future Research Directions . . . . .	178
<b>Chapter 10:</b>	<b>Conclusions . . . . .</b>	<b>181</b>
10.1	Contributions . . . . .	181
10.2	Future Research Directions . . . . .	182
<b>References</b>	<b>. . . . .</b>	<b>199</b>

## LIST OF TABLES

1.1	Thesis outline, and publications contributing to each part. . . . .	2
3.1	Our <b>high-confidence</b> perturbations succeed at attacking at a variety of distances and angles. For each distance-angle combination, we show the detected class and the confidence score. If more than one bounding boxes are detected, we report the highest-scoring one. Confidence values lower than 30% is considered undetected. . . . .	31
3.2	As expected, low-confidence perturbations achieve lower success rates. . . .	32
3.3	Sample high-confidence perturbations from indoor experiments. For complete experiment results, please refer to Table 3.1. . . . .	33
3.4	Black-box transferability of our 3 perturbations. We report the number of images (of the 15 angle-distance images) that failed to transfer to the specified model. We consider the detection of any stop sign a “failure to transfer.” Our perturbations fail to transfer for most models, most likely due to the iterative nature of our attack. . . . .	36
4.1	Summary of model accuracies (in %) for all defenses: SHIELD [20, 40, 60, 80], JPEG, median filter (MF), and total variation denoising (TVD); v/s all attacks: Carlini-Wagner L2 (CW-L2), DeepFool (DF), I-FGSM and FGSM. While all techniques are able to recover accuracies from CW-L2 and DF attacks with lower perturbation strength, the best performing settings are from JPEG (in bold font). SHIELD benefits from the combination of SLQ, vaccination and ensembling, and outperforms all other techniques when facing high perturbation delivered by I-FGSM and FGSM. We use $\kappa = 0$ in <b>CW-L2</b> and $\epsilon = 4$ in <b>FGSM</b> and <b>I-FGSM</b> . . . . .	51

4.2	Comparison of two ensemble schemes with SHIELD, when defending against FGSM. $\mathcal{M}_q \times q$ corresponds to each model $\mathcal{M}_q$ voting on each JPEG quality $q$ from $q \in \{20, 30, 40, \dots, 90\}$ . In $\mathcal{M}_q - q$ , each model $\mathcal{M}_q$ votes only on $q$ , the JPEG quality it was trained on. SHIELD offers a favorable trade-off, providing at least 2x speed-up as compared to larger ensembles, while delivering comparable accuracies. . . . .	54
4.3	JPEG compression as defense does not reduce model accuracy significantly on transferred attacks with low perturbation. Adversarial images crafted using the ResNet-v2 50 model are protected using <i>JPEG</i> alone and <i>Stochastic Local Quantization</i> (SLQ), before being fed into two other models: Inception-v4 (Inc-v4) and ResNet-v2 101 (RN-v2 101). . . . .	55
5.1	Symbols and Definition . . . . .	65
5.2	Class-Feature Matrix. Top: dots mark classes' features. Bottom: four class sets with varying levels of feature overlap. . . . .	78
5.3	Number of images used in training models $K$ and $M$ . . . . .	79
5.4	Number of ImageNet images used to evaluate UNMASK. Only the images that can be successfully perturbed by the attack are used, thus the variations in numbers. We report values for PGD and FGSM with $\epsilon=16$ . The numbers for $\epsilon=8$ are similar. . . . .	79
5.5	Four <i>class sets</i> investigated in our evaluation, with varying number of classes and feature overlap. . . . .	79
5.6	UNMASK's accuracies (in %) in countering three attacks: DeepFool, FGSM, and <i>Projected Gradient Descent</i> (PGD). We test two popular CNN architectures, VGG16 and ResNet50, as unprotected model $M$ , with four class sets with varying numbers of classes and feature overlap. We use $\epsilon = 16$ for FGSM and PGD in this experiment. We show the models' accuracies (1) when not under attack ("No Attk" column); (2) attacked without defense ("DeepFool (No Def)"); for both FGSM and PGD, the accuracies drop to 0 without defense and are omitted in this table; and (3) attacked and defended by UNMASK (the last three columns). . . . .	80
7.1	Average (over 10 trials) error rate (%) and standard deviation on the synthetic dataset . . . . .	121
7.2	Average (over 10 trials) error rate (%) and standard deviation on real-world datasets . . . . .	121

7.3	Average run time (sec) on real-world datasets . . . . .	122
8.1	A long list of inconclusive alerts generated in a real incident of a machine infected by the infamous <i>Zbot Trojan</i> . These alerts overwhelm a cybersecurity analyst, and do not help answer important questions such as: <i>Is this machine compromised? How severe is the attack? What actions should be taken?</i> Our technique, <i>Virtual Product</i> , correctly predicts the presence of the infamous Zbot Trojan, which would have been identified by an AV product, had it been installed. . . . .	127
8.2	Terminology used in this chapter . . . . .	131
8.3	Summary of the training datasets (Jul-Sept) for the top five products that detect the most incidents. . . . .	137
8.4	Summary of validation datasets (Oct-Dec). . . . .	139
8.5	Performance of reconstruction on all five datasets . . . . .	141
8.6	Our approach (VP) detects security incidents with high accuracies (AUCs) across all five datasets, outperforming the baseline model (LR). . . . .	143
8.7	True positive rate (TPR) of incident detection on all five data sets at 10% false positive rate (FPR). Our approach (VP) outperforms the baseline (LR) . . . . .	143
8.8	Average F1 scores of incident categorization on our datasets. We do not include <i>EP2</i> because over 99% of the detected incidents belong to one single incident type. . . . .	146
8.9	Average virtual product training times (over 10 runs). . . . .	147
8.10	Virtual Product correctly predicts that FirewallB would have detected an incident, and 10 of its top 11 predicted alerts coincide with the one that actually occurred, yielding a clearer picture of the artifacts involved in the attack and the vulnerabilities used. The incorrect prediction is shown in <del>strikeout font</del> . . . . .	149
8.11	An attack on a webserver is obviously underway, but was it successful? Virtual Product correctly predicts, with 99.9% confidence, that not only a deployed AV product would detect attacks on the machine, but predict successful infection of the system. . . . .	150

8.12	There are indications of possible ransomware activity, but how did the attack appear on the machine in the first place? Virtual Product correctly indicates that a malicious spreadsheet (detected as Bloodhound.Exploit.170) was at fault, a method by which the Locky RansomWare has been known to propagate.	151
9.1	Summary of inspection and building lists . . . . .	161
9.2	Firebird Data Sources Summary . . . . .	162
9.3	Testing AUC of each year . . . . .	170
9.4	Top-10 features in Random Forest . . . . .	171

## LIST OF FIGURES

1.1	My work on physical adversarial attack discovers a serious vulnerability of DNNs in a more realistic threat model where the attacker does not need to have control over the internal computer vision system pipeline. The crafted physical adversarial objects (e.g., fake stop signs) can fool the state-of-the-art object detectors. . . . .	3
1.2	Snapshots of a drive-by test result. The real stop sign is correctly predicted by Faster R-CNN with high confidence. The adversarial stop sign crafted by <i>ShapeShifter</i> is detected as the target class “person.” . . . .	4
1.3	<i>UnMask</i> combats adversarial attacks (in red) by extracting <i>building-block knowledge</i> (e.g., <i>wheel</i> ) from the image (top, in green), and comparing them to expected features of the classification (“Bird” at bottom) from the unprotected model. Low feature overlap signals attack. <i>UnMask</i> rectifies misclassification using the image’s extracted features. Our approach <i>detects</i> 92.9% of gray-box attacks (at 9.67% false positive rate) and <i>defends</i> the model by correctly classifying up to 92.24% of adversarial images crafted by the strongest attack, Projected Gradient Descent. . . . .	5
1.4	For $\epsilon \in [\frac{1}{n}, 1]$ , we define a probability distribution $P$ to be $\epsilon$ -diversified if $P(i) \leq \frac{1}{\epsilon n}$ for all $i$ . A distribution can be diversified through a Bregman projection onto the set of all $\epsilon$ -diversified distributions. A mixed strategy determined by a diversified distribution is called a diversified (mixed) strategy. We explore properties of such diversified strategies in both zero-sum and general-sum games as well as give algorithmic guarantees. . . . .	6
1.5	Our distributed SmoothBoost algorithm. In each iteration, (1) each machine samples its own data based on the current data distribution and sends it to the Center; (2) Center trains an ML model by using some weak learning algorithm and broadcasts the trained model to all the machines; (3) each machine updates its data distribution (i.e., example weights) based on received model, and performs distributed Bregman projection to ensure the distribution is diversified. All the weak models are combined at the end to obtain a strong model. . . . .	7

1.6	<i>Virtual Product</i> helps our user Sam discover and understand cyber-threats, and informs deployment decisions (e.g., add firewall?) through semi-supervised non-negative matrix factorization on telemetry data from other users (with firewalls deployed). In the data matrix, each row represents a machine-day, and each column a security event's occurrences. Missing events from undeployed products are shown as gray blocks. The last column indicates whether the firewall has detected an incident. Our <i>virtual</i> firewall serves as a proxy to the actual firewall and predicts the occurrence of security events and incidents Sam might observe (dark green block) if he deploys the firewall. . . . .	8
1.7	<b>Firebird Framework Overview.</b> By combining 8 datasets, Firebird identifies new commercial properties for fire inspections. Its fire risk predictive models (SVM, random forest) and interactive map help AFRD prioritize fire inspections and personnel allocation. . . . .	9
3.1	Illustration motivating the need of physical adversarial attack, because attackers typically do not have full control over the computer vision system pipeline. . . . .	20
3.2	Digital perturbations we created using our method. Low confidence perturbations on the top and high confidence perturbations on the bottom. . . . .	29
3.3	<b>Indoor experiment setup.</b> We take photos of the printed adversarial sign, from multiple angles ( $0^\circ$ , $15^\circ$ , $30^\circ$ , $45^\circ$ , $60^\circ$ , from the sign's tangent), and distances (5' to 40'). The camera locations are indicated by the red dots, and the camera always points at the sign. . . . .	30
3.4	Snapshots of the drive-by test results. In (a), the person perturbation was detected 47% of the frames as a person and only once as a stop sign. The perturbation in (b) was detected 36% of the time as a sports ball and never as a stop sign. The untargeted perturbation in (c) was detected as <i>bird</i> 6 times and never detected as a stop sign or anything else for the remaining frames. . . . .	35
3.5	Example stop signs from the MS-COCO dataset. Stop signs can vary by language, by degree of occlusion by stickers or modification by graffiti, or just elements of the weather. Each stop sign in the images is correctly detected by the object detector with high confidence (99%, 99%, 99%, and 64%, respectively). . . . .	36

4.1	SHIELD Framework Overview. SHIELD combats adversarial images (in red) by removing perturbation in real-time using Stochastic Local Quantization (SLQ) and an ensemble of vaccinated models which are robust to the compression transformation. Our approach eliminates up to 98% of gray-box attacks delivered by strong adversarial techniques such as <i>Carlini-Wagner's L2</i> attack and <i>DeepFool</i> . . . . .	40
4.2	SHIELD uses Stochastic Local Quantization (SLQ) to remove adversarial perturbations from input images. SLQ divides an image into $8 \times 8$ blocks and applies a randomly selected JPEG compression quality (20, 40, 60 or 80) to each block to mitigate the attack. . . . .	45
4.3	Carlini-Wagner-L2 (CW-L2) and DeepFool, two recent strong attacks, introduce perturbations that lowers model accuracy to around 10% ( $\emptyset$ ). JPEG compression recovers up to 98% of the original accuracy (with DeepFool), while SHIELD achieves similar performance, recovering up to 95% of the original accuracy (with DeepFool). . . . .	47
4.4	SHIELD recovers the accuracy of the model when attacked with I-FGSM (left) and FGSM (right). Both charts show the accuracy of the model when undefended (gray dotted curve). Applying varying JPEG compression qualities (purple curves) helps recover accuracy significantly, and SHIELD (orange curve) is able to recover more than any single JPEG-defended model. . . . .	49
4.5	Runtime comparison for three defenses: (1) total variation denoising (TVD), (2) median filter (MF), and (3) JPEG compression, timed using the full 50k ImageNet validation images, averaged over 3 runs. JPEG is at least 22x faster than TVD, and 14x faster than MF. . . . .	52
4.6	Vaccinating a model by retraining it with compressed images helps recover its accuracy. Each plot shows the model accuracies when preprocessing with different JPEG qualities with the FGSM attack. Each curve in the plot corresponds to a different model. The gray dotted curve corresponds to the original unvaccinated ResNet-v2 50 model. The orange and purple curves correspond to the models retrained on JPEG qualities 80 and 20 respectively. Retraining on JPEG compressed images and applying JPEG preprocessing helps recover accuracy in a gray-box attack. . . . .	54

5.1	UNMASK framework overview. UNMASK combats adversarial attacks (in red) by extracting <i>building-block knowledge</i> (e.g., <i>wheel</i> ) from the image (top, in green), and comparing them to expected features of the classification (“Bird” at bottom) from the unprotected model. Low feature overlap signals attack. UNMASK rectifies misclassification using the image’s extracted features. Our approach <i>detects</i> 92.9% of gray-box attacks (at 9.67% false positive rate) and <i>defends</i> the model by correctly classifying up to 92.24% of adversarial images crafted by the state-of-the-art Projected Gradient Descent attack. . . . .	62
5.2	UNMASK guards against adversarial image perturbation by extracting building-block features from an image and comparing them to its expected features using Jaccard similarity. If the similarity is below a threshold, UNMASK deems the image adversarial and predicts its class by matching the extracted features with the most similar class. . . . .	67
5.3	UNMASK’s effectiveness in detecting detecting three attacks: DeepFool, FGSM ( $\epsilon=8,16$ ), and PGD ( $\epsilon=8,16$ ). UNMASK’s protection may not be affected strictly based on the number of classes. Rather, an important factor is the <i>feature overlap</i> among classes. UNMASK provides better detection when there are 5 classes (dark orange; 23.53% overlap) than when there are 3 (light blue; 50% overlap). Keeping the number of classes constant and varying their feature overlap also supports our observation about the role of feature overlap (e.g., CS3a at 6.89% vs. CS3b at 50%). . . . .	71
6.1	For $\epsilon \in [\frac{1}{n}, 1]$ , we define a probability distribution $P$ to be $\epsilon$ -diversified if $P(i) \leq \frac{1}{\epsilon n}$ for all $i$ . A distribution can be diversified through Bregman projection into the set of all $\epsilon$ -diversified distributions. A mixed strategy determined by a diversified distribution is called a diversified (mixed) strategy. We explore properties of such diversified strategies in both zero-sum and general-sum games as well as give algorithmic guarantees. . . . .	84
6.2	Braess’ paradox. Here, $k$ players wish to travel from $s$ to $t$ , and requiring all players to use diversified strategies improves the quality of the equilibrium for everyone. . . . .	97
6.3	Simulated results of Braess’ paradox after $T = 10,000$ rounds. A more diversified strategy leads to lower loss. . . . .	106
6.4	Average reward over $T = 10,000$ rounds with different values of $\epsilon$ . When the rare event happens, the non-diversified strategy gains very low (even negative) reward. . . . .	106

8.1	<i>Virtual Product</i> helps our user Sam discover and understand cyber-threats, and informs deployment decisions (e.g., add firewall?) through semi-supervised non-negative matrix factorization on telemetry data from other users (with firewalls deployed). In the data matrix, each row represents a machine-day, and each column a security event's occurrences. Missing events from undeployed products are shown as gray blocks. The last column indicates if the firewall has detected an incident. Our <i>virtual</i> firewall serves as a proxy to the actual product and predicts the output Sam may observe (dark green block) if he deploys it. . . . .	129
8.2	Averaged ROC curves from 10-fold cross-validation of <i>Virtual Product</i> on our top five product datasets. . . . .	144
8.3	ROC curves of the Virtual Product model evaluated using the validation datasets of the five products. . . . .	145
9.1	<b>Firebird Framework Overview.</b> By combining 8 datasets, Firebird identifies new commercial properties for fire inspections. Its fire risk predictive models (SVM, random forest) and interactive map help AFRD prioritize fire inspections and personnel allocation. . . . .	156
9.2	Joining eight datasets using three spatial information types (geocode, address, parcel ID). . . . .	158
9.3	ROC curves of Random Forest and SVM . . . . .	168
9.4	<b>Interactive map of fires and inspections.</b> The colored circles on the map represent fire incidents, currently inspected properties, and potentially inspectable properties in red, green, and blue, respectively. Inspectors can filter the displayed properties based on property usage type, date of fire or inspection, and fire risk score. <i>Callout:</i> activating the Neighborhood Planning Unit overlay allows an inspector to mouse-over a political subdivision of the city to view its aggregate and percentage of the fires, inspections, and potential inspections. . . . .	173

## SUMMARY

While Artificial Intelligence (AI) has tremendous potential as a defense against real-world cybersecurity threats, understanding the capabilities and robustness of AI remains a fundamental challenge. This dissertation tackles problems essential to successful deployment of AI in security settings and is comprised of the following three interrelated research thrusts.

(1) **Adversarial Attack and Defense of Deep Neural Networks:** We discover vulnerabilities of deep neural networks in real-world settings and the countermeasures to mitigate the threat. We develop *ShapeShifter*, the first targeted physical adversarial attack that fools state-of-the-art object detectors. For defenses, we develop *SHIELD*, an efficient defense leveraging stochastic image compression, and *UnMask*, a knowledge-based adversarial detection and defense framework.

(2) **Theoretically-Principled Defense via Game Theory and ML:** We develop new theories that guide defense resources allocation to guard against unexpected attacks and catastrophic events, using a novel online decision-making framework that compels players to employ “diversified” mixed strategies. Furthermore, by leveraging the deep connection between game theory and boosting, we develop a communication-efficient distributed boosting algorithm with strong theoretical guarantees in the agnostic learning setting.

(3) **Using AI to Protect Enterprise and Society:** We show how AI can be used in real enterprise environment with a novel framework called *Virtual Product* that predicts potential enterprise cyber threats. Beyond cybersecurity, we also develop the *Firebird* framework to help municipal fire departments prioritize fire inspections.

Our work has made multiple important contributions to both theory and practice: our distributed boosting algorithm solved an open problem of distributed learning; *ShaperShifter* motivated a new DARPA program (GARD); *Virtual Product* led to two patents; and *Firebird* was highlighted by National Fire Protection Association as a best practice for using data to inform fire inspections.

# CHAPTER 1

## INTRODUCTION

Internet-connected devices, such as mobile phones and smart home systems, have become ubiquitous in our everyday lives. The increased connectivity also presents new cybersecurity challenges and creates significant national risks. The number of cyber incidents on federal systems reported to the U.S. Department of Homeland Security increased more than ten-fold between 2006 and 2015 [1].

To defend against these daunting and ever-increasing attacks, artificial intelligence (AI) and machine learning (ML) have been explored and employed by cybersecurity researchers and practitioners. However, even today, researchers have not yet fully understood the complex ML models and their capabilities in solving various real-world tasks. The goal of this thesis is to gain a deeper understanding of the capabilities and limitations of AI in security-critical tasks, so that we can develop resilient AI-powered next-generation cybersecurity defenses.

### 1.1 Thesis Overview and Main Ideas

Many cybersecurity scenarios can be modeled as a game between the defender and the attacker. To design the best security solution, we need to fully understand the capabilities and limitations from both the defense and attack point of views, and how they interact with each other. Recent advances in AI provide



great opportunities to fortify security-critical applications. However, AI may also pose new threats and challenges. To solve these challenges, my research innovates at the intersection of AI, cybersecurity, and algorithmic game theory. My thesis includes three parts of research,

spanning the theory and application parts of cybersecurity. I make contributions to both the defensive and attacking sides of cybersecurity. Table 1.1 provides a brief overview of my dissertation.

Table 1.1: Thesis outline, and publications contributing to each part.

## **Part I: Adversarial Attack and Defense of Deep Neural Networks (Chapter 3, 4, 5)**

§ ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector.

**Shang-Tse Chen**, Cory Cornelius, Jason Martin, Duen Horng Chau. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018.

§ Shield: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression.

Nilaksh Das, Madhuri Shanbhogue, **Shang-Tse Chen**, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, Duen Horng Chau. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

§ Extracting Knowledge For Adversarial Detection and Defense in Deep Learning.

Scott Freitas, **Shang-Tse Chen**, Duen Horng Chau. In *KDD 2019 Workshop on Learning and Mining for Cybersecurity (LEMINCS)*, 2019.

## **Part II: Theoretically-Principled Defense via Game Theory and ML (Chapter 6, 7)**

§ Diversified Strategies for Mitigating Adversarial Attacks in Multiagent Systems.

Maria-Florina Balcan, Avrim Blum, **Shang-Tse Chen**. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

§ Communication Efficient Distributed Agnostic Boosting.

**Shang-Tse Chen**, Maria-Florina Balcan, Duen Horng Chau. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

## **Part IV: Applying AI to Protect Enterprise and Society (Chapter 8, 9)**

§ Predicting Cyber Threats with Virtual Security Products.

**Shang-Tse Chen**, Yufei Han, Duen Horng Chau, Christopher Gates, Michael Hart, Kevin Roundy. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2017.

§ Firebird: Predicting Fire Risk and Prioritizing Fire Inspections in Atlanta.

Michael Madaio, **Shang-Tse Chen**, Oliver Haimson, Wenwen Zhang, Xiang Cheng, Matthew Hinds-Aldrich, Duen Horng Chau, and Bistra Dilkina. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

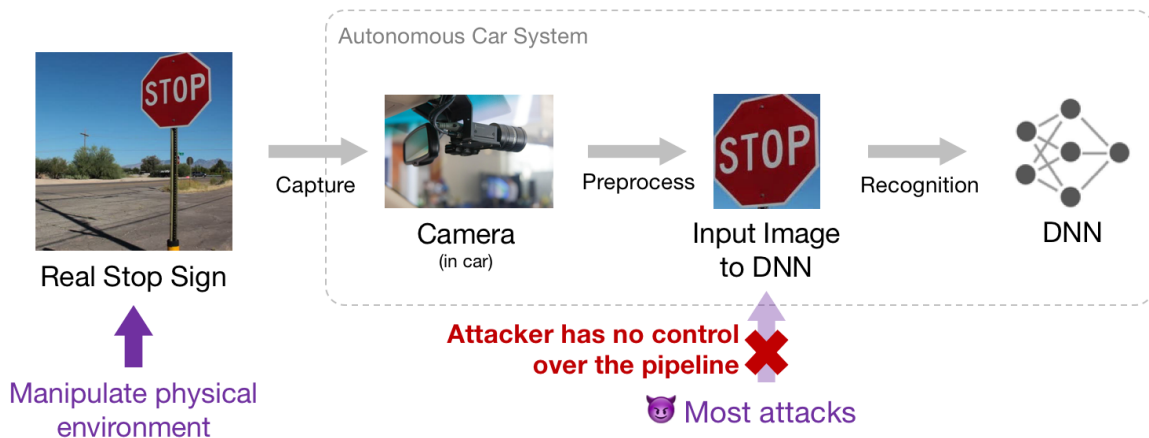


Figure 1.1: My work on physical adversarial attack discovers a serious vulnerability of DNNs in a more realistic threat model where the attacker does not need to have control over the internal computer vision system pipeline. The crafted physical adversarial objects (e.g., fake stop signs) can fool the state-of-the-art object detectors.

### 1.1.1 Part I: Adversarial Attack and Defense of Deep Neural Networks

Recent advances in deep neural networks (DNNs) have generated much optimism about deploying AI in safety-critical applications, such as self-driving cars. However, it has recently been discovered that given the ability to directly manipulate image pixels in the digital input space, an adversary can easily generate imperceptible perturbations to fool a DNN image classifier [2].

Although many adversarial attack algorithms have been proposed [3, 4], attacking a real-world computer vision system is difficult, because attackers usually do not have the ability to directly manipulate data inside such systems (Figure 1.1). To understand the vulnerabilities of DNN-based computer vision systems, I collaborated with Intel and developed **ShapeShifter** [5], the **first targeted physical adversarial attack on the state-of-the-art Faster R-CNN object detectors**.

Attacking an object detector is more difficult than attacking an image classifier, as the attack needs to mislead the classifications of multiple bounding boxes at different scales. Extending a digital attack to the physical world adds another layer of difficulty; this requires the perturbation to be sufficiently robust to survive real-world distortions due to different

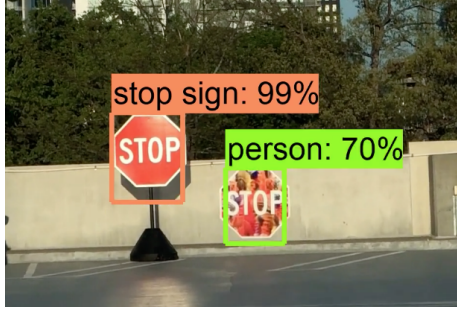


Figure 1.2: Snapshots of a drive-by test result. The real stop sign is correctly predicted by Faster R-CNN with high confidence. The adversarial stop sign crafted by *ShapeShifter* is detected as the target class “person.”

viewing distances and angles, lighting conditions, and camera limitations.

*ShapeShifter* generates adversarial stop signs that were consistently mis-detected by Faster R-CNN as the target objects in real drive-by tests (Figure 1.2), posing a potential threat to autonomous vehicles and other safety-critical computer vision systems. Our code is open-sourced and the drive-by test videos are publicly available<sup>1</sup>. *ShapeShifter* was **highlighted as the state-of-the-art physical adversarial attack in the recent DARPA program “Guaranteeing AI Robustness against Deception” (GARD)** that focuses on defending against such kind of attacks.

there have been many attempts to mitigate the threat.

Although there have been many attempts to mitigate adversarial attacks, completely protecting a DNN model from adversarial attacks remains an open problem. Most methods suffer from significant computational overhead or sacrifice accuracy on benign data. In collaboration with Intel, we developed **SHIELD** [6], a practical defense leveraging stochastic compression that removes adversarial perturbations. *SHIELD* makes multiple positive impacts on Intel’s research and product development plans. Utilizing Intel’s Quick Sync Video (QSV) technology with dedicated hardware for high-speed video processing, we pave the way for real-time defense in safety-critical applications, such as autonomous vehicles. Our research sparked insightful discussion at Intel about *secure deep learning* that necessitates tight integration of practical defense strategies, software platforms, and

<sup>1</sup><https://github.com/shangtse/robust-physical-attack>

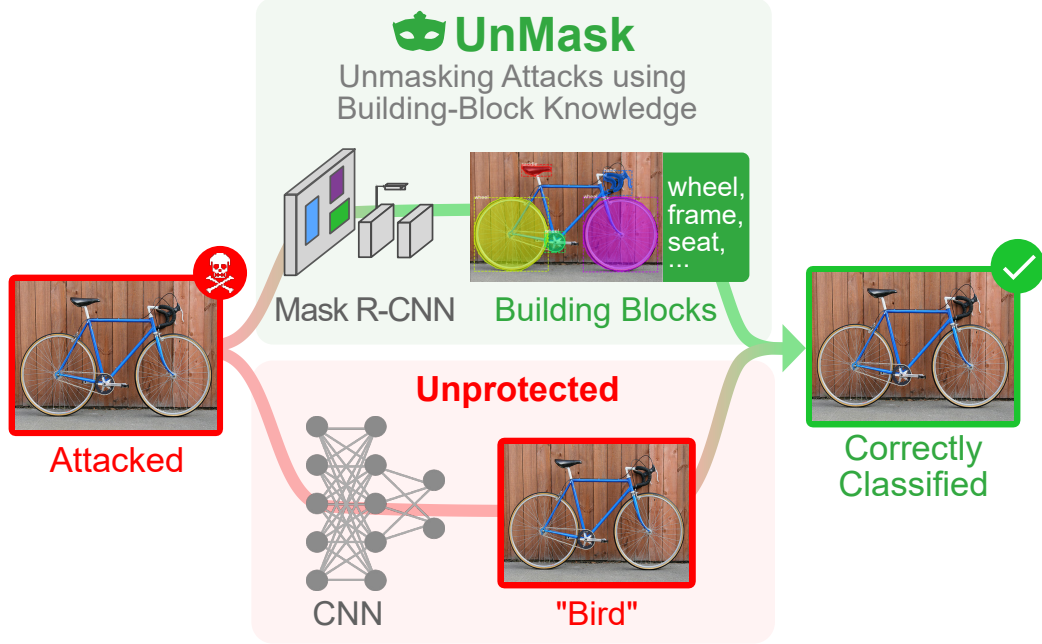


Figure 1.3: *UnMask* combats adversarial attacks (in red) by extracting *building-block knowledge* (e.g., *wheel*) from the image (top, in green), and comparing them to expected features of the classification (“Bird” at bottom) from the unprotected model. Low feature overlap signals attack. *UnMask* rectifies misclassification using the image’s extracted features. Our approach *detects* 92.9% of gray-box attacks (at 9.67% false positive rate) and *defends* the model by correctly classifying up to 92.24% of adversarial images crafted by the strongest attack, Projected Gradient Descent.

hardware accelerators. Our work will accelerate the industry’s emphasis on this important topic. Both *ShapeShifter* and *SHIELD* have been incorporated into **MLsploit** [7], an open-sourced ML evaluation and fortification framework designed for education and research. These two works are also part of the **Intel AI Academy course**.

*Shield* is best suited for defending against imperceptible perturbations. To defend against *ShapeShifter*-style attacks, we developed **UnMask**, a knowledge-based adversarial detection and defense framework. *UnMask* protects models by verifying that an image’s predicted class (e.g., “bird”) contains the expected building blocks (e.g., beak, wings, eyes). For example, if an image is classified as “bird”, but the extracted building blocks are *wheel*, *seat* and *frame*, the model may be under attack. When *UnMask* detects such attacks, it can rectify the misclassification by re-classifying the image based on its extracted building blocks (Figure 1.3).

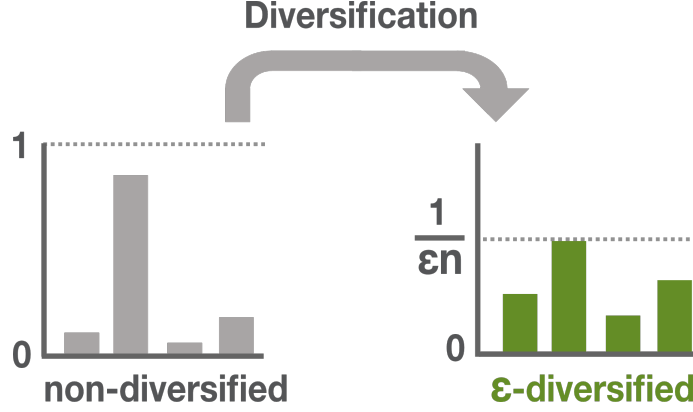


Figure 1.4: For  $\epsilon \in [\frac{1}{n}, 1]$ , we define a probability distribution  $P$  to be  $\epsilon$ -diversified if  $P(i) \leq \frac{1}{\epsilon n}$  for all  $i$ . A distribution can be diversified through a Bregman projection onto the set of all  $\epsilon$ -diversified distributions. A mixed strategy determined by a diversified distribution is called a diversified (mixed) strategy. We explore properties of such diversified strategies in both zero-sum and general-sum games as well as give algorithmic guarantees.

### 1.1.2 Part II: Theoretically-Principled Defense via Game Theory and ML

Defense resource allocation is a well-known and critical task in security. For example, a company that wants to implement security controls with a limited budget needs to make trade-offs in its deployment. I modeled this problem as a two-player zero-sum game between a defender and an attacker, and introduced a novel solution concept called **diversified mixed strategy** [8].

Inspired by the proverb “don’t put all your eggs in one basket,” my new solution concept compels players to employ a “diversified” strategy that does not place too much weight on any one action. I systematically studied properties of diversified strategies in multiple games, and designed efficient algorithms that asymptotically achieve the optimum reward within the family of diversified strategies. As a result, these algorithms limit the exposure to adversarial or catastrophic events while still performing successfully in typical cases.

Leveraging the deep connection between game theory, online learning, and boosting, I proved that the proposed *diversified strategy* concept can also be used to help learn robust and efficient ML models. Specifically, I **solved an open problem** listed in [9] by developing a boosting-based approach [10] in one of the hardest and most general settings in distributed

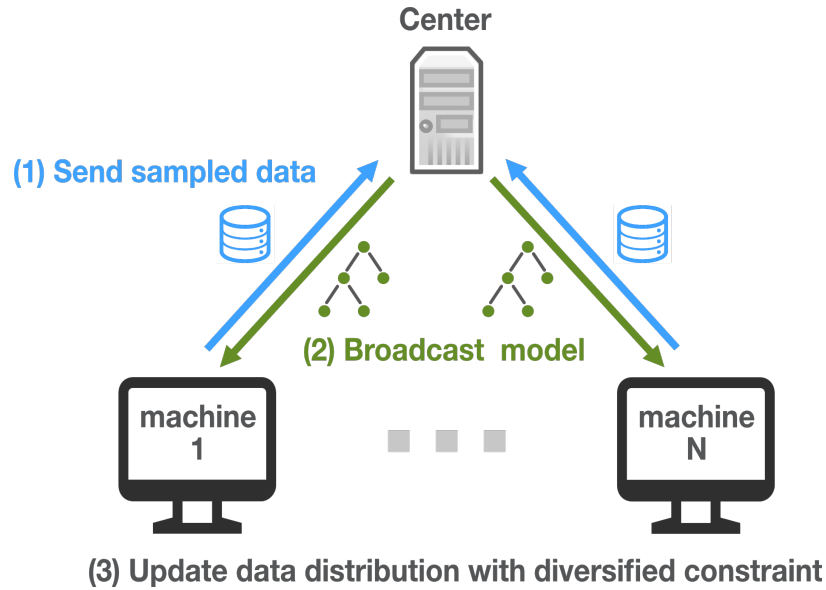


Figure 1.5: Our distributed SmoothBoost algorithm. In each iteration, (1) each machine samples its own data based on the current data distribution and sends it to the Center; (2) Center trains an ML model by using some weak learning algorithm and broadcasts the trained model to all the machines; (3) each machine updates its data distribution (i.e., example weights) based on received model, and performs distributed Bregman projection to ensure the distribution is diversified. All the weak models are combined at the end to obtain a strong model.

learning, where data is adversarially partitioned and distributed across multiple locations, and can have arbitrary forms of noise (Figure 1.5). Succinctly, since boosting algorithms tend to place too much weight on outliers, we can project the weights back to the set of *diversified* distributions at the end of each boosting iteration. Our algorithm is simultaneously noise tolerant, communication efficient, and computationally efficient. This is a significant improvement over prior works that either were only communication efficient in noise-free scenarios or were computationally prohibitive. Our distributed boosting algorithm is not only theoretically principled but also demonstrates excellent accuracy on real-world datasets.

### 1.1.3 Part III: Applying AI to Protect Enterprise and Society

Part I and II provide theories, algorithms, and insight of the capabilities and limitations of AI. But how can we put AI into practice and utilize it to provide solutions that solve real enterprise security problems and create positive societal impacts? In collaboration

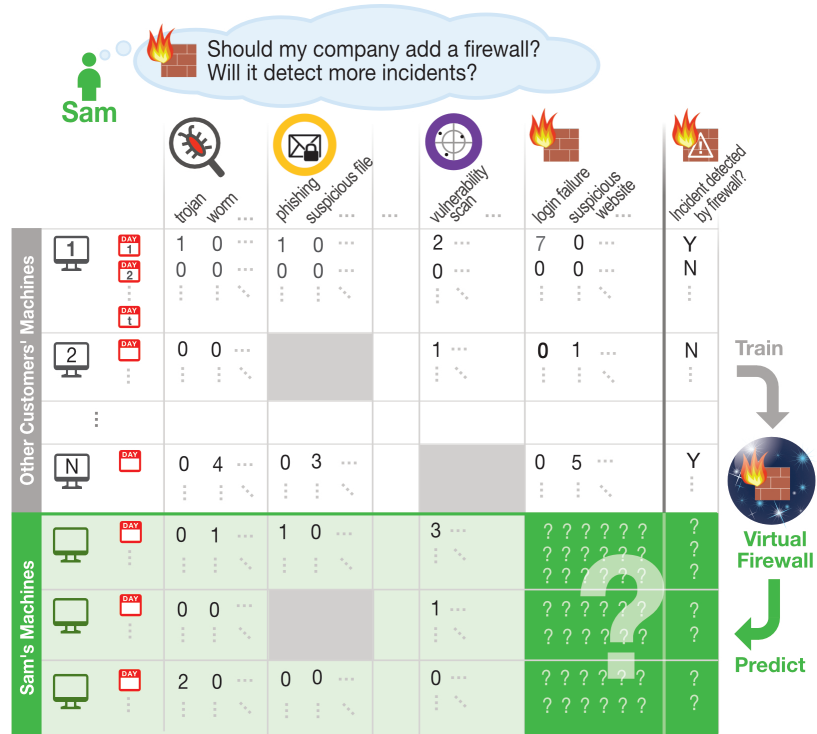


Figure 1.6: *Virtual Product* helps our user Sam discover and understand cyber-threats, and informs deployment decisions (e.g., add firewall?) through semi-supervised non-negative matrix factorization on telemetry data from other users (with firewalls deployed). In the data matrix, each row represents a machine-day, and each column a security event's occurrences. Missing events from undeployed products are shown as gray blocks. The last column indicates whether the firewall has detected an incident. Our *virtual* firewall serves as a proxy to the actual firewall and predicts the occurrence of security events and incidents Sam might observe (dark green block) if he deploys the firewall.

with Symantec, we develop the **patented Virtual Product** framework, the first method to predict security events and high-severity incidents that would have been identified by a security product if it had been deployed. This is made possible by learning from the vast amounts of telemetry data produced by the prevalent defense-in-depth approach to computer security, wherein multiple security products are deployed alongside each other, producing highly correlated alert data. By studying this data, we are able to accurately predict which security alerts a product would have triggered in a particular situation, even though it was not deployed. See Figure 1.6 for the overview of our approach.

Beyond cybersecurity, I further explored novel applications of AI in various domains that create positive societal impacts. In collaboration with the Atlanta Fire Rescue Department,

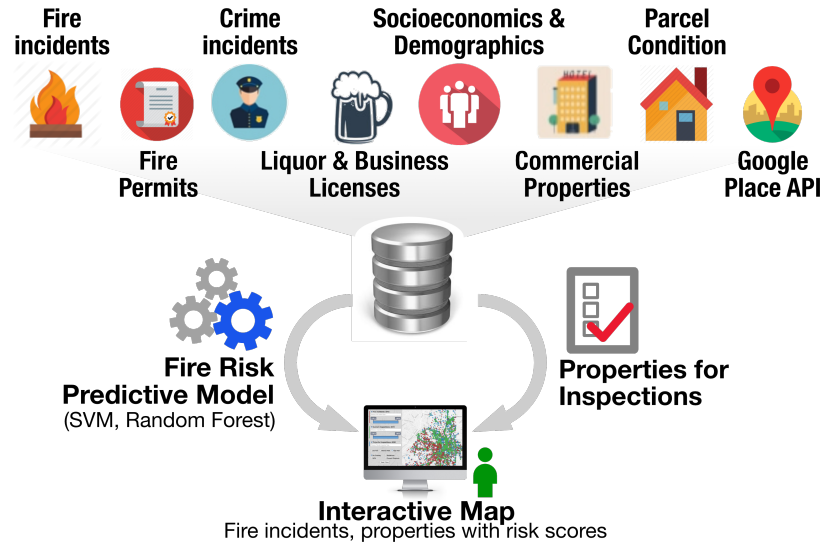


Figure 1.7: **Firebird Framework Overview.** By combining 8 datasets, Firebird identifies new commercial properties for fire inspections. Its fire risk predictive models (SVM, random forest) and interactive map help AFRD prioritize fire inspections and personnel allocation.

we developed the **Firebird** framework [11] (Figure 1.7) that helps municipal fire departments identify and prioritize commercial property fire inspections. *Firebird* computes fire risk scores for over 5,000 buildings in Atlanta, and correctly predicts 71% of fires. *Firebird* won the **Best Student Paper Award Runner-up at KDD 2016** and was highlighted by National Fire Protection Association as a best practice for using data to inform fire inspections.

## 1.2 Thesis Statement

Uniquely combining techniques from AI, cybersecurity, and algorithmic game theory, enables the development of next-generation strong cybersecurity defenses, contributing to:

1. *New theory* that guide defense resources allocation to guard against surprise attacks and catastrophic events;
2. *New scalable and robust machine learning algorithms* for a variety of threat models;
3. *New application of AI* on predicting enterprise cyber threats and prioritizing fire inspections.

### 1.3 Research Contributions

The goal of this thesis is to develop robust AI, and apply AI to solve security-critical and high-stakes problems. Our research contributes in multiple facets of AI and cybersecurity.

#### New Algorithms:

- Our *ShapeShifter* attack is the first robust targeted attack that can fool a state-of-the-art Faster R-CNN object detector. (Chapter 3)
- Our *SHIELD* defense combines image compression and randomization to protect neural networks from adversarial attacks in real-time. (Chapter 4)
- Our distributed boosting algorithm is simultaneously noise tolerant, communication efficient, and computationally efficient. (Chapter 7)

#### New Theories:

- We introduce a new online decision-making setting in game theory where players are compelled to play “diversified” strategies, and give strong guarantees on both the price of anarchy and the social welfare in this setting. (Chapter 6)
- Our distributed boosting algorithm requires exponentially less communication complexity in the agnostic setting, solving an open problem in distributed learning [9]. (Chapter 7)

#### New Applications:

- Our *Virtual Product* framework (Chapter 8) is the first method to predict security events and high-severity incidents identifiable by a security product as if it had been deployed.
- Our *Firebird* framework (Chapter 9) computes fire risk scores for over 5, 000 buildings in the city, with true positive rates of up to 71% in predicting fires.

## 1.4 Impact

This thesis work has made significant impact to society:

- My thesis ideas in developing theoretically principled, practical techniques to defend ML-based systems directly contributed to two funded competitive grant awards:
  - Our theory-guided decision making framework (Chapter 6) laid the foundation of the \$1.2M medium NSF grant *Understanding and Fortifying Machine Learning Based Security Analytics* (NSF CNS 1704701);
  - *ShapeShifter* and *SHIELD* (Chapter 3) were two highlights of the \$1.5M Intel “gift” grant for *Intel Science & Technology Center for Adversary-Resilient Security Analytics* (ISTC-ARSA);
- Our *ShapeShifter* attack, developed with Intel, reveals serious vulnerabilities for autonomous vehicles that use pure vision-based input, and was highlighted as the state-of-the-art physical adversarial attack in the recent DARPA program “Guaranteeing AI Robustness against Deception” (GARD). Our work appeared in the media <sup>2</sup> and is open-sourced at <https://github.com/shangtse/robust-physical-attack>.
- *ShapeShifter* and *SHIELD* have been integrated into the Intel AI Academy course.
- Our *Virtual Product* framework, developed with Symantec, has led to two patents.
- Our *Firebird* project is open-sourced<sup>3</sup> and has been used by the Atlanta Fire Rescue Department to prioritize fire inspections. *Firebird* won the Best Student Paper Award Runner-up at KDD 2016 and was highlighted by National Fire Protection Association as a best practice for using data to inform fire inspections.

---

<sup>2</sup><https://techhq.com/2018/10/study-reveals-new-vulnerability-in-self-driving-cars/>

<sup>3</sup><http://firebird.gatech.edu>

- My thesis research on AI-infused security has been recognized by the 2018 IBM PhD Fellowship.

## CHAPTER 2

### SURVEY

Our survey focuses on two important areas of research related to this thesis: security of ML and applications of ML in cybersecurity.

#### 2.1 Security of Machine Learning

We briefly survey robust machine learning algorithms under various threat models.

**Random Classification Noise.** This is one of the most basic threat models studied in classic learning theory [12]. In this setting, the training and testing data come from the same fixed but unknown distribution. However, the label of each training example presented to the learning algorithm is randomly flipped with probability  $0 \leq \eta < 1/2$ . Here we only consider the binary classification case, and  $\eta$  is a parameter called the classification error rate. It is known that if a training algorithm is in the family of Statistical query (SQ) learning, it can be converted into a noise-tolerant algorithm in the random classification noise setting [13].

**Malicious Noise.** This setting is similar to the random classification noise model, where  $\eta$  fraction of the training examples are changed by the adversary. The only difference is that the adversary can arbitrarily change not only the label but also the features of the training examples, making it a notoriously difficult setting [14]. It has been proved that it is information-theoretically impossible to learn to accuracy  $1 - \epsilon$  if  $\eta > \epsilon/(1 + \epsilon)$  [15]. Most of the positive results require strong assumptions on the underlying data distribution or the target function [16, 17]. For learning linear separators, the current state-of-the-art method is developed by Awasthi et al. [18].

**Agnostic Learning.** This is the setting that we will study in Chapter 7. In the two aforementioned settings, with probability  $1 - \eta$  the examples are labeled by an unknown target function from a known hypothesis set. For example, in the case of learning a linear classifier,  $1 - \eta$  fraction of the examples are linearly separable, and the remaining examples are contaminated by random classification noise or malicious noise, respectively. In contrast, in the agnostic learning setting, we do not make any assumptions on the data distribution nor the target function [19]. Since the target function may not be from the hypothesis set that the training algorithm uses, the goal is to achieve accuracy as close to the best hypothesis in our hypothesis set as possible.

**Adversarial Machine Learning.** This line of research was first studied by cybersecurity researchers in applications such as spam filtering [20], network intrusion detection [21], and malware detection [22]. Depending on the stage at which an attacker can manipulate data, adversarial attacks can be further categorized into *causative attacks* and *exploratory attacks* [23].

Causative attack, also known as poisoning attack, refers to the setting where the attacker can manipulate the training data in order to decrease the accuracy on all or a subset of the test examples. For example, the attacker can add backdoors to a maliciously trained traffic sign image classifier such that it achieves high overall test accuracy but classifies stop signs as speed limit signs when a special sticker is attached to the stop sign [24]. Similarly, one can also train networks for face recognition and speech recognition that only perform malicious behaviors when a specific “trojan” trigger is presented [25].

In an exploratory attack, also called an evasion attack, the attacker can only change the test examples to fool a trained ML model. The success of Deep Neural Networks (DNNs) in computer vision does not exempt them from this threat. It is possible to reduce the accuracy of a state-of-the-art DNN image classifier to zero percent by adding imperceptible adversarial perturbations [2, 26]. Many new attack algorithms have been proposed [27,

28, 29, 30] and applied to other domains such as malware detection [31, 32], sentiment analysis [33], and reinforcement learning [34, 35]. In Chapter 3, we demonstrate a new attack in a slightly different setting called physical adversarial attacks. There have been various attempts to mitigate the threat of adversarial attacks [4, 36], but immunizing a DNN model to adversarial attacks remains an open problem and an active research area. In Chapter 4 and 5, we propose new methods toward this goal.

## 2.2 Applications of Machine Learning to Cybersecurity

**Malware Detection.** Traditional anti-malware software depends heavily on signature-based methods, which use static fingerprints of known malware to detect future malicious files [37]. However, it can only identify “known” malware for which the signatures have been created, and hence can be easily evaded by more advanced attacking techniques like polymorphism and obfuscation [38, 39]. Many machine learning based approaches, using various feature extraction techniques and learning algorithms, have thus been explored [40, 41, 42, 43]. Reputation-based approaches using graph mining is another popular line of research [44, 45].

**Intrusion Detection System.** The main task of an intrusion detection system (IDS) is to monitor a system’s vulnerability exploits and attacks. Similar to malware detection, early work on IDS used signature-based approaches [46], which has limited ability to detect zero-day attacks. Anomaly-based detection models the normal internet traffic or system behavior using machine learning and data mining methods, and detects deviations from the baseline behavior [47, 48].

**Online Fraudulent Behavior Detection.** AI helps many websites provide better services, but it also creates new vulnerabilities. For example, an adversary can create fake accounts and write fraudulent reviews to manipulate reputation-based recommendation system. Researchers have used data mining and machine learning techniques to detect fake reviews [49,

50], internet bots [51], auction fraud [52], insider trading [53], and credit card fraud [54].

A good defense requires a combination of several techniques such as natural language processing, graph mining, and time series analysis.

## **Part I**

# **Adversarial Attack and Defense of Deep Neural Networks**

## OVERVIEW

Deep neural networks (DNNs), although very powerful, are known to be vulnerable to adversarial attacks. In computer vision applications, such attack can be achieved by adding carefully crafted but visually imperceptible perturbations to input images.

The threat of adversarial attack casts a shadow over deploying DNNs in security- and safety-critical applications, such as self-driving cars. To better understand and fix the vulnerabilities, there is a growing body of research on both designing stronger attacks and making DNN models more robust. However, many existing works are “impractical” either because they assume an unrealistic threat model, or the defense is too computationally expensive to be used in practice. In Part I of my thesis, we present the following practical attack and defenses.

- **ShapeShifter** (Chapter 3) is the first “physical” adversarial attack that fools the state-of-the-art object detector.
- **SHIELD** (Chapter 4) is an efficient defense leveraging stochastic image compression
- **UnMask** (Chapter 5) is a knowledge-based adversarial detection and defense framework.

## CHAPTER 3

### SHAPESHIFTER: ROBUST PHYSICAL ADVERSARIAL ATTACK ON OBJECT DETECTOR

Given the ability to directly manipulate image pixels in the digital input space, an adversary can easily generate imperceptible perturbations to fool a Deep Neural Network (DNN) image classifier, as demonstrated in prior work. In this work, we propose *ShapeShifter*, an attack that tackles the more challenging problem of crafting physical adversarial perturbations to fool image-based object detectors like Faster R-CNN. Attacking an object detector is more difficult than attacking an image classifier, as it needs to mislead the classification results in multiple bounding boxes at different scales. Extending a digital attack to the physical world adds another layer of difficulty, because it requires the perturbation to be robust enough to survive real-world distortions like different viewing distances and angles, lighting conditions, and camera limitations. We show that the *Expectation over Transformation* technique, which was originally proposed to enhance the robustness of adversarial perturbations in image classification, can be adapted to the object detection setting. *ShapeShifter* can generate adversarially perturbed stop signs that Faster R-CNN consistently mis-detects as other objects, posing a potential threat to autonomous vehicles and other safety-critical computer vision systems.

#### 3.1 Introduction

Adversarial examples are input instances that are intentionally designed to fool a machine learning model into producing a chosen prediction. The success of DNNs in computer vision does not exempt it from this threat. It is possible to bring the accuracy of a state-of-the-art DNN image classifier down to zero percent by adding imperceptible adversarial perturbations [2, 26]. The existence of adversarial examples not only reveals intriguing theoretical

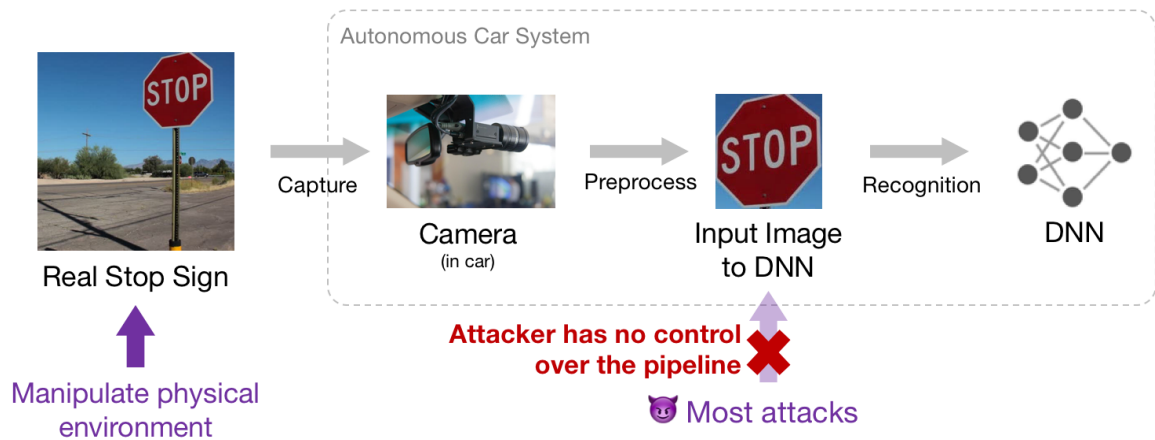


Figure 3.1: Illustration motivating the need of physical adversarial attack, because attackers typically do not have full control over the computer vision system pipeline.

properties of DNN, but also raises serious practical concerns about their deployment in security- and safety-critical systems. Autonomous vehicle is an example application that cannot be fully trusted until DNNs are robust to adversarial attacks. The need to understand robustness of DNNs attracts tremendous interest among machine learning, computer vision, and security researchers.

Although many adversarial attack algorithms have been proposed, using them to attack a real-world computer vision systems is difficult. First of all, many of these existing attack algorithms focus on the image classification task, yet for many real-world use cases there will be more than one object in an image. Object detection, which recognizes and localizes multiple objects in an image, is a more suitable model for many vision-based real-world use cases. Attacking an object detector is more difficult than attacking an image classifier, as it needs to mislead the classification results in multiple bounding boxes at different scales [55].

Further difficulty comes from the fact that a DNN is usually only one component in a complete computer vision system pipeline. For many applications, attackers do not have the ability to directly manipulate data inside the pipeline. Instead, they can only manipulate the things outside of the system, i.e., those things in the physical environment. Figure 3.1 illustrates the intuition behind *physical adversarial attacks*. To be successful attacks, physical adversarial attacks must be robust enough to survive real-world distortions

like different viewing distances and angles, lighting conditions, and camera limitations.

Prior work can either attack object detectors digitally [56], or attack image classifiers physically [3, 57, 58]. However, existing attempts to physically attack object detectors remain unsatisfactory. The perturbed stop sign shown in [59] cannot be detected by the Faster R-CNN object detector [60]. However, the perturbation is very noticeable. The authors tested it against a background with poor texture contrast, making the perturbed stop sign difficult to see even by humans. A concurrent work [61] claims to be able to generate some adversarial stickers that, when attached to a stop sign, can fool the YOLO object detector [62] and Faster R-CNN.

In this work, we propose *ShapeShifter*, the first robust targeted attack that can fool a state-of-the-art Faster R-CNN object detector. To make the attack robust, we adopt the *Expectation over Transformation* technique [63, 64], and adapt it from the image classification task to the object detection task. As a case study, we created adversarial stop signs that are mis-detected by Faster R-CNN in real drive-by tests. Our contributions are summarized below.

### 3.1.1 Our Contributions

- To the best of knowledge, our work presents the first *reproducible* and robust targeted attack against Faster R-CNN [55]. We have open-sourced our code on GitHub<sup>1</sup>.
- We show that the *Expectation over Transformation* technique [63], originally proposed for image classification, can be adapted to the object detection task and can significantly enhance the robustness of the resulting perturbation.
- By carefully studying the Faster R-CNN object detector algorithm, we overcome non-differentiable components in the model, and successfully perform optimization-based attacks using gradient descent and backpropagation.
- We generate perturbed stop signs that can consistently fool Faster R-CNN in real

---

<sup>1</sup><https://github.com/shangtse/robust-physical-attack>

drive-by tests (videos available on the GitHub repository), demonstrating for the need to improve and fortify vision-based object detectors.

## 3.2 Background

This section provides background information of adversarial attacks and briefly describes the Faster R-CNN object detector that we attack in this work.

### 3.2.1 Adversarial Attacks

Given a trained machine learning model  $C$  and a benign instance  $x \in \mathcal{X}$  that is correctly classified by  $C$ , the goal of the untargeted adversarial attack is to find another instance  $x' \in \mathcal{X}$ , such that  $C(x') \neq C(x)$  and  $d(x, x') \leq \epsilon$  for some distance metric  $d(\cdot, \cdot)$  and perturbation budget  $\epsilon > 0$ . For targeted attacks, we further require  $C(x') = y'$  where  $y' \neq C(x)$  is the target class. Common distance metrics  $d(\cdot, \cdot)$  in the computer vision domain are  $\ell_2$  distance  $d(x, x') = \|x - x'\|_2^2$  and  $\ell_\infty$  distance  $d(x, x') = \|x - x'\|_\infty$ .

The work of [2] was the first to discover the existence of adversarial examples for DNNs. Several subsequent works have improved the computational cost of creating adversarial examples and made the perturbations highly imperceptible to humans [27, 28]. Many adversarial attack algorithms assume that the model is differentiable, and use the gradient of the model to change the input towards the desired model output [26]. Sharif et al. [57] first demonstrated a physically realizable attack to fool a face recognition model by wearing an adversarially crafted pair of glasses.

### 3.2.2 Faster R-CNN

Faster R-CNN [60] is a state-of-the-art general object detector. It adopts a two-stage detection strategy. In the first stage, a region proposal network generates several class-agnostic bounding boxes, called region proposals, that may contain objects. In the second stage, a classifier and a regressor output a classification result and refined bounding box

coordinate for each region proposal, respectively. The computation cost is reduced by sharing the convolutional layers between the two stages. Faster R-CNN is difficult to attack, because a single object can be covered by multiple region proposals of different sizes and aspect ratios, and one needs to mislead the classification results in all overlapping region proposals to fool the detection.

### 3.3 Threat Model

Existing methods that generate adversarial examples typically yield imperceptible perturbations that fool a given machine learning model. Our work, following [57], generates perturbations that are perceptible but constrained such that a human would not be easily fooled by such a perturbation. We examine this kind of perturbation in the context of object detection. We chose this use case because of object detector’s possible uses in security-related and safety-related settings (e.g., autonomous vehicles). For example, attacks on traffic sign recognition could cause a car to miss a stop sign or travel faster than legally allowed.

We assume the adversary has *white-box* level access to the machine learning model. This means the adversary has access to the model structure and weights such that the adversary can compute both outputs (i.e., the forward pass) and gradients (i.e., the backward pass). It also means that the adversary does not have to construct a perturbation in real-time. Rather, the adversary can study the model and craft an attack for that model using methods like Carlini-Wagner attack [26]. This kind of adversary is distinguished from one with black-box level of access which is defined as having no access to the model architecture or weights. While our choice of adversary is the most knowledgeable one, existing research has shown it is possible to construct imperceptible perturbations without white-box level access [65]. Whether our method is capable of generating perceptible perturbations with only black-box access remains an open question. Results from Liu et al. [66] suggest that iterative attacks (like ours) tend not to transfer well to other models.

Unlike previous work, we restrict the adversary such that they cannot manipulate the digital values of pixels gathered by the camera used to sense the world. This is an important distinction from existing imperceptible perturbation methods. Because those methods create imperceptible perturbations, there is a high likelihood such imperceptible perturbations would not fool our use cases when physically realized. That is, when printed and then presented to the systems in our use cases, those imperceptible perturbations would have to survive both the printing process and camera sensing pipeline in order to fool the system. This is not an insurmountable task as Kurakin et al. [3] have constructed such imperceptible yet physically realizable adversarial perturbations for image classification systems.

Finally, we also restrict our adversary by limiting the shape of the perturbation the adversary can generate. This is important distinction for our use cases because one could easily craft an oddly-shaped “stop sign” that does not exist in the real world. We also do not give the adversary the latitude of modifying all pixels in an image like Kurakin et al. [3], but rather restrict them to certain pixels that we believe are physically realistic, and whose change is inconspicuous.

### 3.4 Attack Method

Our attack method, *ShapeShifter*, is inspired by the iterative, change-of-variable attack described in [26] and the *Expectation over Transformation* technique [63, 64]. Both methods were originally proposed for the task of image classification. We describe these two methods in the image classification setting before showing how to extend them to attack the Faster R-CNN object detector.

#### 3.4.1 Attacking an Image Classifier

Let  $F : [-1, 1]^{h \times w \times 3} \rightarrow \mathbb{R}^K$  be an image classifier that takes an image of height  $h$  and width  $w$  as input, and outputs a probability distribution over  $K$  classes. The goal of the attacker is to create an image  $x'$  that looks like an object  $x$  of class  $y$ , but will be classified

as another target class  $y'$ .

### *Change-of-variable Attack*

Denote  $L_F(x, y) = L(F(x), y)$  as the loss function that calculates the distance between the model output  $F(x)$  and the target label  $y$ . Given an original input image  $x$  and a target class  $y'$ , the change-of-variable attack [26] propose the following optimization formulation.

$$\arg \min_{x' \in \mathbb{R}^{h \times w \times 3}} L_F(\tanh(x'), y') + c \cdot ||\tanh(x') - x||_2^2. \quad (3.1)$$

The use of *tanh* ensures that each pixel is between  $[-1, 1]$ . The constant  $c$  controls the similarity between the modified object  $x'$  and the original image  $x$ . In practice,  $c$  can be determined by binary search [26].

### *Expectation over Transformation*

The *Expectation over Transformation* [63, 64] idea is simple: adding random distortions in each iteration of the optimization makes the resulting perturbation more robust to these distortions. Given a transformation  $t$  like translation, rotation, or scaling,  $M_t(x_b, x_o)$  is an operation that transforms an object image  $x_o$  using  $t$  and then overlays it onto a background image  $x_b$ .  $M_t(x_b, x_o)$  can also include a masking operation that only keeps a certain area of  $x_o$ . Masking is necessary when one wants to restrict the shape of the perturbation. After incorporating the random distortions, equation (3.1) becomes

$$\arg \min_{x' \in \mathbb{R}^{h \times w \times 3}} \mathbb{E}_{x \sim X, t \sim T} [L_F(M_t(x, \tanh(x')), y')] + c \cdot ||\tanh(x') - x_o||_2^2, \quad (3.2)$$

where  $X$  is the training set of background images. When the model  $F$  is differentiable, this optimization problem can be solved by gradient descent and back-propagation. The expectation can be approximated by the empirical mean.

### 3.4.2 Extension to Attacking Faster R-CNN

An object detector  $F : [-1, 1]^{h \times w \times 3} \rightarrow (\mathbb{R}^{N \times K}, \mathbb{R}^{N \times 4})$  takes an image as input and outputs  $N$  detected objects. Each detection includes a probability distribution over  $K$  pre-defined classification classes as well as the location of the detected object, represented by its 4 coordinates. Note that it is possible for an object detector to output more or fewer detected objects, depending on the input image, but for simplicity we select top- $N$  detected objects ranked by confidence.

As described in subsection 3.2.2, Faster R-CNN adopts a 2-stage approach. The region proposal network in the first stage outputs several region proposals, and the second stage classifier performs classification within each of the region proposals. Let  $rpn(x) = \{r_1, \dots, r_m\}$ , where each  $r_i$  is a region proposal represented as its four coordinates, and let  $x_r$  be a sub-image covered by region  $r$ . Denote  $L_{F_i}(x, y) = L(F(x_{r_i}), y)$ , i.e., the loss of the classification in the  $i$ -th region proposal. We can simultaneously attack all the classifications in each region proposal by optimizing the following:

$$\arg \min_{x' \in \mathbb{R}^{h \times w \times 3}} \mathbb{E}_{x \sim X, t \sim T} \left[ \frac{1}{m} \sum_{r_i \in rpn(M_t(x'))} L_{F_i}(M_t(x'), y') \right] + c \cdot \|\tanh(x') - x_o\|_2^2, \quad (3.3)$$

where we abuse the notation  $M_t(x') = M_t(x, \tanh(x'))$  for simplicity. However, for computational reasons, Faster R-CNN prunes the region proposals by using non-maximum suppression [60]. The pruning operations are usually non-differentiable, making it hard to optimize equation (3.3) end to end. Therefore, we approximately solve this optimization problem by first running a forward pass of the region proposal network, and then fixing the pruned region proposals as constants to the second stage classification problem in each iteration. We empirically find this approximation sufficient to find a good solution.

### 3.5 Evaluation

We evaluate our method by fooling a pre-trained Faster R-CNN model with Inception-v2 [67] convolutional feature extraction component. The model was trained on the Microsoft Common Objects in Context (MS-COCO) dataset [68] and is publicly available in the Tensorflow Object Detection API [69] model zoo repository<sup>2</sup>.

The MS-COCO dataset contains 80 general object classes ranging from people and animals, to trucks and cars, and other common objects. Although our method can potentially be used to attack any class, we chose to focus on attacking the stop sign class due to its importance and relevance to self-driving cars, where a vision-based object detector is used to help make driving decisions. An additional benefit of choosing the stop sign is its flat shape that can easily be printed on paper. Other classes, like dogs, are less likely to be perceived as real objects by human when printed on a paper. While 3D printing adversarial examples for image recognition is possible [63], we leave 3D-printed adversarial examples against object detectors as future work.

#### 3.5.1 Digitally Perturbed Stop Sign

We generate adversarial stop signs by performing the optimization process described in Equation 3.3. The hyperparameter  $c$  is crucial in determining the perturbation strength. A smaller value of  $c$  will result in a more conspicuous perturbation, but the perturbation will also be more robust to real-world distortions when we do the physical attack later.

However, it is hard to choose an appropriate  $c$  when naively using the  $\ell_2$  distance to a real stop sign as regularization. To obtain a robust enough perturbation, a very small  $c$  needs to be used, which has the consequence of creating stop signs that are difficult for humans to recognize. The  $\ell_2$  distance is not a perfect metric for human perception, which tends to be more sensitive to color changes on lighter-colored objects. Due to this observation,

---

<sup>2</sup>[http://download.tensorflow.org/models/object\\_detection/faster\\_rcnn\\_inception\\_v2\\_coco\\_2017\\_11\\_08.tar.gz](http://download.tensorflow.org/models/object_detection/faster_rcnn_inception_v2_coco_2017_11_08.tar.gz)

we only allowed the perturbation to change the red part of the stop sign, while leaving the white text intact. This allows us to generate larger and more robust perturbation, while providing enough contrast between the lettering and red parts so that a human can recognize the perturbation as a stop sign. The adversarial stop sign generated in [59] does not consider this and is visually more conspicuous. Automating this procedure for other objects we leave as future work.

We performed two targeted attacks and one untargeted attack. We choose *person* and *sports ball* as the two target classes because they are relatively similar in size and shape to stop signs. Our method allows attackers to use any target classes, however the perturbation needs to fool the object detector. For some target classes, this may mean creating perturbations so large in deviation that they may appear radically different from the victim class. We also noticed that some classes are easier to be detected at small scales, such as *kite*, while other classes (e.g., *truck*) could not be detected when the object was too small. This may be an artifact of the MS-COCO dataset that the object detector was trained on. Nevertheless, the attacker has a choice in target class and, for many applications, can find the target class that best fools the object detector according to their means.

For each attack, we generated a high confidence perturbation and a low confidence perturbation. The high confidence perturbations were generated using a smaller value of  $c$ , thus making them more conspicuous but also more robust. Depending upon the target class, it may be difficult to generate an effective perturbation. We manually chose  $c$  for each target class so that the digital attack achieves high success rate while keeping the perturbation not too conspicuous, i.e., we tried to keep the color as red as possible. We used  $c = 0.002$  for the high confidence perturbations and  $c = 0.005$  for the low confidence perturbations in the “sports ball” targeted attack and the untargeted attack. We used  $c = 0.005$  and  $c = 0.01$  for the high and low confidence perturbations in the “person” targeted attack, respectively. The 6 perturbations we created are shown in Figure 3.2.

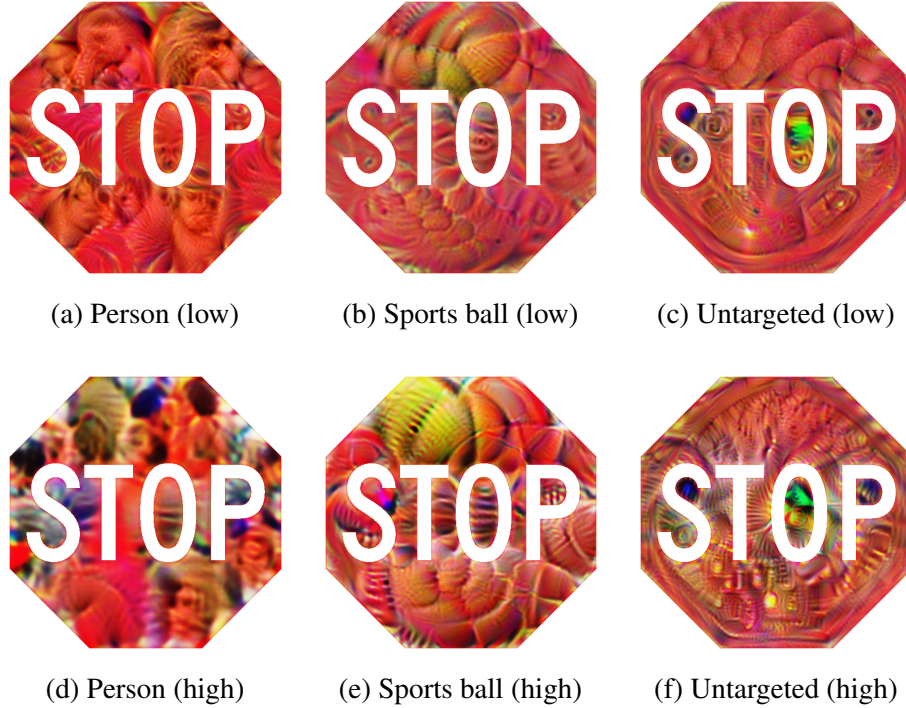


Figure 3.2: Digital perturbations we created using our method. Low confidence perturbations on the top and high confidence perturbations on the bottom.

### 3.5.2 Physical Attack

We performed physical attacks on the object detector by printing out the perturbed stop signs shown in Figure 3.2. We then took photos from a variety of distances and angles in a controlled indoor setting. We also conducted drive-by tests by recording videos from a moving vehicle that approached the signs from a distance. The lighting conditions varied from recording to recording due to the weather at the time.

#### *Equipment*

We used a Canon Pixma Pro-100 photo printer to print out signs with high-confidence perturbations, and an HP DesignJet to print out those with low-confidence perturbations<sup>3</sup>. For static images, we used a Canon EOS Rebel T7i DSLR camera, equipped with a EF-S 18-55mm IS STM lens. The videos in our drive-by tests were shot using an iPhone 8 Plus

<sup>3</sup>We used two printers to speed up our sign production, since a sign can take more than 30 minutes to produce.

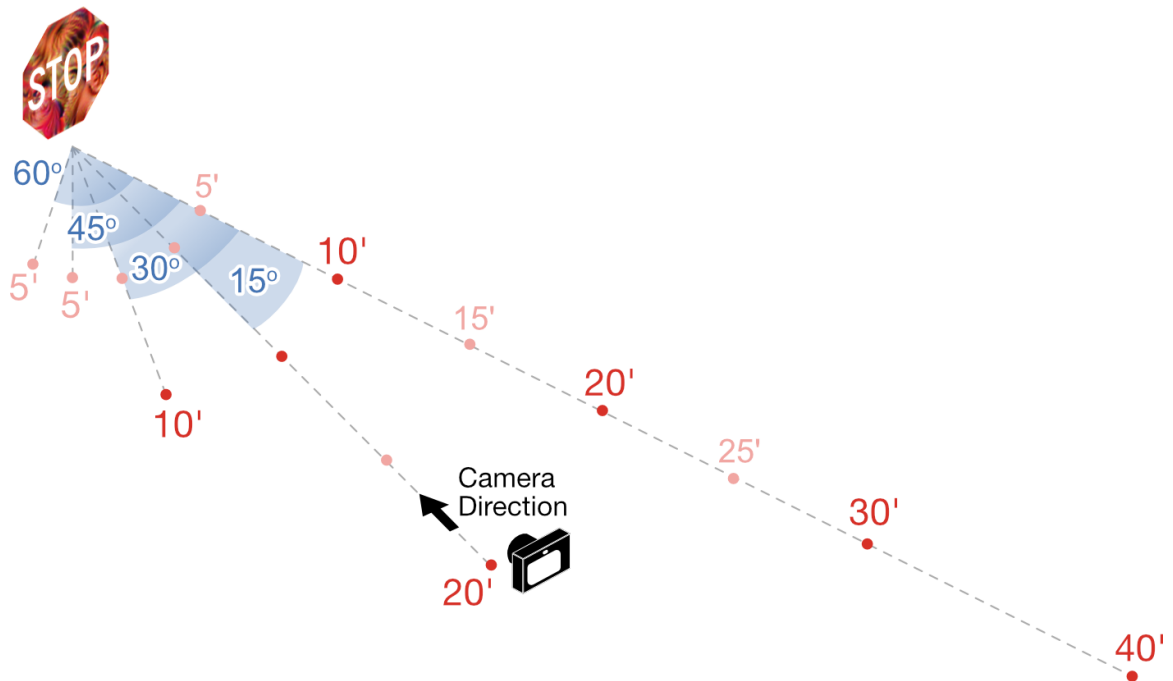


Figure 3.3: **Indoor experiment setup.** We take photos of the printed adversarial sign, from multiple angles ( $0^\circ$ ,  $15^\circ$ ,  $30^\circ$ ,  $45^\circ$ ,  $60^\circ$ , from the sign’s tangent), and distances (5’ to 40’). The camera locations are indicated by the red dots, and the camera always points at the sign.

mounted on the windshield of a car.

### *Indoor Experiments*

Following the experimental setup of [58], we took photos of the printed adversarial stop sign, at a variety of distances (5’ to 40’) and angles ( $0^\circ$ ,  $15^\circ$ ,  $30^\circ$ ,  $45^\circ$ ,  $60^\circ$ , from the sign’s tangent). This setup is depicted in Figure 3.3 where camera locations are indicated by red dots. The camera always pointed at the sign. We chose these distance-angle combinations to mimic a vehicle’s points of view as it would approach the sign [59]. Table 3.1 and Table 3.2 summarize the results for our *high-confidence* and *low-confidence* perturbations, respectively. For each distance-angle combination, we show the detected class and the detection’s confidence score. If more than one bounding box was detected, we report the highest-scoring one. Confidence values lower than 30% were considered undetected; we decided to use the threshold of 30%, instead of the default 50% in the Tensorflow Object Detection API [69], to impose a stricter requirement on ourselves (the “attacker”). Since

Table 3.1: Our **high-confidence** perturbations succeed at attacking at a variety of distances and angles. For each distance-angle combination, we show the detected class and the confidence score. If more than one bounding boxes are detected, we report the highest-scoring one. Confidence values lower than 30% is considered undetected.

Distance	Angle	person	(Conf.)	sports ball	(Conf.)	untargeted	(Conf.)
5'	0°	person	(.77)	sports ball	(.61)	clock	(.35)
5'	15°	person	(.91)	cake	(.73)	clock	(.41)
5'	30°	person	(.93)	cake	(.66)	cake	(.39)
5'	45°	person	(.69)	cake	(.61)	<i>stop sign</i>	(.62)
5'	60°	<i>stop sign</i>	(.93)	<i>stop sign</i>	(.70)	<i>stop sign</i>	(.88)
10'	0°	person	(.55)	cake	(.34)	clock	(.99)
10'	15°	person	(.63)	cake	(.33)	clock	(.99)
10'	30°	person	(.51)	cake	(.55)	clock	(.99)
15'	0°	undetected	—	cake	(.49)	clock	(.99)
15'	15°	person	(.57)	cake	(.53)	clock	(.99)
20'	0°	person	(.49)	sports ball	(.98)	clock	(.99)
20'	15°	person	(.41)	sports ball	(.96)	clock	(.99)
25'	0°	person	(.47)	sports ball	(.99)	<i>stop sign</i>	(.91)
30'	0°	person	(.49)	sports ball	(.92)	undetected	—
40'	0°	person	(.56)	sports ball	(.30)	<i>stop sign</i>	(.30)
Targeted success rate		87%		40%		N/A	
Untargeted success rate		93%		93%		73%	

an object can be detected as a stop sign and the target class simultaneously, we consider our attack to be successful only when the confidence score of the target class is the highest among all of the detected classes.

Table 3.1 shows that our high-confidence perturbations achieve a high attack success rate at a variety of distances and angles. For example, we achieved a targeted success rate 87% in misleading the object detector into detecting the stop sign as a *person*, and an even higher untargeted success rate of 93% when our attack goal is to cause the detector to either fail to detect the stop sign (e.g., at 15' 0°) or to detect it as a class that is *not* a stop sign. The *sports ball* targeted attack has a lower targeted success rate but achieves the same untargeted success rate. Our untargeted attack consistently misleads the detection into the *clock* class in medium distances, but is less robust for longer distances. Overall, the perturbation is less

Table 3.2: As expected, low-confidence perturbations achieve lower success rates.

<b>Distance</b>	<b>Angle</b>	<b>person</b>	<b>(Conf.)</b>	<b>sports ball</b>	<b>(Conf.)</b>	<b>untargeted</b>	<b>(Conf.)</b>
5'	0°	<i>stop sign</i>	(.87)	cake	(.90)	cake	(.41)
5'	15°	<i>stop sign</i>	(.63)	cake	(.93)	cake	(.34)
5'	30°	person	(.83)	cake	(.84)	<i>stop sign</i>	(.48)
5'	45°	<i>stop sign</i>	(.97)	<i>stop sign</i>	(.94)	<i>stop sign</i>	(.82)
5'	60°	<i>stop sign</i>	(.99)	<i>stop sign</i>	(.99)	<i>stop sign</i>	(.89)
10'	0°	<i>stop sign</i>	(.83)	<i>stop sign</i>	(.99)	undetected	—
10'	15°	<i>stop sign</i>	(.79)	<i>stop sign</i>	(.94)	undetected	—
10'	30°	<i>stop sign</i>	(.60)	<i>stop sign</i>	(.98)	<i>stop sign</i>	(.78)
15'	0°	<i>stop sign</i>	(.52)	<i>stop sign</i>	(.94)	<i>stop sign</i>	(.31)
15'	15°	<i>stop sign</i>	(.33)	<i>stop sign</i>	(.93)	undetected	—
20'	0°	<i>stop sign</i>	(.42)	sports ball	(.73)	undetected	—
20'	15°	person	(.51)	sports ball	(.83)	cell phone	(.62)
25'	0°	<i>stop sign</i>	(.94)	sports ball	(.87)	undetected	—
30'	0°	<i>stop sign</i>	(.94)	sports ball	(.95)	<i>stop sign</i>	(.79)
40'	0°	<i>stop sign</i>	(.95)	undetected	—	<i>stop sign</i>	(.52)
Targeted success rate		13%		27%		N/A	
Untargeted success rate		13%		53%		53%	

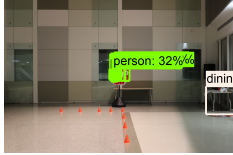
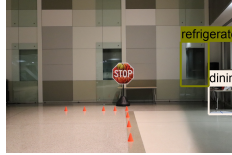




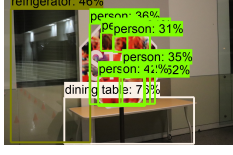
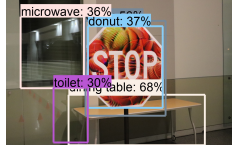
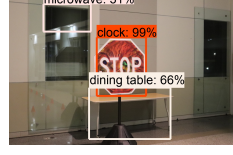

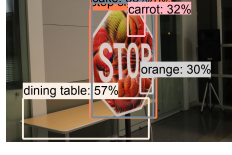
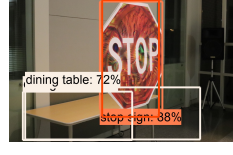
robust to very high viewing angle (60° from the sign’s tangent), because we did not simulate this high viewing angle distortion in the optimization.

The low-confidence perturbations (Table 3.2), as expected, achieve a much lower attack success rate, which informed our use of higher-confidence perturbations when we conducted the more challenging drive-by tests. Table 3.3 shows some high-confidence perturbations from our indoor experiments.

### *Drive-by Tests*

We performed drive-by tests at a parking lot so as not to disrupt other vehicles with our stop signs. We used a real stop sign as a control and put our printed, perturbed stop sign by its side. Starting from about 200 feet away, we slowly drove (between 5 mph to 15 mph) towards the signs while simultaneously recording video from the vehicle’s dashboard at 4K resolution and 24 FPS using an iPhone 8 Plus. We extracted all video frames, and for

Table 3.3: Sample high-confidence perturbations from indoor experiments. For complete experiment results, please refer to Table 3.1.

Dist.	Angle	Target: person	Target: ball	sports	Untargeted
40'	0°				
10'	0°				
10'	30°				
5'	60°				

each frame, we obtained the detection results from Faster R-CNN object detection model. Because our low confidence attacks showed relatively little robustness indoors, we only include the results from our high-confidence attack. Similar to our indoor experiments, we only consider detections that had a confidence score of at least 30%.

In Figure 3.4, we show sample video frames (rectangular images) to show the size of the signs relative to the full video frame; we also show zoomed-in views (square images) that more clearly show the Faster R-CNN detection results.

The *person-perturbation* in Figure 3.4a drive-by totaled 405 frames. The real stop sign in the video was correctly detected in every frame with high confidence. On the other hand, the perturbed stop sign was only correctly detected once, while 190 of the frames identified the perturbed stop sign as a person with medium confidence. For the rest of the 214 frames the object detector failed to detect anything around the perturbed stop sign.

The video we took with the *sports-ball-perturbation* shown in Figure 3.4b had 445

frames. The real stop sign was correctly identified all of the time, while the perturbed stop sign was never detected as a stop sign. As the vehicle (video camera) moved closer to the perturbed stop sign, 160 of the frames were detected as a sports ball with medium confidence. One frame was detected as *apple* and *sports ball* and the remaining 284 frames had no detection around the perturbed stop sign.

Finally, the video of the untargeted perturbation (Figure 3.4c) totaled 367 frames. While the unperturbed stop sign was correctly detected all of the time, the perturbed stop sign was detected as *bird* 6 times and never detected in the remaining 361 frames.

### *Exploring Black-box Transferability*

We also sought to understand how well our high-confidence perturbations could fool other object detection models. For image recognition, it is known that high-confidence targeted attacks fail to transfer to other models [66].

To this end, we fed our high-confidence perturbations into 8 other MS-COCO-trained models from the Tensorflow detection model zoo<sup>4</sup>. Table 3.4 shows how well our perturbation generated from the Faster R-CNN Inception-V2 transfer to other models. To better understand transferability, we examined the worse case. That is, if a model successfully detects a stop sign in the image, we say the perturbation has failed to transfer or attack that model. We report the number of images (of the 15 angle-distance images in our indoor experiments) where a model successfully detected a stop sign with at least 30% confidence. We also report the maximum confidence of all of those detected stop sign.

Table 3.4 shows the lack of transferability of our generated perturbations. The untargeted perturbation fails to transfer most of the time, followed by the sports ball perturbation, and finally the person perturbation. The models most susceptible to transferability were the Faster R-CNN Inception-ResNet-V2 model, followed by the SSD MobileNet-V2 model. Iterative attacks on image recognition also usually fail to transfer [66], so it is not surprising

---

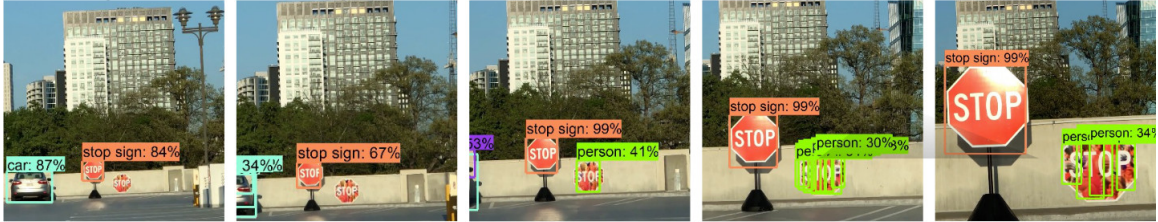
<sup>4</sup>[https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/detection\\_model\\_zoo.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md)

### a. Target Class: Person

Video frames



Zoomed-in



### b. Target Class: Sports Ball

Video frames



Zoomed-in



### c. Untargeted

Video frames



Zoomed-in



Figure 3.4: Snapshots of the drive-by test results. In (a), the person perturbation was detected 47% of the frames as a person and only once as a stop sign. The perturbation in (b) was detected 36% of the time as a sports ball and never as a stop sign. The untargeted perturbation in (c) was detected as *bird* 6 times and never detected as a stop sign or anything else for the remaining frames.

Table 3.4: Black-box transferability of our 3 perturbations. We report the number of images (of the 15 angle-distance images) that failed to transfer to the specified model. We consider the detection of any stop sign a “failure to transfer.” Our perturbations fail to transfer for most models, most likely due to the iterative nature of our attack.

<b>Model</b>	<b>person</b> (conf.)		<b>sports ball</b> (conf.)		<b>untargeted</b> (conf.)	
Faster R-CNN Inception-V2	3	(.93)	1	(.70)	5	(0.91)
SSD MobileNet-V2	2	(.69)	8	(.96)	15	(1.00)
SSD Inception-V2	11	(1.00)	14	(.99)	15	(1.00)
R-FCN ResNet-101	4	(.82)	10	(.85)	15	(1.00)
Faster R-CNN ResNet-50	13	(.00)	15	(1.00)	15	(1.00)
Faster R-CNN ResNet-101	15	(.99)	13	(.97)	15	(1.00)
Faster R-CNN Inc-Res-V2	1	(.70)	0	(.00)	12	(1.00)
Faster R-CNN NASNet	14	(1.00)	15	(1.00)	15	(1.00)

that our attacks fail to transfer as well. We leave the thorough exploration of transferability as future work.

### 3.6 Discussion & Future Work



Figure 3.5: Example stop signs from the MS-COCO dataset. Stop signs can vary by language, by degree of occlusion by stickers or modification by graffiti, or just elements of the weather. Each stop sign in the images is correctly detected by the object detector with high confidence (99%, 99%, 99%, and 64%, respectively).

There is considerable variation in the physical world that real systems will have to deal with. Figure 3.5 shows a curated set of non-standard examples of stop signs from the MS-COCO dataset<sup>5</sup>. The examples show stop signs in a different language, or that have graffiti or stickers applied to them, or that have been occluded by the elements. In each of

<sup>5</sup>Full resolution images of the examples in Figure 3.5 can be found at: <http://cocodataset.org/#explore?id=315605>, <http://cocodataset.org/#explore?id=214450>, <http://cocodataset.org/#explore?id=547465>, and <http://cocodataset.org/#explore?id=559484>

these cases, it is very unlikely a human would misinterpret the sign as anything else but a stop sign. They each have the characteristic octagonal shape and are predominantly red in color. Yet, against our stop signs with similar perturbations, the object detector sees something else.

Unlike previous work on adversarial examples for image recognition, our adversarial perturbations are overt. They, like the examples in Figure 3.5, exhibit large deviations from the standard stop sign. A human would probably notice these large deviations, and a trained human might even guess they were constructed to be adversarial. But they would not be fooled by our perturbations. However an automated-system using an off-the-shelf object detector would be fooled, as our results show. Our perturbation shown in Figure 3.2e does look like a baseball or tennis ball has been painted on the upper right hand corner. Figure 3.4b shows how the object detector detects this part of the image as a sports ball with high confidence. This might seem unfair, but attackers have much more latitude to change the environment when these kind of models are deployed in automated systems. Even in non-automated systems, a human might not think anything of Figure 3.2d because it does not exhibit any recognizable person-like features.

Attackers might also generate perturbations without restricting the shape and color, and attach them to some arbitrary objects, like a street light or a trash bin. An untrained eye might see these perturbations as some kind of artwork, but the autonomous system might see something completely different. This attack, as described in [70], could be extended to object detectors using our method.

Defending against these adversarial examples has proven difficult. Many defenses fall prey to the so-called “gradient masking” or “gradient obfuscating” problem [71]. The most promising defense, adversarial training, has yet to scale up to models with good performance on the ImageNet dataset. Whether adversarial training can mitigate our style of overt, large-deviation (e.g., large  $\ell_p$  distance) perturbations is left as future work.

### 3.7 Conclusion

We show that a state-of-the-art Faster R-CNN object detector, while previously considered more robust to physical adversarial attacks, can actually be attacked with high confidence. Our work demonstrates vulnerability in MS-COCO-learned object detectors and posits that security- and safety-critical systems need to account for the potential threat of adversarial inputs to object detection systems.

Many real-world systems probably do not use an off-the-shelf pre-trained object detector as we do in our work. Why would a system with safety or security implications care to detecting sports balls? Most probably do not. Although it remains to be shown whether our style of attack can be applied to safety or security critical systems that leverage object detectors, our attack provides the means to test for this new class of vulnerability.

## CHAPTER 4

### SHIELD: FAST, PRACTICAL DEFENSE AND VACCINATION FOR DEEP LEARNING USING JPEG COMPRESSION

In the previous chapter, we showed that deep neural networks (DNNs) are highly vulnerable to adversarially generated images. This underscores the urgent need for practical defense techniques that can be readily deployed to combat attacks in real-time. Observing that many attack strategies aim to perturb image pixels in ways that are visually imperceptible, we use JPEG compression at the core of our proposed SHIELD defense framework, utilizing its capability to effectively “compress away” such pixel manipulation. To immunize a DNN model from artifacts introduced by compression, SHIELD “vaccinates” the model by retraining it with compressed images, where different compression levels are applied to generate multiple vaccinated models that are ultimately used together in an ensemble defense. On top of that, SHIELD adds an additional layer of protection by employing randomization at test time that compresses different regions of an image using random compression levels, making it harder for an adversary to estimate the transformation performed. This novel combination of vaccination, ensembling, and randomization makes SHIELD a fortified multi-pronged defense. We conducted extensive, large-scale experiments using the ImageNet dataset, and show that our approaches eliminate up to 98% of gray-box attacks delivered by strong adversarial techniques such as *Carlini-Wagner’s L2* attack and *DeepFool*. Our approaches are fast and work without requiring knowledge about the model.

#### 4.1 Introduction

In computer vision applications, an attacker can add visually imperceptible perturbations to an image and mislead a DNN model into making arbitrary predictions. When the attacker has complete knowledge of a DNN model, these perturbations can be computed by using

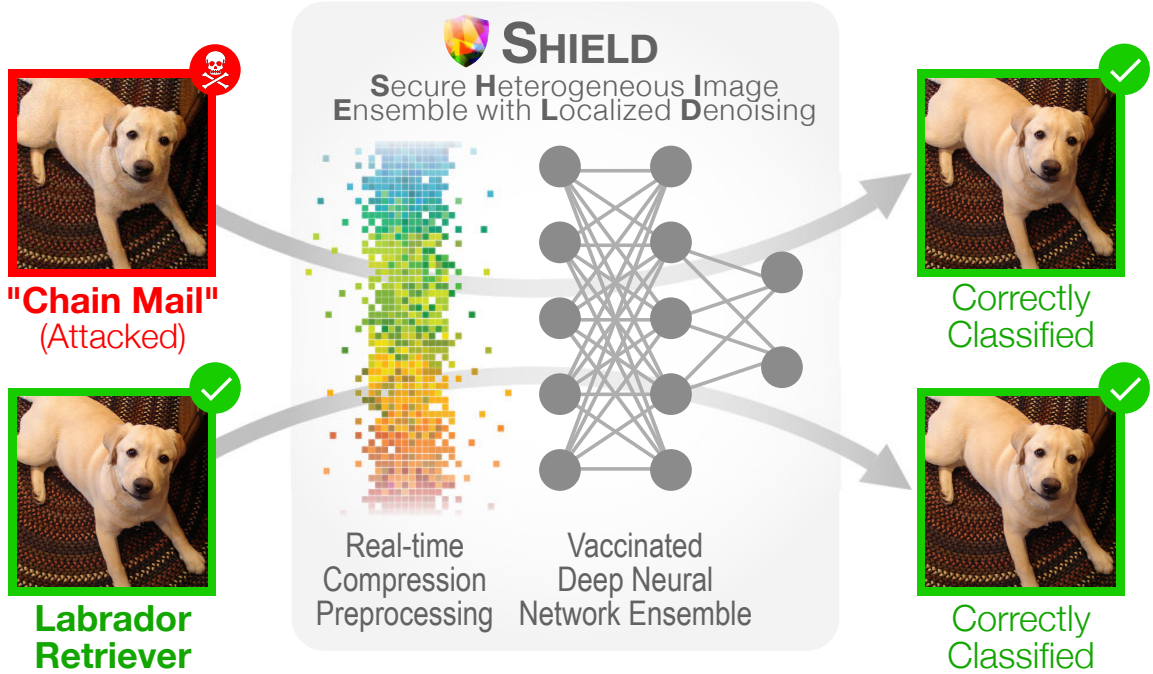


Figure 4.1: SHIELD Framework Overview. SHIELD combats adversarial images (in red) by removing perturbation in real-time using Stochastic Local Quantization (SLQ) and an ensemble of vaccinated models which are robust to the compression transformation. Our approach eliminates up to 98% of gray-box attacks delivered by strong adversarial techniques such as *Carlini-Wagner’s L2* attack and *DeepFool*.

the gradient information of the model. This guides the adversary toward vulnerable regions of the input space that would most drastically affect the model output [27, 30]. But even in a black-box scenario, where the attacker does not know the exact network architecture, one can use a substitute model to craft adversarial perturbations that are transferable to the target model [65].

To better understand and fix these vulnerabilities, there is a growing body of research on defending against various attacks and making DNN models more robust [72, 73, 74]. However, the progress of defense research lags behind the attack side. Moreover, research on defenses rarely focuses on practicality and scalability, both essential for real-world deployment. For example, total variation denoising and image quilting are image pre-processing techniques that have potential in mitigating adversarial perturbations [75], but they incur significant computational overhead, calling into question whether they can be used in practical applications, which often require a fast, real-time defense [58, 76].

#### 4.1.1 Our Contributions and Impact

**1. Compression as Fast, Practical, Effective Defense.** We leverage the idea that *compression* — a central concept that underpins numerous successful data mining techniques — can offer a powerful, scalable, practical, and real-time protection for deep learning models against adversarial image perturbations. Motivated by the observation that many attack strategies aim to perturb images in ways that are visually imperceptible to the naked eye, we show that systematic adaptation of the widely available JPEG compression technique can effectively compress away such pixel “noise”, especially since JPEG was designed to remove image details that are imperceptible to humans. (Section 4.3.1)

**2. SHIELD: Multifaceted Defense Framework.** Building on our principal idea of compression, we contribute the novel SHIELD defense framework that combines *randomization*, *vaccination* and *ensembling* into a fortified multi-pronged defense:

1. We exploit JPEG’s flexibility in supporting varying compression levels to develop strong ensemble models that span the spectrum of compression levels;
2. We “vaccinate” a model by training it on compressed images, increasing its robustness towards compression transformation for both adversarial and benign images;
3. SHIELD employs stochastic quantization that compresses different regions of an image using randomly sampled compression levels, making it harder for the adversary to estimate the transformation performed.

SHIELD does not require any change in the model architecture, and can recover a significant amount of model accuracy lost to adversarial instances, with little effect on the accuracy for benign inputs. SHIELD stands for **Secure Heterogeneous Image Ensemble with Localized Denoising**. (Sections 4.3.2 & 4.3.3)

**3. Extensive Evaluation Against Major Attacks.** We perform extensive experiments using the full ImageNet benchmark dataset with 50K images, demonstrating that our approach

is fast, effective and scalable. Our approaches eliminate up to 98% of gray-box attacks delivered by some of the most recent, strongest attacks, such as *Carlini-Wagner’s L2* attack [26] and *DeepFool* [28]. (Section 4.4)

**4. Impact to Intel and Beyond.** This work is making multiple positive impacts on Intel’s research and product development plans. Introduced with the Sandy Bridge CPU microarchitecture, Intel’s Quick Sync Video (QSV) technology dedicates a hardware core for high-speed video processing, performs JPEG compression up to 24X faster than TensorFlow implementations, paving the way for real-time defense in safety-critical applications, such as autonomous vehicles. This research has sparked insightful discussion among research and development teams at Intel, on the priority of *secure deep learning* that necessitates tight integration of practical defense strategies, software platforms and hardware accelerators. We believe our work will accelerate the industry’s emphasis on this important topic. To ensure reproducibility of our results, we have open-sourced our code on GitHub (<https://github.com/poloclub/jpeg-defense>). (Section 4.5)

## 4.2 Background: Adversarial Attacks

In this section, we describe the major, well-studied attacks in the literature, against which we will evaluate our approach.

**Carlini-Wagner’s  $L_2$  (CW-L2)** [26] is an optimization-based attack that adds a relaxation term to the perturbation minimization problem based on a differentiable surrogate of the model. They pose the optimization as minimizing:

$$\|x - x'\|_2 + \lambda \max \left( -\kappa, Z(x')_k - \max\{Z(x')_{k'} : k' \neq k\} \right) \quad (4.1)$$

where  $\kappa$  controls the confidence with which an image is misclassified by the DNN, and  $Z(\cdot)$  is the output from the logit layer (i.e., last layer before the softmax function is applied for prediction) of  $C$ .

**DeepFool (DF)** [28] constructs an adversarial instance under an  $L_2$  constraint by assuming the decision boundary to be hyperplanar. The authors leverage this simplification to compute a minimal adversarial perturbation that results in a sample that is close to the original instance but orthogonally cuts across the nearest decision boundary. In this respect, *DF* is an untargeted attack. Since the underlying assumption about the decision boundary being completely linear in higher dimensions is an oversimplification of the actual case, *DF* keeps reiterating until a true adversarial instance is found. The resulting perturbations are harder for humans to detect compared to perturbations introduced by other attacks.

**Iterative Fast Gradient Sign Method (I-FGSM)** [3] is the iterative version of the **Fast Gradient Sign Method (FGSM)** [27], which is a fast algorithm that computes perturbations subject to an  $L_\infty$  constraint. *FGSM* simply takes the sign of the gradient of loss function  $J$  w.r.t. the input  $x$ ,

$$x' = x + \epsilon \cdot \text{sign}(\nabla J_x(\theta, x, y)) \quad (4.2)$$

where  $\theta$  is the set of parameters of the model and  $y$  is the true label of the instance. The parameter  $\epsilon$  controls the magnitude of per-pixel perturbation. *I-FGSM* iteratively applies FGSM in each iteration  $i$  after clipping the values appropriately at each step:

$$x^{(i)} = x^{(i-1)} + \epsilon \cdot \text{sign}(\nabla J_{x^{(i-1)}}(\theta, x^{(i-1)}, y)) \quad (4.3)$$

### 4.3 Proposed Method: Compression as Defense

We present our compression-based approach for combating adversarial attacks. In Section 4.3.1, we begin by describing the technical reasons why compression can remove perturbations. As compression modifies the distribution of the input space by introducing some artifacts, in Section 4.3.2, we propose to “vaccinate” the model by training it with compressed images. This increases its robustness towards the compression transformation for

both adversarial and benign images. Finally, in Section 4.3.3, we present our multifaceted SHIELD defense framework that combines random quantization, vaccination and ensembling into a fortified multi-pronged defense.

#### 4.3.1 Preprocessing Images using Compression

Our main idea on rectifying the prediction of a trained model  $C$ , with respect to a perturbed input  $x'$ , is to apply a preprocessing operation  $g(\cdot)$  that brings back  $x'$  closer to the original benign instance  $x$ , which implicitly aims to make  $C(g(x')) = C(x)$ . Constructing such a  $g(\cdot)$  is application dependent. For the image classification problem, we show that JPEG compression is a powerful preprocessing defense technique. JPEG compression mainly consists of the following steps:

1. Convert the given image from  $RGB$  color space to  $YC_bC_r$  (chrominance + luminance) color space.
2. Perform spatial subsampling of the chrominance channels, since the human eye is less susceptible to these changes and relies more on the luminance information.
3. Transform  $8 \times 8$  blocks of the  $YC_bC_r$  channels to a frequency domain representation using Discrete Cosine Transform (DCT).
4. Quantize the blocks in the frequency domain representation according to a quantization table which corresponds to a user-defined quality factor for the image.

The last step is where the JPEG algorithm achieves the majority of compression at the expense of image quality. This step suppresses higher frequencies since these frequencies contribute less to the human perception of the image. Adversarial attacks do not maintain the spectral signature of the image, and they tend to introduce more high frequency components which can be removed via compression. This step also renders the JPEG compression non-differentiable, which makes it non-trivial for an adversary to optimize against, allowing

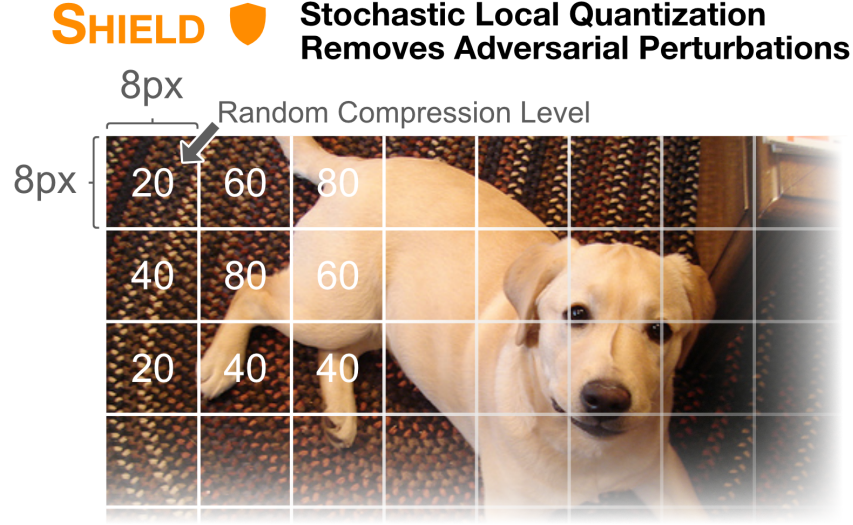


Figure 4.2: SHIELD uses Stochastic Local Quantization (SLQ) to remove adversarial perturbations from input images. SLQ divides an image into  $8 \times 8$  blocks and applies a randomly selected JPEG compression quality (20, 40, 60 or 80) to each block to mitigate the attack.

only estimations to be made of the transformation [77]. We show in our evaluation (Section 4.4.2) that JPEG compression effectively removes adversarial perturbation across a range of compression levels.

#### 4.3.2 Vaccinating Models with Compressed Images

Applying too much compression reduces the model’s accuracy on benign images, due to the artifacts introduced by JPEG compression. We propose to “vaccinate” the model by training it with compressed images, especially at lower JPEG qualities, to increase the model’s robustness towards the compression transformation for both adversarial and benign images. With vaccination, we can apply more aggressive compression to remove more adversarial perturbation. In our evaluation (Section 4.4.3), we show the advantage that our vaccination strategy provides, as it recovers more than 7 *absolute* percentage points in model accuracy for high-perturbation attacks.

### 4.3.3 SHIELD: Multifaceted Defense Framework

To leverage the effectiveness of JPEG compression as a preprocessing technique along with the benefit of vaccinating with JPEG images, we propose a *stochastic variant* of the JPEG algorithm that introduces randomization to the quantization step. This makes it difficult for the adversary to estimate the preprocessing transformation.

Figure 4.2 illustrates our proposed strategy, where we vary the quantization table for each  $8 \times 8$  block in the frequency domain to correspond to a random quality factor from a provided set of qualities, such that the compression level does not remain uniform across the image. This is equivalent to breaking up the image into disjoint  $8 \times 8$  blocks, compressing each block with a random quality factor, and putting the blocks together to re-create the final image. We call this method *Stochastic Local Quantization* (SLQ). As the adversary is free to craft images with varying amounts of perturbation, our defense should offer protection across a wide spectrum. We selected the set of qualities  $\{20, 40, 60, 80\}$  as our randomization candidates, uniformly spanning the range of JPEG qualities from 1 (most compressed) to 100 (least compressed).

Comparing our stochastic approach to taking an average over JPEG compressed images, our method maintains the original semantics of the image in the blocks compressed to higher qualities, while performing localized denoising in the blocks compressed to lower qualities. In the case of an average, perturbations may not be removed at higher qualities and so they could shift the average, and remain adversarial. Introducing localized stochasticity reduces this expectation.

In our evaluation (Section 4.4.3), we show that by using a variety of JPEG compression levels as in SLQ, our model can simultaneously attain a high accuracy on benign images, while being more robust to adversarial perturbations than using a single JPEG quality. Our method is further fortified by using an ensemble of vaccinated models individually trained on the set of qualities picked for SLQ. We show in Section 4.4.3 that our method achieves accuracies comparable to those of much larger ensembles while being significantly faster.

## SHIELD and JPEG Removes Carlini-Wagner-L2 & DeepFool Perturbation

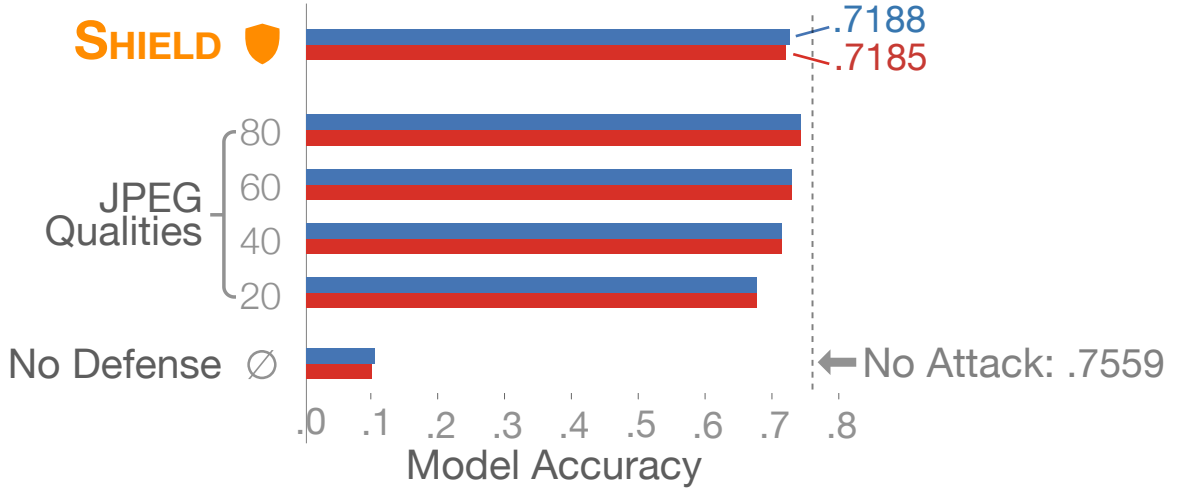


Figure 4.3: Carlini-Wagner-L2 (CW-L2) and DeepFool, two recent strong attacks, introduce perturbations that lowers model accuracy to around 10% ( $\emptyset$ ). JPEG compression recovers up to 98% of the original accuracy (with DeepFool), while SHIELD achieves similar performance, recovering up to 95% of the original accuracy (with DeepFool).

### 4.4 Evaluation

In this section, we show that our approach is scalable, effective, and practical at removing adversarial image perturbations. In our experiments, we consider the following scenarios:

- The adversary has access to the full model, including its architecture and parameters. (Section 4.4.2)
- The adversary has access to the model architecture, but not the exact parameters. (Section 4.4.3)
- The adversary does not have access to the model architecture. (Section 4.4.4)

#### 4.4.1 Experiment Setup

We performed experiments on the full validation set of the *ImageNet* benchmark image classification dataset [78], which consists of 1,000 classes, totaling 50,000 images. We show

the performance of each defense on the *ResNet-v2 50* model obtained from the *TF-Slim* module in *TensorFlow*. We construct the attacks using the popular *CleverHans* package<sup>1</sup>, which contains implementations from the authors of the attacks.

- For *Carlini-Wagner-L2* (CW-L2), we set its parameter  $\kappa = 0$ , a common value used in studies [75], as larger values (higher confidence) incur prohibitively high computation cost.
- *DeepFool* (DF) is a non-parametric attack that optimizes the amount of perturbation required to misclassify an image.
- For *FGSM* and *I-FGSM*, we vary  $\epsilon$  from 0 to 8 in steps of 2.

We compare JPEG compression and SHIELD with two popular denoising techniques that have potential in defending against adversarial attacks [79, 75]. Median filter (MF) collapses a small window of pixels into a single value, and may drop some of the adversarial pixels in the process. Total variation denoising (TVD) aims to reduce the total variation in an image, and may undo the artificial noise injected by the attacks. We varied the parameters of each method to evaluate how their values affect defense performance.

- For JPEG compression, we varied the compression level from quality 100 (least compressed) to 20 (greatly compressed), in decrements of 10.
- For *median filter*, we used window sizes of 3 (smallest possible) and 5. We tested larger window sizes (e.g., 7), which led to extremely poor model accuracies, thus we ruled them out as parameter candidates.
- For *total variation denoising*, we varied its weight parameter from 10 through 40, in increments of 10. Reducing the weight of TVD further produces blurry images that lead to poor model accuracy.

---

<sup>1</sup><https://github.com/tensorflow/cleverhans>

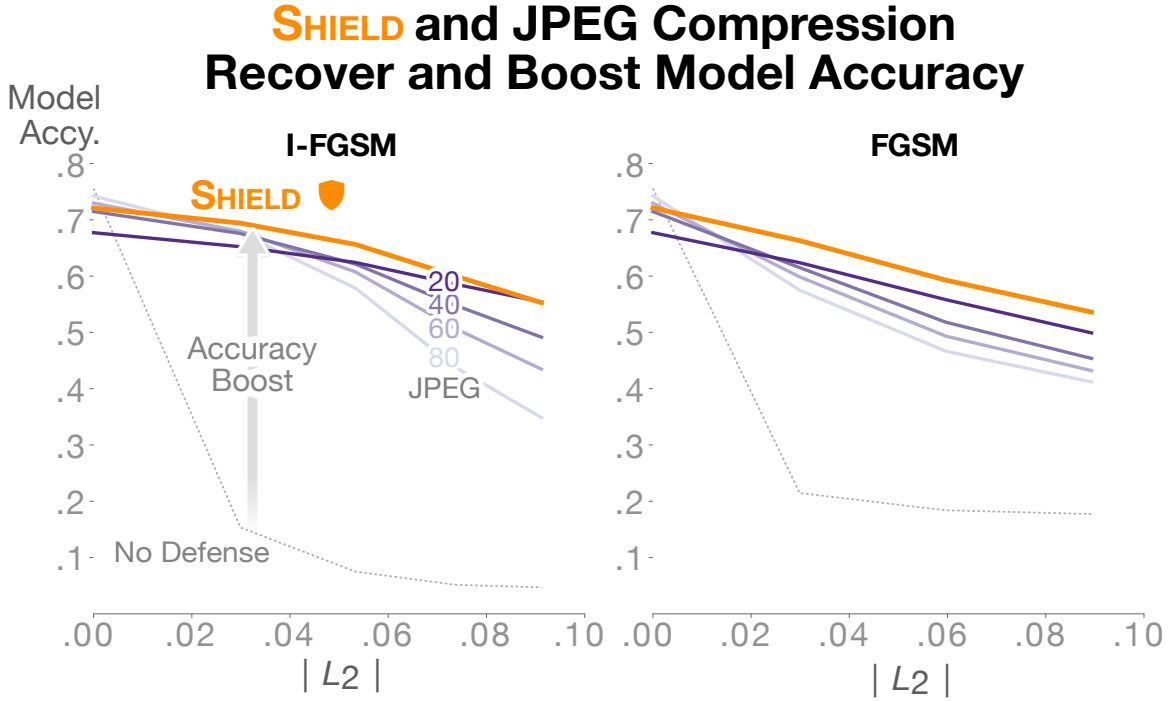


Figure 4.4: SHIELD recovers the accuracy of the model when attacked with I-FGSM (left) and FGSM (right). Both charts show the accuracy of the model when undefended (gray dotted curve). Applying varying JPEG compression qualities (purple curves) helps recover accuracy significantly, and SHIELD (orange curve) is able to recover more than any single JPEG-defended model.

#### 4.4.2 Defending Gray-Box Attacks with Image Preprocessing

In this section, we investigate the setting where an adversary gains access to all parameters and weights of a model that is trained on benign images but is unaware of the defense strategy. This constitutes a *gray-box* attack on the overall classification pipeline.

We show the results of applying JPEG compression at various qualities on images attacked with Carlini-Wagner-L2 (CW-L2) and DeepFool (DF) in Figure 4.3, and on images attacked with I-FGSM and FGSM in Figure 4.4.

**Combating Carlini-Wagner-L2 (CW-L2) & DeepFool (DF).** Although CW-L2 and DF, both considered strong attacks, are highly effective at lowering model accuracies, Figure 4.3 shows that even applying mild JPEG compression (i.e., using higher JPEG qualities) can recover much of the lost accuracy. Since both methods optimize for a lower perturbation to fool the model, the noise introduced by these attacks is imperceptible to the human eye and

lies in the high frequency spectrum, which is destroyed in the quantization step of the JPEG algorithm. SHIELD performs well, and comparably, for both attacks. We do not arbitrarily scale the perturbation magnitude of either attack as in [75], as doing so would violate the attacks’ optimization criteria.

**Combating I-FSGM & FGSM.** As shown in Figure 4.4, JPEG compression also achieves success in countering I-FGSM and FGSM attacks, which introduce higher magnitudes of perturbation than *CW-L2* and *DeepFool* attacks.

As the amount of perturbation increases, the accuracies of models without any protection (gray dotted curves in Figure 4.4) rapidly falls beneath 19%. JPEG recovers significant portions of the lost accuracies (purple curves); its effectiveness also gradually and expectantly declines as perturbation becomes severe. Applying more compression generally recovers more accuracy (e.g., dark purple curve, for JPEG quality 20), but at the cost of losing some accuracy for benign images. SHIELD (orange curve) offers a desirable trade-off, achieving good performance under severe perturbation while retaining accuracies comparable to the original models. Applying less compression (light purple curves) performs well with benign images but is not as effective when perturbation increases.

**Effectiveness and Runtime Comparison against Median Filter (MF) and Total Variation Denoising (TVD).** We compare JPEG compression and SHIELD with MF and TVD, two popular denoising techniques, because they too have potential in defending against adversarial attacks [79, 75]. Like JPEG, both MF and TVD are parameterized. Table 4.1 summarizes the performance of all the image preprocessing techniques under consideration. While all techniques are able to recover accuracies from CW-L2 and DF, both strongly optimized attacks with lower perturbation strength, the best performing settings are from JPEG (bold font in Table 4.1). When faced with large amount of perturbation generated by the I-FGSM and FSGM attacks, SHIELD benefits from the combination of Stochastic Local Quantization, vaccination, and ensembling, outperforming all other techniques.

As developing practical defense is our primary goal, effectiveness, while important, is

Defense	No Attack $ L_2  = 0$	<b>CW-L2</b> $ L_2  = .0025$	<b>DF</b> $ L_2  = .0020$	<b>I-FGSM</b> $ L_2  = .0533$	<b>FGSM</b> $ L_2  = .0597$
<i>No Defense</i>	75.59	10.29	9.78	7.49	18.40
SHIELD	72.11	71.85	71.88	<b>65.63</b>	<b>59.29</b>
JPEG [quality=100]	<b>74.95</b>	74.37	<b>74.41</b>	52.52	44.00
JPEG [quality=90]	74.83	<b>74.43</b>	74.36	55.18	45.12
JPEG [quality=80]	74.23	73.92	73.88	57.86	46.66
JPEG [quality=70]	73.61	73.11	73.17	59.53	47.96
JPEG [quality=60]	72.97	72.46	72.52	60.74	49.33
JPEG [quality=50]	72.32	71.86	71.91	61.47	50.53
JPEG [quality=40]	71.48	71.03	71.05	62.14	51.81
JPEG [quality=30]	70.08	69.63	69.67	62.52	53.51
JPEG [quality=20]	67.72	67.32	67.34	62.43	55.81
MF [window=3]	71.05	70.44	70.42	60.09	51.06
MF [window=5]	58.48	58.19	58.06	53.59	49.71
TVD [weight=10]	69.14	68.69	68.74	62.40	53.56
TVD [weight=20]	71.87	71.44	71.45	61.90	50.26
TVD [weight=30]	72.82	72.34	72.37	60.70	48.18
TVD [weight=40]	73.31	72.90	72.91	59.60	47.07

Table 4.1: Summary of model accuracies (in %) for all defenses: SHIELD [20, 40, 60, 80], JPEG, median filter (MF), and total variation denoising (TVD); v/s all attacks: Carlini-Wagner L2 (CW-L2), DeepFool (DF), I-FGSM and FGSM. While all techniques are able to recover accuracies from CW-L2 and DF attacks with lower perturbation strength, the best performing settings are from JPEG (in bold font). SHIELD benefits from the combination of SLQ, vaccination and ensembling, and outperforms all other techniques when facing high perturbation delivered by I-FGSM and FGSM. We use  $\kappa = 0$  in **CW-L2** and  $\epsilon = 4$  in **FGSM** and **I-FGSM**.

only one part of our desirable solution. Another critical requirement is that our solution be fast and scalable. Thus, we also compare the runtimes of the image processing techniques. Our comparison focuses on the most computationally intensive parts of each technique, ignoring irrelevant overheads (e.g., disk I/O) common to all techniques. All runtimes are averaged over 3 runs, using the full 50k ImageNet validation images, on a dedicated desktop computer equipped with an Intel i7-4770K quad-core CPU clocked at 3.50GHz, 4x8GB RAM, 1TB SSD of Samsung 840 EVO-Series and 2x3TB WD 7200RPM hard disk, running Ubuntu 14.04.5 LTS and Python 2.7. We used the most popular Python implementations of

the image processing techniques. We used JPEG and MF from Pillow 5.0, and TVD from scikit-image.

As shown in Figure 4.5, JPEG is the fastest, spending no more than 107 seconds to compress 50k images (at JPEG quality 80). It is at least 22x faster than TVD, and 14x faster than median filter. We tested the speed of the TensorFlow implementation of SHIELD, which also compresses all images at high speed, taking only 150s.

#### 4.4.3 Black-Box Attack with Vaccination and Ensembling

We now turn our attention to the setting where an adversary has knowledge of the model being used but does not have access to the model parameters or weights. More concretely, we vaccinate the ResNet-v2 50 model by retraining on the ImageNet training set and preprocessing the images with JPEG compression while training. This setup constitutes a *black-box* attack, as the attacker only has access to the original model but not the vaccinated model being used.

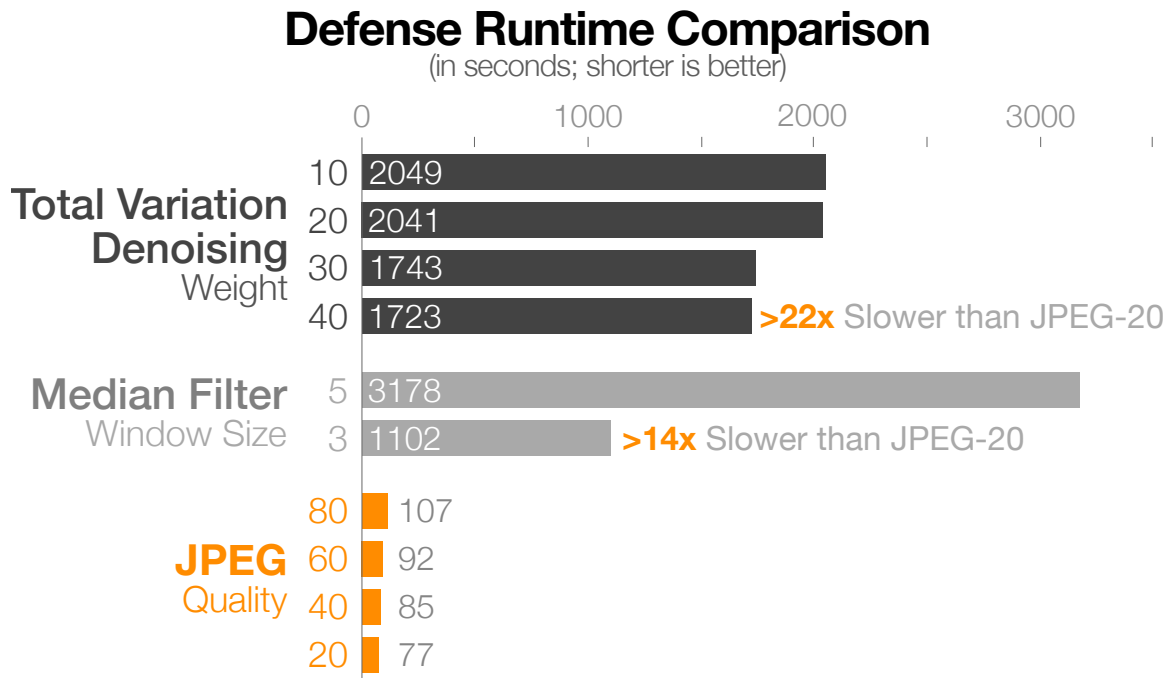


Figure 4.5: Runtime comparison for three defenses: (1) total variation denoising (TVD), (2) median filter (MF), and (3) JPEG compression, timed using the full 50k ImageNet validation images, averaged over 3 runs. JPEG is at least 22x faster than TVD, and 14x faster than MF.

We denote the original ResNet-v2 50 model as  $\mathcal{M}$ , which the adversary has access to. By retraining on images of a particular JPEG compression quality  $q$ , we transform  $\mathcal{M}$  to  $\mathcal{M}_q$ , e.g., for JPEG-20 vaccination, we retrain  $\mathcal{M}$  on JPEG-compressed images at quality 20 and obtain  $\mathcal{M}_{20}$ . When retraining the ResNet-v2 50 models, we used stochastic gradient descent (SGD) with a learning rate of  $5 \times 10^{-3}$ , with a decay of 94% over  $25 \times 10^4$  iterations. We conducted the retraining on a GPU cluster with 12 NVIDIA Tesla K80 GPUs. We trained 8 models from quality 20 through quality 90 in increments of 10 ( $\mathcal{M}_{20}, \mathcal{M}_{30}, \mathcal{M}_{40} \dots \mathcal{M}_{90}$ ) to cover a wide spectrum of JPEG qualities. Figure 4.6 shows the results of model vaccination against FGSM attacks, whose parameter  $\epsilon$  ranges from 0 (no perturbation) to 8 (severe perturbation), in steps of 2. The plots show that retraining the model helps recover more model accuracy than using JPEG preprocessing alone (compare the *unvaccinated* gray dotted curve vs. the *vaccinated* orange and purple curves in Figure 4.6). We found that a given model  $\mathcal{M}_q$  performed best when tested with JPEG-compressed images of the same quality  $q$ .

We tested these models in an ensemble with two different voting schemes. The first ensemble scheme, denoted as  $\mathcal{M}_q \times q$ , corresponds to each model  $\mathcal{M}_q$  casting a vote on every JPEG quality  $q$  from  $q \in \{20, 30, 40, \dots, 90\}$ . This has a total cost of 64 votes from which we select the majority vote. In the second scheme, denoted by  $\mathcal{M}_q - q$ , each model  $\mathcal{M}_q$  votes only on  $q$ , the JPEG quality it was trained on. This incurs a cost of 8 votes.

Table 4.2 compares the accuracies (against FGSM) and computation costs of these two schemes with those of SHIELD, which also utilizes an ensemble ( $\mathcal{M}_{20}, \mathcal{M}_{40}, \mathcal{M}_{60}, \mathcal{M}_{80}$ ) with a total of 4 votes. SHIELD achieves very similar performance as compared to the vaccinated models, at half the cost when compared to  $\mathcal{M}_q - q$ . Hence, SHIELD offers a favorable trade-off in terms of scalability with minimal effect on accuracy.

## Vaccinating Models with Compressed Images Improves Accuracies

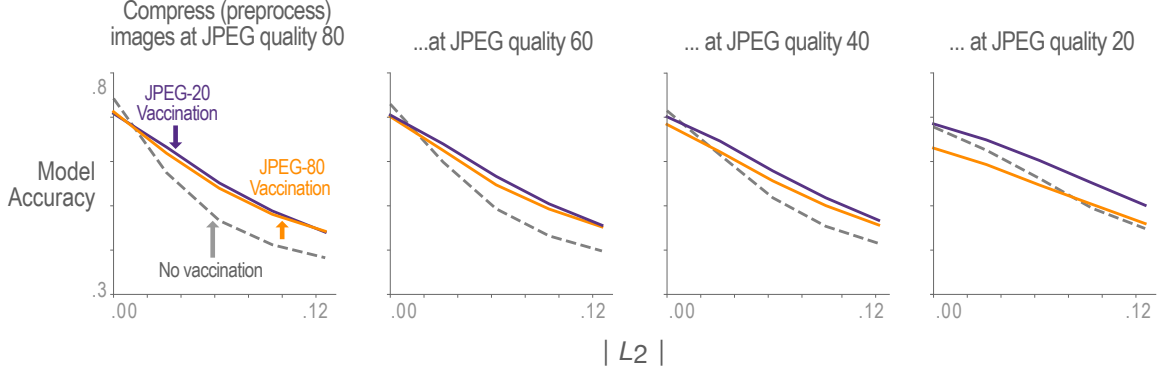


Figure 4.6: Vaccinating a model by retraining it with compressed images helps recover its accuracy. Each plot shows the model accuracies when preprocessing with different JPEG qualities with the FGSM attack. Each curve in the plot corresponds to a different model. The gray dotted curve corresponds to the original unvaccinated ResNet-v2 50 model. The orange and purple curves correspond to the models retrained on JPEG qualities 80 and 20 respectively. Retraining on JPEG compressed images and applying JPEG preprocessing helps recover accuracy in a gray-box attack.

### 4.4.4 Transferability in Black-Box Setting

In this setup, we evaluated the transferability of attacked images generated using ResNet-v2 50 on ResNet-v2 101 and Inception-v4. The attacked images were preprocessed using JPEG compression and Stochastic Local Quantization. In Table 4.3, we show that JPEG compression as a defense does not significantly reduce model accuracies on low perturbation attacks like DF and CW-L2. For higher-perturbation attacks, the accuracy of Inception-v4 lowers by a maximum of 10%.

Ensemble	Cost	$\epsilon = 0$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 6$	$\epsilon = 8$
$\mathcal{M}_q \times q$	64	73.90	67.72	60.13	54.44	49.84
$\mathcal{M}_q - q$	8	73.54	67.06	59.86	53.91	49.40
SHIELD	4	72.11	66.30	59.29	53.60	48.63

Table 4.2: Comparison of two ensemble schemes with SHIELD, when defending against FGSM.  $\mathcal{M}_q \times q$  corresponds to each model  $\mathcal{M}_q$  voting on each JPEG quality  $q$  from  $q \in \{20, 30, 40, \dots, 90\}$ . In  $\mathcal{M}_q - q$ , each model  $\mathcal{M}_q$  votes only on  $q$ , the JPEG quality it was trained on. SHIELD offers a favorable trade-off, providing at least 2x speed-up as compared to larger ensembles, while delivering comparable accuracies.

Attack	Defense	Inc-v4 (80.2%)		RN-v2 101 (77.0%)	
		Accuracy	(Qual.)	Accuracy	(Qual.)
None	JPEG	79.05	(100)	76.48	(100)
	SLQ	75.90	-	73.70	-
CW-L2	JPEG	79.00	(100)	76.20	(100)
	SLQ	75.80	-	73.60	-
DF	JPEG	78.91	(100)	76.19	(100)
	SLQ	76.29	-	73.70	-
I-FGSM	JPEG	74.84	(100)	70.06	(70)
	SLQ	73.20	-	69.40	-
FGSM	JPEG	71.00	(100)	64.18	(40)
	SLQ	70.01	-	64.64	-

Table 4.3: JPEG compression as defense does not reduce model accuracy significantly on transferred attacks with low perturbation. Adversarial images crafted using the ResNet-v2 50 model are protected using *JPEG* alone and *Stochastic Local Quantization* (SLQ), before being fed into two other models: Inception-v4 (Inc-v4) and ResNet-v2 101 (RN-v2 101).

#### 4.4.5 NIPS 2017 Competition Results

In addition to the experiment results shown above, we also participated in the NIPS 2017 competition on Defense Against Adversarial Attack using a version of our approach that included JPEG compression and *vaccination* to defend against attacks “in the wild.” With only an ensemble of three JPEG compression qualities (90, 80, 70), our entry received a silver badge in the competition, ranking 16th out of more than 100 submissions.

## 4.5 Significance and Impact

This work has been making multiple positive impacts on Intel’s research and product development plans. In this section, we describe such impacts in detail, and also describe how they may more broadly influence deep learning and cybersecurity. We then discuss our work’s scope, limitations, and additional practical considerations.

#### 4.5.1 Software and Hardware Integration Milestones

As seen in Section 4.4, JPEG compression is much faster than other popular preprocessing techniques; even commodity implementations from Pillow are fast. However, in order to be deployed into a real defense pipeline, we need to evaluate its computational efficiency with tighter software and hardware integration. Fortunately, JPEG compression is a widely-used and mature technique that can be easily deployed in various platforms, and due to its widespread usage, we can use off-the-shelf optimized software and hardware for such testing. One promising milestone we reached, utilized Intel’s hardware Quick Sync Video (QSV) technology: a hardware core dedicated and optimized for video encoding and decoding. It was introduced with Sandy Bridge CPU microarchitecture and exists currently in various Intel platforms. From our experiments, JPEG compression by Intel QSV is up to 24 times faster than the Pillow and TensorFlow implementations when evaluated on the same ImageNet validation set of 50,000 images. This computational efficiency is desirable for applications that need real-time defense, such as autonomous vehicles. In the future, we plan to explore the feasibility of our approach on more hardware platforms, such as the Intel Movidius Compute Stick<sup>2</sup>, which is a low power USB-based deep learning inference kit.

#### 4.5.2 New Computational Paradigm: Secure Deep Learning

This research has sparked insightful discussion with teams of Intel QSV, Intel Deep Learning SDK, and Intel Movidius Compute Stick. This work provides opportunities to advance deep learning software and hardware development to incorporate adversarial machine learning defenses. For example, almost all defenses incur certain levels of computational overhead. This may be due to image preprocessing techniques [75, 80], using multiple models for model ensembles [81], the introduction of adversarial perturbation detectors [74, 79], or the increase in training time for adversarial training [27]. However, while hardware and system improvement for fast deep learning training and inference remains an active area of

---

<sup>2</sup><https://developer.movidius.com>

research, secure machine learning workloads still receive relatively less attention, suggesting room for improvement. We believe this will accelerate the positive shift of thinking in the industry in the near future, from addressing problems like “*How do we build deep learning accelerators?*” to problems such as “*How do we build deep learning accelerators that are not only fast but also secure?*”. Understanding such hardware implications are important for microprocessor manufacturers, equipment vendors and companies offering cloud computing services.

#### 4.5.3 Scope and Limitations

In this work, we focus on systematically studying the benefit of compression on its own. As myriads of newer and stronger attack strategies are continuously discovered, limitations in existing, single defenses are revealed. Our approach is not a panacea to defend all possible (future) attacks. For example, because JPEG compression mainly removes imperceptible noises, SHIELD is not the best defense against *ShapeShifter* attack introduced in the previous chapter, which uses large and noticeable perturbations. We do not expect or intend for SHIELD to be used in isolation of other techniques. Rather, our methods should be used together with other defense techniques, to potentially develop an even stronger defense. Using multi-layered protection is a proven, long-standing defense strategy that has been pervasive in security research and in practice [45, 82]. Fortunately, since our approach primarily involves preprocessing, it is easy to integrate it into many other defense techniques such as adversarial retraining.

### **4.6 Related Work**

Due to intriguing theoretical properties and practical importance, there has been a surge in the number of papers in the past few years attempting to find countermeasures against adversarial attacks. These include detecting adversarial examples before performing classification [74, 36], modifying network architecture and the underlying primitives used [83, 84,

85], modifying the training process [27, 72], and using preprocessing techniques to remove adversarial perturbations [86, 73, 80, 75]. The preprocessing approach is most relevant to our work. Below, we describe two methods in this category—median filter and total variation denoising, which we compared against in Section 4.4. We then discuss some recent attacks that claim to break preprocessing defenses.

#### 4.6.1 Image Preprocessing as Defense

**Median Filter.** This method uses a sliding window over the image and replaces each pixel with the median value of its neighboring pixels to spatially smooth the image. The size of the sliding window controls the smoothness, for example, a larger window size produces blurrier images. This technique has been used in multiple prior defense works [75, 79].

**Total Variation Denoising.** The method is based on the principle that images with higher levels of (adversarial) noise tend to have larger total variations: the sum of the absolute difference between adjacent pixel values. Denoising is performed by reducing the total variation while keeping the denoised image close to the original one. A weighting parameter is used as a trade-off between the level of total variation and the distance from the original image. Compared with median filter, this method is more effective at removing adversarial noise while preserving image details [75].

#### 4.6.2 Attacks against Preprocessing Techniques

One reason why adding preprocessing steps increases attack difficulty is that many preprocessing operations are non-differentiable, thus restricting the feasibility of gradient-based attacks. In JPEG compression, the quantization in the frequency domain is a non-differentiable operation.

Shin and Song [77] propose a method that approximates the quantization in JPEG with a differentiable function. They also optimize the perturbation over multiple compression qualities to ensure an adversarial image is robust at test time. We evaluated their attacking

strategy against SHIELD under different threat models [87]. We found that SHIELD performs best for defending against targeted attacks in the gray-box setting where the attacker does not know all the models used in the ensemble.

Backward Pass Differentiable Approximation [71] is another potential approach to bypass non-differentiable preprocessing techniques. To attack JPEG preprocessing, it performs forward propagation through the preprocessing and DNN combination but in the backward pass, the method differentiates with respect to the JPEG compressed image. This is based on the intuition that the compressed image should look similar to the original one, so the operation can be approximated by the identity function. However, we believe this assumption only holds for higher compression qualities. Since the work did not report the compression quality used in the experiments, the conclusion remains open for debate.

## 4.7 Conclusion

In this work, we highlighted the urgent need for practical defense for deep learning models that can be readily deployed. We drew inspiration from JPEG image compression, a well-known and ubiquitous image processing technique, and placed it at the core of our new deep learning model defense framework: SHIELD. Since many attack strategies aim to perturb image pixels in ways that are visually imperceptible, the SHIELD defense framework utilizes JPEG compression to effectively “compress away” such pixel manipulation. SHIELD immunizes DNN models from being confused by compression artifacts by “vaccinating” a model: re-training it with compressed images, where different compression levels are applied to generate multiple vaccinated models that are ultimately used together in an ensemble defense. Furthermore, SHIELD adds an additional layer of protection by employing randomization at test time by compressing different regions of an image using random compression levels, making it harder for an adversary to estimate the transformation performed. This novel combination of vaccination, ensembling and randomization makes SHIELD a fortified multi-pronged defense, while remaining fast and successful without requiring knowledge about

the model. We conducted extensive, large-scale experiments using the ImageNet dataset, and showed that our approaches eliminate up to 98% of gray-box attacks delivered by the recent, strongest attacks. To ensure reproducibility of our results, we have open-sourced our code on GitHub<sup>3</sup>.

Although we only tested SHIELD for the image classification task, it is possible to use SHIELD for other computer vision tasks, such as object detection. However, as mentioned in section 4.5.3, SHIELD works best in defending against perturbations that are visually imperceptible. When defending against large and noticeable perturbations generated by *ShapeShifter*, SHIELD needs to use aggressive compression, which can hurt the benign accuracy. In the next chapter, we propose a different defense called UNMASK that uses additional domain knowledge to protect DNNs.

---

<sup>3</sup><https://github.com/poloclub/jpeg-defense>

## CHAPTER 5

### UNMASK: ADVERSARIAL DETECTION AND DEFENSE IN DEEP LEARNING THROUGH BUILDING-BLOCK KNOWLEDGE EXTRACTION

*Shield* (Chapter 4) is best suited for defending against imperceptible perturbations. To better defend against *ShapeShifter*-style attacks (Chapter 3), we further develop UNMASK, a knowledge-based adversarial detection and defense framework. The core idea behind UNMASK is to protect models by verifying that an image’s predicted class (e.g., “bird”) contains the expected building blocks (e.g., beak, wings, eyes). For example, if an image is classified as “bird”, but the extracted building blocks are *wheel*, *seat* and *frame*, the model may be under attack. UNMASK detects such attacks and defends the model by rectifying the misclassification, re-classifying the image based on its extracted building blocks. Our extensive evaluation shows that UNMASK (1) *detects* up to 92.9% of attacks, with a false positive rate of 9.67% and (2) *defends* the model by correctly classifying up to 92.24% of adversarial images produced by the state-of-the-art Projected Gradient Descent attack in the gray-box setting. UNMASK is architecture agnostic and fast.

#### 5.1 Introduction

Adversarial attacks on deep neural networks highlight a critical issue with modern computer vision systems: these deep learning systems do not distinguish objects in ways that humans would [88, 89]. For example, when humans see a bicycle, we see its handlebar, frame, wheels, chains, and pedals (Fig. 5.1, top). Through our visual perception and cognition, we synthesize these detection results with our knowledge to determine that we are actually seeing a bicycle. However, when an image of a bicycle is adversarially perturbed to fool the model into misclassifying it as a bird (by manipulating pixels, as in Fig. 5.1, bottom), to humans, we still see the bicycle’s **building-block features** (e.g., handlebar). On the

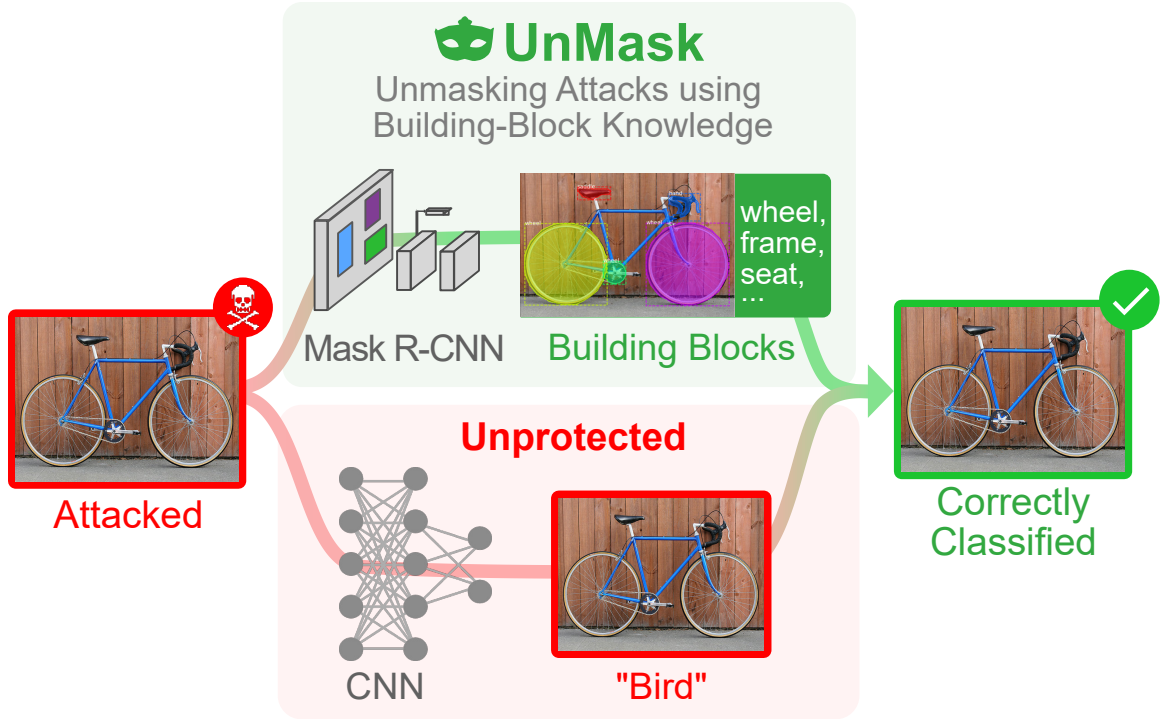


Figure 5.1: UNMASK framework overview. UNMASK combats adversarial attacks (in red) by extracting *building-block knowledge* (e.g., *wheel*) from the image (top, in green), and comparing them to expected features of the classification (“Bird” at bottom) from the unprotected model. Low feature overlap signals attack. UNMASK rectifies misclassification using the image’s extracted features. Our approach *detects* 92.9% of gray-box attacks (at 9.67% false positive rate) and *defends* the model by correctly classifying up to 92.24% of adversarial images crafted by the state-of-the-art Projected Gradient Descent attack.

other hand, the attacked model fails to perceive these building blocks, and is tricked into misclassifying the image. How do we incorporate this intuitive detection capability natural to human beings, into deep learning models to protect them from harm?

There has been a rich body of research studying detection and defense for deep learning, including adversarial training [3, 90], distillation [72] and image pre-processing [6, 73]. However, these approaches have not *explicitly* considered incorporating the extraction of building-block knowledge from images to protect deep learning models. Furthermore, research has shown that optimization based learning methods often fail to learn representations of objects that strongly align with humans’ intuitive perception of those objects [27]. To fill this critical research gap in adversarial machine learning, we propose **UNMASK** (Figure 5.1), a novel method to protect deep learning models from adversarial perturbations by extracting

building-block knowledge from images.

### 5.1.1 Contributions

**1. Building-Block Knowledge Extraction.** We contribute the major idea that *building-block knowledge extraction* offers a powerful, explainable and practical method of detecting and defending against adversarial perturbations in deep learning models. *Building-block knowledge extraction* extracts higher-level information out of images—extending the core concept of *feature extraction* that is central to numerous successful data mining techniques.

A significant advantage of our proposed knowledge extraction concept is that while an attacker may be able to manipulate the class label by subtly changing pixel values, it is much more challenging for such perturbation to simultaneously manipulate all the individual features that jointly compose the image. We demonstrate that by adapting the Mask R-CNN architecture [91], we can effectively extract higher-level *building-block knowledge feature* contained in images to detect and defend against adversarial attacks.

**2. UNMASK: Detection & Defense Framework.** Building on our core concept of building-block knowledge extraction, we propose UNMASK as a framework to detect and defeat adversarial image perturbation by quantifying the similarity between the image’s extracted features with the expected features of its predicted class. To the best of our knowledge, UNMASK is the first framework that utilizes the concept of building-block knowledge extraction to combat adversarial perturbations.

We illustrate how UNMASK works in Figure 5.1, where a bicycle image has been perturbed such that it would fool an unprotected model into misclassifying it as a bird. For a real “bird” image, we would expect to see features such as *beak*, *wing* and *tail*. However, UNMASK would (correctly) extract bike features: *wheel*, *frame*, and *pedals*. UNMASK quantifies the similarity between the *extracted* features (of a bike) with the *expected* features (of a bird), in this case zero. This comparison gives us the dual ability to both **detect**

adversarial perturbations by selecting a similarity threshold for which we classify an image as adversarial, and to **defend** the model by predicting a corrected class that best matches the extracted features. Since we are presenting a new category of detection and defense research, our results are the first presented in this line of research.

**3. Extensive Evaluation.** We evaluate UNMASK’s effectiveness using the large UNMASK DATASET that we curated, with over 18k images in total. We test multiple factors, including: 3 attacks, including the state-of-the-art *Projected Gradient Descent* (PGD) attack; 2 popular CNN architectures, VGG16 [92] and ResNet50 [93]; and multiple combinations of varying numbers of classes and feature overlaps. Experiments demonstrate that our approach *detects* up to 92.9% of gray-box attacks with a false positive rate of 9.67% and (2) *defends* the model by correctly classifying up to 92.24% of adversarial images crafted by PGD. (Section 5.3)

To enhance readability of this chapter, we list and define the terminology used throughout the chapter in Table 5.1. We use the terms “adversarial attack” and “adversarial perturbation” interchangeably to refer to attacks on images. We abbreviate “building-block features” as “features”, and “building-block knowledge extraction model” ( $K$ ) as “building-block model,” when their meanings are clear from context.

## 5.2 UNMASK: Detection and Defense Framework

In this section, we present our building-block knowledge extraction based approach to combating adversarial perturbations (Figure 5.1). The objective is to defend a vulnerable deep learning model  $M$  (Figure 5.1, bottom) using our UNMASK defense framework  $D$ , where the adversary has full access to  $M$  but is unaware of the defense strategy  $D$ . This constitutes a *gray-box* attack on the overall classification pipeline [6].

In Section 5.2.1, we provide the intuition of why building-block knowledge extraction may be well suited for combating adversarial perturbations. Then in Section 5.2.2, we

Symbol	Definition
$X$	Training images
$Y$	Training classification labels
$S$	Training building-block segmentation masks
$C$	Set of possible classes
$V$	Class-feature matrix
$x$	Test image
$\hat{y}$	class prediction from model $M$
$K$	Building-block knowledge extraction model
$M$	Unprotected model
$D$	UNMASK Defense framework
$f_e$	Extracted building-block features from image, by $K$
$f_a$	Expected features of image classified by $M$
$J(f_e, f_a)$	Jaccard similarity between $f_e$ and $f_a$
$s$	similarity score
$d$	distance score (1- $s$ )
$t$	Adversarial-benign classification threshold
$z$	Determination of adversarial or benign
$p$	Class prediction, by UNMASK

Table 5.1: Symbols and Definition

describe how our UNMASK framework leverages this knowledge extraction concept as a new way for detection and defense. We formally define the UNMASK *detection & defense problem* as:

**Given:**

- Training images  $X$ , which contains corresponding classification labels  $Y$  and building-block segmentation masks  $S$ .
- Set of classes  $C$  (e.g., bike,...) and class feature matrix  $V$  (see Table. 5.2). Each class  $c \in C$  is associated with features  $V[c]$  (e.g., wheel,...).

**Output:**

- Detection: adversarial or benign determination  $z \in \{0, 1\}$
- Defense: predicted class label  $p \in C$

### 5.2.1 Intuition: Protection via Building-Block Knowledge Extraction

Our main idea to combat adversarial image perturbations with respect to an input image  $x$ , is to extract building-block features  $f_e$  using a **building-block knowledge extraction model**  $K$ ,  $f_e = K(x)$  (see Figure 5.2 for an example). These extracted building blocks, and their collective composition, forges a layer of protection around the model by disrupting the traditional pixel-centric attacks [4, 94, 95]. Our building-block defensive layer forces the adversary to now solve a more complex problem of manipulating both the class label and all of the image’s constituent parts. For example, in Figure 5.2 the attacker needs to fool the defensive layer into misclassifying the bike as a bird by changing the class label and manipulating the bike building-block features (e.g., *wheel*, *seat*, *handlebar*) into bird features (e.g., *beak*, *wing*, *tail*).

### 5.2.2 Overview of UNMASK

Leveraging the concept of building-block knowledge extraction, we introduce UNMASK as a detection and defense framework ( $D$ ). Figure 5.1 summarizes how our method works at the high level, for an *unprotected* model  $M$  (Figure 5.1, bottom). The adversary crafts an *attacked* image by carefully manipulating its pixel values using an adversarial technique (e.g., Projected Gradient Descent [4]). This attacked image then fools model  $M$  into misclassifying the image, as shown in Figure 5.1. To guard against this kind of attack on  $M$ , we use our UNMASK framework  $D$  in conjunction with the *building-block knowledge extraction model*  $K$  (Figure 5.1, top). Model  $K$  processes the same image, which may be benign or attacked, and extracts the building-block features from the image to compare to the images’ expected features. Figure 5.2 shows an example, where an attacked *bike* image has fooled the unprotected model  $M$  to classify it as a *bird*. We would *expect* the building-block features to include *head*, *claw*, *wing*, and *tail*. However, from the same (attacked) image, UNMASK’s building-block model  $K$  extracts *wheels*, *handle* and *seat*. Comparing the set of *expected* features and the actual *extracted* features (which do not overlap in this example),

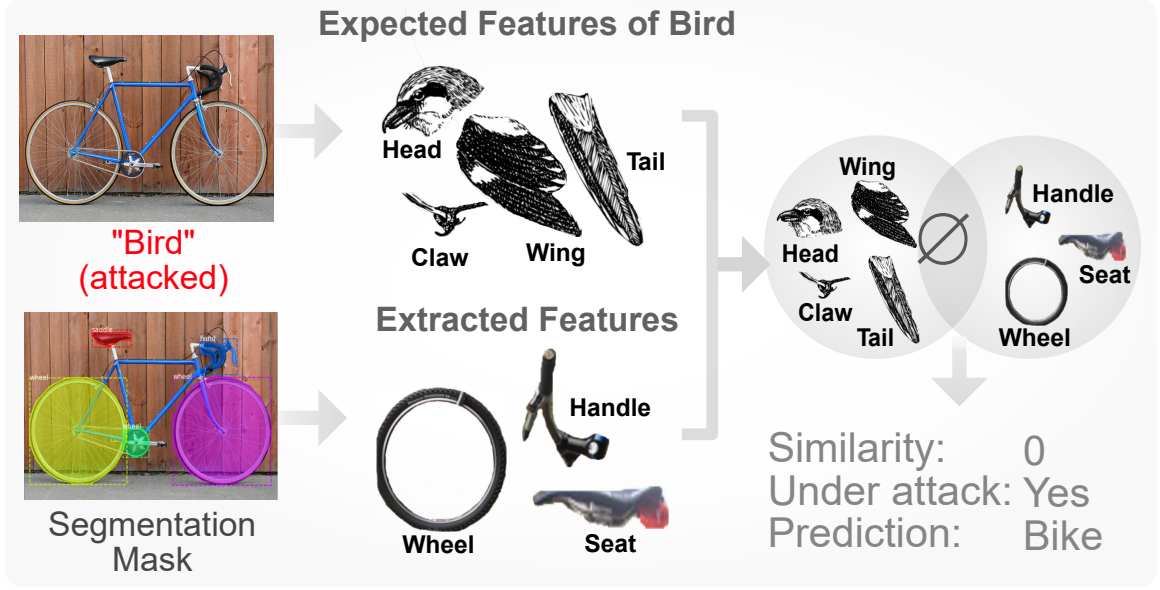


Figure 5.2: UNMASK guards against adversarial image perturbation by extracting building-block features from an image and comparing them to its expected features using Jaccard similarity. If the similarity is below a threshold, UNMASK deems the image adversarial and predicts its class by matching the extracted features with the most similar class.

UNMASK determines the image was attacked, and predicts its class to be *bike* based on the extracted features.

### 5.2.3 Technical Walk-Through of UNMASK

Now, we detail UNMASK’s technical operations and algorithm for detection and defense (Algorithm 1). Its major steps are:

**1. Classify input.** Given an input image  $x$ , UNMASK obtains its class prediction  $\hat{y}$  from (unprotected) model  $M$ , i.e.,  $\hat{y} = M(x)$ . At this point, UNMASK does not know if image  $x$  is adversarial or not.

**2. Extract building-block features.** UNMASK extracts  $x$ ’s features  $f_e$  using *building-block knowledge extraction model*  $K$ , i.e.,  $f_e = K(x)$ . Armed with these features  $f_e$ , UNMASK can utilize them to both detect if model  $M$  is under attack, and to rectify misclassification caused by the attack. We considered multiple approaches for  $K$ , and decided to adopt Mask R-CNN for its ability to leverage image segmentation masks to learn and identify

coherent image regions that closely resemble building-blocks that would appear semantically and visually meaningful to humans [91]. Different from conventional image classification models or object detectors, the annotations used to train our building-block extractor  $K$  are *segmented* object parts instead of the whole objects. For example, for the *wheel* feature, an instance of training data would consist of a bike image and a *segmentation mask* indicating which region of that image represents a wheel. Technically, this means  $K$  uses only a part of an image, and not the whole image, for training. Furthermore, while an image may consist of multiple image parts,  $K$  treats them independently.

**3. Detect attack.** UNMASK measures the similarity between the set of *extracted* features  $f_e$  and the set of *expected* features of  $\hat{y}$  (obtained through matrix  $V[\hat{y}]$ ), by calculating the Jaccard similarity score  $s = J(f_e, f_a)$ . If similarity score  $s$  is greater than the threshold parameter  $t$ , input image  $x$  is deemed benign, otherwise adversarial. Adjusting  $t$  would allow us to assess the trade-off between sensitivity and specificity, which we describe in detail in Section 5.3.

**4. Defend and rectify.** Determining an image to be adversarial also means that model  $M$  is under attack and is giving unreliable classification output. Thus, we need to rectify the misclassification. UNMASK accomplishes this by comparing the extracted features  $f_e$  to every set of class features in  $V$ , outputting class  $\hat{y}$  that contains the highest feature similarity  $s$ ,  $0 \leq s \leq 1$ .

### 5.3 Evaluation

We extensively evaluate UNMASK’s effectiveness in **defending** and **detecting** adversarial perturbations, using:

- 3 attacks, including the state-of-the-art *Projected Gradient Descent* (PGD) attack;
- 2 popular CNN architectures, VGG16 [92] and ResNet50 [93], as unprotected models  $M$ ; and

- multiple combinations of varying numbers of classes and feature overlaps.

To the best of our knowledge, this work proposes the first building-block knowledge extraction to detect and defend against adversarial perturbation for deep learning. We present the first results in this new line of work.

### 5.3.1 Experiment Setup

#### *Software and Hardware*

We develop all experiment code in Python 3.6 on Linux. We use open-source libraries Keras, Tensorflow, Foolbox [96] and Matterport [97]; and GPUs that include two Nvidia Titan X’s, a Titan RTX and a cluster of 24 K40s.

#### *Adversarial Attacks*

We evaluated UNMASK against three attacks, where we detail the parameter selection below:

- **DeepFool (DF)**  $L_2$ : a non-parametric attack that optimizes the amount of perturbation required to misclassify an image[98]; we set the update steps to 100.
- **Fast Gradient Sign (FGSM)**: we set  $\epsilon = 8, 16$ —two common parameters for this attack [3].
- **Projected Gradient Descent with Random Start (PGD)**: PGD is the current strongest first-order attack [4]. Its key parameter  $\epsilon$  represents how much each pixel may be changed by PGD in intensity, e.g.,  $\epsilon = 4$  means changing up to 4 units of intensity (out of 255). It is common to evaluate up to a value of 16 [3, 6] (as perturbation becomes visible) with a stepsize of 0.01 and 40 iterations.

#### UNMASK DATASET

We curated the UNMASK DATASET for our evaluation, which consists of three component datasets—PASCAL-Part, PASCAL VOC 2010 and a subset of ImageNet—as seen in Tables

5.3 and 5.4. The impetus for our curation is to (i) collect all of the data used in our evaluation as a single source to promote ease of reproducibility by our research community, and (ii) to increase the number of images available for evaluating the performance of the deep learning models and the UNMASK defense framework. We designed multiple *class sets* with varying number of classes and feature overlap (e.g., CS3a, in Table 5.2, bottom; and Table 5.5), to study how they would affect detection and defense effectiveness. We further discuss the utilization of the data in Sections 5.3.1 and 5.3.1 below.

### ***Training Building-Block Model $K$***

As illustrated in Figure 5.2 and Section 5.2.3, the building-block knowledge extraction model  $K$  takes an image as input (e.g., bike) and outputs a set of building-block features (e.g., wheel,...). To train  $K$ , we use the PASCAL-Part dataset [99], which consists of 180,423 feature segmentation masks over 9,323 images across the 44 building-block features. The original dataset contains very fine-grained features, such as 18 types of “legs” (e.g., right front lower leg, left back upper leg), while for our purposes we only need the abstraction of “leg”. Therefore, we combined these fine-grained features into more generalized ones (shown as rows in Table 5.2).

We followed a similar procedure described in [97], training  $K$  for 40 epochs. We use a ratio of 80/10/10 for training, validating and testing the model respectively (see Table 5.3). Our work is the first adaptation of Mask R-CNN model for the PASCAL-Part dataset. As such, there are no prior results for comparison. We computed model  $K$ ’s mAP (mean Average Precision), which estimates  $K$ ’s ability to extract features. The model attains an mAP of 0.56, in line with Mask R-CNN on other datasets [99]. Model  $K$  processes up to 4 images per second with a single Nvidia Titan X, matching the speeds reported in [97]. This speed can be easily raised through parallelism by using more GPUs. As building-block extraction is the most time-intensive process of the UNMASK framework, its speed is representative of the overall speed of the framework.

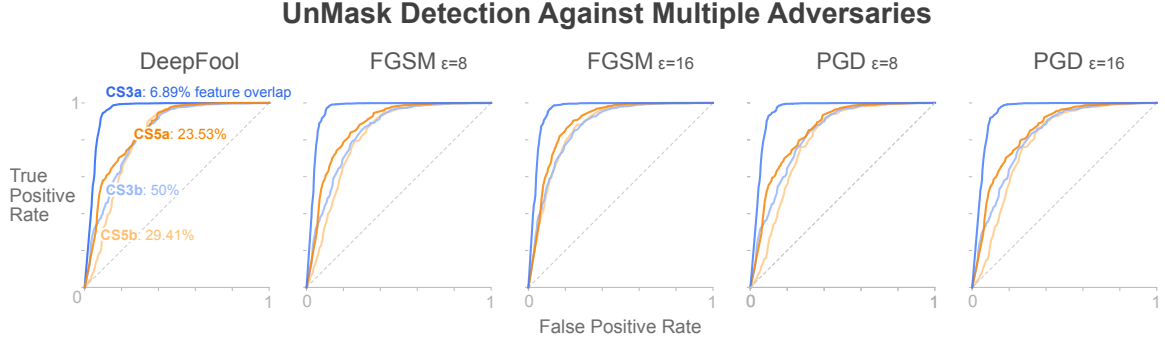


Figure 5.3: UNMASK’s effectiveness in detecting detecting three attacks: DeepFool, FGSM ( $\epsilon=8,16$ ), and PGD ( $\epsilon=8,16$ ). UNMASK’s protection may not be affected strictly based on the number of classes. Rather, an important factor is the *feature overlap* among classes. UNMASK provides better detection when there are 5 classes (dark orange; 23.53% overlap) than when there are 3 (light blue; 50% overlap). Keeping the number of classes constant and varying their feature overlap also supports our observation about the role of feature overlap (e.g., CS3a at 6.89% vs. CS3b at 50%).

### ***Training Unprotected Model $M$***

As described in Section 5.2,  $M$  is the model under attack, and is what UNMASK aims to protect. Our evaluation studies two popular deep learning architectures — VGG16 [92] and ResNet50 [93] — however, UNMASK supports other architectures. Training these models from scratch is generally computationally expensive and requires large amount of data. To reduce such need for computation and data, we adopt the approach described in [97], where we leverage a model pre-trained on ImageNet images, and *replace* its dense layers (i.e., the fully connected layers) to enable us to work with various class sets (e.g., CS3a) In detail, the training process for  $M$  is as follows:

1. Load weights from model pre-trained on ImageNet data.
2. Replace dense layers of the model with new dense layers, allowing us to specify a variable number of classes.
3. Freeze all of the model weights except for the newly-added dense layers, allowing us to preserve the ImageNet features contained in the early layers while training the new dense layers on our data.

We chose to train the new dense layers using the PASCAL VOC 2010 dataset [100] for its desirable connection to the Pascal-Part dataset — PASCAL-Part uses the images from PASCAL VOC and adds *segmentation masks* for those images to describe image parts. Thus, we can readily ensure that the classes of model  $M$  and model  $K$  are at parity. In practice, other datasets containing the classes from Table 5.2 may also be used. Refer to Table 5.3, for a full breakdown of the data used for training and evaluation.

### 5.3.2 Evaluating UNMASK Defense and Detection

The key research questions that our evaluation aims to address is how effectively UNMASK can **detect** adversarial images, and **defend** against attacks by rectifying misclassification through inferring the actual class label. Most image datasets containing the classes from Table 5.2 may be used. However, we use ImageNet data (see Table 5.4) as it matches our class sets and has a large number of available images. We note that the evaluation is focused on images containing a single-class (i.e., no “person” and “car” in same image) as this allows for a more controlled environment.

#### *Evaluating Detection of Attacks*

To evaluate UNMASK’s effectiveness in detecting adversarial images against attacks (DF, FGSM, PGD), we use a contamination level of 0.5—meaning half of the images are benign and the other half are adversarial. Figure 5.3 summarizes UNMASK’s detection effectiveness, using *receiver operating characteristics* (ROC) curves constructed by varying the adversarial-benign threshold  $t$ . The curves show UNMASK’s performances across operating points as measured by the tradeoff between *true positive* (TP) and *false positive* (FP) rates.

An interesting characteristic of UNMASK’s protection is that its effectiveness may not be affected strictly based on the number of classes in the dataset as in conventional classification tasks. Rather, an important factor is how much *feature overlap* there is among the classes. The ROC curves in Figure 5.3 illustrate this phenomenon, where UNMASK

provides better detection when there are 5 classes (Figure 5.3, dark orange) than when there are 3 classes (light blue). As shown in Table 5.5, the 5-class setup (CS5a—dark orange) has a feature overlap of 23.53% across the 5 classes’ 34 unique features, while the 3-class setup (CS3b—light blue) has 50% overlap. Keeping the number of classes constant and varying their feature overlap also supports this observation about the role of feature overlap (e.g., CS3a vs. CS3b in Figure 5.3). We call each combination of *class count* and *feature overlap* a “class set,” abbreviated as “CS.” CS3 thus means a class set with 3 classes. CS3a and CS3b have the same number of classes, with different feature overlap. Table 5.4 details the number of images used across the four class sets we investigated.

For a given feature overlap level, UNMASK performs similarly across attack methods. When examining feature overlap 6.89% (CS3a) on VGG16, UNMASK attains an AUC scores of 0.952, 0.96, 0.959, 0.951 and 0.949 on attacks DF, FGSM ( $\epsilon=8,16$ ) and PGD ( $\epsilon=8,16$ ), respectively. This result is significant because it highlights the ability of UNMASK to operate against multiple strong attack strategies to achieve high detection success rate. As a representative ROC operating point for the attack vectors, we use PGD ( $\epsilon=8$ ), on feature overlap 6.89%. In this scenario, UNMASK is able to detect up to 92.67% of attacks with a false positive rate of 9.67%. We believe that performing well in a low feature overlap environment is all that is required. This is because in many circumstances it is not important to distinguish the exact true class (e.g., dog or cat) of the image, but whether the image is being completely misclassified (e.g., car vs. person). Therefore, in practice, classes can be selected such that feature overlap is minimized.

### ***Evaluating Defense and Rectification***

Detecting an attack is only the first step of UNMASK’s protection. It also rectifies the misclassification by comparing the extracted features  $f_e$  to every set of class features in  $V$ , and outputting class  $c$  that contains the highest feature similarity. As the evaluation focus is on rectifying misclassification, our test images have a contamination level of 1—meaning

all of the images are adversarial. We evaluate UNMASK’s rectification capability on:

- 2 neural network models (VGG16, ResNet50)
- 3 attacks (DF, FGSM, PGD)
- 4 class sets (CS3a, CS3b, CS5a, CS5b)

Table 5.6 shows that UNMASK is agnostic to the deep learning model being protected, as measured by the ability of UNMASK to infer an adversarial images’ actual class. This can be seen when comparing the results across each attack on VGG16 and ResNet50.

In addition, we find that the results from Table 5.6 support our observation that *feature overlap* is the dominant factor in determining the effectiveness of UNMASK, as opposed to the number of classes. When examining DeepFool on class set CS3b (3 classes; feature overlap 50%), UNMASK is able to determine the underlying class 85.62% of the time. At class set CS5a (5 classes; feature overlap 23.53%) we obtain an accuracy of 91.11%, highlighting the important role that feature overlap plays in UNMASK’s defense ability.

It is interesting to note that FGSM is more effective at attacking our UNMASK defense than the other two attacks. We believe this is due to the single-step attacks’ better transferability, which has been reported in prior work [3]. Given this transferability property of FGSM, we believe UNMASK provides a significant defense.

We also mention the fact that UNMASK’s accuracy can be higher than the un-attacked model  $M$  due to the fact that, in some instances, model  $K$  learned a better representation of the data through the feature masks as opposed to model  $M$ , which trained on the images directly. This occurs on multiple occasions in Table 5.6.

## 5.4 Conclusion & Discussion

In this work, we have introduced a new fundamental concept of *building-block knowledge extraction*, and showed how it protects deep learning models against adversarial attacks through the UNMASK detection and defense framework. We draw inspiration from humans’

natural ability to make robust classification decisions through the detection and synthesis of contextual building-block knowledge contained in images. We designed and developed our UNMASK framework to simulate such capability, so it can detect adversarial pixel-centric manipulations targeting a deep learning model, and defend the model against attacks by rectifying the classification. Through extensive evaluation on large-scale real-world image data, we showcase the merits of our ideas through UNMASK’s ability to *detect* up to 92.9% of attacks with a false positive rate of 9.67% and *defend* deep learning models by correctly classifying up to 92.24% of adversarial images in the gray-box scenario. Our proposed method is fast and architecture-agnostic.

In this work, we direct our efforts to studying the efficacy of UNMASK and the concept of building-block knowledge extraction on their own. As myriads of newer and stronger attack strategies are continuously discovered, our approach is not a panacea to defend all possible (future) attacks, and we do not intend for it to be used in isolation of other techniques. Rather, we believe that detection and defense strategies should be combined. We expect our approach to be one of multiple techniques that are used in concert to provide comprehensive protection. Multi-pronged protection is a proven, long-standing defense strategy pervasive in security research and in practice [82, 45]. Fortunately, our proposed technique can be readily integrated with many existing techniques, as it operates in parallel to the deep learning model that it aims to protect (see Figure 5.1).

We note that UNMASK has the potential vulnerability to attacks that simultaneously target and manipulate all building-block features, e.g., changing every “bike” parts in a bike image, into “bird” parts (bike wheel→bird wing; bike handlebar→bird tail). Such simultaneous, multi-part attack could be challenging to formulate and execute. To the best of our knowledge, we have not yet encountered it in research or practice.

Future research directions include extending UNMASK to the object detection task. That is, we protect an object detector with an auxiliary detector that detects object parts. Such defense has the potential to mitigate *ShapeShifter*-style attacks in the black-box or gray-box

settings. We did not evaluate UNMASK directly against *ShapeShifter* because *ShapeShifter* currently only works in the white-box setting and has limited black-box transferability. We also want to improve UNMASK by reducing the dependency on object part labeling.

---

**Algorithm 1: UNMASK**

---

**Input:** Training images  $X$ , labels  $Y$ , segmentation masks  $S$ , set of possible classes  $C$ , attribute matrix  $V$ , threshold  $t$ , test image  $x$

**Result:** adversarial prediction  $z \in \{0, 1\}$ , predicted class  $p$

**Train unprotected classification model  $M$ :**

$M = \text{NeuralNet}(X, Y);$

$\hat{y} = M(x);$

**Train building-block extraction model  $K$ :**

$K = \text{Mask-RCNN}(X, S);$

$f_e = K(x);$

(extracted building blocks)

$f_a = V[\hat{y}];$

(expected building blocks)

**Detection:**

$s = J(f_e, f_a); d = 1 - s;$

$$z = \begin{cases} 0 \text{ (benign)}, & \text{if } d < t \\ 1 \text{ (adversarial)}, & \text{if } d \geq t \end{cases}$$

**Defense:**

$$p = \begin{cases} \hat{y}, & \text{if } z = 0 \\ \underset{c \in C}{\operatorname{argmin}} J(f_e, V[c]), & \text{if } z = 1 \end{cases}$$

**return**  $z, p;$

---

Features	Airplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	Television
Arm	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Beak	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Body	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Cap	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Coach	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Door	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Engine	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Ear	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Eye	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Eyebrow	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Foot	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Front side	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Hair	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Hand	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Head	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Headlight	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Hoof	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Horn	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Leg	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
License plate	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Mirror	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Mouth	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Muzzle	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Neck	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Nose	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Paw	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Plant	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Pot	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Saddle	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Screen	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Stern	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Tail	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Torso	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Vehicle	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Wheel	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Window	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Wing	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Class Set (CS)																				
CS3a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CS3b	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CS5a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CS5b	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Table 5.2: Class-Feature Matrix. Top: dots mark classes’ features. Bottom: four class sets with varying levels of feature overlap.

Setup		PASCAL-Part			PASCAL VOC 2010		
Model	Classes	Train	Val	Test	Train	Val	Test
<b>K</b>	44	7,457	930	936	-	-	-
<b>M</b>	CS3a	-	-	-	1,750	350	1,400
	CS3b	-	-	-	2,104	421	1,684
	CS5a	-	-	-	2,264	452	1,812
	CS5b	-	-	-	2,501	500	2,001

Table 5.3: Number of images used in training models  $K$  and  $M$ .

Class Set	Defense			Detection
	DF	FGSM	PGD	All Attacks
CS3a	3,485	2,823	3,494	3,494
CS3b	4,749	4,161	4,764	4,764
CS5a	5,827	5,252	5,849	5,849
CS5b	6,728	5,883	6,747	6,747

Table 5.4: Number of ImageNet images used to evaluate UNMASK. Only the images that can be successfully perturbed by the attack are used, thus the variations in numbers. We report values for PGD and FGSM with  $\epsilon=16$ . The numbers for  $\epsilon=8$  are similar.

Class Set	Classes	Unique Parts	Overlap
CS3a	3	29	6.89%
CS3b	3	18	50.00%
CS5a	5	34	23.53%
CS5b	5	34	29.41%

Table 5.5: Four *class sets* investigated in our evaluation, with varying number of classes and feature overlap.

Model $M$	Class Set	Overlap	No Attk	DeepFool (No Def)	DeepFool	FGSM	PGD
<b>VGG16</b>	CS3a	6.89%	87.00	5.13	94.33	73.44	89.89
	CS3b	50.00%	89.13	3.47	85.62	60.11	75.19
	CS5a	23.53%	80.35	3.91	91.11	65.86	82.65
	CS5b	29.41%	81.36	3.04	87.17	62.88	77.02
<b>ResNet50</b>	CS3a	6.89	86.64	4.51	95.04	74.42	90.81
	CS3b	50.00	85.75	3.28	86.12	66.71	78.55
	CS5a	23.53	80.35	3.91	91.11	65.86	82.65
	CS5b	29.41	79.91	3.33	87.57	65.19	80.01

Table 5.6: UNMASK’s accuracies (in %) in countering three attacks: DeepFool, FGSM, and *Projected Gradient Descent* (PGD). We test two popular CNN architectures, VGG16 and ResNet50, as unprotected model  $M$ , with four class sets with varying numbers of classes and feature overlap. We use  $\epsilon = 16$  for FGSM and PGD in this experiment. We show the models’ accuracies (1) when not under attack (“No Attk” column); (2) attacked without defense (“DeepFool (No Def)”); for both FGSM and PGD, the accuracies drop to 0 without defense and are omitted in this table; and (3) attacked and defended by UNMASK (the last three columns).

## **Part II**

# **Theoretically-Principled Defense via Game Theory and ML**

## OVERVIEW

Most non-trivial security problems require some kind of decision making and resource allocation. For example, a company that wants to implement security controls with a limited budget needs to make trade-offs in its deployment. I modeled this problem as a two-player zero-sum game between a defender and an attacker, and introduced a novel solution concept called **diversified mixed strategy** (Chapter 6).

Inspired by the proverb “Don’t put all your eggs in one basket,” my new solution concept compels players to employ a “diversified” strategy that does not place too much weight on any one action. Furthermore, by leveraging the deep connection between game theory and boosting, we develop a communication-efficient **distributed boosting** algorithm with strong theoretical guarantees (Chapter 7) in the agnostic learning setting where the data can contain arbitrary noise. Our algorithm achieves exponential improvement in communication complexity over prior work and solves an open problem in distributed learning.

## CHAPTER 6

### DIVERSIFIED STRATEGIES FOR MITIGATING ADVERSARIAL ATTACKS IN MULTIAGENT SYSTEMS

In this chapter we consider online decision-making in settings where players want to guard against possible adversarial attacks or other catastrophic failures. To address this, we propose a solution concept in which players have an additional constraint that at each time step they must play a *diversified* mixed strategy: one that does not put too much weight on any one action. This constraint is motivated by applications such as finance, routing, and resource allocation, where one would like to limit one’s exposure to adversarial or catastrophic events while still performing well in typical cases. We explore properties of diversified strategies in both zero-sum and general-sum games, and provide algorithms for minimizing regret within the family of diversified strategies as well as methods for using taxes or fees to guide standard regret-minimizing players towards diversified strategies. We also analyze equilibria produced by diversified strategies in general-sum games. We show that surprisingly, requiring diversification can actually lead to higher-welfare equilibria, and give strong guarantees on both price of anarchy and the social welfare produced by regret-minimizing diversified agents. We additionally give algorithms for finding optimal diversified strategies in distributed settings where one must limit communication overhead.

#### 6.1 Introduction

A common piece of advice when one needs to make decisions in the face of unknown future events is “Don’t put all your eggs in one basket.” This is especially important when there can be an adversarial attack or catastrophic failure. We consider game-theoretic problems from this perspective, design online learning algorithms with good performance subject to such exposure-limiting constraints on behavior, and analyze the effects of these constraints

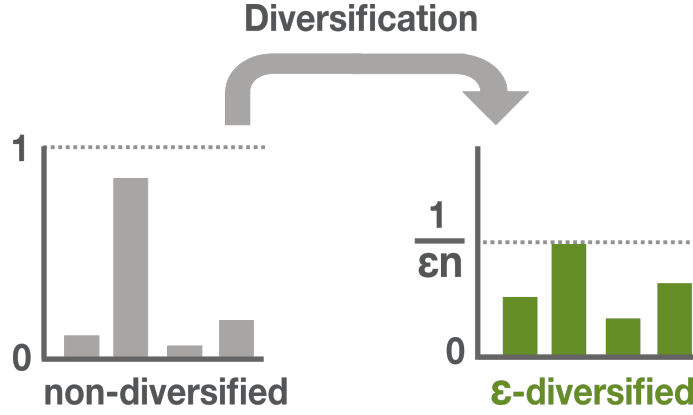


Figure 6.1: For  $\epsilon \in [\frac{1}{n}, 1]$ , we define a probability distribution  $P$  to be  $\epsilon$ -diversified if  $P(i) \leq \frac{1}{\epsilon n}$  for all  $i$ . A distribution can be diversified through Bregman projection into the set of all  $\epsilon$ -diversified distributions. A mixed strategy determined by a diversified distribution is called a diversified (mixed) strategy. We explore properties of such diversified strategies in both zero-sum and general-sum games as well as give algorithmic guarantees.

on the expected value obtained (in zero-sum games) and the overall social welfare produced (in general-sum games).

As an example, consider a standard game-theoretic scenario: an agent must drive from point A to point B and has  $n$  different routes it can take. We could model this as a game  $M$  where rows correspond to the  $n$  routes, columns correspond to  $m$  possible traffic patterns, and entry  $M(i, j)$  is the cost for using route  $i$  under traffic pattern  $j$ . However, suppose the agent is carrying valuable documents and is concerned an adversary might try to steal them. In this case, to reduce the chance of this happening, we might require that no route have more than (say) 10% probability. The agent then wants to minimize expected travel time subject to this requirement. Or in an investment scenario, if rows correspond to different investments and columns to possible market conditions, we might have an additional worry that perhaps one of the investment choices is run by a crook. In this case, we may wish to restrict the strategy space to allocations of funds that are not too concentrated.

To address such scenarios, for  $\epsilon \in [\frac{1}{n}, 1]$  let us define a probability distribution (or allocation)  $P$  to be  $\epsilon$ -diversified if  $P(i) \leq \frac{1}{\epsilon n}$  for all  $i$ . For example, for  $\epsilon = \frac{1}{n}$  this is no restriction at all, for  $\epsilon = 1$  this requires the uniform distribution, and for intermediate values of  $\epsilon$  this requires an intermediate level of diversification. We then explore properties of such

diversified strategies in both zero-sum and general-sum games as well as give algorithmic guarantees.

For zero-sum games, define  $v_\epsilon$  to be the minimax-optimal value of the game in which the row player is restricted to playing  $\epsilon$ -diversified mixed strategies. Natural questions we address are: Can one design adaptive learning algorithms that maintain  $\epsilon$ -diversified distributions and minimize regret within this class so they never perform much worse than  $v_\epsilon$ ? Can a central authority “nudge” a generic non-diversified regret-minimizer into using diversified strategies via fines or taxes (extra loss vectors strategically placed into the event stream) such that it maintains low-regret over the original sequence? And for reasonable games, how much worse is  $v_\epsilon$  compared to the non-diversified minimax value  $v$ ? We also consider a dual problem of producing a strategy  $Q$  for the column player that achieves value  $v_\epsilon$  against all but an  $\epsilon$  fraction of the rows (which an adversary can then aim to attack).

One might ask why not model such an adversary directly within the game, via additional columns that each give a large loss to one of the rows. The main reason is that these would then dominate the minimax value of the game. (And they either would not have values within the usual  $[0, 1]$  range assumed by regret-minimizing learners, or, if they were scaled to lie in this range, they would cause all other events to seem roughly the same). Instead, we want to consider learning algorithms that optimize for more common events, while keeping to the constraint of maintaining diversified strategies. We also remark that one could also make diversification a soft constraint by adding a loss term for not diversifying.

We next consider general-sum games, such as routing games and atomic congestion games, in which  $k$  players interact in ways that lead to various costs being incurred by each player. We show that surprisingly, requiring a player to use diversified strategies can actually improve its performance in equilibria in such games. We then study the  *$\epsilon$ -diversified price of anarchy*: the ratio of the social cost of the worst equilibrium subject to all players being  $\epsilon$ -diversified to the social cost of the socially-best set of  $\epsilon$ -diversified strategies. We show that in some natural games, even requiring a small amount of diversification can

dramatically improve the price of anarchy of the game, though we show there also exist games where diversification can make the price of anarchy worse. We also bring several threads of this investigation together by showing that for the class of *smooth games* defined by Roughgarden [101], for any diversification parameter  $\epsilon \in [\frac{1}{n}, 1]$ , the  $\epsilon$ -diversified price of anarchy is no worse than the smoothness of the game, and moreover, players using diversified regret-minimizing strategies will indeed approach this bound. Thus, we get strong guarantees on the quality of interactions produced by self-interested diversified play. Finally, we consider how much diversification can hurt *optimal* play, showing that in random unit-demand congestion games, diversification indeed incurs a low penalty.

Lastly, we consider an information-limited, distributed, “big-data” setting in which the number of rows and columns of the matrix  $M$  is very large and we do not have it explicitly. Specifically, we assume the  $n$  rows are distributed among  $r$  processors, and the only access to the matrix  $M$  we have is via an oracle for the column player that takes in a sample of rows and outputs the column player’s best response. What we show is how in such a setting to produce near optimal strategies for each player in the sense described above, from very limited communication among processors.

In addition to our theoretical results, we also present experimental simulations for both zero-sum and general-sum games.

### 6.1.1 Related Work

There has been substantial work on design of “no-regret” learning algorithms for repeated play of zero-sum games [102, 103, 104]. *Multiplicative Weight Update* methods [105, 106] are a specific type of no-regret algorithm that have received considerable attention in game theory [102, 107], machine learning [102, 108], and many other research areas [109, 110], due to their simplicity and elegance.

We consider the additional constraint that players play diversified mixed strategies, motivated by the goal of reducing exposure to adversarial attacks. The concept of diversified

strategies, sometimes called “smooth distributions”, appears in a range of different areas [109, 111, 112]. [113] considers a somewhat related notion where there is a penalty for deviation from a given fixed strategy, and shows existence of equilibria in such games. Also related is work on adversarial machine learning, e.g., [114, 115, 116]; however, in this work we are instead focused on decision-making scenarios.

Our distributed algorithm is inspired by prior work in distributed machine learning [117, 118, 108], where the key idea is to perform weight updates in a communication efficient way. Other work on the impact of adversaries in general-sum games appears in [119, 120, 121].

## 6.2 Zero-Sum Games

We begin by studying two-player zero-sum games. Recall that a two-player zero-sum game is defined by a  $n \times m$  matrix  $M$ . In each round of the game, the row player chooses a distribution  $P$  over the rows of  $M$ , and the column player chooses a distribution  $Q$  over the columns of  $M$ . The expected loss of the row player is

$$M(P, Q) = P^T M Q = \sum_{i,j} P(i) M(i, j) Q(j),$$

where  $M(i, j) \in [0, 1]$  is the loss suffered by the row player if the row player plays row  $i$  and the column player plays column  $j$ . The goal of the row player is to minimize its loss, and the goal of the column player is to maximize this loss. The minimax value  $v$  of the game is:

$$v = \min_P \max_Q M(P, Q) = \max_Q \min_P M(P, Q).$$

### 6.2.1 Multiplicative Weights and Diversified Strategies

We now consider row players restricted to only playing diversified distributions, defined as follows.

**Definition 1.** A distribution  $\mathbf{p} \in \Delta_n$  is called  $\epsilon$ -diversified if  $\max_i p_i \leq \frac{1}{\epsilon n}$ .

Let  $\mathcal{P}_\epsilon$  be the set of all  $\epsilon$ -diversified distributions, and let  $v_\epsilon$  be the minimax value of the game subject to the row player restricted to playing in  $\mathcal{P}_\epsilon$ . Note that the range of  $\epsilon$  is between  $1/n$  and 1. It is easy to verify that  $\mathcal{P}_\epsilon$  is a convex set. As a result, the minimax theorem applies to  $\mathcal{P}_\epsilon$  [122], and we call the minimax value  $v_\epsilon$ :

$$v_\epsilon = \min_{P \in \mathcal{P}_\epsilon} \max_Q M(P, Q) = \max_Q \min_{P \in \mathcal{P}_\epsilon} M(P, Q).$$

The multiplicative weights update algorithm [105, 123] can be naturally adapted to maintain diversified strategies by projecting its distributions into the class  $\mathcal{P}_\epsilon$  if they ever step outside of it. This is shown in Algorithm 2. By adapting the analysis of [123] to this case, we arrive at the following regret bound.

**Theorem 1.** For any  $0 < \gamma \leq 1/2$  and any positive integer  $T$ , Algorithm 2 generates distributions  $P^{(1)}, \dots, P^{(T)} \in \mathcal{P}_\epsilon$  to responses  $j_1, \dots, j_T$ , such that for any  $P \in \mathcal{P}_\epsilon$ ,

$$\sum_{t=1}^T M(P^{(t)}, j_t) \leq (1 + \gamma) \sum_{t=1}^T M(P, j_t) + \frac{RE(P \| P^{(1)})}{\gamma},$$

where  $RE(p \| q) = \sum_i p_i \ln(p_i/q_i)$  is relative entropy.

By combining Algorithm 2 with a best-response oracle for the column player, and applying Theorem 1 and a standard argument [107, 106] we have:

**Theorem 2.** Running Algorithm 2 for  $T$  steps against a best-response oracle, one can construct mixed strategies  $\bar{P}$  and  $\bar{Q}$  s.t.

$$\max_Q M(\bar{P}, Q) \leq v_\epsilon + \Delta_T \quad \text{and} \quad \min_{P \in \mathcal{P}_\epsilon} M(P, \bar{Q}) \geq v_\epsilon - \Delta_T,$$

for  $\Delta_T = 2\sqrt{\frac{\ln(1/\epsilon)}{T}}$ , where  $\bar{P} = \frac{1}{T} \sum_{t=1}^T P^{(t)}$  and  $\bar{Q} = \frac{1}{T} \sum_{t=1}^T j_t$ .

---

**Algorithm 2** Multiplicative Weights Update algorithm with Restricted Distributions
 

---

**Initialization:** Fix a  $\gamma \leq \frac{1}{2}$ . Set  $P^{(1)}$  to be the uniform distribution.

**For**  $t = 1, 2, \dots, T$ :

1. Choose distribution  $P^{(t)}$
2. Receive the pure strategy  $j_t$  for the column player
3. Compute the multiplicative update rule

$$\hat{P}_i^{(t+1)} = P_i^{(t)}(1 - \gamma)^{M(i, j_t)} / Z^{(t)}$$

where  $Z^{(t)} = \sum_i P_i^{(t)}(1 - \gamma)^{M(i, j_t)}$  is the normalization factor.

4. Project  $\hat{P}^{(t+1)}$  into  $\mathcal{P}_\epsilon$

$$P^{(t+1)} = \arg \min_{P \in \mathcal{P}_\epsilon} RE(P \parallel \hat{P}^{(t+1)})$$


---

*Proof.* We can sandwich the desired inequalities inside a proof of the minimax theorem as follows:

$$\begin{aligned} \min_{P \in \mathcal{P}_\epsilon} \max_Q M(P, Q) &\leq \max_Q M(\bar{P}, Q) = \max_Q \frac{1}{T} \sum_{t=1}^T M(P^{(t)}, Q) \\ &\leq \frac{1}{T} \sum_{t=1}^T \max_Q M(P^{(t)}, Q) = \frac{1}{T} \sum_{t=1}^T M(P^{(t)}, j_t) \\ &\leq \min_{P \in \mathcal{P}_\epsilon} \frac{1 + \gamma}{T} \sum_{t=1}^T M(P, j_t) + \frac{\ln(1/\epsilon)}{\gamma T} \\ &\leq \min_{P \in \mathcal{P}_\epsilon} M(P, \bar{Q}) + \gamma + \frac{\ln(1/\epsilon)}{\gamma T} \\ &\leq \max_Q \min_{P \in \mathcal{P}_\epsilon} M(P, Q) + \gamma + \frac{\ln(1/\epsilon)}{\gamma T} \end{aligned}$$

If we set  $\gamma = \sqrt{\frac{\ln(1/\epsilon)}{T}}$ , then  $\Delta_T = \gamma + \frac{\ln(1/\epsilon)}{\gamma T} = 2\sqrt{\frac{\ln(1/\epsilon)}{T}}$ . The two inequalities in the theorem follow by skipping the first and the last inequalities from the proof above, respectively.  $\square$

The next theorem shows that the distribution  $\bar{Q}$  in Theorem 2 is also a good mixed strategy for the column player against any row-player strategy if we remove a small fraction of the rows.

**Theorem 3.** *By running Algorithm 2 for  $T$  steps against a best-response oracle, we can construct a mixed strategy  $\bar{Q}$  such that for all but an  $\epsilon$  fraction of the rows  $i$ ,  $M(i, \bar{Q}) \geq v_\epsilon - \gamma$ . Moreover we can do this with at most  $T = O\left(\frac{\log(1/\epsilon)}{\gamma^2(1+\gamma-v_\epsilon)}\right)$  oracle calls.*

*Proof.* We generate distributions  $P^{(1)}, \dots, P^{(T)} \in \mathcal{P}_\epsilon$  by using Algorithm 2. Let  $j_t$  be the column returned by the oracle with the input  $P^{(t)}$ . After  $T = \left\lceil \frac{\log(1/\epsilon)}{\gamma^2(1+\gamma-v_\epsilon)} \right\rceil + 1$  rounds, we set the mixed strategy  $\bar{Q} = \frac{1}{T} \sum_{t=1}^T j_t$ . Set  $E = \{i | M(i, \bar{Q}) < v_\epsilon - \gamma\}$ . Suppose for contradiction that  $|E| \geq \epsilon n$ . Let  $P = u_E$ , the uniform distribution on  $E$  and 0 elsewhere. It is easy to see that  $u_E \in \mathcal{P}_\epsilon$ , since  $|E| \geq \epsilon n$ .

By the assumption of the oracle, we have  $v_\epsilon T \leq \sum_{t=1}^T M(P^{(t)}, j_t)$ . In addition, by Theorem 1, we have

$$\sum_{t=1}^T M(P^{(t)}, j_t) \leq (1 + \gamma) \sum_{t=1}^T M(P, j_t) + \frac{RE(P \parallel P^{(1)})}{\gamma}.$$

For any  $i \in E$ ,  $\sum_{t=1}^T M(i, j_t) = T \cdot M(i, \bar{Q}) < (v_\epsilon - \gamma)T$ . Since  $P$  is the uniform distribution on  $E$ , we have  $\sum_{t=1}^T M(P, j_t) < (v_\epsilon - \gamma)T$ . Furthermore, since  $|E| \geq \epsilon n$ , we have

$$RE(P \parallel P^{(1)}) = RE(u_E \parallel u) \leq \ln(1/\epsilon).$$

Putting these facts together, we get  $v_\epsilon T \leq (1 + \gamma)(v_\epsilon - \gamma)T + \frac{\ln(1/\epsilon)}{\gamma}$ , which implies  $T \leq \frac{\ln(1/\epsilon)}{\gamma^2(1+\gamma-v_\epsilon)}$ , a contradiction.  $\square$

### 6.2.2 Diversifying Dynamics

Theorem 1 shows that it is possible for a player to maintain an  $\epsilon$ -diversified distribution at all times while achieving low regret with respect to the entire family  $\mathcal{P}_\epsilon$  of  $\epsilon$ -diversified distributions. However, suppose a player, who is allocating an investment portfolio among  $n$  investments, does not recognize the need for maintaining a diversified distribution and simply uses the standard multiplicative-weights algorithm to minimize regret. For example, the player might not realize that the matrix  $M$  only represents “typical” behavior of investments,

---

**Algorithm 3** Multiplicative Weights Update algorithm with Interventions
 

---

**Initialization:** Fix a  $\gamma \leq \frac{1}{2}$ . Set  $P^{(1)}$  to be the uniform distribution.

**For**  $t = 1, 2, \dots, T$ :

1. Choose distribution  $P^{(t)}$
2. Receive the pure strategy  $j_t$  for the column player
3. Compute the multiplicative update rule

$$P_i^{(t+1)} = P_i^{(t)}(1 - \gamma)^{M(i, j_t)} / Z^{(t)}$$

where  $Z^{(t)} = \sum_i P_i^{(t)}(1 - \gamma)^{M(i, j_t)}$  is the normalization factor.

4. While  $P^{(t+1)}$  is not  $(1 - \gamma)\epsilon$ -diversified, run multiplicative update (Step 3) on fake loss vector  $\ell$  defined as:

$$\ell_i = \begin{cases} 1 & \text{if } P_i^{(t+1)} > \frac{1}{(1-\gamma)\epsilon n} \\ 0 & \text{if } P_i^{(t+1)} \leq \frac{1}{(1-\gamma)\epsilon n} \end{cases}$$


---

and that a crooked portfolio manager or clever hacker could cause an entire investment to be wiped out. This player might quickly reach a dangerous non-diversified portfolio in which nearly all of its weight is just on one row.

Suppose, however, that an investment advisor or helpful authority has the ability to charge fees on actions whose weights are too high, that can be viewed as inserting fake loss vectors into the stream of loss vectors observed by the player's algorithm. We show here that by doing so in an appropriate manner, this advisor or authority can ensure that the player both (a) maintains diversified distributions, and (b) incurs low regret with respect to the family  $\mathcal{P}_\epsilon$  over the sequence of real loss vectors. Viewed another way, this can be thought of as an alternative to Algorithm 2 with slightly weaker guarantees but that does not require the projection. The algorithm remains efficient.

**Theorem 4.** *Algorithm 3 generates distributions  $P^{(1)}, \dots, P^{(T)}$  such that*

- (a)  $P^{(t)} \in \mathcal{P}_{(1-\gamma)\epsilon}$  for all  $t$ , and

(b) for any  $P \in \mathcal{P}_\epsilon$  we have

$$\sum_{t=1}^T M(P^{(t)}, j_t) \leq (1 + \gamma) \sum_{t=1}^T M(P, j_t) + \frac{RE(P \parallel P^{(1)})}{\gamma}.$$

*Proof.* For part (a) we just need to show that the while loop in Step 4 of the algorithm halts after a finite number of loops. To show this, we show that each time a fake loss vector is applied, the gap between the maximum and minimum total losses (including both actual losses and fake losses) over the rows  $i$  is reduced. In particular, the multiplicative-weights algorithm has the property that the probability on an action  $i$  is proportional to  $(1 - \gamma)^{L_{total}^i}$  where  $L_{total}^i$  is the total loss (actual plus fake) on action  $i$  so far; so, the actions of highest probability are also the actions of lowest total loss. This means that in Step 4, there exists some threshold  $\tau$  such that  $\ell_i = 1$  for all  $i$  of total loss at most  $\tau$  and  $\ell_i = 0$  for all  $i$  of total loss greater than  $\tau$ . Since we are adding 1 to those actions of total loss at most  $\tau$ , this means that the gap between the maximum and minimum total loss over all the actions is decreasing, so long as that gap was greater than 1. However, note that if  $P^{(t+1)}$  is not  $(1 - \gamma)\epsilon$ -diversified then the gap between maximum and minimum total loss must be greater than 1, by definition of the update rule and using the fact that  $\epsilon \leq 1$ . Therefore, the gap between maximum and minimum total loss is strictly reduced on each iteration (and reduced by at least 1 if any row is ever updated twice) until  $P^{(t+1)}$  becomes  $(1 - \gamma)\epsilon$ -diversified.

For part (b), define  $L_{actual}^{alg} = \sum_{t=1}^T M(P^{(t)}, j_t)$  to be the actual loss of the algorithm and define  $L_{actual}^P = \sum_{t=1}^T M(P, j_t)$  to be the actual loss of some  $\epsilon$ -diversified distribution  $P$ . We wish to show that  $L_{actual}^{alg}$  is not too much larger than  $L_{actual}^P$ . To do so, we begin with the fact that, by the usual multiplicative weights analysis, the algorithm has low regret with respect to any fixed strategy over the entire sequence of loss vectors (actual and fake). Say the algorithm's total loss is  $L_{total}^{alg} = L_{actual}^{alg} + L_{fake}^{alg}$  and the total loss of  $P$  is  $L_{total}^P = L_{actual}^P + L_{fake}^P$ . We know that

$$L_{total}^{alg} \leq (1 + \gamma)L_{total}^P + \frac{RE(P \parallel P^{(1)})}{\gamma},$$

which we can rewrite as:

$$L_{actual}^{alg} + L_{fake}^{alg} \leq (1 + \gamma)L_{actual}^P + (1 + \gamma)L_{fake}^P + \frac{RE(P\|P^{(1)})}{\gamma}.$$

Thus, to prove part (b) it suffices to show that  $L_{fake}^{alg} \geq (1 + \gamma)L_{fake}^P$ . But notice that on each fake loss vector, for each index  $i$  such that  $\ell_i = 1$ , the algorithm has strictly more than  $\frac{1}{(1-\gamma)\epsilon n} > \frac{1+\gamma}{\epsilon n}$  probability mass on row  $i$ . In contrast,  $P$  has at most  $\frac{1}{\epsilon n}$  probability mass on row  $i$ , since  $P$  is  $\epsilon$ -diversified. Therefore  $L_{fake}^{alg} \geq (1 + \gamma)L_{fake}^P$  and the proof is complete.  $\square$

This analysis can be extended to the case of an advisor who only periodically monitors the player's strategy. If the advisor monitors the strategy every  $k$  steps, then in the meantime the maximum probability that any row  $i$  can reach is  $\frac{1}{(1-\gamma)^k \epsilon n}$ . So, part (a) of Theorem 4 would need to be relaxed to  $P^{(t)} \in \mathcal{P}_{(1-\gamma)^k \epsilon}$ . However, part (b) of Theorem 4 holds as is.

### 6.2.3 How Close Is $v_\epsilon$ to $v$ ?

Restricting the row player to play  $\epsilon$ -diversified strategies can of course increase its minimax loss, i.e.,  $v_\epsilon \geq v$ . In fact, it is not hard to give examples of games where the gap is quite large. For example, suppose the row player has one action that always incurs loss 0, and the remaining  $n - 1$  actions always incur loss 1 (whatever the column player does). Then  $v = 0$  but for  $\epsilon \in [\frac{1}{n}, 1]$ ,  $v_\epsilon = 1 - \frac{1}{\epsilon n}$ .

However, we show here that for *random* matrices  $M$ , the gap between the two is quite small. I.e., the additional loss incurred due to requiring diversification is low. A related result, in a somewhat different model, appears in [124].

**Theorem 5.** *Consider a random  $n \times n$  game  $M$  where each entry  $M(i, j)$  is drawn i.i.d. from some distribution  $D$  over  $[0, 1]$ . With probability  $\geq 1 - \frac{1}{n}$ , for any  $\epsilon \leq 1$ , we have  $v_\epsilon - v = O\left(\sqrt{\frac{\log n}{n}}\right)$ .*

*Proof.* Let  $\mu = \mathbf{E}_{x \sim D}[x]$  be the mean of distribution  $D$ . We will show that  $v$  and  $v_\epsilon$  are both close to  $\mu$ . To argue this, we will examine the value of the uniform distribution  $P_{\text{unif}}$  for the row player, and the value of the uniform distribution  $Q_{\text{unif}}$  for the column player. In particular, notice that  $v \geq \min_i M(i, Q_{\text{unif}})$  because  $Q_{\text{unif}}$  is just one possible strategy for the column player, and by definition,  $v = \min_i M(i, Q^*)$  where  $Q^*$  is the minimax optimal strategy for the column player, and the row player's loss under  $Q^*$  is greater than or equal to the row player's loss under  $Q_{\text{unif}}$  since the column player is trying to maximize the row player's loss. Similarly,  $v_\epsilon \leq \max_j M(P_{\text{unif}}, j)$  since  $P_{\text{unif}}$  is just one possible diversified strategy for the row player, and by definition  $v_\epsilon = \max_j M(P^*, j)$  where  $P^*$  is the minimax optimal diversified strategy for the row player and the row player is trying to minimize loss. So, we have

$$\min_i M(i, Q_{\text{unif}}) \leq v \leq v_\epsilon \leq \max_j M(P_{\text{unif}}, j).$$

Thus, if we can show that with high probability  $\min_i M(i, Q_{\text{unif}})$  and  $\max_j M(P_{\text{unif}}, j)$  are both close to  $\mu$ , then this will imply that  $v$  and  $v_\epsilon$  are close to each other.

Let us begin with  $P_{\text{unif}}$ . Notice that  $M(P_{\text{unif}}, j)$  is just the average of the entries in the  $j$ th column. So, by Hoeffding bounds, there exists a constant  $c$  such that for any given column  $j$ ,

$$\Pr \left[ M(P_{\text{unif}}, j) > \mu + c\sqrt{\frac{\log n}{n}} \right] \leq \frac{1}{2n^2},$$

where the probability is over the random draw of  $M$ . By the union bound, with probability at least  $1 - \frac{1}{2n}$ , this inequality holds simultaneously for all columns  $j$ . Since  $P_{\text{unif}}$  is  $\epsilon$ -diversified, as noted above this implies that  $v_\epsilon \leq \mu + c\sqrt{(\log n)/n}$  with probability at least  $1 - \frac{1}{2n}$ .

On the other hand, by the same reasoning, with probability at least  $1 - \frac{1}{2n}$  the uniform distribution  $Q_{\text{unif}}$  for the column player has the property that for all rows  $i$ ,  $M(i, Q_{\text{unif}}) \geq \mu - c\sqrt{\frac{\log n}{n}}$ . This implies as noted above that  $v \geq \mu - c\sqrt{\frac{\log n}{n}}$ . Therefore, with probability at least  $1 - \frac{1}{n}$ ,  $v_\epsilon - v \leq 2c\sqrt{\frac{\log n}{n}}$  as desired.  $\square$

### 6.3 General-Sum Games

We now consider  $k$ -player general-sum games. Instead of minimax optimality, the natural solution concept now is a Nash equilibrium. We begin by showing that unlike zero-sum games, it is now possible for the payoff of a player at equilibrium to actually be improved by requiring it to play a diversified strategy. This is a bit peculiar because constraining a player is actually helping it.

We then consider the relationship between the social cost at equilibrium and the optimal social cost, when all players are required to use diversified strategies. We call the ratio of these two quantities the *diversified price of anarchy* of the game, in analogy to the usual *price of anarchy* notion when there is no diversification constraint. We show that in some natural games, even requiring a small amount of diversification can significantly improve the price of anarchy of the game, though there also exist games where diversification can make the price of anarchy worse. Finally, we bring several threads of this investigation together by showing that for the class of *smooth games* defined by Roughgarden [101], for any diversification parameter  $\epsilon \in [\frac{1}{n}, 1]$ , the  $\epsilon$ -diversified price of anarchy is no worse than the smoothness of the game, and moreover that players using diversified regret-minimizing strategies (such as those in Sections 6.2.1 and 6.2.2) will indeed approach this bound.

#### 6.3.1 The Benefits of Diversification

First, let us formally define the notion of a *Nash equilibrium subject to a (convex) constraint*  $\mathcal{C}$ , where  $\mathcal{C}$  could be a constraint such as “the row player must use an  $\epsilon$ -diversified strategy”.

**Definition 2.** A set of mixed strategies  $(P_1, \dots, P_k)$  is a Nash equilibrium subject to constraint  $\mathcal{C}$  if no player can unilaterally deviate to improve its payoff without violating constraint  $\mathcal{C}$ . We will just call this a Nash equilibrium when  $\mathcal{C}$  is clear from context.

We now consider the case of  $k = 2$  players, and examine how requiring the row player to diversify can affect its payoff at equilibrium. For zero-sum games, the value  $v_\epsilon$  was always

no better than the minimax value  $v$  of the game, since constraining the row player can never help it. We show here that this is not the case for general-sum games: requiring a player to use a diversified strategy can in some games *improve* its payoff at equilibrium.

**Theorem 6.** *There exist 2-player general-sum games for which a diversification constraint on the row player lowers the row player's payoff at equilibrium, and games for which such a constraint increases the row player's payoff at equilibrium.*

*Proof.* Consider the following two bimatrix games (entries here represent payoffs rather than losses):

Game $A$ :	2, 2	1, 1
	1, 1	0, 0

Game $B$ :	1, 1	3, 0
	0, 0	1, 3

In Game  $A$ , the unique Nash equilibrium has payoff of 2 to each player, and requiring the row player to be diversified strictly lowers both player's payoffs. On the other hand, diversification helps the row player in Game  $B$ . Without a diversification constraint, in Game  $B$  the row player will play the top row and the column player will therefore play the left column, giving both players a payoff of 1. However, requiring the row-player to put probability  $\frac{1}{2}$  on each row will cause the column player to choose the right column, giving the row player a payoff of 2 and the column player a payoff of 1.5.  $\square$

Routing games [125] are an interesting class of many-player games where requiring all players to diversify can actually improve the quality of the equilibrium for everyone. An example is Braess' paradox [126] shown in Figure 6.2. In this example,  $k$  players need to travel from  $s$  to  $t$  and wish to take the cheapest route. Edge costs are given in the figure, where  $k_e$  is the number of players using edge  $e$ . At Nash equilibrium, all players choose the route  $s$ - $a$ - $b$ - $t$  and incur a cost of 2. However, if they must put equal probability on the three routes they can choose from, the expected cost of each player approaches only  $\frac{1}{3}(\frac{2}{3} + 1) + \frac{1}{3}(\frac{2}{3} + \frac{2}{3}) + \frac{1}{3}(1 + \frac{2}{3}) = 1 + \frac{5}{9}$ . Thus, even though from an individual player's perspective, diversification is a restriction that increases robustness at the expense of

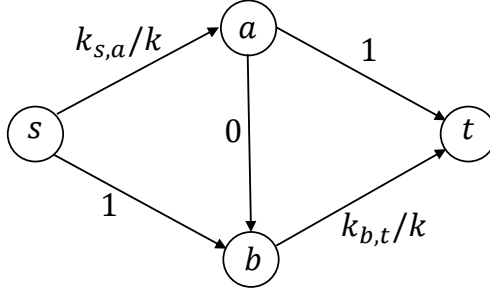


Figure 6.2: Braess' paradox. Here,  $k$  players wish to travel from  $s$  to  $t$ , and requiring all players to use diversified strategies improves the quality of the equilibrium for everyone.

higher average loss, overall, diversification can actually improve the quality of the resulting equilibrium state. In the next section, we discuss the social cost of diversified equilibria in many-player games in more detail, analyzing what we call the *diversified price of anarchy* as well as the social cost that results from all players using diversified regret-minimizing strategies.

### 6.3.2 The Diversified Price of Anarchy

We now consider structured general-sum games with  $k \geq 2$  players. In these games, each player  $i$  chooses some strategy  $s_i$  from a strategy space  $\mathcal{S}_i$ . The combined choice of the players  $\mathbf{s} = (s_1, \dots, s_k)$ , which we call the *outcome*, determines the cost that each player incurs. Specifically, let  $\text{cost}_i(\mathbf{s})$  denote the cost incurred by player  $i$  under outcome  $\mathbf{s}$ , and let  $\text{cost}(\mathbf{s}) = \sum_{i=1}^k \text{cost}_i(\mathbf{s})$  denote the overall social cost of  $\mathbf{s}$ . Let  $\mathbf{s}^* = \text{argmin}_{\mathbf{s}} \text{cost}(\mathbf{s})$ , i.e., the outcome of optimum social cost. The *price of anarchy* of a game is defined as the maximum ratio  $\text{cost}(\mathbf{s})/\text{cost}(\mathbf{s}^*)$  over all Nash equilibria  $\mathbf{s}$ . A low price of anarchy in a game means that all Nash equilibria have social cost that is not too much worse than the optimum. We can analogously define the  $\epsilon$ -diversified price of anarchy:

**Definition 3.** Let  $\mathbf{s}_\epsilon^*$  denote the outcome of optimum social cost subject to each player choosing an  $\epsilon$ -diversified strategy. The  $\epsilon$ -diversified price of anarchy is the maximum ratio  $\text{cost}(\mathbf{s}_\epsilon)/\text{cost}(\mathbf{s}_\epsilon^*)$  over all outcomes  $\mathbf{s}_\epsilon$  that are Nash equilibria subject to all players playing

$\epsilon$ -diversified strategies.

Note that for any game, the 1-diversified price of anarchy equals 1, because players are all required to play the uniform distribution. This suggests that as we increase  $\epsilon$ , the  $\epsilon$ -diversified price of anarchy should drop, though as we show, in some games it is not monotone.

**Examples.** In *consensus games*, each player  $i$  is a distinct node in a  $k$ -node graph  $G$ . Players each choose one of two colors, red or blue, and the cost of player  $i$  is the number of neighbors it has of color different from its own. The social cost is the sum of the players' costs, and to keep ratios finite we add 1 to the total. The optimal  $s^*$  is either “all blue” or “all red” in which each player has a cost of 0, so the social cost is 1. However, if the graph is a complete graph minus a matching, then there exists an equilibrium in which half of the players choose red and half the players choose blue. Each player has  $\frac{k}{2} - 1$  red neighbors and  $\frac{k}{2} - 1$  blue neighbors, so the social cost of this equilibrium is  $\Theta(k^2)$ . This means the price of anarchy is  $\Theta(k^2)$ . However, if we require players to play  $\epsilon$ -diversified strategies for any constant  $\epsilon > \frac{1}{2}$  (i.e., they cannot play pure strategies), then for any  $m$ -edge graph  $G$ , even the optimum outcome has cost  $\Omega(m)$  since every edge has a constant probability of contributing to the cost. So the diversified price of anarchy is  $O(1)$ .

As another example, consider *atomic congestion games* [127]. Here, we have a set  $R$  of *resources* (e.g., edges in a graph  $G$ ) and each player  $i$  has a strategy set  $\mathcal{S}_i \subseteq 2^R$  (e.g., all ways to select a path between two specified vertices in  $G$ ). The cost incurred by a player is the sum of the costs of the resources it uses (the cost of its path). Each resource  $j$  has a cost function  $c_j(k_j)$  where  $k_j$  is the number of players who are using resource  $j$ . The cost functions  $c_j$  could be increasing, such as in packet routing where latency increases with the number of users of an edge, or decreasing, such as players splitting the cost of a shared printer. When examining diversified strategies, we sometimes view players as making fractional choices, such as sending half their packets down one path and half of them down another. The quantity  $k_j$  then denotes the total fractional usage of resource  $j$  (or

equivalently, the expected number of users of that resource).

**Non-monotonicity.** An example of an atomic congestion game where some diversification can initially increase the price of anarchy is the following. Suppose there are four resources, and each player just needs to choose one of them. The costs of the resources behave as follows:

$$c_1(k_1) = 1, \quad c_2(k_2) = 5, \quad c_3(k_3) = 6/k_3, \quad c_4(k_4) = 6/k_4.$$

Assume the total number of players  $k$  is at least 13. The optimal outcome  $\mathbf{s}^*$  is for all players to choose resource 3 (or all choose resource 4) for a total social cost of 6. The optimal  $\epsilon$ -diversified outcome for  $\epsilon = \frac{1}{2}$  (i.e., each player can put weight at most  $\frac{1}{2}$  on any given resource) is for all players to put half their weight on strategy 3 and half their weight on strategy 4, for a total cost of 12. The worst Nash equilibrium is for all players to choose strategy 1, for a total cost of  $k$ , giving a price of anarchy of  $k/6$ . However if we require players to be  $\epsilon$ -diversified for  $\epsilon = \frac{1}{2}$ , there is now a worse equilibrium where each player puts half its weight on strategy 1 and half its weight on strategy 2, for a total cost of  $3k$  and a diversified price of anarchy of  $3k/12 = k/4$ . So, increasing  $\epsilon$  from  $\frac{1}{4}$  up to  $\frac{1}{2}$  increases the price of anarchy, and then increasing  $\epsilon$  further to 1 will then decrease the price of anarchy to 1.

### *General Bounds*

We now present a general bound on the diversified price of anarchy for games, as well as for the social welfare when all players use diversified regret-minimizing strategies such as given in Sections 6.2.1 and 6.2.2, using the smoothness framework of Roughgarden [101].

**Definition 4.** [101] A general-sum game is  $(\lambda, \mu)$ -smooth if for any two outcomes  $\mathbf{s}$  and  $\mathbf{s}^*$ ,

$$\sum_{i=1}^k \text{cost}_i(s_i^*, \mathbf{s}_{-i}) \leq \lambda \text{cost}(\mathbf{s}^*) + \mu \text{cost}(\mathbf{s}).$$

Here,  $(s_i^*, \mathbf{s}_{-i})$  means the outcome in which player  $i$  plays its action in  $\mathbf{s}^*$  but all other players play their action in  $\mathbf{s}$ .

**Theorem 7.** *If a game is  $(\lambda, \mu)$ -smooth, then for any  $\epsilon$ , the  $\epsilon$ -diversified price of anarchy is at most  $\frac{\lambda}{1-\mu}$ .*

*Proof.* Let  $\mathbf{s} = \mathbf{s}_\epsilon$  be some Nash equilibrium subject to all players playing  $\epsilon$ -diversified strategies, and let  $\mathbf{s}^* = \mathbf{s}_\epsilon^*$  be an outcome of optimum social cost subject to all players choosing  $\epsilon$ -diversified strategies. Since  $\mathbf{s}$  is an equilibrium, no player wishes to deviate to their action in  $\mathbf{s}^*$ ; here we are using the fact that  $\mathbf{s}^*$  includes only  $\epsilon$ -diversified strategies, so such a deviation would be legal. Therefore  $\text{cost}(\mathbf{s}) \leq \sum_{i=1}^k \text{cost}_i(s_i^*, \mathbf{s}_{-i}) \leq \lambda \text{cost}(\mathbf{s}^*) + \mu \text{cost}(\mathbf{s})$ . Rearranging, we have  $(1 - \mu)\text{cost}(\mathbf{s}) \leq \lambda \text{cost}(\mathbf{s}^*)$ , so  $\text{cost}(\mathbf{s})/\text{cost}(\mathbf{s}^*) \leq \frac{\lambda}{1-\mu}$ .  $\square$

Roughgarden [101] shows that atomic congestion games with affine cost functions, i.e., cost functions of the form  $c_j(k_j) = a_j k_j + b_j$ , are  $(\frac{5}{3}, \frac{1}{3})$ -smooth. So, their  $\epsilon$ -diversified price of anarchy is at most 2.5. We now adapt the proof in [101] to show that players with vanishing regret with respect to diversified strategies will also approach the bound of Theorem 7.

**Theorem 8.** *Suppose that in repeated play of a  $(\lambda, \mu)$ -smooth game, each player  $i$  uses a sequence of mixed strategies  $s_i^{(1)}, \dots, s_i^{(T)}$  such that for any  $\epsilon$ -diversified strategy  $s_i^*$  we have:*

$$\frac{1}{T} \sum_{t=1}^T \text{cost}_i(\mathbf{s}^{(t)}) \leq \frac{1}{T} \sum_{t=1}^T \text{cost}_i(s_i^*, \mathbf{s}_{-i}^{(t)}) + \Delta_T.$$

*Then the average social cost over the  $T$  steps satisfies*

$$\frac{1}{T} \sum_{t=1}^T \text{cost}(\mathbf{s}^{(t)}) \leq \frac{\lambda}{1-\mu} \text{cost}(\mathbf{s}^*) + \frac{k\Delta_T}{1-\mu}.$$

*In particular, if  $\Delta_T \rightarrow 0$  then the average social cost approaches the bound of Theorem 7.*

*Proof.* Combining the assumption of the theorem, the definition of social cost, and the smoothness definition we have:

$$\begin{aligned}
\frac{1}{T} \sum_{t=1}^T \text{cost}(\mathbf{s}^{(t)}) &= \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^k \text{cost}_i(\mathbf{s}^{(t)}) \\
&\quad \text{(definition of social cost)} \\
&\leq \sum_{i=1}^k \left[ \frac{1}{T} \sum_{t=1}^T \text{cost}_i(s_i^*, \mathbf{s}_{-i}^{(t)}) + \Delta_T \right] \\
&\quad \text{(assumption of theorem)} \\
&= \frac{1}{T} \sum_{t=1}^T \left[ \sum_{i=1}^k \text{cost}_i(s_i^*, \mathbf{s}_{-i}^{(t)}) \right] + k\Delta_T \\
&\quad \text{(rearranging)} \\
&\leq \frac{1}{T} \sum_{t=1}^T [\lambda \text{cost}(\mathbf{s}^*) + \mu \text{cost}(\mathbf{s}^{(t)})] + k\Delta_T. \\
&\quad \text{(applying smoothness)}
\end{aligned}$$

Rearranging, we have:

$$(1 - \mu) \frac{1}{T} \sum_{t=1}^T \text{cost}(\mathbf{s}^{(t)}) \leq \lambda \text{cost}(\mathbf{s}^*) + k\Delta_T,$$

which immediately yields the result of the theorem.  $\square$

### 6.3.3 The Cost of Diversification

We now complement the above results by considering how much worse  $\text{cost}(\mathbf{s}_\epsilon^*)$  can be compared to  $\text{cost}(\mathbf{s}^*)$  in natural games. We focus here on *unit-demand* congestion games where each strategy set  $\mathcal{S}_i \subseteq R$ ; that is, each player  $i$  selects a single resource in  $\mathcal{S}_i$ . In particular, we focus on two important special cases: (a)  $c_j(k_j) = 1/k_j \ \forall j$  (players share the cost of their resource equally with all others who make the same choice; this can be viewed as a game-theoretic distributed hitting-set problem), and (b)  $c_j(k_j) = k_j \ \forall j$ , i.e., linear congestion games. To avoid unnecessary complication, we assume all  $\mathcal{S}_i$  have the

same size  $n$ , i.e., every player has  $n$  choices. We will also think of the number of choices per player  $n$  as  $O(1)$ , whereas the number of players  $k$  and the total number of resources  $R$  may be large.

Unfortunately, in both cases (a) and (b), the cost of diversification can be very high in the worst case. For case (a) (cost sharing), a bad scenario is if there is a single element  $j^*$  such that  $\mathcal{S}_i \cap \mathcal{S}_{i'} = j^*$  for all pairs  $i \neq i'$ . Here,  $\text{cost}(s^*) = 1$  since all players can choose  $j^*$ , but for any  $\epsilon \in [\frac{2}{n}, 1]$ , we have  $E[\text{cost}(s_\epsilon^*)] = \Omega(k)$ , since even in the best solution each player has a 50% chance of choosing a resource that no other player chose. For case (b) (linear congestion), a bad scenario is if there are  $n - 1$  elements  $j_1^*, \dots, j_{n-1}^*$  such that  $\mathcal{S}_i \cap \mathcal{S}_{i'} = \{j_1^*, \dots, j_{n-1}^*\}$  for all pairs  $i \neq i'$ . Here,  $\text{cost}(s^*) = k$  since each player can choose a distinct resource, but for any  $\epsilon \in [\frac{2}{n}, 1]$ , we have  $E[\text{cost}(s_\epsilon^*)] = \Omega(k^2/(n - 1))$ , which is  $\Omega(k^2)$  for  $n = O(1)$ . So, in both cases, the ratio  $\text{cost}(s_\epsilon^*)/\text{cost}(s^*) = \Omega(k)$ .

However, in the *average case* (each  $\mathcal{S}_i$  consists of  $n$  random elements from  $R$ ) the cost of diversification is only  $O(1)$ .

**Theorem 9.** *For both (a) unit-demand cost-sharing and (b) unit-demand linear congestion games, with  $n = O(1)$  strategies per player and random strategy sets  $\mathcal{S}_i$ ,  $E[\text{cost}(s_\epsilon^*)] = O(E[\text{cost}(s^*)])$ .*

*Proof.* Let us first consider (a) unit-demand cost-sharing. One lower bound on  $\text{cost}(s^*)$  is that it is at least the cardinality of the largest collection of disjoint strategy sets  $\mathcal{S}_i$ ; for  $n = 2$  this is the statement that the smallest vertex cover in a graph is at least the size of the maximum matching. Now consider selecting the random sets  $\mathcal{S}_i$  one at a time. For  $i \leq R/n^2$ , the first  $i$  sets cover at most  $R/n$  resources, so set  $\mathcal{S}_{i+1}$  has at least a constant probability of being disjoint from the first  $i$ . This means that the expected size of the largest collection of disjoint strategy sets is at least  $\Omega(\min\{k, R/n^2\})$ . On the other hand, a trivial upper bound on  $\text{cost}(s_\epsilon^*)$ , even for  $\epsilon = 1$ , is  $\min\{k, R\}$ , since at worst each player takes a separate resource until all resources are used. Thus, for  $n = O(1)$ , we have  $E[\text{cost}(s_\epsilon^*)] = O(E[\text{cost}(s^*)])$ .

Now let us consider (b) unit-demand linear congestion. In this case, a lower bound on  $\text{cost}(s^*)$  is the best-case allocation of all resources equally divided. In this case we have  $k/R$  usage per resource for a total cost of  $R \times (k/R)^2 = k^2/R$ . Another lower bound is simply  $k$ , so we have  $\text{cost}(s^*) \geq \max\{k, k^2/R\}$ . On the other hand, we can notice that  $s_\epsilon^*$  for a fully-diversified  $\epsilon = 1$  and random sets  $\mathcal{S}_i$  is equivalent to players choosing resources independently at random. In this case, the social cost is identical to the analysis of random hashing:  $E[\text{cost}(s_1^*)] = E[\sum_j k_j^2] = k + k(k-1)/R$ . Thus,  $E[\text{cost}(s_\epsilon^*)] = O(E[\text{cost}(s^*)])$  as desired.  $\square$

## 6.4 Distributed Setting

We now consider a distributed setting where the actions of the row player are partitioned among  $k$  entities, such as subdivisions within a company or machines in a distributed system. At each time step, the row player asks for a number of actions from each entity, and plays a mixed strategy over them. However, asking for actions requires communication, which we would like to minimize.

Our aim is to obtain results similar to Theorem 3 with low communication complexity, measured by the number of actions requested and any additional constant-sized words communicated. Let  $d \leq \log m$  denote the VC-dimension or pseudo-dimension of the set of columns  $H$ , viewing each row as an example and each column as a hypothesis. A baseline approach is that the row player samples  $O(\frac{d}{\epsilon^2} \log \frac{1}{\epsilon})$  times, from the uniform multinomial distribution over  $\{1, \dots, k\}$  and asks each entity to send the corresponding number of actions to the row player. The row player can then use Algorithm 2 as in the centralized setting over the sampled actions, and will lose only an additional  $\epsilon$  in the value of its strategy. The communication complexity of this method is  $O(\frac{d}{\epsilon^2} \log \frac{1}{\epsilon})$  actions plus  $O(k)$  additional words. Here, we provide an algorithm that reduces communication to  $O(d \log(1/\epsilon))$ . The idea is to show that in Algorithm 2, each iteration of the multiplicative weight update can be simulated in the distributed setting with  $O(d)$  communication. Then, since there are at most

$O(\log(1/\epsilon))$  iterations, the desired result follows. More specifically, we show that in each iteration, we can do the following two actions communication efficiently:

1. For any distribution  $P$  over the rows partitioned across  $k$  entities, obtain a column  $j$  such that  $M(P, j) \geq v_\epsilon$ .
2. Update the distribution using the received column  $j$ .

To achieve the first statement, assume there is a centralized oracle, which for any  $\epsilon$ -diversified distribution  $P$  returns a column  $j$  such that  $M(P, j) \geq v_\epsilon$ . For any distribution  $P$  partitioned across  $k$  entities, each agent first sends its sum of weights to the row player. Then, the row player samples  $O(\frac{d}{(1-\alpha)^2 v_\epsilon^2})$  actions ( $0 < \alpha < 1$ ) across the  $k$  agents proportional to their sum of weights, where  $d$  is the VC-dimension of  $H$ . By the standard VC-theory, a mixed strategy  $P'$  of choosing a uniform distribution over the sampled actions is a  $(1 - \alpha)v_\epsilon$ -approximation for  $H$ , i.e.  $M(P', j) \geq M(P, j) - (1 - \alpha)v_\epsilon \geq \alpha v_\epsilon$  for all column  $j \in H$ . The communication complexity of this step is  $O(\frac{d}{(1-\alpha)^2 v_\epsilon^2})$  actions plus  $O(k)$  additional words. For (2), we show steps 3 and 4 in Algorithm 2 can be simulated with low communication. Step 3 is easy: just send column  $j$  to all entities, and each entity then updates its own weights. What is left is to show that the projection step in Algorithm 2 can be simulated in the distributed setting. Fortunately, this projection step has been studied before in the distributed machine learning literature [108], where an efficient algorithm with  $O(k \log^2(d/\epsilon))$  words of communication is proposed. We summarize our results for the distributed setting with the following theorem.

**Theorem 10.** *Given a centralized oracle, which for any  $\epsilon$ -diversified distribution  $P$  returns a column  $j$  such that  $M(P, j) \geq v_\epsilon$ . If the actions of the row players are distributed across  $k$  entities, there is an algorithm that constructs a mixed strategy  $Q$  such that for all but an  $\epsilon$  fraction of the rows  $i$ ,  $M(i, Q) \geq \alpha v_\epsilon - \gamma$ ,  $0 < \alpha < 1$ . The algorithm requests at most  $O(\frac{\log(1/\epsilon)}{\gamma^2(1+\gamma-\alpha v_\epsilon)} \cdot \frac{d}{(1-\alpha)^2 v_\epsilon^2})$  actions and uses an additional  $O(\frac{\log(1/\epsilon)}{\gamma^2(1+\gamma-\alpha v_\epsilon)} \cdot k \log^2(d/\epsilon))$  words of communication.*

## 6.5 Experiments

To better understand the benefit of diversified strategies, we give some empirical simulations for both two-player zero-sum games and general-sum games. For all the experiments, we fix  $\gamma = 0.2$  and show the results of using different values of  $\epsilon$ .

**Two-player zero-sum games.** The row player has  $n = 10$  actions to choose from, where each round, each action  $a_i$  returns a uniformly random reward  $r_i \in [i/n, 1]$ . The game is played for  $T = 10,000$  rounds. Note that the  $n$ -th action has the highest expected reward.

We consider two scenarios in which a rare but catastrophic event occurs. The first scenario is that at time  $T$ , the cumulative reward gained from choosing the  $n$ -th action becomes zero. The second scenario is that the  $n$ -th action incurs a large negative reward of  $-T$  in time step  $T$ . Both of these can be viewed as different ways of simulating a bad event where, for instance, the shares of a company become worthless when the company goes bankrupt.

The results for both scenarios, averaged over 10 independent trials, are shown in Figure 6.4. One can see that as expected, in the normal situation, the diversified strategy gains less reward. However, when the rare event happens, the non-diversified strategy gains very low reward. In both cases, a modest value of  $\epsilon = 0.4$  achieves a high reward whether the bad event happens or not.

**General-sum games.** We play the routing game defined in Braess' paradox (see Figure 6.2). Each player has three routes to choose from ( $s-a-b-t$ ,  $s-a-t$ , and  $s-b-t$ ) in each round, so  $\epsilon \in [1/3, 1]$ . As analyzed in Section 6.3.1, without the diversified constraint (i.e.,  $\epsilon = 1/3$ ), the game quickly converges to the Nash equilibrium where all players choose the route  $s-a-b-t$  and incur a loss of 2. The best strategy in this case is to play the 1-diversified strategy, which incur a lower loss of about 1.55. See Figure 6.3 for the results using other  $\epsilon$  values.

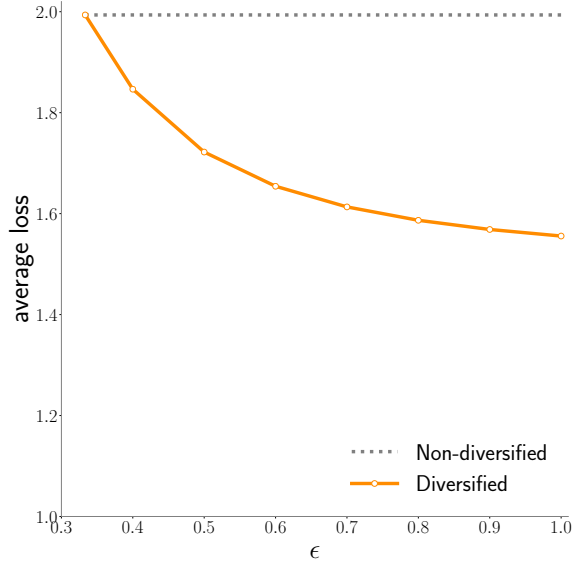
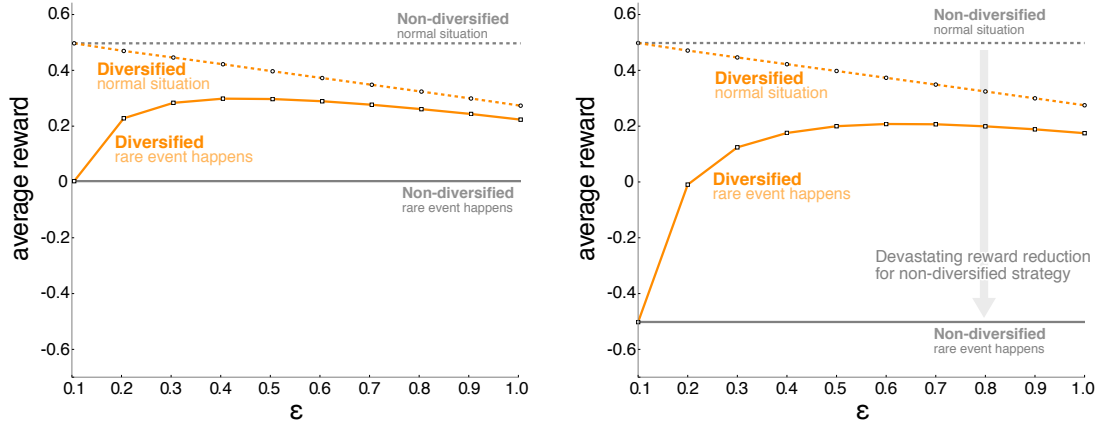


Figure 6.3: Simulated results of Braess' paradox after  $T = 10,000$  rounds. A more diversified strategy leads to lower loss.



(a) Rare event removes all the reward gained from the  $n$ -th action.

(b) Rare event changes the reward of the  $n$ -th action to  $-T$  in the last round.

Figure 6.4: Average reward over  $T = 10,000$  rounds with different values of  $\epsilon$ . When the rare event happens, the non-diversified strategy gains very low (even negative) reward.

## 6.6 Conclusion

We consider games in which one wants to play well without choosing a mixed strategy that is too concentrated. We show that such a diversification restriction has a number of benefits, and give adaptive algorithms to find diversified strategies that are near-optimal, also showing how taxes or fines can be used to keep a standard algorithm diversified. Further, our algorithms are simple and efficient, and can be implemented in a distributed setting. We also analyze properties of diversified strategies in both zero-sum and general-sum games, and give general bounds on the diversified price of anarchy as well as the social cost achieved by diversified regret-minimizing players.

## CHAPTER 7

### COMMUNICATION EFFICIENT DISTRIBUTED AGNOSTIC BOOSTING

In this chapter, we show how the online decision-making framework introduced in the previous chapter can also help train a robust and scalable machine learning model. Specifically, we consider the problem of learning from distributed data in the agnostic setting, i.e., in the presence of arbitrary forms of noise. Our main contribution is a general distributed boosting-based procedure for learning an arbitrary concept space, that is simultaneously noise tolerant, communication efficient, and computationally efficient. This improves significantly over prior works that were either communication efficient only in noise-free scenarios or computationally prohibitive. Empirical results on large synthetic and real-world datasets demonstrate the effectiveness and scalability of the proposed approach.

#### 7.1 Introduction

Distributed machine learning has received an increasing amount of attention in this “big data” era [128]. The most common use case of distributed learning is when the data cannot fit into a single machine, or when one wants to speed up the training process by utilizing parallel computation of multiple machines [129, 130, 131]. In these cases, one can usually freely distribute the data across entities, and an evenly distributed partition would be a natural choice.

In this work, we consider a different setting where the data is inherently distributed across different locations or entities. Examples of this scenario include scientific data gathered by different teams, or customer information of a multinational corporation obtained in different countries. The goal is to design an efficient learning algorithm with a low generalization error over the union of the data. Note that the distribution of the data from each source may be very different. Therefore, to deal with the worst-case situation, we

assume the data can be adversarially partitioned. This scenario has been studied for different tasks, such as supervised learning [118, 132, 117], unsupervised learning [133, 134], and optimization [135, 136].

Traditional machine learning algorithms often only care about sample complexity and computational complexity. However, since the bottleneck in the distributed setting is often the communication between machines [118], the theoretical analysis in this work will focus on communication complexity. A baseline approach in this setting would be to uniformly sample examples from each entity and perform centralized learning at the center. By the standard VC-theory, a sampling set of size  $O(\frac{d}{\epsilon^2} \log \frac{1}{\epsilon})$  is sufficient. The communication complexity of this approach is thus  $O(\frac{d}{\epsilon^2} \log \frac{1}{\epsilon})$  examples.

More advanced algorithms with better communication complexities have been proposed in recent works [118, 117]. For example, [118] proposes a generic distributed boosting algorithm that achieves communication with only logarithmic dependence on  $1/\epsilon$  for any concept class. Unfortunately, their method only works in the standard realizable PAC-learning setting, where the data can be perfectly classified by a function in the hypothesis set and is noiseless. This is because many boosting algorithms are vulnerable to noise [137, 138]. The realizable case is often unrealistic in real-world problems. Therefore, we consider the more general agnostic learning setting [139], where there is no assumption on the target function. Since it is impossible to achieve an arbitrary error rate  $\epsilon$ , the goal in this setting is to find a hypothesis with error rate close to  $\text{opt}(H)$ , the minimum error rate achievable within the hypothesis set  $H$ . The error bound is often in the form of  $O(\text{opt}(H)) + \epsilon$ . Balcan et al. [118] propose an algorithm based on the robust generalized halving algorithm with communication complexity of  $\tilde{O}(k \log(|H|) \log(1/\epsilon))$  examples. However, the algorithm works only for a finite hypothesis set  $H$  and is computationally inefficient.

We propose a new distributed boosting algorithm that works in the agnostic learning setting. While our algorithm can handle this much more difficult and more realistic scenario, it enjoys the same communication complexity as in [118] that is logarithmic in  $1/\epsilon$  and

exponentially better than the natural baselines. The algorithm is computationally efficient and works for any concept class with a finite VC-dimension. The key insight, inspired by [118], is that a constant (independent of  $\epsilon$ ) number of examples suffice to learn a weak hypothesis, and thus if the boosting algorithm only needs  $O(\log \frac{1}{\epsilon})$  iterations, we obtain the desired result.

A key challenge in this approach is that most agnostic boosting algorithms either have poor error bound guarantees or require too many iterations. The first agnostic boosting algorithm was proposed in [140]. Although the number of iterations is  $O(\log \frac{1}{\epsilon})$  and is asymptotically optimal, their bound on the final error rate is much weaker: instead of  $O(\text{opt}(H)) + \epsilon$ , the bound is  $O(\text{opt}(H)^{c(\beta)} + \epsilon$ , where  $c(\beta) = 2(1/2 - \beta)^2 / \ln(1/\beta - 1)$ . Some subsequent works [141, 111] significantly improve the bound on the error rate. However, their algorithms all require  $O(1/\epsilon^2)$  iterations, which can in turn result in  $O(1/\epsilon^2)$  communication in the distributed setting. Fortunately, we identify a very special boosting algorithm [142] that runs in  $O(\log \frac{1}{\epsilon})$  iterations. This algorithm was analyzed in the realizable case in the original paper, but has later been noted to be able to work in the agnostic setting [143]. We show how to adapt it to the distributed setting and obtain a communication efficient distributed learning algorithm with good agnostic learning error bound. Our main contributions are summarized as follows.

- We identify a centralized agnostic boosting algorithm and show that it can be elegantly adapted to the distributed setting. This results in the first algorithm that is both computationally efficient *and* communication efficient to learn a general concept class in the distributed agnostic learning setting.
- Our proposed algorithm, which is a boosting-based approach, is flexible in that it can be used with various weak learners. Furthermore, the weak learner only needs to work in the traditional centralized setting rather than in the more challenging distributed setting. This makes it much easier to design new algorithms for different concept classes in the distributed setting.

- We confirm our theoretical results by empirically comparing our algorithm to the existing distributed boosting algorithm [118]. It does much better on the synthetic dataset and achieves promising results on real-world datasets as well.

## 7.2 Problem Setup

We first introduce agnostic learning as a special case of the general statistical learning problem. Then, we discuss the extension of the problem to the distributed setting, where the data is adversarially partitioned.

### 7.2.1 Statistical Learning Problem

In statistical learning, we have access to a sampling oracle according to some probability distribution  $D$  over  $X \times \{-1, 1\}$ . The goal of a learning algorithm is to output a hypothesis  $h$  with a low error rate with respect to  $D$ , defined as  $err_D(h) = E_{(x,y) \sim D}(h(x) \neq y)$ . Often, we compare the error rate to the minimum achievable value within a hypothesis set  $H$ , denoted by  $err_D(H) = \inf_{h' \in H} err_D(h')$ . More precisely, a common error bound is in the following form.

$$err_D(h) \leq c \cdot err_D(H) + \epsilon, \quad (7.1)$$

for some constant  $c \geq 1$  and an arbitrary error parameter  $\epsilon > 0$ .

Many efficient learning algorithms have been proposed for the realizable case, where the target function is in  $H$  and thus  $err_D(H) = 0$ . In this work, we consider the more general case where we do not have any assumption on the value of  $err_D(H)$ . This is often called the agnostic learning setting [139]. Ideally, we want  $c$  in the bound to be as close to one as possible. However, for some hypothesis set  $H$ , achieving such a bound with  $c = 1$  is known to be NP-hard [144].

### 7.2.2 Extension to the Distributed Setting

In this work, we consider the agnostic learning problem in the distributed learning framework proposed by [118]. In this framework, we have  $k$  entities. Each entity  $i \in [k]$  has access to a sampling oracle according to a distribution  $D_i$  over  $X \times \{-1, 1\}$ . There is also a *center* which can communicate with the  $k$  entities and acts as a coordinator. The goal is to learn a good hypothesis with respect to the overall distribution  $D = \frac{1}{k} \sum_{i=1}^k D_i$  without too much communication among entities. It is convenient to calculate the communication by *words*. For example, a  $d$ -dimensional vector counts as  $O(d)$  words.

**Main goal.** The problem we want to solve in this work is to design an algorithm that achieves error bound (7.1) for a general concept class  $H$ . The communication complexity should depend only logarithmically on  $1/\epsilon$ .

### 7.3 Distributed Agnostic Boosting

In this work, we show a distributed boosting algorithm for any concept class with a finite VC-dimension  $d$ . In the realizable PAC setting, the boosting algorithm is assumed to have access to a  $\gamma$ -weak learner that, under any distribution, finds a hypothesis with error rate at most  $1/2 - \gamma$ . This assumption is unrealistic in the agnostic setting since even the best hypothesis in the hypothesis set can perform poorly. Instead, following the setting of [140], the boosting algorithm is assumed to have access to a  $\beta$ -weak agnostic learner defined as follows.

**Definition 5.** A  $\beta$ -weak agnostic learner, given any probability distribution  $D$ , will return a hypothesis  $h$  with error rate

$$err_D(h) \leq err_D(H) + \beta.$$

Detailed discussion of the existence of such weak learners can be found in [140]. Since

error of  $1/2$  can be trivially achieved, in order for the weak learner to convey meaningful information, we assume  $err_D(H) < 1/2 - \beta$ . Some prior works use different definitions. For example, [145] uses the definition of  $(\alpha, \gamma)$ -weak learner. That definition is stronger than ours, since an  $(\alpha, \gamma)$ -weak learner in that paper implies a  $\beta$ -weak learner in our work with  $\beta = \alpha - \gamma$ . Therefore, our results still hold by using their definition. Below we show an efficient agnostic boosting algorithm in the centralized setting.

### 7.3.1 Agnostic Boosting: Centralized Version

The main reason why many boosting algorithms (including AdaBoost [102] and weight-based boosting [146, 147]) fail in the agnostic setting is that they tend to update the example weights aggressively and may end up putting too much weight on noisy examples.

Inspired by the “diversified” distribution introduced in the previous chapter, we consider a smoothed boosting algorithm [142], shown in Algorithm 4. This algorithm uses at most  $O(\log 1/\epsilon)$  iterations and enjoys a nice “smoothness” property, which is shown to be helpful in the agnostic setting [111]. The algorithm was originally analyzed in the realizable case but has later been noted to be able to work in the agnostic setting [143]. Below, for completeness we show the analyses of the algorithm in both the realizable and agnostic settings.

The boosting algorithm adjusts the example weights using the standard multiplicative weight update rule. The main difference is that it performs an additional Bregman projection step of the current example weight distribution into a convex set  $\mathcal{P}$  after each boosting iteration. The Bregman projection is a general projection technique that finds a point in the feasible set with the smallest “distance” to the original point in terms of *Bregman divergence*. Here we use a particular Bregman divergence called *relative entropy*  $RE(p \parallel q) = \sum_i p_i \ln(p_i/q_i)$  for two distributions  $p$  and  $q$ . To ensure that the boosting algorithm always generates a “diversified” distribution, we set the feasible set  $\mathcal{P}$  to be the set of all  $\epsilon$ -diversified distributions as defined in Definition 1.

The complete boosting algorithm is shown in Algorithm 4 and the theoretical guarantee

---

**Algorithm 4** Centralized Smooth Boosting algorithm [142]

---

**Initialization:** Fix a  $\gamma \leq \frac{1}{2}$ . Let  $D^{(1)}$  to be the uniform distribution over the dataset  $S$ .  
**For**  $t = 1, 2, \dots, T$ :

1. Call the weak learner with distribution  $D^{(t)}$  and obtain a hypothesis  $h^{(t)}$
2. Update the example weights

$$\hat{D}^{(t+1)}(i) = D^{(t)}(i) \cdot (1 - \gamma)^{\ell_i^{(t)}} / Z^{(t)}$$

where  $\ell_i^{(t)} = \mathbf{1}[h^{(t)}(x_i) \neq y_i]$  and  $Z^{(t)} = \sum_i D^{(t)}(i) \cdot (1 - \gamma)^{\ell_i^{(t)}}$  is the normalization factor.

3. Project  $\hat{D}^{(t+1)}$  into the feasible set  $\mathcal{P}$  of  $\epsilon$ -diversified distributions

$$D^{(t+1)} = \arg \min_{D \in \mathcal{P}} RE(D \parallel \hat{D}^{(t+1)})$$

**Output:** The hypothesis  $\text{sign} \left( \frac{1}{T} \sum_{t=1}^T h^{(t)} \right)$

---

in Theorem 11. The proof is similar to the one in [142], except that they use real-valued weak learners, whereas here we only consider binary hypotheses for simplicity.

**Theorem 11.** *Given a sample  $S$  and access to a  $\gamma$ -weak learner, Algorithm 4 makes at most  $T = O\left(\frac{\log(1/\epsilon)}{\gamma^2}\right)$  calls to the weak learner with  $\epsilon$ -diversified distributions and achieves error rate  $\epsilon$  on  $S$ .*

Note that in Theorem 11, it is not explicitly assumed to be in the realizable case. In other words, If we have a  $\gamma$ -weak learner in the agnostic setting, we can achieve the same guarantee. However, in the agnostic setting, we only have access to a  $\beta$ -weak agnostic learner, which is a much weaker and more realistic assumption. The next theorem shows the error bound we get under this usual assumption in the agnostic setting.

**Theorem 12.** *Given a sample  $S$  and access to a  $\beta$ -weak agnostic learner, Algorithm 4 uses at most  $O\left(\frac{\log(1/\epsilon)}{(1/2-\beta)^2}\right)$  iterations and achieves an error rate  $\frac{2\text{err}_S(H)}{1/2-\beta} + \epsilon$  on  $S$ , where  $\text{err}_S(H)$  is the optimal error rate on  $S$  achievable using the hypothesis class  $H$ .*

*Proof.* We show that as long as the boosting algorithm always generates some  $\epsilon'$ -diversified distributions, the  $\beta$ -weak agnostic learner is actually a  $\gamma$ -weak learner for some  $\gamma > 0$ , i.e.,

it achieves error rate  $1/2 - \gamma$  for any  $\epsilon'$ -diversified distributions. In each iteration  $t$ , the  $\beta$ -weak agnostic learner, given  $S$  with distribution  $D^{(t)}$ , returns a hypothesis  $h^{(t)}$  such that

$$\begin{aligned} \text{err}_{D^{(t)}}(h^{(t)}) &\leq \text{err}_{D^{(t)}}(H) + \beta \\ &\leq \frac{1}{\epsilon'} \text{err}_S(H) + \beta. \end{aligned}$$

The second inequality utilizes the  $\epsilon'$ -diversifiedness property of  $D^{(t)}$ . The reason is that if  $h$  is the optimal hypothesis on  $S$ , we have

$$\text{err}_{D^{(t)}}(H) \leq \text{err}_{D^{(t)}}(h) \leq \frac{\text{\#mistakes on } S}{\epsilon'|S|} = \frac{1}{\epsilon'} \text{err}_S(h) = \frac{1}{\epsilon'} \text{err}_S(H).$$

Let  $\frac{1}{\epsilon'} \text{err}_S(H) + \beta = \frac{1}{2} - \gamma$ , or equivalently  $\gamma = (\frac{1}{2} - \beta) - \frac{1}{\epsilon'} \text{err}_S(H)$ . Then, if  $\epsilon' \geq \frac{2\text{err}_S(H)}{1/2 - \beta}$ , we have  $\gamma \geq \frac{1}{2}(1/2 - \beta) > 0$ . Therefore, we can use Theorem 11, and achieves error rate  $\epsilon'$  on  $S$  by using  $O(\frac{\log(1/\epsilon')}{(1/2 - \beta)^2})$  iterations. Alternatively, it achieves error rate  $\frac{2\text{err}_S(H)}{1/2 - \beta} + \epsilon$  by using  $O(\frac{\log(1/\epsilon)}{(1/2 - \beta)^2})$  iterations.  $\square$

Next, we show how to adapt this algorithm to the distributed setting.

### 7.3.2 Agnostic Boosting: Distributed Version

The technique of adapting a boosting algorithm to the distributed setting is inspired by [118]. They claim that any weight-based boosting algorithm can be turned into a distributed boosting algorithm with communication complexity that depends linearly on the number of iterations in the original boosting algorithm. However, their result is not directly applicable to our boosting algorithm due to the additional projection step. We will describe our distributed boosting algorithm by showing how to simulate the three steps in each iteration of Algorithm 4 in the distributed setting with  $O(d)$  words of communication. Then, since there are at most  $O(\log(1/\epsilon))$  iterations, the desired result follows.

In step 1, in order to obtain a  $2\beta$ -weak hypothesis (we use  $2\beta$  instead of  $\beta$  for conve-

nience, which only affects the constant terms), the center calls the  $\beta$ -weak agnostic learner on a dataset sampled from  $D^{(t)} = \frac{1}{k} \sum_{i=1}^k D_i^{(t)}$ . The sampling procedure is as follows. Each entity first sends its sum of weights to the center. Then, the center samples  $O(\frac{d}{\beta^2} \log \frac{1}{\beta})$  examples in total across the  $k$  entities proportional to their sum of weights. By the standard VC-theory, the error rate of any hypothesis on the sample is within  $\beta$  to the true error rate with respect to the underlying distribution, with high probability. It is thus sufficient to find a hypothesis with error within  $\beta$  to the best hypothesis, which can be done thanks to the assumed  $\beta$ -weak learner.

Step 2 is relatively straightforward. The center broadcasts  $h^{(t)}$  and each entity updates its own internal weights independently. Each entity then sends the summation of internal weights to the center for the calculation of the normalization factor. The communication in this step is  $O(kd)$  for sending  $h^{(t)}$  and some numbers. What is left is to show that the projection in step 3 can be done in a communication efficient way. As shown in [123], the projection using relative entropy as the distance into  $\mathcal{P}$ , the set of all  $\epsilon$ -diversified distributions, can be done by the following simple algorithm.

For a fixed index  $m$ , we first clip the largest  $m$  coordinates of  $p$  to  $\frac{1}{\epsilon n}$ , and then rescale the rest of the coordinates to sum up to  $1 - \frac{m}{\epsilon n}$ . We find the least index  $m$  such that the resulting distribution is in  $\mathcal{P}$ , i.e. all the coordinates are at most  $\frac{1}{\epsilon n}$ . A naive algorithm by first sorting the coordinates takes  $O(n \log n)$  time, but it is communicationally inefficient.

Fortunately, [123] also proposes a more advanced algorithm by recursively finding the median. The idea is to use the median as the threshold, which corresponds to a potential index  $m$ , i.e.,  $m$  is the number of coordinates larger than the median. We then use a binary search to find the least index  $m$ . The distributed version of the algorithm is shown in Algorithm 5.

**Theorem 13.** *Algorithm 5 projects a  $n$ -dimensional distribution into the set of all  $\epsilon$ -diversified distributions  $\mathcal{P}$  with  $O(k \log^2(n))$  words of total communication complexity.*

*Proof.* Since Algorithm 5 is a direct adaptation of the centralized projection algorithm in

---

**Algorithm 5** Distributed Bregman projection algorithm
 

---

**Input:**

**Center:**  $n_0 = n$ ;  $C = 0$ ;  $C^w = 0$

**Each entity  $i$ :** a disjoint subset  $\mathcal{W}_i$  of  $\mathcal{W} = \{w_1, \dots, w_n\}$

**While**  $n_0 \neq 0$ :

Distributedly find the median  $\theta$  of  $(\mathcal{W}_1, \dots, \mathcal{W}_k)$

**Each entity  $i$ :**

$$\begin{aligned}\mathcal{L}_i &= \{w : w < \theta, w \in \mathcal{W}_i\}; & L_i^w &= \sum_{w \in \mathcal{L}_i} w \\ \mathcal{M}_i &= \{w : w = \theta, w \in \mathcal{W}_i\}; & M_i^w &= \sum_{w \in \mathcal{M}_i} w \\ \mathcal{H}_i &= \{w : w > \theta, w \in \mathcal{W}_i\}; & H_i^w &= \sum_{w \in \mathcal{H}_i} w\end{aligned}$$

**Center:**

$$\begin{aligned}L &= \sum_i |\mathcal{L}_i|; & L^w &= \sum_i L_i^w \\ M &= \sum_i |\mathcal{M}_i|; & M^w &= \sum_i M_i^w \\ H &= \sum_i |\mathcal{H}_i|; & H^w &= \sum_i H_i^w \\ m_0 &= \frac{1 - (C + H) \frac{1}{\epsilon n}}{1 - (C^w + H^w)} \text{ and broadcasts it}\end{aligned}$$

**if**  $\theta m_0 > \frac{1}{\epsilon n}$  **then**

$$C = C + H + M; \quad C^w = C^w + H^w + M^w$$

**if**  $L = 0$  **then**  $\theta = \max(w : w < \theta, w \in \mathcal{W})$

set  $n_0 = L$

notify each entity  $i$  to set  $\mathcal{W}_i = \mathcal{L}_i$

**else**

set  $n_0 = H$

notify each entity  $i$  to set  $\mathcal{W}_i = \mathcal{H}_i$

**Center:**  $m_0 = \frac{1 - C \frac{1}{\epsilon n}}{1 - C^w}$  and broadcasts it

**Each entity  $i$ :** set each coordinate as

$$w'_i = \begin{cases} \frac{1}{\epsilon n} & \text{if } w_i > \theta \\ w_i m_0 & \text{if } w_i \leq \theta \end{cases}$$


---

[123], we omit the proof of its correctness. Because we use a binary search over possible thresholds, the algorithm runs at most  $O(\log(n))$  iteration. Therefore, it suffices to show that the communication complexity of finding the median is at most  $O(k \log n)$ . This can be done by the iterative procedure shown in Algorithm 6. Each entity first sends its own median to the center. The center identifies the maximum and minimum local medians, denoted as  $\overline{m}$  and  $\underline{m}$ , respectively. The global median must be between  $\overline{m}$  and  $\underline{m}$ , and removing the same number of elements larger than or equal to  $\overline{m}$  and less than  $\underline{m}$  will not change the median. Therefore, the center can notify the two corresponding entities and let them

---

**Algorithm 6** Distributedly finding the median

---

**Input:**

**Each entity**  $i$ : a disjoint subset  $\mathcal{W}_i$  of  $\mathcal{W} = \{w_1, \dots, w_n\}$

**Each entity**  $i$ : Send the median  $m_i$  of  $\mathcal{W}_i$  to the center

**While**  $|\mathcal{W}_i| > 1$  for some  $i \in [k]$ :

**Center:**

Find the maximum and minimum of the  $k$  medians, denoted by  $\overline{m}$  and  $\underline{m}$  and notify the corresponding entities, denoted by  $A$  and  $B$ .

**Entity**  $A$ : Send  $\overline{n} = |i : w_i \in \mathcal{W}_A, w_i \geq \overline{m}|$  to the center

**Entity**  $B$ : Send  $\underline{n} = |i : w_i \in \mathcal{W}_B, w_i < \underline{m}|$  to the center

**Center:** Send  $r = \min\{\overline{n}, \underline{n}\}$  to entity  $A$  and  $B$

**Entity**  $A$ : Remove the largest  $r$  elements in  $\mathcal{W}_A$

**Entity**  $B$ : Remove the smallest  $r$  elements in  $\mathcal{W}_B$

**Entity**  $A$  and  $B$ : Send the new median to the center

---

remove the same number of elements. At least one entity will reduce its size by half, so the algorithm stops after  $O(k \log n)$  iterations. Note that except for the first round, we only need to communicate the updated medians of two entities at each round, so the overall communication complexity is  $O(k \log n)$  words.

In practice, it is often easier and more efficient to use a *quickselect*-based distributed algorithm to find the median. The idea is to randomly select and broadcast a weight at each iteration. This, in expectation, can remove half of the possible median candidates. This approach achieves the same communication complexity in expectation.  $\square$

The complete distributed agnostic boosting algorithm is shown in Algorithm 7. We summarize our theoretical results in the next Theorem.

**Theorem 14.** *Given access to a  $\beta$ -weak agnostic learner, Algorithm 7 achieves error rate  $\frac{2err_D(H)}{1/2-\beta} + \epsilon$  by using at most  $O(\frac{\log(1/\epsilon)}{(1/2-\beta)^2})$  rounds, each involving  $O((d/\beta^2) \log(1/\beta))$  examples and an additional  $O(kd \log^2(\frac{d \log(1/\epsilon)}{(1/2-\beta)\epsilon}))$  words of communication per round.*

*Proof.* The boosting algorithm starts by drawing from  $D$  a sample  $S$  of size  $n = \tilde{O}(\frac{\log(1/\epsilon)d}{(1/2-\beta)^2\epsilon^2})$  across the  $k$  entities without communicating them. If  $S$  is a centralized dataset, then by Theorem 12 we know that Algorithm 4 achieves error rate  $\frac{2err_S(H)}{1/2-\beta} + \frac{\epsilon}{2}$  on  $S$  using  $O(\frac{\log(1/\epsilon)}{(1/2-\beta)^2})$  iterations. We have shown that Algorithm 7 is a correct simulation of Algorithm 4 in

---

**Algorithm 7** Distributed agnostic boosting algorithm

---

**Initialization:**

**Center:** Access to a  $\beta$ -agnostic weak learner. Set  $\gamma = \frac{1}{2}(\frac{1}{2} - \beta)$

**Each entity  $i$ :**

Sample  $S_i$  drawn from  $D_i$  such that  $S = \cup_i S_i$  with size  $n = \tilde{O}(\frac{\log(1/\epsilon)d}{(1/2-\beta)^2\epsilon^2})$

Set weights  $v_{i,x}^{(1)} = 1/|S_i|$  for each  $(x, y) \in S_i$

**For  $t = 1, 2, \dots, T$ :**

**Each entity  $i$ :** Send  $w_i^{(t)} = \sum_{x \in S_i} v_{i,x}^{(t)}$  to the center

**Center:** Let  $W^{(t)} = \sum_i w_i^{(t)}$ . Determine the number of examples  $n_i^{(t)}$  to request from each entity  $i$  by sampling  $O(\frac{d}{\beta^2} \log \frac{1}{\beta})$  times from the multinomial distribution  $w_i^{(t)}/W^{(t)}$ , and then send each number  $n_i^{(t)}$  to entity  $i$ .

**Each entity  $i$ :** sample  $n_i^{(t)}$  times from  $S_i$  proportional to  $v_{i,x}^{(t)}$ ; send them to center

**Center:** run the  $\beta$ -agnostic weak learner on the union of the received  $O(\frac{d}{\beta^2} \log \frac{1}{\beta})$  examples, and then broadcast the returned hypothesis  $h^{(t)}$

**Each entity  $i$ :** update the weight of each example  $(x, y)$

$$\hat{v}_{i,x}^{(t+1)} = \begin{cases} v_{i,x}^{(t)}(1 - \gamma) & \text{if } h^{(t)}(x) = y \\ v_{i,x}^{(t)} & \text{otherwise} \end{cases}$$

Distributedly normalize and then project the weights by Algorithm 5

**Output:** The hypothesis  $\text{sign}\left(\frac{1}{T} \sum_{t=1}^T h^{(t)}\right)$

---

the distributed setting, and thus we achieve the same error bound on  $S$ . The number of communication rounds is the same as the number of iterations of the boosting algorithm. And in each round, the communication includes  $O(d/\beta^2 \log(1/\beta))$  examples for finding the  $\beta$ -weak hypothesis,  $O(kd)$  words for broadcasting the hypothesis and some numbers, and  $O(k \log^2(n))$  words for the distributed Bregman projection.

So far we only have the error bound of  $\frac{2err_S(H)}{1/2-\beta} + \frac{\epsilon}{2}$  on  $S$ . To obtain the generalization error bound, note that with  $n = \tilde{O}(\frac{\log(1/\epsilon)d}{(1/2-\beta)^2\epsilon^2})$  and by the standard VC-dimension argument, we have that with high probability  $err_S(H) \leq err_D(H) + \frac{(1/2-\beta)\epsilon}{8}$ , and the generalization error of our final hypothesis deviates from the empirical error by at most  $\epsilon/4$ , which completes the proof with the desired generalization error bound.  $\square$

## 7.4 Experiments

In this section, we compare the empirical performance of the proposed distributed boosting algorithms with two other algorithms on synthetic and real-world datasets. The first one is distributed AdaBoost [118], which is similar to our algorithm but without the projection step. The second one is the distributed logistic regression algorithm available in the MPI implementation of the Liblinear package [148]. We choose it as a comparison to a non-boosting approach. Note that Liblinear is a highly-optimized package while our implementation is not, so the comparison in terms of speed is not absolutely fair. However, we show that our approach, grounded in a rigorous framework, is comparable to this leading method in practice.

### 7.4.1 Experiment Setup

All three algorithms are implemented in C using MPI, and all the experiments are run on Amazon EC2 with 16 m3.large machines. The data is uniformly partitioned across 16 machines. All the results are averaged over 10 independent trials. Logistic regression is a deterministic algorithm, so we do not show the standard deviation of the error rate. We however still run it for 10 times to get the average running time. Since each algorithm has different number of parameters, for fairness, we do not tune the parameters. For the two boosting algorithms, we use  $T = 100$  decision stumps as our weak learners and set  $\beta = 0.2$  and  $\epsilon = 0.1$  in all experiments. For logistic regression, we use the default parameter  $C = 1$ .

### 7.4.2 Synthetic Dataset

We use the synthetic dataset from [138]. This dataset has an interesting theoretical property that although it is linearly separable, by randomly flipping a tiny fraction of labels, all convex potential boosting algorithms, including AdaBoost, fail to learn well. A random example is generated as follows. The label  $y$  is randomly chosen from  $\{-1, +1\}$  with equal

Table 7.1: Average (over 10 trials) error rate (%) and standard deviation on the synthetic dataset

Noise	Dist.AdaBoost	Dist.SmoothBoost	Liblinear-LR
0.1%	11.64 $\pm$ 3.82	4.28 $\pm$ 0.66	0.00
1%	25.97 $\pm$ 1.56	13.38 $\pm$ 4.66	0.00
10%	28.04 $\pm$ 0.94	27.07 $\pm$ 1.60	37.67

Table 7.2: Average (over 10 trials) error rate (%) and standard deviation on real-world datasets

Dataset	#examples	# features	Dist.AdaBoost	Dist.SmoothBoost	Liblinear-LR
Adult	48,842	123	15.71 $\pm$ 0.16	<b>15.07</b> $\pm$ 2.32	15.36
Ijcnn1	141,691	22	5.90 $\pm$ 0.10	<b>4.33</b> $\pm$ 0.18	7.57
Cod-RNA	488,565	8	<b>6.12</b> $\pm$ 0.09	6.51 $\pm$ 0.11	11.79
Covtype	581,012	54	24.98 $\pm$ 0.22	24.68 $\pm$ 0.30	<b>24.52</b>
Yahoo	3,251,378	10	37.08 $\pm$ 0.15	<b>36.86</b> $\pm$ 0.27	39.15

odds. The feature  $x = \langle x_1, \dots, x_{21} \rangle$ , where  $x_i \in \{-1, +1\}$ , is sampled from a mixture distribution: **1)** With probability 1/4, set all  $x_i$  to be equal to  $y$ . **2)** With probability 1/4, set  $x_1 = x_2 = \dots = x_{11} = y$  and  $x_{12} = x_{13} = \dots = x_{21} = -y$ . **3)** With probability 1/2, randomly set 5 coordinates from the first 11 and 6 coordinates from the last 10 to be equal to  $y$ . Set the remaining coordinates to  $-y$ .

We generate 1,600,000 examples in total for training on 16 machines and test on a separate set of size 100,000. The results are shown in Table 7.1. One can see that our approach (Dist.SmoothBoost), is more resistant to noise than Dist.AdaBoost and significantly outperforms it for having upto 1% noise. In high noise setting (10%), Liblinear performs poorly, while our approach achieves the best error rate.

### 7.4.3 Real-world Datasets

We run the experiments on 5 real-world datasets with sizes ranging from 50 thousands to over 3 millions: ADULT, IJCNN1, COD-RNA, and COVTYPE from the LibSVM data

Table 7.3: Average run time (sec) on real-world datasets

<b>Dataset</b>	<b>Dist.AdaBoost</b>	<b>Dist.SmoothBoost</b>	<b>Liblinear-LR</b>
Adult	5.02	15.54	0.06
Ijcnn1	0.76	9.19	0.10
Cod-RNA	1.08	10.11	0.12
Covtype	3.71	6.48	0.31
Yahoo	3.37	3.79	1.37

repository <sup>1</sup>; YAHOO from the Yahoo! WebScope dataset [149]. The Yahoo dataset is used for predicting whether a user will click the news article on their front page. It contains user click logs and is extremely imbalanced. We trim down this dataset so that the number of positive and negative examples are the same. The detailed information of the datasets are summarized in Table 7.2. Each dataset is randomly split into 4/5 for the training set and 1/5 for the testing set.

The average error rate and the total running time are summarized in Table 7.2 and Table 7.3, respectively. The bold entries indicates the best error rate. Our approach outperforms the other two on 3 datasets and performs competitively on the other 2 datasets. In terms of running time, Liblinear is the fastest on all datasets. However, the communication of our algorithm only depends on the dimension  $d$ , so even for the largest dataset (YAHOO), it can still finish within 4 seconds. Therefore, our algorithm is suitable for many real-world situations where the number of examples is much larger than the dimension of the data. Furthermore, our algorithm can be used with more advanced weak learners, such as distributed logistic regression, to further reduce the running time.

## 7.5 Conclusions

We propose the first distributed boosting algorithm that enjoys strong performance guarantees, being simultaneously noise tolerant, communication efficient, and computationally efficient; furthermore, it is quite flexible in that it can be used with a variety of weak learners.

<sup>1</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets>.

This improves over the prior work of [118, 117] that were either communication efficient only in noise-free scenarios or computationally prohibitive. While enjoying nice theoretical guarantees, our algorithm also shows promising empirical results on large synthetic and real-world datasets.

Finally, we raise some related open questions. In this work we assumed a star topology, i.e., the center can communicate with all players directly. An interesting open question is to extend our results to general communication topologies. Another concrete open question is reducing the constant in our error bound while maintaining good communication complexity. Finally, our approach uses centralized weak learners for learning general concept classes, so the computation is mostly done in the center. Are there efficient distributed weak learners for some specific concept classes? That could provide a more computation balanced distributed learning procedure that enjoys strong communication complexity as well.

## **Part III**

# **Applying AI to Protect Enterprise and Society**

## OVERVIEW

Part I and II of my thesis provide theories, algorithms, and insight of the capabilities and limitations of AI. But how can we deploy AI in practice and utilize it to provide solutions that solve real enterprise security problems and create positive societal impacts? I believe the key to success is through deep communication and collaboration with domain experts. In Part III of my thesis, I present two projects that utilize AI to help the security industry and for broader social good through deep collaboration with industry and government partners.

- **Virtual Product** (Chapter 8) is a patented enterprise cyber threat detection method developed with Symantec.
- **Firebird** (Chapter 9) is an open-source framework, developed with the Atlanta Fire Rescue Department, to help municipal fire departments identify and prioritize commercial property fire inspections.

## CHAPTER 8

### PREDICTING CYBER THREATS WITH VIRTUAL SECURITY PRODUCTS

Cybersecurity analysts are often presented suspicious machine activity that does not conclusively indicate compromise, resulting in undetected incidents or costly investigations into the most appropriate remediation actions. There are many reasons for this: deficiencies in the number and quality of security products that are deployed, poor configuration of those security products, and incomplete reporting of product-security telemetry. Managed Security Service Providers (MSSP's), which are tasked with detecting security incidents on behalf of multiple customers, are confronted with these data quality issues, but also possess a wealth of cross-product security data that enables innovative solutions. We use MSSP data to develop *Virtual Product*, which addresses the aforementioned data challenges by predicting what security events would have been triggered by a security product if it had been present. This benefits the analysts by providing more context into existing security incidents (albeit probabilistic) and by making questionable security incidents more conclusive. We achieve up to 99% AUC in predicting the incidents that some products would have detected had they been present.

#### 8.1 Introduction

Security products often are primed to detect certain threats extremely well. In other contexts, they will generally provide less than conclusive or no evidence of attacks. This motivates a defense in depth strategy that advocates for deploying multiple kinds of security devices to provide the most robust defense. Clearly it is infeasible to deploy every single security product to maximally protect every single device in an organization. Cybersecurity analysts, therefore, must contend with suboptimal context regarding potential attacks because the products that are providing telemetry are not well suited for a potential attack. Their

Product Type	Alert Description (Event)
Gateway	TCP Urgent Data Enforcement
Gateway	TCP anomaly
Gateway	TCP Out of Sequence
Gateway	ICMP Echo Request
Windows	Cryptographic operation
Windows	Attempt to unprotect auditable protected data
Windows	Logon attempt using explicit credentials
Windows	Key file operation
Windows	Filter Manager Event 1
Windows	Attempt to register a security event source
Windows	Attempt to unregister a security event source
Windows	Special privileges assigned to new logon
Windows	A privileged service was called
Windows	A network share object was accessed
Firewall	TCP Connection
Firewall	UDP Connection
Proxy	TCP Cache Hit
Proxy	TCP Cache Miss: Non-Cacheable Object

Table 8.1: A long list of inconclusive alerts generated in a real incident of a machine infected by the infamous *Zbot Trojan*. These alerts overwhelm a cybersecurity analyst, and do not help answer important questions such as: *Is this machine compromised? How severe is the attack? What actions should be taken?* Our technique, *Virtual Product*, correctly predicts the presence of the infamous *Zbot Trojan*, which would have been identified by an AV product, had it been installed.

confidence in either pursuing or ignoring a potential compromise is often less than ideal.

The key to improving detection rates in this environment is to learn from the vast amounts of telemetry produced by the prevalent defense-in-depth approach to computer security, wherein multiple security products are deployed alongside each other, producing highly correlated alert data. By studying this data, we are able to accurately predict which security alerts a product would have triggered in a particular situation, even though it was not deployed. A representative example is shown in Table 8.1, wherein security alerts produced by several products hint at the possibility of a security problem, but do not present conclusive evidence. Our models, however, are able to correctly predict the presence of the Zeus (also known as the Zbot) trojan, as the cause of the anomalous system and network behavior on the machine.

We introduce and formulate the novel problem of *Virtual Product*, the first known attempt to predict the security events and high-severity incidents that would have been identified by a product if it had been deployed. Given sufficient data from many organizations deploying different sets of security products, we posit it should be possible to predict the events that would have been reported by additional security products that were not deployed. This analysis benefits from the observations that many security products detect the same threats, and that attacks are typically automated and therefore proceed in predictable sequences of behavior.

Figure 8.1 shows how *Virtual Product* works. We formulate incident data as a large matrix. Each row, called a *machine-day*, tracks all of the security events that were observed on a particular machine, on a given date. Although many entries will be empty since machines are at most protected by a handful of products, we can predict the likely events that would have been triggered by those products that were not deployed. The security officers can then hopefully make a more informed decision about the trade off of cost and value of what other security products would provide. For the analyst, *Virtual Product* enriches each incident (i.e., row) with more context to understand the severity of the threat posed by the

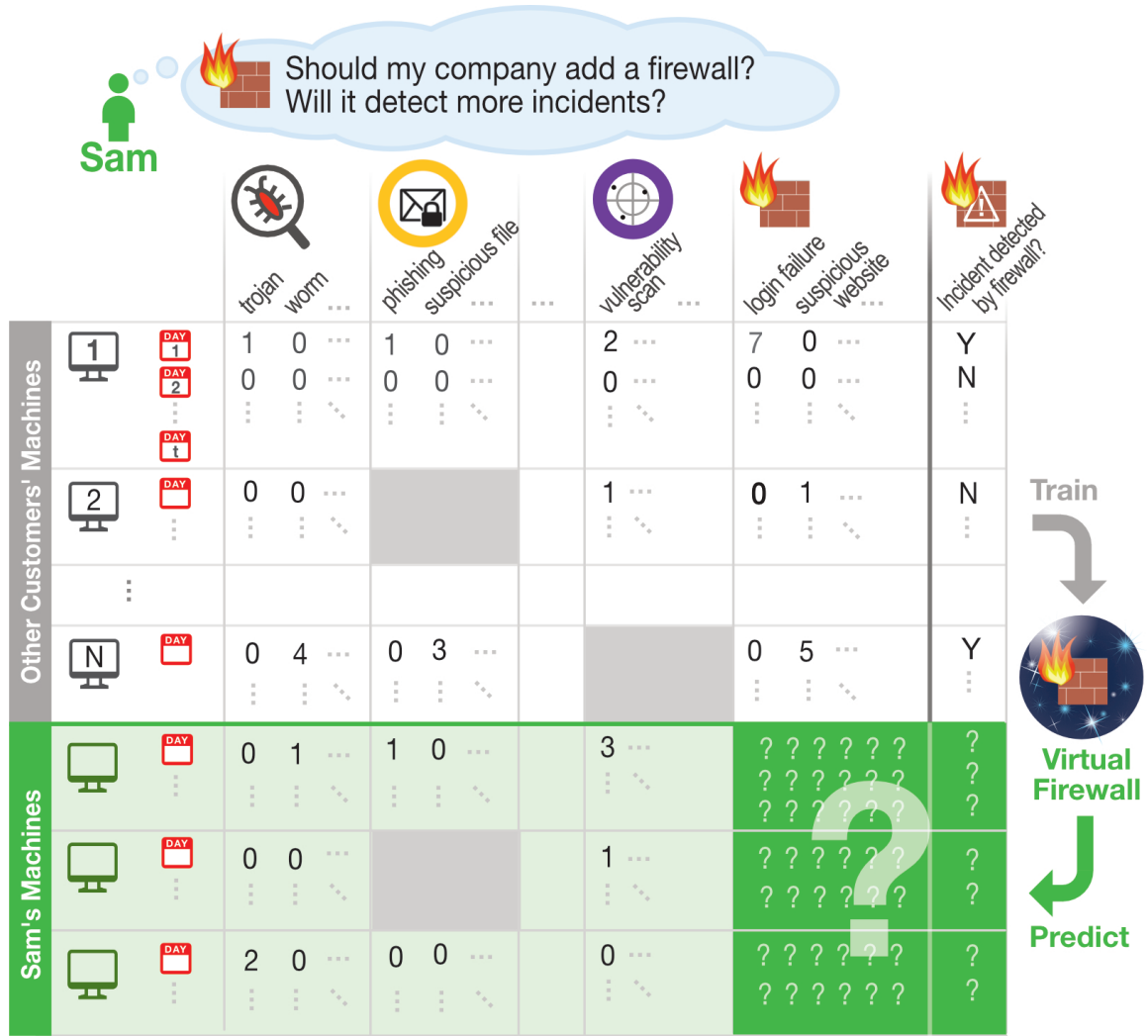


Figure 8.1: *Virtual Product* helps our user Sam discover and understand cyber-threats, and informs deployment decisions (e.g., add firewall?) through semi-supervised non-negative matrix factorization on telemetry data from other users (with firewalls deployed). In the data matrix, each row represents a machine-day, and each column a security event's occurrences. Missing events from undeployed products are shown as gray blocks. The last column indicates if the firewall has detected an incident. Our *virtual* firewall serves as a proxy to the actual product and predicts the output Sam may observe (dark green block) if he deploys it.

observed activity. Our work makes the following contributions:

- **Novel Idea of Virtual Product.** We introduce the problem of simulating a security product's individual security events and the security incidents that these events would have raised, had it actually been deployed. We formulate techniques by which the security data managed by Security Incident and Event Managers (SIEM's) and Managed Security Service Providers (MSSP's) on behalf of multiple products can be used for this purpose

(see Table 8.2 for definitions of these terms).

- **Effective Approach.** We provide a practical implementation for this problem by adapting semi-supervised non-negative matrix factorization techniques, which simultaneously addresses the problem of security incident and event prediction for the absent products, with high accuracy.
- **Impact to Security Industry.** Our Virtual Product model will impact the security industry by increasing company security at significantly reduced costs. We are working towards making *Virtual Product* events and security incidents available to customers of an MSSP. By deploying *Virtual Product* on behalf of customers, we provide a new way for them to experience the potential benefits of security products without deploying them, allowing them to make more informed purchasing decisions.

To enhance readability of this chapter, we have listed the terminology used in this chapter in Table 8.2. The reader may want to return to this table throughout this chapter for technical terms' meanings and synonyms used in various contexts of discussion. We proceed by discussing related work in Section 8.2, and present our proposed *Virtual Product* model in Section 8.3. We evaluate the performance of our algorithm in Section 8.4. Next, we discuss the expected impact of *Virtual Product* and concrete deployment plans studies in Section 8.5. Finally, we discuss our findings and conclude in Section 8.6.

## 8.2 Related Work

There has been growing interest in applying machine learning and data mining techniques to detect cyber-threats, such as malicious files [45, 44], malicious websites [150, 151], and online fraudulent behaviour [52], using approaches range from Naive Bayes [152], to neural networks [153], decision trees [154], to large-scale graph-based inference [45, 44]. In contrast to prior work, instead of predicting cyber-threats directly, we formulate and tackle the novel *Virtual Product* problem of predicting how a security product would work

Technical term	Meaning
Virtual product	A machine learning model used to reconstruct a real security product's behavior
Machine-day	A machine on a particular day
Security event	A description of activity recorded by a security product, not necessarily malicious, e.g., login failure
Incident	A serious security threat, evidenced by one or more events, warranting attention, e.g., unblocked malware
SIEM	Security Incident and Event Managers, which manage security events produced by products, that they analyze to detect and report security incidents
MSSP	Managed Security Service Providers, which run a SIEM on behalf of multiple customers

Table 8.2: Terminology used in this chapter

in a customer's specific environment, had it been deployed. Not only do we predict the incident triggering behaviour of a product, but also reconstruct all the security events it detects, tackling both tasks simultaneously using matrix factorization methods. To the best of our knowledge, *Virtual Product* addresses a novel problem, one that provides additional context and detection capabilities by predicting the incidents and individual security events that would be provided by security products had they been deployed.

Matrix factorization [155, 156, 157] exploits latent features of a data matrix by decomposing the matrix into a series of low-rank factor matrices. These factor matrices, though additional constraints can be enforced on them, are learned by minimizing a generalized Bregman divergence [156, 158] between the original data matrix and the dot product of the low-rank factor matrices. Matrix factorization has been popularly used in collaborative filtering [159, 160, 161] of highly sparse user-item rating records to predict users' preferences and recommend unrated products. Document clustering is another well-studied research domain that uses matrix factorization. A common approach is to apply non-negative matrix factorization (*NMF*) [162, 163] on sparse bag-of-words features of documents and group

documents using the derived non-negative factors [162, 163, 164]. Supervision in the context of matrix factorization introduces class separating structure into the factor matrices, which enforces linear separation of classes in the linear projection of data [159, 165]. The objective function of the factorization has been designed to enforce specific properties of the latent projected data. Previous efforts on supervised and weakly supervised matrix factorization can be found in [159, 166, 167, 168]. Most of them focus on decomposing densely valued data matrices to improve clustering accuracy. Our algorithm extends and adapts prior work to the classification problem of predicting attacks by reconstructing missing signals on extremely sparse data.

### 8.3 Proposed Model: Virtual Product

Given a security product  $P$ , our *Virtual Product* model aims to detect and categorize incidents for customers who have not deployed  $P$ . We formulate the construction of *Virtual Product* as a classification problem, training it on machine-day observations collected from machines that have deployed  $P$ . The training process learns the functional mapping between the event occurrence patterns of other products and the incident class labels reported by  $P$ . During testing, the *Virtual Product* model takes as input the observed event occurrence patterns from products except  $P$ , and produces incident detection and categorization results.

A main challenge for *Virtual Product* is in training and applying it with incomplete event occurrence patterns as input. Events may be missing either because their corresponding products are not deployed, or due to data corruption at the telemetry data collection process.

To address this issue, we propose a semi-supervised non-negative matrix factorization method (*SSNMF*) as a core computation technique for *Virtual Product*. It extracts an unified discriminative feature representation of the event occurrence records from both the training and testing datasets. We conduct incident detection and categorization in *Virtual Product* by feeding the learned feature representations as input to any standard supervised classifiers. *Virtual Product* denotes the process of conducting *SSNMF* on event occurrence

data, followed by training a supervised classifier on the output of *SSNMF*.

Another contribution of *Virtual Product* is to estimate event occurrence patterns that are missing from the observed data. It is helpful for security analysts to understand relations between event occurrence profiles and reported security incidents. The *SSNMF* well matches this requirement, as it is intrinsically equipped with the capability of reconstructing the missing event values through inner product of low-rank matrices.

### 8.3.1 Semi-Supervised Non-negative Matrix Factorization (SSNMF)

We use a non-negative data matrix  $X \in R^{N \times M}$  to denote the aggregation of both training and testing event occurrence data. Each row in  $X$ , noted as  $X_{i,:}$ , denotes occurrence counts of different events around a machine-day. Without loss of generality, the first  $N_1$  rows of  $X$  belong to the training event occurrence data. They are equipped with corresponding incident class labels reported by the target product  $P$ . The remaining  $N - N_1$  rows of  $X$  are the testing data corresponding to event occurrence data collected from customers' machines without  $P$  deployed.

Non-negative Matrix Factorization reconstructs a non-negative data matrix  $X \in R^{N \times M}$  using the dot product of two non-negative factors  $U \in R^{N \times k}$  and  $V \in R^{M \times k}$ , where  $k$  is the number of latent features that is often determined by cross-validation. As shown in Equation (8.1), the latent factors are learned by minimizing the reconstruction error on the observed events in our data.

$$U, V = \underset{U, V \geq 0}{\operatorname{argmin}} \|X - UV^T\|_o^2 \quad (8.1)$$

The norm  $\|\cdot\|_o$  indicates the aggregated reconstruction error on the observed entries of  $X$ . Each row in  $U$ ,  $U_{i,:}$ , represents the linear projection of  $X_{i,:}$ , which formulates a new feature representation of machine-day observations in a low-dimensional space. Column vectors of  $V$  are the projection bases spanning the projection space.

To integrate supervision information into the matrix factorization process, we introduce a class-sensitive loss into the objective function of matrix factorization, in order to force

machine-day observations of different classes to be separated from each other in the projected space. Equation (8.2) and Equation (8.3) give the formulation of the discriminative loss functions defined for binary and multi-class classification scenarios, respectively.

$$F(\hat{Y}, U, W) = - \sum_{i=1}^N \hat{Y}_i \log \frac{1}{1 + \exp(-U_{i,:} W^T)} + (1 - \hat{Y}_i) \log \frac{1}{1 + \exp(U_{i,:} W^T)} \quad (8.2)$$

$$F(\hat{Y}, U, W) = - \sum_{i=1}^N \sum_{j=1}^C \hat{Y}_{i,j} \log \frac{\exp(U_{i,:} W_{j,:}^T)}{\sum_{j'} \exp(U_{i,:} W_{j',:}^T)} \quad (8.3)$$

where  $W \in R^{1 \times k}$  stores the regression coefficients.  $\hat{Y}_i$  represents the class label of each machine-day observation. For labeled machine-days,  $\hat{Y}_i$  is either 1 or 0, depending on whether  $X_{i,:}$  belongs to positive or negative class. For unlabeled machine-days,  $\hat{Y}_i$  represents any plug-in estimator of probabilistic confidence of  $X_{i,:}$  belonging to positive class. In the multi-class version of the loss function,  $C$  denotes the number of classes in the labeled dataset. As a result,  $W$  becomes a  $R^{C \times k}$  matrix. Each row in  $W$  corresponds to the regression coefficients for each class.  $\hat{Y}_{i,j}$  of labeled data is defined following one-hot encoding scheme. For unlabeled data,  $\hat{Y}_{i,j}$  represents the probabilistic class membership of each  $X_{i,:}$ .  $U$  is the common factor shared by both matrix factorization in Equation (8.1) and the class-sensitive loss function defined in Equation (8.2) and (8.3). This design guarantees the feature representation  $U$  preserves the class separating structure of the training data.

$\hat{Y}$  for unlabeled data can be initialized using external oracles with probabilistic output, such as gradient boosting and logistic regression. In this work, we treat  $\hat{Y}$  as one variable to learn and estimate it by jointly optimizing the objective function with respect to  $U$ ,  $V$ ,  $W$  and  $\hat{Y}$ . We assume that unlabeled data points with similar profiles are likely to share similar soft class label  $\hat{Y}$ . By enforcing such assumption to the objective function design, we explicitly inject supervised information into the projection of both labeled and unlabeled

machine-day observations. The complete optimization problem of *SSNMF* is shown in the following equation.

$$\begin{aligned}
U, V, W, \hat{Y} = & \underset{U, V \geq 0, W, 1 \geq \hat{Y} \geq 0}{\operatorname{argmin}} \|X - UV^T\|_o^2 + \alpha F(\hat{Y}, U, W) \\
& + \beta \operatorname{Tr}(\hat{Y}^T L \hat{Y}) + \gamma(\|U\|^2 + \|V\|^2) + \rho\|W\|^2 \\
& s.t. \hat{Y}_i = Y_i \text{ if } X_{i,:} \text{ is labeled}
\end{aligned} \tag{8.4}$$

The constraint in the objective function requires strict consistency between  $\hat{Y}$  and the true class labels on labeled machine-day observations.  $L$  is the graph laplacian matrix defined based on  $K$ -nearest neighbor graph of the whole data matrix  $X$ . Minimizing the trace function  $\operatorname{Tr}(\hat{Y}^T L \hat{Y})$  propagates the confidence of class membership from true class label of labeled machine-days to unlabeled machine-days. It embeds class-separating information into the projection  $U$  of unlabeled machine-days. Regularization terms  $\gamma(\|U\|^2 + \|V\|^2)$  and  $\rho\|W\|^2$  are added to prevent over-fitting.

### 8.3.2 Optimization Algorithm

We use coordinate descent to optimize Equation 8.4. During each iteration,  $U$ ,  $V$ ,  $W$  and  $\hat{Y}$  are updated alternatively. One of the four variables are updated while all the others are fixed. Iterations continue until the objective value cannot be further improved.  $U$ ,  $V$  are updated using multiplicative update [163], which is a popular optimization technique for solving many variants of *NMF*. Equation (8.5) gives the formulations of multiplicative update of  $U$  and  $V$

$$\begin{aligned}
U^{t+1} &= U^t \odot \frac{[(X \odot M)V]^+ + [(UV^T \odot M)V]^- + \alpha[\hat{Y}W]^+ + \alpha[RW]^-}{[(X \odot M)V]^- + [(UV^T \odot M)V]^+ + \alpha[\hat{Y}W]^- + \alpha[RW]^+ + \gamma U} \\
V^{t+1} &= V^t \odot \frac{(X \odot M)^T U}{(UV^T \odot M)^T U + \gamma V}
\end{aligned} \tag{8.5}$$

where  $[A]^+ = (|A| + A)/2$  and  $[A]^- = (|A| - A)/2$ .  $R$  is the output from the sigmoid

function (binary classification)  $R_i = \frac{1}{1+\exp(-U_{i,:}W^T)}$  or the softmax function (multi-class classification)  $R_{i,j} = \frac{\exp(U_{i,:}W_{j,:}^T)}{\sum_{j'=1}^C \exp(U_{i,:}W_{j',:}^T)}$ . The operation  $\odot$  indicates Hadamard product between matrices.  $M$  is a entry-wise weight matrix.  $M_{i,j} = 1$  if the entry  $X_{i,j}$  is observed, and  $M_{i,j} = 0$  otherwise.

Updating  $\hat{Y}$  consists of two components. For one aspect, the learning of  $\hat{Y}$  is based on supervision information propagation. For the other aspect, estimates of  $\hat{Y}$  depends on the output from the sigmoid or softmax function, which encodes the retraction from data reconstruction penalty in the objective function. Equation (8.6) and Equation (8.7) define how to estimate  $\hat{Y}$  in binary and multi-class classification scenarios:

$$\begin{aligned} \hat{Y}_i &= Y_i \text{ if } X_{i,:} \text{ is labeled} \\ \hat{Y}^{t+1} &= \hat{Y}^t \odot \frac{\alpha \log(1 + \exp(UW^T)) + 2\beta S\hat{Y}}{\alpha \log(1 + \exp(-UW^T)) + 2\beta D\hat{Y}} \end{aligned} \quad (8.6)$$

$$\begin{aligned} \hat{Y}_i &= Y_i \text{ if } X_{i,:} \text{ is labeled} \\ \hat{Y}^{t+1} &= \hat{Y}^t \odot \frac{\alpha \log \hat{R} + 2\beta S\hat{Y}}{2\beta D\hat{Y}} \end{aligned} \quad (8.7)$$

where  $\hat{R}_{i,j} = \frac{\exp(U_{i,:}W_{j,:}^T)}{\sum_{j'=1}^C \exp(U_{i,:}W_{j',:}^T)}$ .  $S$  is weight matrix of  $K$ -nearest neighbor graph.  $D$  is a diagonal matrix with  $D_{i,i}$  defined as  $\sum_{j=1}^N S_{i,j}$ .

By removing terms without  $W$  in Equation (8.4), the left terms of the objective function formulate a  $L2$ -penalised logistic regression with soft class labels  $\hat{Y}$  and training data points in the projected space  $U$ . Therefore, learning  $W$  given  $U$  and  $\hat{Y}$  fixed can be performed through iterative gradient descent until convergence. We found that the number of iterations can be dramatically reduced by choosing the step size of gradient in an adaptive way using

Dataset	#Machine Days	#Detected Incidents	#Events	Sparsity	#Incident Type
<i>FW1</i> : Firewall 1	4506	770	1011	98%	10
<i>FW2</i> : Firewall 2	9254	3093	1927	99%	12
<i>FW3</i> : Firewall 3	4477	1274	2019	98%	10
<i>EP1</i> : Endpoint Protection 1	18983	4128	2409	99%	30
<i>EP2</i> : Endpoint Protection 2	8006	904	988	97%	5

Table 8.3: Summary of the training datasets (Jul-Sept) for the top five products that detect the most incidents.

AdaGrad [169], as shown below:

$$W^{t+1} = W^t + \lambda \frac{G_{W^t}}{\sqrt{\sum_{t'=1}^{t-1} G_{W^{t'}}}} \quad (8.8)$$

where  $G_W$  is the gradient of Equation (8.4) with respect to  $W$ .

When  $U$ ,  $V$  and  $W$  converge,  $U_{1:N_1,:}$  and  $U_{N_1:N,:}$  are used as low-dimensional feature representations of the training and testing data, respectively. We then train a logistic regression on the row vector space of  $U$  to conduct incident detection and categorization in *Virtual Product*. Note that we are not restricted to logistic regression. We choose it due to its simplicity and probabilistic decision output. Despite its simplicity, it shows superior performance thanks to the learned feature representation  $U$ , as reported in the experimental study.

## 8.4 Evaluation

### 8.4.1 Data Collection

Our evaluation uses telemetry data sent from a leading Managed Security Service Provider (MSSP), which supports roughly 80 security products from different vendors. Customers send telemetry from their deployed products to the MSSP, which analyzes the telemetry to identify and report incidents. Due to space constraints, we show the results of the top five products that detect the most incidents: three firewalls (*FW1*, *FW2*, *FW3*) and two endpoint

protection products ( $EP1$ ,  $EP2$ ).

To evaluate our approach’s prediction performance for a specific product  $P$ , we derived an anonymized dataset from the telemetry data. This derived dataset consists of data contributed by the machines that have deployed  $P$ , which allows us to extract ground truth labels. When performing prediction, we do not use any events from  $P$ . In other words, we pretend that product  $P$  is not deployed and hide all its events.

The dataset is represented as a  $N$ -by- $M$  matrix  $X$  (see Figure 8.1). Each row  $X_{i,:}$  is an instance that represents a machine-day. Each column  $X_{:,j}$  is a feature that corresponds to the number of occurrences of a event from different products except  $P$ . To prevent numerical overflow during computation, we take the logarithm of each event occurrence count in  $X$ . Since events relevant to an incident may not appear within a single machine-day, when counting occurrences, we consider the period that spans three days before and after the machine-day. Note that our task is not to predict an incident before it happens, so we also collect events observed after the machine-day. To prevent duplicate and similar instances, we only use machine-days from the same machine that are at least one week apart from other machine-days.

The matrix is extremely sparse, and each machine-day typically only has a few observed events. Events may be missing if their corresponding products are not deployed. They may also be caused by data corruption when the products report them. To avoid machine-days with zero or very few observed events, which are nearly impossible to perform prediction, we filter out all machine-days with fewer than 20 observed events.

There are two sets of labels associated with each machine-day, one for binary classification of whether there is an incident, and the other one for multi-class classification of the incident type. The positive and negative machine-days are collected as follows. For each incident reported in our system, we label a machine-day as **positive** if an incident is detected by product  $P$ . For *negative* machine-days, we use the same set of machines (as when collecting positive instances). A machine-day is labeled **negative** if no incidents have

<b>Dataset</b>	<b>#Machine-days</b>	<b>#Detected Incidents</b>	<b>Sparsity level</b>
<i>FW1</i>	3090	355	98%
<i>FW2</i>	6515	2830	98%
<i>FW3</i>	2660	253	97%
<i>EP1</i>	8222	2377	98%
<i>EP2</i>	2275	754	98%

Table 8.4: Summary of validation datasets (Oct-Dec).

been detected by any products within a one-month period (15 days before to 15 days after). This binary label definition is used in experiments, in order to evaluate the capability of the proposed method for detecting malicious incidents. We also include a multi-class definition of incident labels. The multi-class incident label denotes multiple categories of detected incidents, valued as  $\{-1, 1, 2, \dots, C\}$ , where  $C$  is the number of incident categories, and  $-1$  means “no incident”.

The same data collection process is performed over two independent time periods. The first dataset was collected from July to September in 2016 — we call this the **training** dataset (summarized in Table 8.3), on which we conduct cross-validation to verify theoretical validity of the algorithmic design in Section 8.4.4 and evaluate reconstruction performances in Section 8.4.3. The second dataset was collected from October to December in 2016 — we call this the **validation** dataset (summarized in Table 8.4). We use the *training* dataset to tune the parameters of our model, then apply it on the *validation* dataset to evaluate incident classification accuracy in real-world applications

In Table 8.3, we also show the sparsity level of each product’s dataset, to highlight how sparse the data matrices are in our study. The sparsity level of a given data matrix is defined as the fraction of unobserved entries in the matrix.

## 8.4.2 Experiment Setup and Overview

Our experiment consists of three parts.

1. In Section 8.4.3, benefiting from matrix factorization, the proposed method estimates

count values of security events produced by those products whose events were withheld from the dataset. The reconstructed event counts will later help us determine whether a security incident would have been raised by the events produced by the withheld product. Since security incident are formulated as collections of relevant events, reconstructing the missing events is essential for incident reproduction based on the occurrence pattern of the corresponding incident. Furthermore, the individual events provide important insights and context into the nature of the security incident, which frequently enable improved triage and remediation of the incident. We evaluate our proposed event-reconstruction model by measuring reconstruction error between ground truth event counts and our estimated values.

2. In Section 8.4.4, we evaluate the performance of our proposed method for detecting security incidents. This output of Virtual Product’s methods allows us to build a incident detector based on incomplete event information. The test is conducted on the training data matrices and the incident labels of all five products. Both binary and multi-class incident labels are produced for this test.
3. In Section 8.4.5, we investigate the computational complexity and empirical scalability of our proposed *Virtual Product* model, noted as *VP*.

### 8.4.3 Evaluation on Reconstruction Accuracy

We validate in this section that the proposed model can compensate for missing events. We investigate the reconstruction performance of *Virtual Product* with respect to the occurrence counts of withheld event observations. Reconstruction capability is a key function of the proposed model, since knowing which events were responsible for triggering a predicted security incidents is essential to the understanding of that incident, and to its remediation. Accordingly, we evaluate the reconstruction capability of the proposed method. We randomly select 50% of the observed entries and take the event count of these entries as ground truth.

Dataset	R-squared Score		Percentage	
	Mean	Std	Mean	Std
<i>FW1</i>	0.8493	0.0036	0.9819	0.0009
<i>FW2</i>	0.7300	0.0032	0.9834	0.0002
<i>FW3</i>	0.8408	0.0023	0.9858	0.0007
<i>EP1</i>	0.7356	0.0013	0.9801	0.0001
<i>EP2</i>	0.8193	0.0014	0.9832	0.0004

Table 8.5: Performance of reconstruction on all five datasets

After that, we hold out the ground truth counts and apply our matrix factorization method to derive the estimated count values of the masked entries. To measure reconstruction accuracy we use the R-squared score between the ground truth and the estimated values. To remove randomness introduced by sampling, we repeatedly sample the observe entries 10 times. The average and standard deviation of the derived R-squared scores from different sampling rounds are used as a comprehensive evaluation metric of the reconstruction performance.

The average and standard deviation of R-squared scores derived on the five datasets are shown in Table 8.5. We also provide a statistical summary of our reconstruction results in Table 8.5. In addition, for each dataset, we count the percentage of the entries in which the reconstructed event occurrence counts are larger than 50% of the corresponding ground truth occurrence count values, as noted as *Percentage* in Table 8.5. This statistical summary provides an intuitive understanding on the reported reconstruction accuracy. In practice, if the reconstructed occurrence count of a given event is close enough to its ground truth, the reconstruction is precise enough to estimate whether this event was triggered by the corresponding product. The results show that *Virtual Product* is able to reconstruct event occurrence patterns with precision for the security products. As seen in the results, almost all masked event occurrence patterns are perfectly recovered through the matrix factorization process embedded using our proposed method. As we will see in Section 8.4.6, these recovered security events enable machine learning models to perform improved incident detection. The true value of this work, however, is perhaps best illustrated by the case studies shown in Section 8.5.1.

#### 8.4.4 Evaluation: Incident Detection and Categorization

We perform 10-fold Monte Carlo cross-validation, where each randomly samples 70% of the machine-days from the training dataset collected from July to September in 2016. The remaining 30% is left for testing.

We set up a baseline model (shorthand: **LR**) by training a logistic regression classifier directly on the event count matrix  $X$ , with missing entries filled with zeros. In our approach (shorthand: **VP**), we train a logistic regression classifier on the low-dimensional feature representation of  $X$  produced by *Virtual Product* model.

The purpose of introducing the baseline model is two-folds. Firstly, we use the results from the baseline model to further validate our initial assumption: it is possible to predict the events that would have been reported by additional security products that were not deployed. The baseline model conducts classification using only the observed events from the deployed products. No reconstructed event information is embedded. Therefore, if the baseline method can detect or categorize incidents with an acceptable accuracy, we have strong reason to believe the proposed *Virtual Product* model can perform even better by incorporating the reconstructed event counts into the classifier design. Secondly, we aim to conduct a fair comparative study for our proposed methodology, though we note that the baseline model is not a comparison to prior art, as *Virtual Product* addresses a novel problem of not only predicting the incidents but also recovering the associated security events. The objective function of *SSNMF*, used in *Virtual Product*, can be roughly understood as construction of a logistic regression classifier on the projected space of the original data. This comparative study aims to verify the benefits gained from the algorithmic design of *Virtual Product* for classification with missing features.

To allow fine-grained comparison, we compute the mean and standard deviation of the *Area-Under-Curve* (AUC) and the *True Positive Rate* (TPR) across 10 folds, and display them in Table 8.6 and Table 8.7, respectively. As we can see in the two tables, both the baseline and the proposed *Virtual Product* method present good classification performances over *training*

Dataset	VP AUC		LR AUC	
	Mean	Std	Mean	Std
<i>FW1</i>	<b>0.9831</b>	0.0041	0.9695	0.0055
<i>FW2</i>	<b>0.9900</b>	0.0018	0.9810	0.0029
<i>FW3</i>	<b>0.9200</b>	0.0070	0.8761	0.0131
<i>EP1</i>	<b>0.8218</b>	0.0066	0.8076	0.0072
<i>EP2</i>	<b>0.8962</b>	0.0083	0.8306	0.0164

Table 8.6: Our approach (VP) detects security incidents with high accuracies (AUCs) across all five datasets, outperforming the baseline model (LR).

Dataset	VP TPR		LR TPR	
	Mean	Std	Mean	Std
<i>FW1</i>	<b>0.9724</b>	0.0114	0.9661	0.0078
<i>FW2</i>	<b>0.9820</b>	0.0057	0.9810	0.0074
<i>FW3</i>	<b>0.7879</b>	0.0157	0.7608	0.0228
<i>EP1</i>	<b>0.5200</b>	0.0175	0.5016	0.0268
<i>EP2</i>	<b>0.5897</b>	0.0293	0.5663	0.0399

Table 8.7: True positive rate (TPR) of incident detection on all five data sets at 10% false positive rate (FPR). Our approach (VP) outperforms the baseline (LR)

datasets of all five security products. It indicates that counts of events collected from different organizations are able to predict occurrence of incidents that would have been reported by undeployed products. Furthermore, the result unveils consistently superior incident detection precision of the proposed *Virtual Product* model over the baseline method across the *training* datasets of different products. Figure 8.2 shows the average ROC curve and AUC derived from the cross-validation test, offering a global and intuitive view of incident detection performances over *training* datasets of different products using the proposed *Virtual Product* model. All obtained results support the design of the proposed *Virtual Product* method. Embedding matrix completion into classification helps extract correlation among observed events of different products, which increases available information to boost classification precision.

Additionally, test on the validation datasets follows a standard training-testing process of machine learning models in real-world applications. Classification model built with the

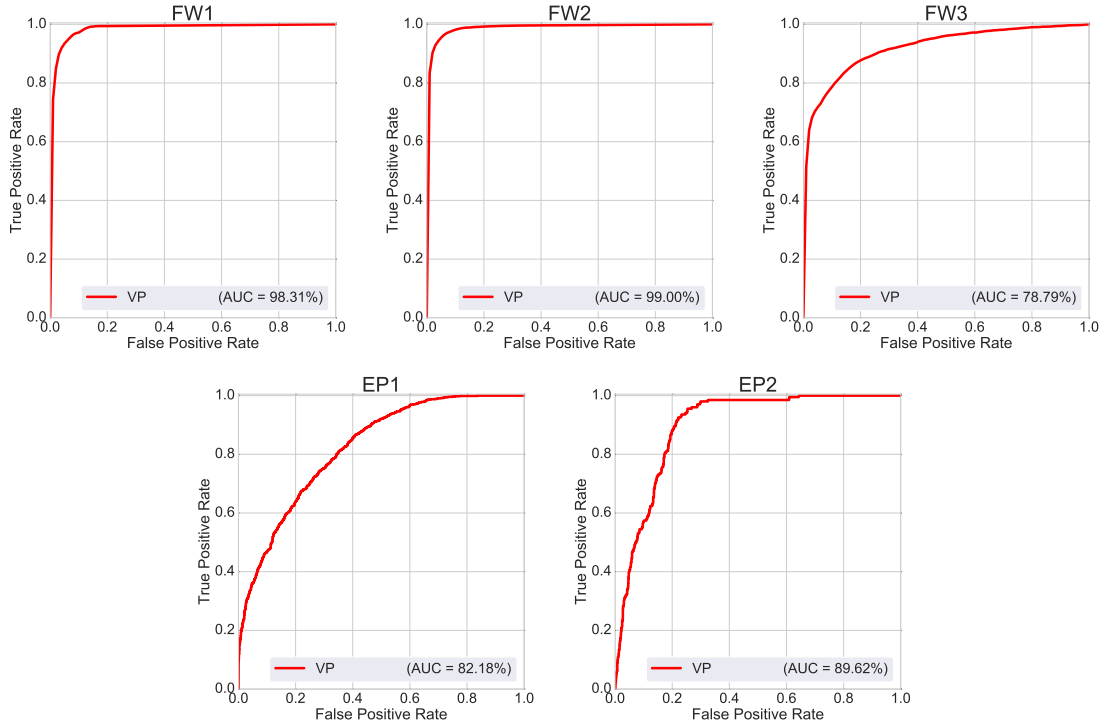


Figure 8.2: Averaged ROC curves from 10-fold cross-validation of *Virtual Product* on our top five product datasets.

*training* dataset collected within the precedent time period is used to detect incidents on the *validation* dataset formulated within the current time slot.

Interestingly, as shown in Figure 8.3, incident detection result using the proposed *Virtual Product* model presents consistent high detection accuracy over *validation* datasets of different products. The reported detection accuracy confirms the robustness of the proposed *Virtual Product* model.

As described in Section 8.3, the proposed *Virtual Product* can be seamlessly extended for incident categorization, which classifies detected incident at a finer scale. Without major modification, the proposed *Virtual Product* is able to achieve both incident detection and categorization (multi-class classification) at the same time. Table 8.8 shows the average F1-score of incident categorization on *training* datasets of different products using *Virtual Product*. As we can see, *Virtual Product* can achieve almost perfect incident categorization on the *FW1* and *FW2* datasets. In the *EP2* dataset, over 99% of detected incidents belong

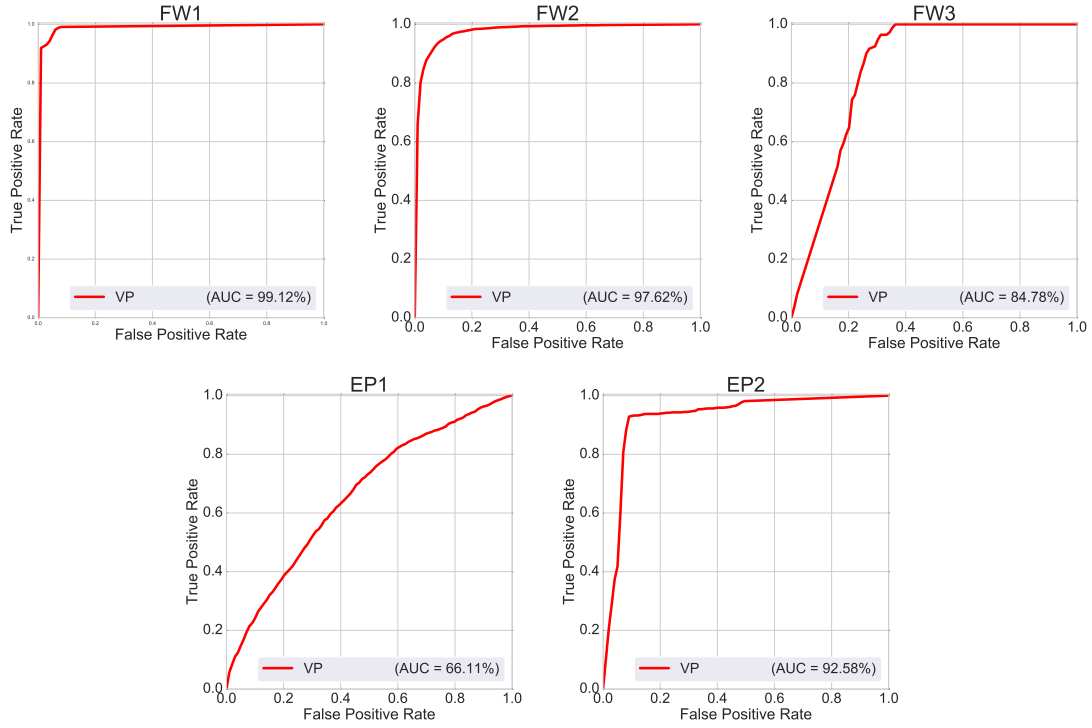


Figure 8.3: ROC curves of the Virtual Product model evaluated using the validation datasets of the five products.

to a single incident type. Severe class imbalance makes any classifier built on the dataset statistically unstable, so we chose not to include the *EP2* dataset in the experimental study of incident categorization. The categorization precisions on *EP1* and *FW3* are relatively lower. This is mainly due to class imbalance among different incident categories in these two datasets, particularly in the case of the *EP1* training dataset, for which nearly half of the 30 incident types are minority classes. Each of these minority classes contains fewer than 10 machine-day observations, which increases the difficulty of categorization. The impact of class imbalance is also confirmed by the baseline *LR* method. Nevertheless, even in this extreme situation, the proposed *Virtual Product* still obtains improvements compared to the baseline model.

In general, all experimental results in this section verify the effectiveness of *Virtual Product*. By jointly conducting matrix factorization and discriminative model learning, the proposed model makes full use of inter-event correlation to compensate information missing

	VP	LR
<i>FW1</i>	<b>0.9927</b>	0.9910
<i>FW2</i>	<b>0.9425</b>	0.9338
<i>FW3</i>	0.8005	<b>0.8043</b>
<i>EPI</i>	<b>0.7501</b>	0.7220

Table 8.8: Average F1 scores of incident categorization on our datasets. We do not include *EP2* because over 99% of the detected incidents belong to one single incident type.

due to the extremely sparse data structure. As a result, it provides a good reconstruction of the classification boundary from highly incomplete event occurrence data.

#### 8.4.5 Evaluation of Computational Cost

**Time Complexity Analysis.** The training of *Virtual Product*’s model consists of two parts. First, we construct a k-nearest neighbor ( $K$ -NN) graph in an offline manner. Nearest neighbor searching generally requires a cost of  $O(N^2M)$ , which is quadratic to the size of the dataset. Since the machine-day event count data is high-dimensional and highly sparse, we use an approximate  $K$ -NN method [170] tailored for sparse data. This reduces the cost of  $K$ -NN searching to  $O(DN\log N)$  in the worst case, where  $D$  is the number of feature dimensions. Next, we perform multiplicative updates in  $O(TNMk) + O(TNCk)$  time, where  $T$  is the number of iterations and  $k$  is the dimension of the projected representation  $U$ . The total cost of the proposed model is therefore at most  $O(DN\log N) + O(TNMk) + O(TNCk)$ . For all five datasets, we observed that 200 iterations ( $T = 200$ ) were sufficient to achieve convergence.

**Empirical Scalability.** We conducted experiments to study how our proposed model scales with increasing volumes of data. We report the average training runtime of the proposed *Virtual Product* model on *EPI* and *FW2* (across 10 runs). Our machine is a 64-bit Linux laptop (Ubuntu 14.0) with an Intel Core i7 quad-core CPU running at 2.5GHz, 16GB RAM and 500GB disk. The *Virtual Product* model is implemented in Python 2.7, with API provided in python scientific computing packages, numpy 1.13 and scikit-learn 0.18.1.

<b>Dataset</b>	<b>#Machine-days</b>	<b>Runtime (minutes)</b>
<i>FW2</i>	9,254	10
<i>EP1</i>	18,983	25
Large <i>FW2</i>	92,540	45
Large <i>EP1</i>	189,830	57

Table 8.9: Average virtual product training times (over 10 runs).

*EP1* and *FW2* datasets contain approximately  $19k$  and  $9k$  machine-days, representing real-world medium-scale applications. To study large-scale scenarios, we enlarge *EP1* and *FW2* datasets by 10 folds by replicating real machine-days contained in the original datasets. The enlarged *EP1* and *FW2* datasets (we call them Large *EP1* and Large *FW2*) contain  $190k$  and  $90k$  machine-days respectively. Table 8.9 shows the average runtimes of our proposed model across the datasets.

Our MSS service currently monitors about 80 products. Since each virtual product can be trained independently from each other, we can easily speed up overall computation through parallelization (e.g., distributed computation using Spark; more discussion in Section 8.5.2). For the evaluation in this section, we were able to train *Virtual Product* in under an hour, even on a commodity computer of modest power with an unoptimized software implementation.

#### 8.4.6 Improvement in Analyst Response Predictions

As additional evidence to support the utility of *Virtual Product*, we measure event reconstruction’s ability to improve the accuracy of a model that the internal Managed Security Services analysts use in determining whether to publish incidents to customers or suppress them as false positives. We use a recent version of this model that is trained with no interaction or influence from *Virtual Product*, and whose primary task is to recommend whether incidents should be published to customers, or suppressed.

We took the *FW1* dataset and removed all events from *FW1*, while keeping the events of other devices. We call this dataset  $X_{none}$ . We take  $X_{none}$  and create a new dataset  $X_{top2}$  from it, for which we include the top two predicted events for *FW1*. We choose the two

events with the highest predicted instance count, normalized by the average instance count for that event. Although we could expand the number of predicted events beyond two, we believe that the simplicity afforded by this heuristic will include the most salient missing context versus the complexity of determining which predicted events to include.

Our model achieved 94.7% accuracy on  $X_{none}$  and 97.6% on  $X_{top2}$ . The 2.9% improvement in accuracy results in halving the error rate. Although the accuracy on  $X_{none}$  is quite good already, we must consider that the model is used to increase the productivity of security analysts. The median salary for an analyst is \$90.1K US dollars [171] and therefore making them individually more efficient is desirable.

## 8.5 Impact and Deployment

This section will illustrate how *Virtual Product* empowers MSSP customers to identify security incidents by adding additional context to make more confident incident response decisions. To provide concrete illustrations of this, we present two case studies and discuss other areas of expected impact. We then proceed to a discussion of our current efforts, and future plans to integrate *Virtual Product* into the infrastructure used by our Managed Security Services.

### 8.5.1 Case Studies and Impact

As in Table 8.1, in this section we present two additional real-world incidents and the event predictions identified by *Virtual Product* for these incidents as examples of its positive impact on the incident response process.

**Example 1.** One of our customers, whom we will call Alice, has an important server that is protected by many network security products, as shown in Table 8.10. What value is FirewallB providing? Let us imagine that FirewallB is not deployed. Alice observes several suspicious events output from the deployed products. FirewallA detects an HTTP beacon from the HiKit exploit kit and the proxy also detects visits to suspicious websites.

Product	Event Description
<i>Seen Indicators (security events)</i>	
Proxy	Suspicious connection
FirewallA	WebVPN Authentication Rejected
FirewallA	WebVPN session created
FirewallA	WebVPN session terminated
FirewallA	WebVPN session deleted
FirewallA	WebVPN session started
FirewallA	WebVPN Authentication success
FirewallA	SSL handshake completed
FirewallA	Teardown TCP connection
FirewallA	TCP connection
FirewallA	Session disconnected
IPS	SQL Query in HTTP Request
IPS	RookIE/1.0 malicious user-agent string
IPS	Angler exploit kit exploit download attempt
IPS	Known malicious user agent - mozilla
IPS	HiKit initial HTTP beacon
IPS	TeamViewer remote administration tool outbound connection attempt
Router	Flow session close
<i>Top Predicted Primary Indicators</i>	
FirewallB	Windows Executable
FirewallB	Malicious File
FirewallB	SQL Injection Attempt
FirewallB	Phishing Webpage
FirewallB	RIG Exploit Kit
FirewallB	Windows DLL
FirewallB	Heartbleed Malformed OpenSSL Heartbeat
FirewallB	Microsoft Indexing Service UTF-7 Cross-Site Scripting Vulnerability
FirewallB	<del>Microsoft IIS HTR Request Parsing Buffer Overflow Vulnerability</del>
FirewallB	/etc/passwd Access Attempt

Table 8.10: Virtual Product correctly predicts that FirewallB would have detected an incident, and 10 of its top 11 predicted alerts coincide with the one that actually occurred, yielding a clearer picture of the artifacts involved in the attack and the vulnerabilities used. The incorrect prediction is shown in strikeout font.

Product	Event Description
<i>Seen Indicators (security events)</i>	
Firewall	Bad TCP Header length
Firewall	P2P Outbound GNUTella client request
Firewall	wu-ftp bad file completion attempt
Firewall	DNS zone transfer via TCP detected
Firewall	SNMP possible reconnaissance, private access udp
Firewall	ICMP PATH MTU denial of service attempt
Firewall	FTP format string attempt
Firewall	SMTP expn root
Firewall	SMTP vrfy root
Firewall	Server netcat (nc.exe) attempt
Firewall	philboard.admin.asp auth bypass attempt
Firewall	SSLv2 Challenge Length overflow attempt
Firewall	OpenSSL KEY_ARG buffer overflow attempt
Firewall	proxystylesheet arbitrary arbitrary command attempt
Firewall	Oracle ONE JSP src-code disclosure attempt
Firewall	JBoss admin-console access
Firewall	RevSlider information disclosure attempt
Firewall	Accellion FTA arbitrary file read attempt
Firewall	Apache Tomcat directory traversal attempt
Firewall	Apache non-SSL conn. to SSL port DoS attempt
Firewall	Windows NAT helper components tcp DoS attempt
Firewall	Multiple SQL injection attempts
Firewall	Bash CGI environment variable inject attempt
Firewall	Suspicious .tk dns query
Firewall	Suspicious .pw dns query
Firewall	ColdFusion admin interface access attempt
Firewall	Windows Terminal server RDP attempt
Firewall	Suspicious DNS request for 360safe.com
Gateway	Connectra Request Accepted
Gateway	ICMP: Timestamp Request
Gateway	Possible IP spoof
Router	Admin Authentication Failed
<i>Top Predicted Primary Indicators</i>	
AV	CVE-2012-4933 ZENWorks Asset Mgmt Exploit
AV	Post-Compromise PHP Shell Command Execution
AV	CVE-2015-1635 OS attack, HTTP.sys Remote Code Execution Exploit

Table 8.11: An attack on a webserver is obviously underway, but was it successful? Virtual Product correctly predicts, with 99.9% confidence, that not only a deployed AV product would detect attacks on the machine, but predict successful infection of the system.

Product	Event Description
<i>Seen Indicators (security events)</i>	
Firewall	Microsoft Windows 98 User-Agent string
Firewall	SMTP: Attempted response buffer overflow
Windows	Encrypted data recovery policy was changed.
Windows	A cryptographic self test was performed.
Windows	Cryptographic operation.
Windows	MSI Installer
Windows	Key file operation.
Windows	A logon was attempted using explicit credentials.
Windows	An attempt was made to reset an account's password.
Windows	Special privileges assigned to new logon.
Windows	System audit policy was changed.
Windows	A user account was changed.
Windows	A security-enabled local group was changed.
Windows	An account failed to logon
Proxy	TCP Cache Miss: Non-Cacheable Object
Gateway	Connectra Request Accepted
<i>Top Predicted Primary Indicators</i>	
AV	Bloodhound.Exploit.170

Table 8.12: There are indications of possible ransomware activity, but how did the attack appear on the machine in the first place? Virtual Product correctly indicates that a malicious spreadsheet (detected as Bloodhound.Exploit.170) was at fault, a method by which the Locky RansomWare has been known to propagate.

No incident was generated by these security products, indicating that without the evidence from FirewallB, the remaining events are insufficiently threatening to warrant attention. Based on evidence from the “virtual” FirewallB, however, Alice finds that there is likely an incident, with 95% confidence.

To further understand the cause of the potential incident, Alice takes a deeper look at FirewallB’s predicted events, which include malicious Windows executables, SQL injection attempts, a visit to a phishing webpage, and attacks on several recognized vulnerabilities. This additional telemetry gives Alice clarity on the used avenues of attack, which she can use to prioritize patching updates to prevent a recurrence of the attack. It also suggests possible data leaks through SQL injection and visits to phishing websites, enabling Alice to take action that could prevent a serious data breach.

For this particular incident, 11 events were triggered by the actual FirewallB product, and we list the top 11 reconstructed events identified by *Virtual Product*. These predictions are prioritized by dividing the events’ reconstructed instance count by the average instance count for that event, which is akin to TF-IDF normalization in statistical language model. In actual deployment, *Virtual Product* users can customize its confidence thresholds based on whether they wish *Virtual Product* to provide only highly confident event reconstructions or a broader list that is more likely to include erroneous predictions, but that may include valuable information that would otherwise have been suppressed.

**Example 2.** In some cases, while existing security events may make it quite obvious that an attack has taken place, they may leave a vital question unanswered, *Was the attack successful?*. This is a vital question, since most web servers are constantly exposed to attacks, and yet most attacks do not succeed in compromising the machine, both because the machine is often not vulnerable to the attempted attack, and because the network devices that report attack events are often able to block them. Table 8.11 illustrates such an example, in which Virtual Product is able to determine that an AV product would have detected a serious incident with 99.9% probability. The reconstructed AV events further indicate that

the attack is very likely to have been successful, and they give further insight into the nature of the predicted attack.

**Example 3.** Virtual Product is often able to provide context that outlines appropriate remediative and preventative actions. In the product events seen in Table 8.12, an observant analyst may see hints of a possible Ransomware attack, but the initial method of attack is not clear. Virtual Product correctly indicates that a malicious spreadsheet was at fault, a method by which the Locky Ransomware has been known to propagate, and therefore, reveals a possible social engineering campaign that the company's security department should investigate.

As is evident in these three case studies, and in the case study shown in Table 8.1, *Virtual Product* helps security analyst by providing context that helps them answer vital questions, such as: Is this machine compromised or just displaying unusual behavior? Was the attack that I see on this machine successful? How should I go about cleaning up this infected machine? How can I prevent a recurrence of a similar attack on this or other machines in my environment? By answering these questions for MSSP customers, Virtual Product significantly facilitates the security analyst's core tasks.

### 8.5.2 Deployment

We are currently working towards delivering an initial version of this technology to our Managed Security Services Product (MSSP), which will run on the Amazon Web Services platform. At present, we process data in batches, because telemetry data is uploaded every 15 minutes from our Security Operations Centers.

To integrate virtual product into the existing customer interface, we both introduce entirely new security incidents that are identified on the basis of Virtual Product's missing signal detection, and enrich existing incidents with additional context from virtual products. Our current system is a hybrid of components that are coupled with services that publish and subscribe to various streaming pipelines. Because of the flexibility that will be afforded by

cloud platforms, we will schedule and provision resources to perform matrix completion and will leverage the existing pipelines for incident generation and enrichment. The interactions that customers and MSS analysts have with *Virtual Product* will be fully captured, as at present, allowing us to tune the parameters of our algorithm.

## **8.6 Conclusions and Discussion**

We have presented *Virtual Product*, a novel technology that allows us to predict events from devices that are not currently deployed. Our evaluation shows that *Virtual Product* can significantly improve our ability to detect incidents. The business value of *Virtual Product* affects multiple levels of the enterprise. Cybersecurity analysts can leverage *Virtual Product* to enrich events coming from machines to make a better determination if and what kind of attack is being conducted. For security officers, *Virtual Product* empowers these decision makers to make informed purchasing decisions based on the additive value of potential products. If this technology is broadly adopted, it could create pressure on security product vendors to focus more on differentiation through actual capability and not through naming conventions. Future applications of this technology include providing product recommendations to our customers, particularly if we can perform “attack forecasting” to identify the likely attacks a customer would experience and how well they are defended and detected by existing products.

## **CHAPTER 9**

### **FIREBIRD: PREDICTING FIRE RISK AND PRIORITIZING FIRE INSPECTIONS IN ATLANTA**

The Atlanta Fire Rescue Department (AFRD), like many municipal fire departments, actively works to reduce fire risk by inspecting commercial properties for potential hazards and fire code violations. However, AFRD's fire inspection practices relied on tradition and intuition, with no existing data-driven process for prioritizing fire inspections or identifying new properties requiring inspection. In collaboration with AFRD, we developed the *Firebird* framework to help municipal fire departments identify and prioritize commercial property fire inspections, using machine learning, geocoding, and information visualization. Firebird computes fire risk scores for over 5,000 buildings in the city, with true positive rates of up to 71% in predicting fires. It has identified 6,096 new potential commercial properties to inspect, based on AFRD's criteria for inspection. Furthermore, through an interactive map, Firebird integrates and visualizes fire incidents, property information and risk scores to help AFRD make informed decisions about fire inspections. Firebird has already begun to make positive impact at both local and national levels. It is improving AFRD's inspection processes and Atlanta residents' safety, and was highlighted by National Fire Protection Association (NFPA) as a best practice for using data to inform fire inspections.

#### **9.1 Introduction**

In 2014 alone, there were 494,000 structure fires in the United States, causing 2,800 civilian deaths and \$9.8 billion in property damage [172]. Municipal fire departments, as the Authority Having Jurisdiction (AHJ), are responsible for enforcing applicable fire codes to reduce the risk of structure fires. The City of Atlanta Fire Rescue Department (AFRD), like many other fire departments, conducts regular commercial property inspections to ensure that

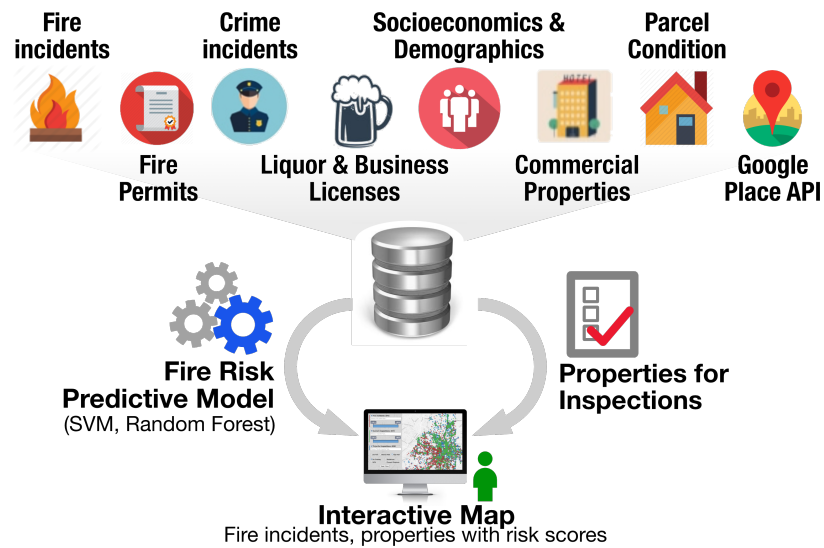


Figure 9.1: **Firebird Framework Overview.** By combining 8 datasets, Firebird identifies new commercial properties for fire inspections. Its fire risk predictive models (SVM, random forest) and interactive map help AFRD prioritize fire inspections and personnel allocation.

they comply with the city’s Code of Ordinances [173] for fire prevention and safety. With an annual average of nearly 650 structure fires and 2,573 annual commercial inspections, the AFRD Community Risk Reduction Section wanted to both identify uninspected properties and ensure that the properties being inspected were those at greatest risk of fire. Through a partnership between the City of Atlanta and the Data Science for Social Good (Atlanta) program, our research team developed the *Firebird* framework for identifying and prioritizing property fire inspections, based on fire department criteria and historical fire risk, tackling two important challenges:

**Challenge 1: Property Identification.** The AFRD Community Risk Reduction Section knew that the 2,573 annually inspected commercial properties were not all of the commercial properties in the city of Atlanta, but they did not have a way to obtain a more complete list of commercial properties that potentially needed inspection. The existing process for AFRD’s property inspections involved a legacy system of paper file records and inspections conducted on the basis of pre-existing permits, without a robust process for identification, selection, and prioritization of new properties to inspect. In addition, the variety of data sources AFRD had compiled to inform their inspections were inconsistent, incomplete,

and were often at different levels of granularity. Thus, cleaning and merging the datasets to identify which inspectable properties in the city had fallen through the cracks required significant effort. By integrating data from a variety of government and commercial sources, we discovered 19,397 potential new commercial properties to inspect, based on the property usage types that the Atlanta Code of Ordinances specifies require inspection.

**Challenge 2: Fire Risk Prediction.** Because 19,397 new commercial property inspections is far greater than the current number of annual commercial property inspections, and far more than AFRD’s current staff of fire inspectors can reasonably inspect, we developed a method to prioritize those inspections based on their fire risk. First, we created a joined dataset of building- and parcel-level information variables, for 8,223 commercial properties<sup>1</sup>. Then, we built predictive models of fire risk using machine learning approaches, including Support Vector Machine (SVM) [174] and Random Forest [175]. These models achieve true positive rates (TPRs) of up to 71.36% (in predicting fires) at a false positive rate (FPR) of 20%. As our most important goal is to save lives, a higher TPR outweighs the increase in FPR. The resulting fire risk scores were then assigned to over 5,000 commercial properties to help AFRD prioritize inspections.

**Contributions & Impact.** With Firebird, AFRD can now use data about historical fires to inform their fire inspections and more efficiently utilize their inspection personnel capacity. The challenges that Firebird addresses are not unique to AFRD or the City of Atlanta; many municipal agencies across the country work to integrate a variety of data sources to inform decision-making at all levels of governance. Specifically, many fire safety departments are seeking effective prioritization of property inspections and allocation of inspection resources, given limited inspection personnel and large numbers of inspectable properties. Firebird has already begun to improve AFRD’s inspection processes. Its major

---

<sup>1</sup>We will be referring to *buildings* and *properties* as two distinct concepts throughout this chapter. The AFRD conducts property inspections and issues permits to the owners of those “inspectable spaces,” which are properties. However, it is the physical structure of buildings that catch fire, and thus, when we built predictive models, we did so with information about the buildings themselves. This is significant because one property may contain multiple buildings, while another building may contain multiple properties.

contributions include:

- **Discovering new properties.** Firebird improves the safety of Atlanta residents and visitors by identifying 19,397 previously unidentified inspectable commercial properties.
- **Predictive fire risk model.** Firebird correctly predicts more than 70% of commercial fires (at 20% FPR), and applies the resulting fire risk scores to over 5,000 properties to help ARFD prioritize inspections.
- **Impact to Atlanta: Firebird at work.** Through an interactive map, Firebird integrates and visualizes fire incidents, property information and inspections, and risk scores to inform the decision-making processes of AFRD fire inspectors, executive staff, and their Community Risk Reduction Section for inspection prioritization and inspection personnel allocation.
- **National impact: reusable end-to-end framework for inspection prioritization.** Firebird provides an explicated model for other municipalities and agencies to use to identify new properties and prioritize commercial property inspections based on fire risk. This project was highlighted by the *National Fire Protection Association* (NFPA) at the *Smart Enforcement Workshop* for fire service professionals across North America as a best practice for using data to inform fire inspections.

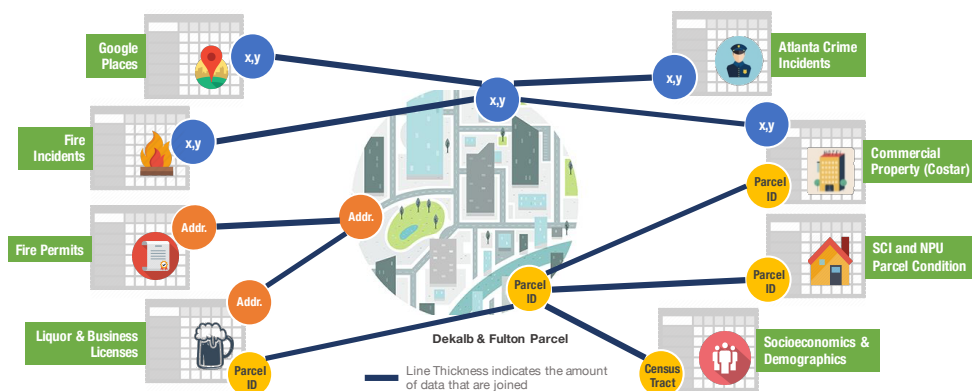


Figure 9.2: Joining eight datasets using three spatial information types (geocode, address, parcel ID).

## 9.2 Related Work

Risk prediction models have been widely used in many domains, including health care [176], student performance evaluation [177], and accounting fraud detection [178]. However, urban fire risk prediction has received relatively less attention, despite its obvious importance.

**Forest fire prediction.** Much of the prior work on data-driven fire risk prediction has targeted woodland and forest fires, such as in Italy [179], Greece [180], and Portugal [181]. They used different methods, such as neural networks [181], fuzzy algebra [180], and decision trees [179] to support the allocation of firefighting, fire prevention, and foliage recuperation resources to the areas of highest fire risk. The features they used, such as soil type and topography, are very different from the ones typically used in urban fire prediction like construction material and property usage type.

**Community-level urban fire prediction.** Prior work in data-driven urban fire risk prediction tends to work at the region or community level [182, 183], rather than the property- or building-level, which is the unit that the Atlanta fire inspectors are assigned to inspect. For instance, [182] undertook a randomized controlled trial of community fire risk education efforts, targeting high-risk residential communities. However, their method for identifying the high-risk areas was to create a point-distribution map of residential structure fires and draw ellipses to capture the areas of densest concentration of fire incidents. A more methodologically rigorous approach, as seen in [183]’s work on optimizing smoke-alarm inspections, joins data from the American Community Survey and American Housing Survey to predict municipal blocks most likely to have homes without functioning smoke alarms, using a Random Forest. Our work similarly uses publicly available datasets to predict properties most likely to be in need of inspection, but differs in that we offer a fire risk prediction score for individual commercial properties, rather than municipal residential blocks.

**Property-level urban fire prediction.** There is limited work on predicting fire risk at

the property or building level. In British Columbia, [184] developed a risk-based model for determining the frequency of commercial property fire inspections, using static and dynamic building-level characteristics. They scored each property by its level of compliance on prior inspections and by a set of risk metric components such as building classification, age, and presence of sprinklers. However, as they acknowledge, the weights and selection of those components were based on their fire code, and not on historical data on features that were highly predictive of fire, such as we utilize in our work.

The nearest precedent for our research with AFRD is the recent work from the New York Mayor’s Office of Data Analytics (MODA) with the Fire Department of New York (FDNY) to build a “Risk-Based Inspection System” (RBIS) [185]. They built a data-driven model to identify structures at greatest fire risk, to better prioritize FDNY’s inspection process, using a set of structural and behavioral information about those properties. However, due to a lack of detailed information on their technical approach, it is unclear how it may apply to AFRD’s scenario.

In both the FDNY RBIS initiative and our work with AFRD, a key challenge emerged: the difficulty of joining disparate datasets about commercial properties, gathered from various city departments without a shared convention for building ID numbers, consistent address formats, or strict internal quality control practices to ensure the datasets are accurate and up-to-date. We differ from [185] and [184] by providing a clear method for identifying new inspectable commercial properties that the fire department is not already aware of. Further, our work goes beyond [185] by presenting a detailed comparison of the performance of several machine learning algorithms for predicting the fire risk of commercial properties, and by incorporating them into an interactive GIS visualization for use by the AFRD fire inspectors and Community Risk Reduction Section, following [186].

### 9.3 Data Description

An essential step before identifying and prioritizing potential properties to inspect is to join the data about commercial properties from multiple sources. This was done to construct as complete a picture as possible for the properties in Atlanta needing inspection, as required by the Atlanta Code of Ordinances. After the data joining, we identified 19,397 new potential commercial properties to inspect, through a process of property discovery that utilized AFRD and City of Atlanta fire code criteria. See Table 9.1 for a summary of the different lists of total commercial property inspections and commercial buildings we will be referring to throughout this chapter.

Name	Count
<b>Current</b> annual inspections	2,573
<b>Long list</b> of potential new inspections <sup>2</sup>	19,397
<b>Short list</b> of potential new inspections	6,096
<b>Current</b> + <b>short list</b> inspections	8,669
<b>Current</b> + <b>short list</b> inspections with risk score	5,022
Properties for building predictive model	8,223

Table 9.1: Summary of inspection and building lists

#### 9.3.1 Data Sources

Firebird uses data from multiple sources, as tabulated in Table 9.2. AFRD provided us with a dataset of 2,543 historical fire incidents from July 2011 to March 2015, of which 34.3% were commercial fires. This includes information about fire incidents, such as time, location, type, and cause of fire. AFRD also provided a dataset of fire inspections, with 32,488 inspection permit records from 2012 to 2015. The inspection data includes information such as inspected property types, address, and time of inspections. We also obtained structural information about commercial properties from a dataset purchased by AFRD from the

<sup>2</sup>We provided AFRD with two lists of potential properties: one longer list that was the most extensive that we could provide, and another shorter list that was more manageable to display on a map, refined using the most frequently inspected property usage types.

Source	Name	Description
Atlanta Fire Rescue Department	Fire Incidents	Fire incidents from 2011 - 201
	Fire Permits	All permits filed by AFRD in 2012-2015
City of Atlanta	Parcel	Basic information for each parcel in Atlanta
	Strategic Community Investigation	Information regarding parcel conditions
	Business Licenses	All the business licenses issued in Atlanta
Atlanta Police Department	Crime	2014 crime in Atlanta
	Liquor Licenses	All filed liquor licenses by Police Department
Atlanta Regional Commission	Neighborhood Planning Unit	Boundary data for each Atlanta neighborhood
U.S. Census Bureau	Demographic	Household number, population by race and age
	Socioeconomic	Household median income
CoStar Group, Inc	CoStar Properties	Commercial property information
Google Place APIs	Google Place	Information regarding places from Google Maps

Table 9.2: Firebird Data Sources Summary

CoStar Group, a commercial real estate agency. This dataset includes building-level features such as year built, building material, number of floors and units, building condition and other information. A total of 8,223 commercial properties are documented by the CoStar Group in the City of Atlanta.

While CoStar offers building-level information, parcel data from Atlanta's Office of Buildings provides parcel-level information, such as property value, square footage, address, and other information about each parcel (a unit of land surrounding building(s)). The business license dataset obtained from the City of Atlanta's Office of Revenue provides information about businesses that own commercial properties. The business licenses dataset has 20,020 records with over 20 features including business type, business name, address, owner, etc. For non-business commercial properties (e.g., schools, churches, daycare

centers), we obtained such data from Google Places API and State of Georgia Government.

To offer more information about properties for building a predictive risk model, we also obtained socioeconomic and demographic data from the U.S. Census Bureau, liquor license and 2014 crime data from the Atlanta Police Department, and Certificate of Occupancy (CO) data from the Atlanta Office of Buildings. All of these data sources contributed to discovering new inspections and developing our predictive model for commercial fire risk estimation.

### 9.3.2 Data Joining

A critical step of this study was to join different datasets together so that data from different sources about the same building or property could be unified to create the most complete picture of a given property. For instance, by joining fire incident and commercial property data together, we can obtain a general idea regarding which commercial properties caught fire in the past five years. Furthermore, by joining commercial property data with data from the commercial real estate reports like the CoStar Group or the SCI Report, we can generate a more comprehensive view regarding specific characteristics of buildings, such as the structure and parcel condition, and even vacancy information.

We joined the datasets together based primarily on spatial location information. There are three types of spatial or location information in our datasets: longitude and latitude, address information, and the parcel identification number, which is a unique ID number created by Fulton and DeKalb county<sup>3</sup> for tax purposes. We then performed a location join based on the above three types of location information. The variety of spatial information types, and our method for joining them is illustrated in Figure 9.2. One obstacle we encountered was that spatial information had different formatting standards across the datasets. For example, the addresses from the CoStar Group were all in lowercase, with road names abbreviated instead

---

<sup>3</sup>The City of Atlanta is comprised of two separate counties, Fulton and Dekalb. Although both county governments provided building information, their parcel ID numbering schemes were not consistent. Thus, building information had to be joined using addresses and coordinates.

of fully spelled out, while datasets from the multiple departments of the City of Atlanta tend to use a more consistent address format. Therefore, a spatial information cleaning process was conducted before joining the datasets directly. The address location information from different datasets was first validated using Google Geocoding API. The API can auto-correct some misspellings of address information. After validation, addresses were then reformatted using US Postal Service's address validation API. The coordinate information was processed in ESRI ArcGIS software to filter out data points falling outside of the City of Atlanta. The cleaned datasets were then joined together based on the formatted addresses from the USPS API and the coordinate information from ArcGIS.

#### **9.4 Identifying New Properties Needing Inspection**

To discover new properties, we first needed to understand what types of properties currently required fire inspections according to the Fire Code [173], and we then identified other similar properties. In the current fire inspection permit dataset, we found more than 100 unique occupancy usage types, such as restaurants, motor vehicle repair facilities, textile storage, schools, children's day care centers, etc. To identify other similar commercial properties, we joined the list of currently inspected properties with the Atlanta Business License data by matching both the spatial location (identified through the joining process explained in Section 3.2) and the business name.

We discovered that, in addition to the 2,573 currently inspected properties, there were approximately 19,397 properties of the same occupancy usage types as the city's current inspections. For instance, the Fire Code of Ordinances [173] stipulates that motor vehicle repair facilities require inspection, due to the presence of flammable or combustible materials. However, only 186 of a total of 507 of those facilities in the city were on the list of current annual property inspections, suggesting that many or all of 321 remaining facilities should be inspected. However, because some occupancy types, such as "miscellaneous business service," may have many properties that are not actually required for inspection, we created

a shorter, more refined list of 6,096 new potential property inspections (instead of the 19,397 mentioned above), including only the top 100 most frequently inspected property usage types. We discovered these properties from a variety of data sources, including the Atlanta Department of Revenue’s Business License dataset, the liquor license dataset from the Atlanta Police Department, the Georgia Department of Education’s child care and preschool database, and Google Places API. We used the Google Places API to supplement the other datasets primarily because it provided more up-to-date information about some of our most commonly inspected property types, such as restaurants, bars, nightclubs, schools, churches, gas stations, etc, and because it proved especially useful for discovering properties that required inspection, but were not in the Business License dataset as they did not belong to any “business” category (e.g., churches). Google Places API served as a “bridge” to cross-check properties from different datasets, increasing the accuracy of our property discovery process.

In identifying new properties needing inspection, the most challenging part was to determine how buildings with different names (or IDs, or address formats) in various datasets actually refer to the same building. We had to ensure that properties on our new inspectable property list were unique and not already on the list of currently inspected properties, after the aforementioned datasets were joined together. Different approaches were attempted to ensure the uniqueness and novelty of properties on our potential list. The most reliable and efficient method was found to be joining different datasets in pairs using geocoding and approximate (“fuzzy”) string matching to approximately match both the business name and the address. We used Google Maps Geocoding API for geocoding and a Python library [187] to match the strings based on the edit distance. From the joined dataset, a final property list was extracted that contained information from all the available data sources.

## 9.5 Predictive Model of Fire Risk

However, 19,397 new properties (or even the shorter list of 6,096) is far more than AFRD is able to add to their annual property inspections, and not all of those properties are likely to need inspection at the same priority. We therefore created a predictive model to generate a fire risk score based on the building- and parcel-level characteristics of properties that had fire incidents in the last five years. This model was built using the scikit-learn machine learning package in Python [188]. The model uses 58 independent variables to predict fire as an outcome variable for each property.

### 9.5.1 Data Cleaning

After joining various datasets together to obtain building- and parcel-level information, significant data cleaning still needed to occur. The bulk of the data cleaning process involved finding the extent of the missing data and deciding how to deal with that missingness. Our missingness procedures were designed to minimize deletion of properties with missing data, because a significant number of the properties in our model had NA values (not available) for many variables (such as the structure condition of a building, which is only known if the building was inspected by the CoStar Group before). For each property with missing data for a particular feature, we replaced missing values with 0 when appropriate. We also included a binary feature indicating whether each property had missing data for each feature. We used log transformation for variables with a large numerical range, such as the “for sale” price of properties.

### 9.5.2 Feature Selection

After merging datasets, we had a total of 252 variables for each property. We manually examined each variable to determine whether it may be relevant to fire prediction, and excluded many obviously non-predictive variables in this initial process (such as the phone

number of the property owner, or property ID numbers). We then used forward and backward feature selection processes to determine each variable’s contribution to the model, and removed the variables that did not contribute to higher predictive accuracy. Our final model includes only 58 variables. We then expanded categorical variables into binary features. For example, the zip code variable was expanded into 37 binary features, and for each property only one zip code was coded as 1 (all zip codes were designated as 0 if a property’s zip code data was missing). After expansion, we had 1127 features in total.

### 9.5.3 Evaluation of the Models

We chose to validate our model using a time-partitioned approach. A fire risk model would ideally be tested in practice by predicting which properties would have a fire incident in the following year, and then waiting a year to verify which properties actually did catch fire. Because we wanted to effectively evaluate the accuracy of our model without waiting a year to collect data on new fires, we simulated this approach by using data from fire incidents in July 2011 to March 2014 as training data to predict fires in the last year of our data, April 2014 to March 2015.

We used grid search with 10-fold cross validation on the training dataset to select the best models and parameters. The models we tried included *Logistic Regression* [189], *Gradient Boosting* [190], *Support Vector Machine* (SVM) [174], and *Random Forest* [175]. SVM and Random Forest performed the best, with comparable performances (see Table 9.3). For SVM, the best configuration is using RBF kernel with  $C = 0.5$  and  $\gamma = \frac{10}{\text{\#features}}$ . For Random Forest, restricting the maximum depth of each tree to be 10 gave the best performance. Increasing the number of trees in general improves the performance, but we only used 200 trees since adding more trees only obtained insignificant improvement.

We then trained SVM and Random Forest on the whole training set using the best parameters and generated predictions on the testing set. Note that training and testing sets include the same set of properties, but different labels correspond to fires in different periods

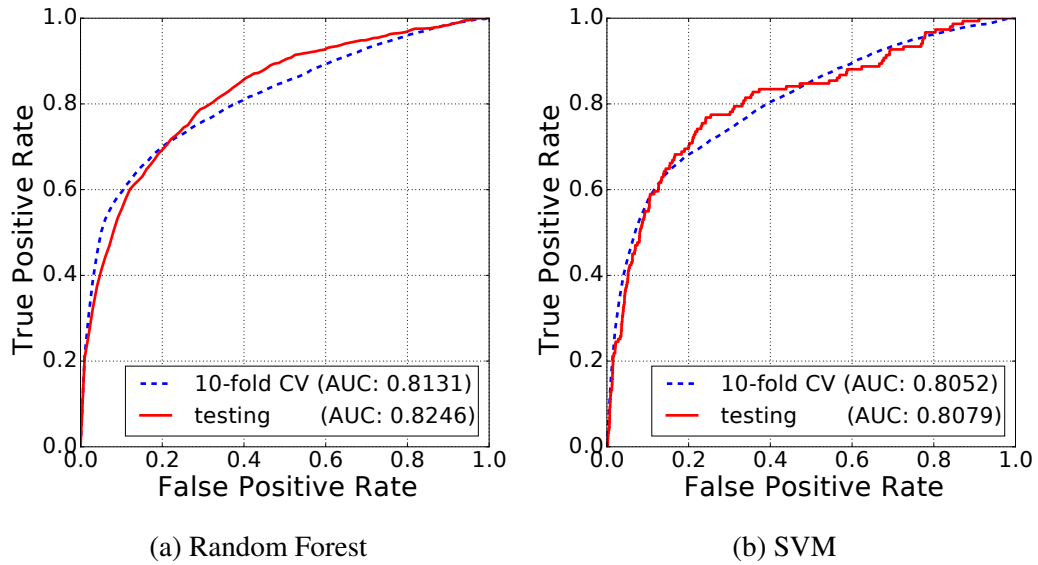


Figure 9.3: ROC curves of Random Forest and SVM

of time. This is a valid approach because we didn't use information that we would only know after the training period, i.e., fires in 2015.

The ROC curves for the training and testing performances are shown in Figure 9.3. All the results are averaged over 10 trials. The most important metric in this case is the true positive rate (TPR), i.e., how many fires were correctly predicted as positive in our model. The SVM model was able to predict 71.36% of the fires in 2014-2015, at a false positive rate (FPR) of 20%, which was deemed practically useful for AFRD — potential to save lives (by achieving a higher TPR) significantly outweighs the increase in FPR. At the same time, a high FPR facilitates more inspections of risky buildings, which is also beneficial. In practice, AFRD can adjust the TPR/FPR ratio to match their risk aversity and inspection capacity. The Random Forest model achieved a slightly lower TPR of 69.28% at the same FPR, but had a higher area under the ROC curve (AUC). Considering how few fires occur (only about 6% of the properties in our total dataset had fires), these results are much more predictive than guessing by chance.

False positives (FPs) provide important information to AFRD. As our testing period was the final year in our dataset, it is possible that some of those FP properties may actually catch fire in the near future. These properties share many characteristics with those that did

catch fire, and should likely be inspected by AFRD.

#### 9.5.4 Further Discussion of the Models

In this section, we discuss some insight we obtained while conducting the experiments. First, there is a mismatch between the meaning of labels in the training and testing datasets. The training labels represent fires that happened in a relatively long period of time, whereas the testing labels represent fires in a single year. One way to address this issue would be to expand each properties into multiple examples, one for each year. Each example is then a properties for a particular year, and the corresponding label indicates whether there was a fire in that year. Using this approach, however, did not improve the performance in our experiments. The reason is that most of our variables are static, such as floor size and zip code, and only a few variables are time-dependent, such as the age of the building and the time since last inspection. Therefore, expanding the properties only gives us many similar examples. However, this approach would potentially be helpful after collecting other dynamic information in the future, such as violations of health codes, sanitation ordinances, or other information from relevant city agencies.

Another important issue is whether the performance of predicting fires is consistent in different testing time periods. To test this, we tried different time windows for training, and for each window, we evaluated its prediction performance for the subsequent year. For each time window, we repeated the process described in Section 9.5.3, including grid search and cross validation, and finally used the best model to predict fires in the following year. The results are shown in Table 9.3. The performances decrease slightly for shorter training periods. This is due to fewer positive training examples, especially in the period of 2011-2012, which only consists of eight months of data (July 2011 to March 2012). However, this is still significantly better than guessing by chance, which demonstrates that we were not just “lucky” in predicting fires for a particular year.

Finally, it is helpful for us and for AFRD to know which features are the most effective

Training window	Testing AUC of the following year	
	Random Forest	SVM
2011-2012	0.7624	0.7614
2011-2013	0.8030	0.7914
2011-2014	0.8246	0.8079

Table 9.3: Testing AUC of each year

predictors. The Random Forest model presents a natural way to evaluate feature importance: for each decision tree in the Random Forest, the importance of a feature is calculated by the ratio of examples split by it. The final importance is then averaged among all trees. The top ten most predictive features are displayed in Table 9.4. Collectively, they capture the intuitive insight that buildings of a larger size or those containing more units (thus more people) would have higher probability of catching fire, and those of higher appraised value and higher taxes would have a lower probability of catching fire. The impact of higher appraised property value may be due to more developed fire prevention practices or infrastructure, but this hypothesis has not been empirically validated.

We also tried logistic regression, a linear model, to estimate each feature’s importance based on the corresponding weight coefficient in the model. We found that the top features in the logistic regression were very different from the ones in Random Forest. All were binary features indicating either a particular neighborhood or property owner. Some neighborhoods have either very high or low fire rates, and logistic regression tends to assign large positive or negative weights to them, respectively. However, since each of these features is only good at predicting a small number of properties within a certain area but does not predict well on the overall data, they are not chosen in the first few iterations of a decision tree.

#### 9.5.5 Assignment of Risk Scores

After we built the predictive model, we then applied the fire risk scores of each property to the list of current and potential inspectable properties, so that AFRD could focus on inspecting the properties most at risk of fire. To do this, we first computed the raw output

Top 10 features	
1	floor size
2	land area
3	number of units
4	appraised value
5	number of buildings
6	total taxes
7	property type is multi-family
8	lot size
9	number of living units
10	percent leased

Table 9.4: Top-10 features in Random Forest

of our predictive model for the list of properties we used to train and test the model. This generated a score between 0 and 1, which we then mapped to the discrete range of 1 to 10 that is easier for our AFRD colleagues to work with. Then, based on visual examination of the clustering of risk scores, we categorized the scores into low risk (1), medium risk (2-5), and high risk (6-10). These risk categorizations were intended to assign a manageable amount of medium risk ( $N = 402$ ) and high risk properties ( $N = 69$ ) for AFRD to prioritize.

We then needed to find out which of the properties with risk scores were in the lists of 2,573 current annually inspected properties and 6,096 potentially inspectable properties. Because of the lack of a consistent property ID across the various datasets used to develop the risk model, the currently inspected and potentially inspectable properties were spatially joined with the properties in the risk model, based on their geo-coordinates or addresses. After joining, we were able to assign risk scores to 5,022 of the 8,669 total commercial properties on the inspection list (both currently inspected [2,573] and potentially inspectable [6,096]).

## **9.6 Impact On AFRD and Atlanta**

### 9.6.1 Previous Inspection Process

Our goal in developing the Firebird framework was to help the Atlanta Fire Rescue Department (AFRD) and other municipal fire departments improve their identification and prioritization of commercial property inspections. Before considering the impact our work had on that process, it is important to first describe the previous process of commercial fire inspections in Atlanta. First, fire inspectors at AFRD received a list of properties to inspect every month, which had been inspected during that same month in the previous year. The existing process for adding new commercial properties to the list of required inspections was extremely ad hoc, without a formal notification process from other city departments when new buildings were built or occupied, or new businesses registered. It was largely the responsibility of individual fire inspectors to notice new inspectable properties and initiate an inspection process while driving to another inspection site. Moreover, there was no formal process used by the inspectors to prioritize their monthly inspections based on risk, or even to schedule their daily inspections based on proximity to other inspections. In other words, it is very possible that an inspector could return to the same business complex multiple times throughout the year conducting inspections on adjacent properties, which is not the most effective use of municipal resources. In addition, at present, the City of Atlanta Code of Ordinances does not specify the frequency of inspections based upon risk or other factors. As a result, inspections are effectively binary; regardless of potential fire risk, a property either gets an annual inspection in the same month every year, or it is unlikely to be inspected at all.

### 9.6.2 Technology Transfer to AFRD

After developing the Firebird framework, we first provided AFRD's executive staff and Community Risk Reduction Section with a dataset of all commercial properties in Atlanta

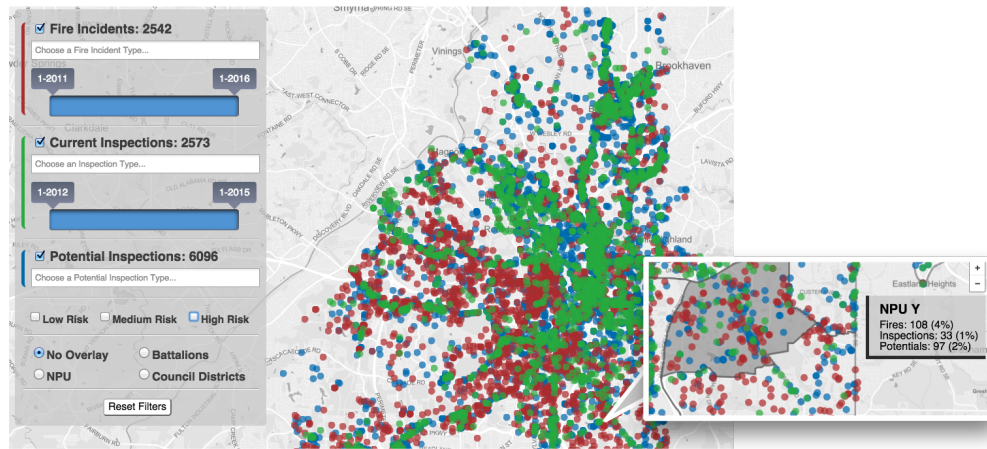


Figure 9.4: **Interactive map of fires and inspections.** The colored circles on the map represent fire incidents, currently inspected properties, and potentially inspectable properties in red, green, and blue, respectively. Inspectors can filter the displayed properties based on property usage type, date of fire or inspection, and fire risk score. *Callout:* activating the Neighborhood Planning Unit overlay allows an inspector to mouse-over a political subdivision of the city to view its aggregate and percentage of the fires, inspections, and potential inspections.

that fit their criteria for inspection. This included a shorter list of 6,096 new inspectable properties which are of the top 100 currently inspected property usage types (e.g., restaurants, motor vehicle repair facilities, etc), and a longer list of all commercial properties (19,397) that fit any property usage type that had been previously inspected. This dataset included the associated building- and parcel-level information for those properties in the form of a CSV file, with a subset of those properties (5,022) assigned a fire risk score. We then provided AFRD with an interactive map-based visualization tool, as part of the Firebird framework, for the fire inspectors and AFRD executive staff to use to augment their policy and decision-making processes. The map in Figure 9.4 was made using the open source map-making tools Mapbox and Leaflet to create the base map layer. Then, using the Javascript visualization library D3.js, we displayed differently colored circles on the map to represent fire incidents, currently inspected properties, and potentially inspectable properties in red, green, and blue, respectively, using their longitude and latitude coordinates.

We also built a user interface for the Firebird map developed through discussions with the AFRD Assessment and Planning Section, and refined by incorporating feedback from fire

inspectors and AFRD executive staff. The map includes an information panel for displaying property information when hovering over a property on the map, such as its business name, address, occupancy usage type, date since fire incident or inspection, and fire risk score, if available. The map also includes a user interface panel with the ability to filter the fire incidents, the currently inspected, and the potentially inspectable properties according to their property usage type, the date of fire incident or inspection, and their risk score. Finally, we incorporated a set of regional overlays requested by the AFRD executive staff, including the AFRD battalions, and the Atlanta Neighborhood Planning Units (NPU) and Council Districts, which are both political subdivisions of the city. We included dynamically updated counts and percentages for the displayed fire incidents, current inspections, and potential inspections for each regional overlay (Figure 4), so the AFRD executive staff could make decisions at a battalion, NPU, or Council District level.

This map, and the Firebird framework in general, could be used as a powerful tool for supporting data-driven conversations about personnel and resource allocation and inspection decisions, and may even be used to inform decisions regarding community education programs for fire safety and prevention. To ensure AFRD could update the risk model and property visualization as new fire incident and inspection data becomes available, we shared our source code and process with AFRD's Assessment and Planning Section, and made it publicly available on Github.<sup>4</sup>

### 9.6.3 Impact on AFRD Processes

After receiving the dataset of properties needing inspection, prioritized according to their fire risk score, AFRD has begun integrating the results of the analytics into their fire inspection process. Increasing the number of annual inspections by 6,096 (237%) overnight was not feasible without significant changes in organizational processes, local ordinances, or increased staffing. As an initial effort, AFRD assigned the 69 high-risk properties to the

---

<sup>4</sup><https://github.com/DSSG-Firebird>

inspectors covering those respective areas. Of those, 27 had current or out-of-date fire safety permits that required re-inspection, 13 properties required new permits, and 15 properties recently went out of business. The remaining properties were found to not require a fire safety permit. Most significantly, the inspectors assigned to review these properties found a total of 48 violations that needed to be addressed to meet the Fire Code. As AFRD continues working through the list of potentially inspectable properties in descending risk order, the sheer number of additional inspections (increased workload) and potential violations identified and mitigated (positive outcomes) has already had a transformative impact on the daily operations of AFRD's fire inspection process.

In addition to the immediate impact on the daily property inspections, the results of this work have stimulated important conversations within the executive leadership of AFRD and the Assessment and Planning Section about 1) how to more effectively allocate inspection personnel; 2) how to update and utilize the model to provide dynamic risk data in real time (e.g., on a monthly basis when new inspection assignments are given to the inspectors); 3) how to motivate increased data sharing between various government departments such as the Office of Buildings and AFRD; 4) how to give teams of firefighters access to fire safety permit and violation information when they respond to a fire emergency at that commercial property; and 5) how to extend the risk prioritization to residential properties using more behavioral data such as noise or sanitation ordinance violations, and consumer data from companies like Experian or ESRI.

Though there are many more inspectable properties than AFRD currently has the personnel capacity to handle, AFRD has already begun to take steps toward a more efficient use of their existing personnel, by discussing how to assign inspectors to regions with a higher proportion of properties requiring inspection, rather than by the geographical assignment to fire battalions currently in use. In addition, they have begun to discuss altering properties' inspection frequencies to reflect their fire risk levels. By prioritizing future inspections and more efficiently allocating inspection personnel to target the commercial properties most at

risk of fire, we hope this work will lead to a reduction in the frequency and severity of fire incidents in Atlanta. We also hope that this framework can be instructive for other municipal fire departments to improve their fire inspection processes.

## **9.7 Challenges**

As fire departments in many municipalities embark on more data-driven fire risk inspection policies and practices, several challenges we encountered could prove instructive for others. As with many governmental data science initiatives, the practical application of the predictive model has been contingent on local politics, organizational inertia, and existing policies, or what [191] calls the “politics of place.” While AFRD was an active and engaged partner throughout this project, securing access to clean and usable data proved a challenge. At the time of this writing, the City of Atlanta’s Office of Buildings, Office of Housing, and AFRD do not share a unified database of buildings or a shared building identification numbering convention, and thus, the process of joining various datasets was more technologically difficult than it might otherwise have been. Even the seemingly simple task of discovering which properties in the Office of Building’s dataset were also in the AFRD dataset required a rather elaborate process of fuzzy text-matching and address verification. In addition, our ability to leverage regularly updated dynamic data was similarly hindered by the difficulty of data sharing among city departments. For instance, the Office of Building’s Business License database may very well be updated regularly, but without a pipeline in place for those updated data to be used by AFRD, businesses may close without AFRD knowing, causing inspectors to waste time attempting to inspect closed businesses. These challenges could be mitigated if each department with a vested interest in municipal commercial properties and structures worked more closely to share their data and information.

Entrenched organizational processes may similarly hinder the adoption of new methods of identifying properties to inspect. While, in theory, the fire inspection process targets properties that are required by city ordinance to have a fire safety permit (e.g., restaurants

over a certain capacity, auto repair facilities, etc.), in practice there is no existing organizational procedure at AFRD and in many other cities to systematically add properties to the list of regular inspections, or to determine their frequency of inspection [184]. In this work, while we have created and employed an innovative method of identifying new properties to inspect, until inter-departmental data-sharing becomes widespread, this process would need to be redone on a regular basis as new businesses open, close, or change usage type. Further, though we created an interactive map for visualization of various types of property inspections, such a tool has not previously been part of AFRD fire inspectors' regular workflow, and this novelty presents a barrier for adoption, as seen in [186]. Finally, with the number of inspectable properties increasing by up to 237%, there is no clear incentive for the fire inspectors to work more efficiently to increase their individual number of property inspections per month.

From a policy standpoint, the existing municipal Fire Code in Atlanta [173] requires that inspections occur regularly for a specified set of commercial property types, but it is not clear that those property types require inspection with equal priority or frequency, or that these property types are the most in need of inspection. After using the results of this work for determining individual property fire risk, the AFRD should begin a conversation about how best to revise their municipal fire code to reflect differences in inspection type, priority, and frequency due to the fire risk associated with various property types. Prior work in [184] has similarly suggested revisions of the British Columbian fire code from being a reactive, inflexible document based on tradition and intuition to a data-driven, responsive, and pro-active document that incorporates information about fire risk. Finally, AFRD's policies for property fire inspections are primarily geared towards commercial properties, yet the majority of the fires in Atlanta occur in residential properties. This will require additional rethinking of their residential community fire safety and prevention education programs.

As in many municipalities, fully leveraging the power of analytics to improve fire

safety in Atlanta will require a significant rethinking of how to approach and manage city operations such as fire inspections, and how to best facilitate data sharing practices between different city agencies.

## **9.8 Conclusions and Future Research Directions**

Due to the large number of commercial properties in Atlanta potentially requiring inspection and the limited inspection personnel capacity of the Atlanta Fire Rescue Department (AFRD), as in many other municipalities, there is a need for a data-driven prioritization of commercial property inspections. In this work, we provide the Firebird framework: a re-usable method for municipal fire departments to identify and prioritize their commercial property fire inspections based on each property's fire risk. Our work first provides a clear process for joining disparate data sources from multiple municipal departments and private sources to identify new inspectable properties based on currently inspected property types. We were able to identify 6,096 new inspectable properties, comprised of the top 100 property types currently inspected by the AFRD, and a total of 19,397 new inspectable properties comprised of all currently inspected property types by AFRD.

We next present a method for predicting fire risk for each commercial property. Our models used 58 building- and parcel-level variables to predict fires in 8,223 properties, 5,022 of which are on the list of properties requiring inspection. Specifically, we trained SVM and Random Forest models using data from 2011-2014 to predict fires in 2015. At a false positive rate of 20%, the SVM and Random Forest models were able to predict 71.36% and 69.28% of the fires in that year, respectively. Furthermore, even the false positives provided valuable insight, since they represent properties with high risk of catching fire, that likely should be inspected by AFRD. We also identified features that are highly related to fires. From the Random Forest model, we learned that features related to building size, number of units, and value were most predictive. On the other hand, the logistic regression model revealed certain neighborhoods and property owners that associate with very high or low

fire rates. We then converted these results to a risk score for each property, and were able to apply these scores to 5,022 currently inspected and potentially inspectable properties (1,975 currently annually inspected and 3,047 potentially inspectable), with 454 of those properties having a medium or high risk score (188 currently inspected and 266 potentially inspectable properties). Finally, we incorporated those scores into our joined dataset of property inspections and visualized each of the properties on an interactive map, with their associated property information and risk score, for use by AFRD to augment their inspection decision processes.

**Research Directions.** Future research should seek to refine, expand, and further validate our prediction model. Due to missing or erroneous entries in the data sources, we could only incorporate 8,223 properties into our predictive fire risk model, out of the more than 20,000 commercial properties in the city. In addition, because of the lack of integration across city department datasets and a lack of completeness in many of our datasets, we could only provide risk scores for 5,022 of the 8,669 current and potentially inspectable properties in the dataset we provided to AFRD. Researchers working with other municipal fire departments might train their models on a dataset that has fewer building- or parcel-level information variables, but may be applicable to more properties. Other work could improve the accuracy of the model by incorporating additional dynamic sources of data, such as violations of prior fire inspections, data from the Department of Health and Wellness inspections, information from the Certificates of Occupancy, or other, more behavioral sources, such as sanitation or noise violations, as seen in [185], rather than the largely static building- and parcel-level data that we used. In addition, more research needs to be done on the usefulness and usability of an interactive map to display inspection, and how the inspectors or executive staff of a fire department could use it in different ways to inform their day-to-day planning, decisions, and operations. One step that municipal government agencies can take towards implementing this framework is to generate a unique Building Identification Number (BIN), used by all stakeholders, such as the Office of Buildings,

Office of Housing or city planning departments, as well as the Fire and Police Departments. This would allow for easier joining of disparate sources of data, without the need for address validation, text matching, and other complex and potentially error-generating processes for joining datasets.

Identification, selection, and prioritization of risky properties for fire inspection can be difficult for cities that do not have an integrated data platform, because some municipal agencies may have relevant property and structure information that is isolated from other local data sources, and which may not have a regular, timely process for updating. Our framework outlined here can be a model for improving the complex process of property inspection, identification, and prioritization. Additionally, our experience joining isolated datasets from different government departments, commercial data, and open data sources could be invaluable for many cities that want to begin utilizing data science for a smarter city, without requiring a significant financial investment. We hope the impact from our work may further promote the beneficial use of open public sector data, both in the city of Atlanta and elsewhere.

## CHAPTER 10

### CONCLUSIONS

While Artificial Intelligence has tremendous potential as a defense against real-world cybersecurity threats, understanding the capabilities and robustness of AI remains a fundamental challenge. This dissertation tackles problems essential to successful deployment of AI in security settings.

#### 10.1 Contributions

We contribute at the intersection of AI, cybersecurity, algorithmic game theory:

- **New Algorithms:** We developed *ShapeShifter* (Chapter 3), the first targeted physical adversarial attack that fools state-of-the-art object detectors. We also developed practical defenses including *SHIELD* (Chapter 4), an efficient defense leveraging stochastic image compression, and *UnMask* (Chapter 5), a knowledge-based adversarial detection and defense framework. Both *ShapeShifter* and *SHIELD* are open-source and have been integrated into an Intel AI Academy course. Our distributed boosting algorithm is simultaneously noise tolerant, communication efficient, and computationally efficient (Chapter 3).
- **New Theories:** We introduce a new online decision-making setting in game theory where players are compelled to play “diversified” strategies, and give strong guarantees on both the price of anarchy and the social welfare in this setting. (Chapter 6). Our distributed boosting algorithm requires exponentially less communication complexity in the agnostic setting, solving an open problem in distributed learning [9]. (Chapter 7)
- **New Applications:** Our *Virtual Product* framework has led to two patents and is the first method to predict security events and high-severity incidents identifiable by a

security product as if it had been deployed. Our *Firebird* framework computes fire risk scores for over 5,000 buildings in the city, with true positive rates of up to 71% in predicting fires. *Firebird* open sourced and has been used by the Atlanta Fire Rescue Department to prioritize fire inspections. *Firebird* won the Best Student Paper Award Runner-up at KDD 2016 and was highlighted by National Fire Protection Association as a best practice for using data to inform fire inspections.

## 10.2 Future Research Directions

This thesis takes an important step in designing practical, robust ML algorithms with strong theoretical guarantees, to reliably solve high-stakes societal problems, such as safe-guarding security-critical systems. Moving forward, I hope to broaden and deepen this investigation, extending my work to more theoretical frameworks and applications. Initially, I will focus on the following three interrelated research directions.

**Physical Attacks against Real ML Systems and Countermeasures.** To the best of our knowledge, there is no reported physical attack yet that affects ML systems in the real-world. This is because all the existing physical attacks, including *ShapeShifter*, only work in the white-box setting, where the attacker has full access to the ML model and all the deployed defensive techniques. To raise the awareness of the security issues in ML, I would like to design black-box physical attacks that break real ML systems, such as a security surveillance system or a smart home system. Depending on the applications, the perturbations may not need to be imperceptible and can be arbitrarily large. For example, the attacker can wear a t-shirt with a large color perturbation to fool an object detection system [192]. There are many possible methods to defend against such attacks in practice. One way is by utilizing multi-modal sensing, such as using RGB and depth sensing at the same time. How to efficiently incorporate multi-modal information is a challenging and important research problem.

**Fraud Detection with ML and Game Theory.** I believe AI has the potential to protect

people from a wide range of cyber harms that affect our everyday lives. For example, *fraud detection* is an important adversarial ML application, where the fraudster creates a benign facade to evade detection. I have worked on detecting fraudulent users and reviews on Yelp by using graph mining techniques like dense graph extraction [50]. However, there is much more information to be utilized to improve detection. I plan to combine techniques in ML, graph mining, natural language processing, and time series analysis to incorporate information from different data sources. I also plan to better understand how and why fraudsters work in particular ways, by using game theory, and ultimately design a framework that discourages people to conduct fraud by better mechanism design.

**Model and Data Privacy.** Keeping an ML model secret is crucial for defending against black-box adversarial attacks. I plan to design practical ML algorithms that are hard to reverse-engineer, such as dynamic random ensemble using *online boosting* [112]. I will also formalize the benefit of randomization as a defense to adversarial attacks by incorporating the techniques in cryptography. Besides model privacy, I also aim to study the problem of data privacy, which has been receiving an increasing amount of attention in the past few years, due to some high-profile data leaks in industry. *Differential privacy* is the most popular theoretical framework for data privacy, wherein a typical method is by adding random noise to the original data. I believe our *diversified strategy* concept (Chapter 6) is also helpful to preserving differential privacy.

## REFERENCES

- [1] U.S. Department of Homeland Security, *U.S. department of homeland security cybersecurity strategy*, [https://www.dhs.gov/sites/default/files/publications/DHS-Cybersecurity-Strategy\\_1.pdf](https://www.dhs.gov/sites/default/files/publications/DHS-Cybersecurity-Strategy_1.pdf), 2018.
- [2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *International Conference on Learning Representations*, 2014.
- [3] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in *International Conference on Learning Representations (Workshop)*, 2017.
- [4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *International Conference on Learning Representations*, 2018.
- [5] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, “Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector,” in *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018.
- [6] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, “Shield: Fast, practical defense and vaccination for deep learning using jpeg compression,” 2018.
- [7] C. J.J.J.S.-T.C.C.Y.E.D.H.P.E.Y.L.C.M.K.R.S.D.D.S.B.P.C.T.K.W. L. Nilaksh Das Siwei Li, “Mlsploit: A cloud-based framework for adversarial machine learning research,” in *KDD project showcase*, 2019.
- [8] M.-F. Balcan, A. Blum, and S.-T. Chen, “Diversified strategies for mitigating adversarial attacks in multiagent systems,” in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 407–415.
- [9] H. Daume III, J. Phillips, A. Saha, and S. Venkatasubramanian, “Protocols for learning classifiers on distributed data,” in *Artificial Intelligence and Statistics*, 2012, pp. 282–290.
- [10] S.-T. Chen, M.-F. Balcan, and D. H. Chau, “Communication efficient distributed agnostic boosting,” in *Artificial Intelligence and Statistics*, 2016, pp. 1299–1307.

- [11] M. Madaio, S.-T. Chen, O. L. Haimson, W. Zhang, X. Cheng, M. Hinds-Aldrich, D. H. Chau, and B. Dilkina, “Firebird: Predicting fire risk and prioritizing fire inspections in atlanta,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 185–194.
- [12] D. Angluin and P. Laird, “Learning from noisy examples,” *Machine Learning*, vol. 2, no. 4, pp. 343–370, 1988.
- [13] M. Kearns, “Efficient noise-tolerant learning from statistical queries,” *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 983–1006, 1998.
- [14] L. G. Valiant, “Learning disjunction of conjunctions..”
- [15] M. Kearns and M. Li, “Learning in the presence of malicious errors,” *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993.
- [16] A. T. Kalai, A. R. Klivans, Y. Mansour, and R. A. Servedio, “Agnostically learning halfspaces,” *SIAM Journal on Computing*, vol. 37, no. 6, pp. 1777–1805, 2008.
- [17] Y. Mansour and M. Parnas, “Learning conjunctions with noise under product distributions,” *Information Processing Letters*, vol. 68, no. 4, pp. 189–196, 1998.
- [18] P. Awasthi, M. F. Balcan, and P. M. Long, “The power of localization for efficiently learning linear separators with noise,” in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, ACM, 2014, pp. 449–458.
- [19] M. J. Kearns, R. E. Schapire, and L. M. Sellie, “Toward efficient agnostic learning,” *Machine Learning*, vol. 17, no. 2-3, pp. 115–141, 1994.
- [20] T. A. Meyer and B. Whateley, “SpamBayes: Effective open-source, Bayesian based, email classification system,” in *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.
- [21] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 34, 2004, pp. 219–230.
- [22] S. J. Stolfo, S. Hershkop, C.-W. Hu, W.-J. Li, O. Nimeskern, and K. Wang, “Behavior-based modeling and its application to email analysis,” *ACM Transactions on Internet Technology (TOIT)*, vol. 6, no. 2, pp. 187–221, 2006.
- [23] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, “Adversarial machine learning,” in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, ACM, 2011, pp. 43–58.

- [24] T. Gu, B. Dolan-Gavitt, and S. Garg, “Badnets: Identifying vulnerabilities in the machine learning model supply chain,” *arXiv preprint arXiv:1708.06733*, 2017.
- [25] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-22, 2018*, The Internet Society, 2018.
- [26] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *Proceedings of the 38th IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.
- [27] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *International Conference on Learning Representations*, 2015.
- [28] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: A simple and accurate method to fool deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.
- [29] S. M. Moosavi Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *CVPR*, 2017.
- [30] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 372–387.
- [31] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial perturbations against deep neural networks for malware classification,” *arXiv preprint arXiv:1606.04435*, 2016.
- [32] W. Hu and Y. Tan, “Generating adversarial malware examples for black-box attacks based on gan,” *arXiv preprint arXiv:1702.05983*, 2017.
- [33] N. Papernot, P. D. McDaniel, A. Swami, and R. E. Harang, “Crafting adversarial input sequences for recurrent neural networks,” in *2016 IEEE Military Communications Conference, MILCOM*, 2016, pp. 49–54.
- [34] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, “Tactics of adversarial attack on deep reinforcement learning agents,” *arXiv preprint arXiv:1703.06748*, 2017.
- [35] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial attacks on neural network policies,” *arXiv preprint arXiv:1702.02284*, 2017.

- [36] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, “Detecting adversarial samples from artifacts,” *arXiv preprint arXiv:1703.00410*, 2017.
- [37] K. Griffin, S. Schneider, X. Hu, and T.-C. Chiueh, “Automatic generation of string signatures for malware detection,” in *International workshop on recent advances in intrusion detection*, Springer, 2009, pp. 101–120.
- [38] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, “Static analyzer of vicious executables (save),” in *Computer Security Applications Conference, 2004. 20th Annual*, IEEE, 2004, pp. 326–334.
- [39] P. Beaucamps and É. Filiol, “On the possibility of practically obfuscating programs towards a unified perspective of code protection,” *Journal in Computer Virology*, vol. 3, no. 1, pp. 3–21, 2007.
- [40] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, “Automated classification and analysis of internet malware,” in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2007, pp. 178–197.
- [41] B. Anderson, C. Storlie, and T. Lane, “Improving malware classification: Bridging the static/dynamic gap,” in *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, ACM, 2012, pp. 3–14.
- [42] O. E. David and N. S. Netanyahu, “Deepsign: Deep learning for automatic malware signature generation and classification,” in *Neural Networks (IJCNN), 2015 International Joint Conference on*, IEEE, 2015, pp. 1–8.
- [43] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, “Deep learning for classification of malware system call sequences,” in *Australasian Joint Conference on Artificial Intelligence*, Springer, 2016, pp. 137–149.
- [44] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, “Polonium: Tera-scale graph mining and inference for malware detection,” in *Proceedings of the 2011 SIAM International Conference on Data Mining*, 2011, pp. 131–142.
- [45] A. Tamersoy, K. Roundy, and D. H. Chau, “Guilt by association: Large scale malware detection by mining file-relation graphs,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 1524–1533.
- [46] E. Biermann, E. Cloete, and L. M. Venter, “A comparison of intrusion detection systems,” *Computers & Security*, vol. 20, no. 8, pp. 676–683, 2001.

- [47] W. Lee, S. J. Stolfo, and K. W. Mok, “A data mining framework for building intrusion detection models,” in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, IEEE, 1999, pp. 120–132.
- [48] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan, “Using artificial anomalies to detect unknown and known network intrusions,” *Knowledge and Information Systems*, vol. 6, no. 5, pp. 507–527, 2004.
- [49] M. Rahman, B. Carbunar, J. Ballesteros, G. Burri, and D. H. Chau, “Turning the tide: Curbing deceptive yelp behaviors,” in *Proceedings of the 2014 SIAM International Conference on Data Mining*, SIAM, 2014, pp. 244–252.
- [50] P. Jain, S.-T. Chen, M. Azimpourkivi, D. H. Chau, and B. Carbunar, “Spotting suspicious reviews via (quasi-) clique extraction,” *arXiv preprint arXiv:1509.05935*, 2015.
- [51] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, “Botfinder: Finding bots in network traffic without deep packet inspection,” in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, ACM, 2012, pp. 349–360.
- [52] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, “Netprobe: A fast and scalable system for fraud detection in online auction networks,” in *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, pp. 201–210.
- [53] A. Tamersoy, B. Xie, S. L. Lenkey, B. R. Routledge, D. H. Chau, and S. B. Navathe, “Inside insider trading: Patterns & discoveries from a large scale exploratory analysis,” in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ACM, 2013, pp. 797–804.
- [54] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, “Feature engineering strategies for credit card fraud detection,” *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.
- [55] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, “No need to worry about adversarial examples in object detection in autonomous vehicles,” *arXiv:1707.03501*, 2017.
- [56] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. L. Yuille, “Adversarial examples for semantic segmentation and object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1378–1387.
- [57] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1528–1540.

- [58] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, “Robust physical-world attacks on machine learning models,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [59] J. Lu, H. Sibai, and E. Fabry, “Adversarial examples that fool detectors,” *arXiv:1712.02494*, 2017.
- [60] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [61] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno, “Physical adversarial examples for object detectors,” in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.
- [62] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6517–6525.
- [63] A. Athalye and I. Sutskever, “Synthesizing robust adversarial examples,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- [64] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, “Adversarial patch,” *arXiv preprint arXiv:1712.09665*, 2017.
- [65] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 12th ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.
- [66] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” in *International Conference on Learning Representations*, 2017.
- [67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.
- [68] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *Proceedings of the 13th European conference on computer vision*, 2014, pp. 740–755.
- [69] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern

- convolutional object detectors,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3296–3297.
- [70] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, “Darts: Deceiving autonomous cars with toxic signs,” *arXiv preprint arXiv:1802.06430*, 2018.
  - [71] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018.
  - [72] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *IEEE Symposium on Security and Privacy*, 2016, pp. 582–597.
  - [73] A. N. Bhagoji, D. Cullina, and P. Mittal, “Dimensionality reduction as a defense against evasion attacks on machine learning classifiers,” *arXiv preprint arXiv:1704.02654*, 2017.
  - [74] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, “On detecting adversarial perturbations,” in *ICLR*, 2017.
  - [75] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, “Countering adversarial images using input transformations,” *International Conference on Learning Representations*, 2018.
  - [76] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, D. Song, T. Kohno, A. Rahmati, A. Prakash, and F. Tramèr, “Note on attacking object detectors with adversarial stickers,” *arXiv:1712.08062*, 2017.
  - [77] R. Shin and D. Song, “Jpeg-resistant adversarial images,” *NIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.
  - [78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
  - [79] W. Xu, D. Evans, and Y. Qi, “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks,” in *Proceedings of the 2018 Network and Distributed Systems Security Symposium (NDSS)*, 2018.
  - [80] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, “Foveation-based mechanisms alleviate adversarial examples,” *arXiv preprint arXiv:1511.06292*, 2015.

- [81] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer, “Ensemble methods as a defense to adversarial perturbations against deep neural networks,” *arXiv preprint arXiv:1709.03423*, 2017.
- [82] S.-T. Chen, Y. Han, D. H. Chau, C. Gates, M. Hart, and K. A. Roundy, “Predicting cyber threats with virtual security products,” in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACM, 2017, pp. 189–199.
- [83] S. Gu and L. Rigazio, “Towards deep neural network architectures robust to adversarial examples,” *arXiv preprint arXiv:1412.5068*, 2014.
- [84] D. Krotov and J. J. Hopfield, “Dense associative memory is robust to adversarial inputs,” *arXiv preprint arXiv:1701.00939*, 2017.
- [85] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, “Improving network robustness against adversarial attacks with compact convolution,” *arXiv preprint arXiv:1712.00699*, 2017.
- [86] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, “A study of the effect of jpg compression on adversarial images,” *arXiv preprint arXiv:1608.00853*, 2016.
- [87] S.-T.C.L.C.-M. K. Cory Cornelius Nilaksh Das and D. H. Chau, “The efficacy of shield under different threat models,” *KDD 2019 Workshop on Learning and Mining for Cybersecurity (LEMINCS)*, 2019.
- [88] F. Hohman, N. Hodas, and D. H. Chau, “Shapeshop: Towards understanding deep learning representations via interactive experimentation,” in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, 2017, pp. 1694–1699.
- [89] A. Cabrera, F. Hohman, J. Lin, and D. H. Chau, “Interactive classification for deep learning interpretation,” *Demo, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [90] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in *International Conference on Learning Representations*, 2018.
- [91] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [92] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

- [93] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [94] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *IEEE Transactions on Evolutionary Computation*, 2019.
- [95] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” in *International Conference on Learning Representations*, 2018.
- [96] J. Rauber, W. Brendel, and M. Bethge, “Foolbox: A python toolbox to benchmark the robustness of machine learning models,” *arXiv preprint arXiv:1707.04131*, 2017.
- [97] W. Abdulla, *Mask r-cnn for object detection and instance segmentation on keras and tensorflow*, [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN), 2017.
- [98] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: A simple and accurate method to fool deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.
- [99] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, “Detect what you can: Detecting and representing objects using holistic models and body parts,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [100] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results*, <http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html>, Accessed: 2019-01-01.
- [101] T. Roughgarden, “Intrinsic robustness of the price of anarchy,” *J. ACM*, vol. 62, no. 5, 32:1–32:42, 2015.
- [102] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [103] A. Blum and Y. Mansour, “Learning, regret minimization, and equilibria,” in, N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Eds. Cambridge University Press, 2007, pp. 79–102.
- [104] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

- [105] N. Littlestone and M. K. Warmuth, “The weighted majority algorithm,” in *30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989, pp. 256–261.
- [106] S. Arora, E. Hazan, and S. Kale, “The multiplicative weights update method: A meta-algorithm and applications,” *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.
- [107] Y. Freund and R. E. Schapire, “Adaptive game playing using multiplicative weights,” *Games and Economic Behavior*, vol. 29, no. 1, pp. 79–103, 1999.
- [108] S.-T. Chen, M.-F. Balcan, and D. Chau, “Communication efficient distributed agnostic boosting,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, 2016, pp. 1299–1307.
- [109] R. Impagliazzo, “Hard-core distributions for somewhat hard problems,” in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, IEEE, 1995, pp. 538–545.
- [110] S. Arora, E. Hazan, and S. Kale, “Fast algorithms for approximate semidefinite programming using the multiplicative weights update method,” in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS’05)*, IEEE, 2005, pp. 339–348.
- [111] D. Gavinsky, “Optimally-smooth adaptive boosting and application to agnostic learning,” *J. Mach. Learn. Res.*, vol. 4, pp. 101–117, 2003.
- [112] S.-T. Chen, H.-T. Lin, and C.-J. Lu, “An online boosting algorithm with theoretical justifications,” in *Proceedings of ICML*, 2012, pp. 1007–1014.
- [113] I. Caragiannis, D. Kurokawa, and A. D. Procaccia, “Biased games,” in *AAAI*, 2014, pp. 609–615.
- [114] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, “Adversarial classification,” in *KDD*, 2004.
- [115] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *ICLR*, 2014.
- [116] M. Hein and M. Andriushchenko, “Formal guarantees on the robustness of a classifier against adversarial manipulation,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2263–2273.
- [117] H. Daumé, J. M. Phillips, A. Saha, and S. Venkatasubramanian, “Efficient protocols for distributed classification and optimization,” in *Proceedings of the 23rd Interna-*

- tional Conference on Algorithmic Learning Theory*, ser. ALT'12, 2012, pp. 154–168.
- [118] M.-F. Balcan, A. Blum, S. Fine, and Y. Mansour, “Distributed learning, communication complexity and privacy,” in *Proceedings of COLT*, 2012.
  - [119] M. Babaioff, R. Kleinberg, and C. H. Papadimitriou, “Congestion games with malicious players,” in *Proceedings of the 8th ACM conference on Electronic commerce*, ACM, 2007, pp. 103–112.
  - [120] M.-F. Balcan, A. Blum, and Y. Mansour, “The price of uncertainty,” in *Proceedings of the 10th ACM conference on Electronic commerce*, ACM, 2009, pp. 285–294.
  - [121] M.-F. Balcan, F. Constantin, and S. Ehrlich, “The snowball effect of uncertainty in potential games,” in *International Workshop on Internet and Network Economics*, Springer, 2011, pp. 1–12.
  - [122] M. Sion, “On general minimax theorems,” *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 171–176, 1958.
  - [123] M. Herbster and M. K. Warmuth, “Tracking the best linear predictor,” *J. Mach. Learn. Res.*, vol. 1, pp. 281–309, 2001.
  - [124] P. N. Panagopoulou and P. G. Spirakis, “Random bimatrix games are asymptotically easy to solve (a simple proof),” *Theory of Computing Systems*, vol. 54, no. 3, pp. 479–490, 2014.
  - [125] T. Roughgarden, “Routing games,” *Algorithmic game theory*, vol. 18, pp. 459–484, 2007.
  - [126] D. Braess, “Über ein paradoxon aus der verkehrsplanung,” *Unternehmensforschung*, vol. 12, pp. 258–268, 1968.
  - [127] R. W. Rosenthal, “A class of games possessing pure-strategy nash equilibria,” *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.
  - [128] M. Jordan and T. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
  - [129] M. Li, D. Andersen, A. Smola, and K. Yu, “Parameter server for distributed machine learning,” in *Proceedings of NIPS*, 2014.
  - [130] Y. Zhang, J. Duchi, M. Jordan, and M. Wainwright, “Information-theoretic lower bounds for distributed statistical estimation with communication constraints,” in *Proceedings of NIPS*, 2013.

- [131] Y. Zhang, J. C. Duchi, and M. Wainwright, “Communication-efficient algorithms for statistical optimization,” in *Proceedings of NIPS*, 2012.
- [132] A. Bellet, Y. Liang, A. B. Garakani, M.-F. Balcan, and F. Sha, “Distributed frank-wolfe algorithm: A unified framework for communication-efficient sparse learning,” in *Proceedings of SDM*, 2015.
- [133] M.-F. Balcan, S. Ehrlich, and Y. Liang, “Distributed k-means and k-median clustering on general communication topologies,” in *Proceedings of NIPS*, 2013.
- [134] M.-F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff, “Improved distributed principal component analysis,” in *Proceedings of NIPS*, 2014.
- [135] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
- [136] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, “Communication-efficient distributed dual coordinate ascent,” in *Proceedings of NIPS*, 2014, pp. 3068–3076.
- [137] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.
- [138] P. Long and R. A. Servedio, “Random classification noise defeats all convex potential boosters,” in *Proceedings of ICML*, ACM, 2008, pp. 608–615.
- [139] M. Kearns, R. Schapire, and L. Sellie, “Toward efficient agnostic learning,” *Machine Learning*, vol. 17, no. 2-3, pp. 115–141, 1994.
- [140] S. Ben-David, P. M. Long, and Y. Mansour, “Agnostic boosting,” in *Computational Learning Theory*, Springer, 2001, pp. 507–516.
- [141] V. Kanade and A. Kalai, “Potential-based agnostic boosting,” in *Advances in Neural Information Processing Systems 22*, 2009, pp. 880–888.
- [142] S. Kale, “Boosting and hard-core set constructions: A simplified approach,” *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 14, no. 131, 2007.
- [143] V. Feldman, “Distribution-specific agnostic boosting,” in *ICS*, 2010, pp. 241–250.
- [144] V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu, “Agnostic learning of monomials by halfspaces is hard,” in *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS ’09, 2009, pp. 385–394.

- [145] A. T. Kalai, Y. Mansour, and E. Verbin, “On agnostic boosting and parity learning,” in *Proceedings of STOC*, ACM, 2008, pp. 629–638.
- [146] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT press, 2012.
- [147] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. Cambridge University Press., 2014.
- [148] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, “Distributed newton methods for regularized logistic regression,” in *Proceedings of the PAKDD*, 2015, pp. 690–703.
- [149] W. Chu, S.-T. Park, T. Beaupre, N. Motgi, A. Phadke, S. Chakraborty, and J. Zachariah, “A case study of behavior-driven conjoint analysis on yahoo!: Front page today module,” in *Proceedings of KDD*, ACM, 2009, pp. 1097–1104.
- [150] D. Sahoo, C. Liu, and S. C. Hoi, “Malicious url detection using machine learning: A survey,” *arXiv preprint arXiv:1701.07179*, 2017.
- [151] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: Learning to detect malicious web sites from suspicious urls,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1245–1254.
- [152] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” in *2001 IEEE Symposium on Security and Privacy*, 2001, pp. 38–49.
- [153] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, “Large-scale malware classification using random projections and neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3422–3426.
- [154] Y. Ye, D. Wang, T. Li, and D. Ye, “Imds: Intelligent malware detection system,” in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2007, pp. 1043–1047.
- [155] K.R.Gabriel and S. Zamir, “Lower rank approximation of matrices by least squares with any choice of weights,” *Technometrics*, vol. 21, no. 4, pp. 489–498, 1979.
- [156] A. P. Singh and G. J. Gordon, “A unified view of matrix factorization models,” in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2008, pp. 358–373.
- [157] N. S. Nati and T. Jaakkola, “Weighted low-rank approximations,” in *In 20th International Conference on Machine Learning*, 2003, pp. 720–727.

- [158] I. S. Dhillon and S. Sra, “Generalized nonnegative matrix approximations with bregman divergences,” in *Neural Information Processing Systems (NIPS)*, 2005, pp. 283–290.
- [159] J. R. Nathan Srebro Nati and T. Jaakkola, “Maximum margin matrix factorizations,” in *Proceesings of Advances in Neural Information Processing Systems (NIPS)*, 2004.
- [160] S. Rendle and L. Schmidt-Thieme, “Online-updating regularized kernel matrix factorization models for large-scale recommender systems,” in *Proceedings of the 2008 ACM Conference on Recommender Systems*, Lausanne, Switzerland, 2008, pp. 251–258, ISBN: 978-1-60558-093-7.
- [161] A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2008, pp. 650–658.
- [162] T. L. Ding Chris and W. Peng, “Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic and a hybrid method,” in *Processings of the 21st AAAI Conference on Artificial Intelligence*, 2006.
- [163] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Neural Information Processing Systems (NIPS)*, MIT Press, 2000, pp. 556–562.
- [164] J. H. Deng Cai Xiaofei He and T. S. Huang, “Graph regularized nonnegative matrix factorization for data representation.,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.
- [165] F. Pereira and G. Gordon, “The support vector decomposition machine,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 689–696.
- [166] L. Z. Changhu Wang Shuicheng Yan and H. Zhang, “Non-negative semi-supervised learning,” in *Proceedings of the 12th AISTATS*, 2009, pp. 575–582.
- [167] A. N. Josif Grabocka and L. Schmidt-Thieme, “Classification of sparse time series via supervised matrix factorization,” in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012, pp. 928–934.
- [168] S. H.P. H. Yulei Niu Zhiwu Lu and J.-R. Wen, “Weakly supervised matrix factorization for noisily tagged image parsing,” in *Proceedings of International Joint Conference on Artificial Intelligence 2015*, 2015, pp. 3749–3755.
- [169] E. H. John Duchi and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

- [170] T. H.J.W.Z. H. Jijie Wang Lei Lin, “Efficient k-nearest neighbor join algorithms for high dimensional sparse data,” *arXiv:1011.2807*, 2010.
- [171] U. S. D. of Labor. (2017). Information security analysts, (visited on 02/17/2017).
- [172] *National Fire Protection Association. Fires in the U.S.* <http://www.nfpa.org/research/reports-and-statistics/fires-in-the-us>, Accessed: 2016-02-09.
- [173] *City of Atlanta. Code of ordinances: Fire prevention and protection*, [https://www.municode.com/library/codes\\_of\\_ordinances](https://www.municode.com/library/codes_of_ordinances), 2015.
- [174] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [175] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [176] D. Kansagara, H. Englander, A. Salanitro, D. Kagen, C. Theobald, M. Freeman, and S. Kripalani, “Risk prediction models for hospital readmission: A systematic review,” *Jama*, vol. 306, no. 15, pp. 1688–1698, 2011.
- [177] H. Lakkaraju, E. Aguiar, C. Shan, D. Miller, N. Bhanpuri, R. Ghani, and K. L. Addison, “A machine learning framework to identify students at risk of adverse academic outcomes,” in *KDD*, ACM, 2015, pp. 1909–1918.
- [178] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos, “Snare: A link analytic system for graph labeling and risk detection,” in *KDD*, ACM, 2009, pp. 1265–1274.
- [179] A. Lapucci, S. Lombardo, M. Petri, and A. Santucci, “A KDD based multicriteria decision making model for fire risk evaluation,” in *Conference on GIS-cience (AGILE)*, 2005.
- [180] L. S. Iliadis, “A decision support system applying an integrated fuzzy model for long-term forest fire risk estimation,” *Environmental Modelling & Software*, vol. 20, no. 5, pp. 613–621, 2005.
- [181] M. P. de Vasconcelos, S. Silva, M. Tome, M. Alvim, and J. C. Pereira, “Spatial prediction of fire ignition probabilities: Comparing logistic regression and neural networks,” *Photogrammetric engineering and remote sensing*, vol. 67, no. 1, pp. 73–81, 2001.
- [182] J. Clare, L. Garis, D. Plecas, and C. Jennings, “Reduced frequency and severity of residential fires following delivery of fire prevention education by on-duty fire

- fighters: Cluster randomized controlled study,” *Journal of safety research*, vol. 43, no. 2, pp. 123–128, 2012.
- [183] M. DaCosta, J. Krinsley, and B. Abelson, “Optimizing local smoke alarm inspections with federal data,” *Bloomberg Data for Good Exchange*, 2015.
  - [184] L. Garis and J. Clare, “A dynamic risk–based framework for redesigning the scheduling of fire safety inspections,” 2014.
  - [185] E. Copeland, “Big data in the big apple,” *Capital City Foundation.*, 2015.
  - [186] L. Dobrică, T. Ionescu, L. Dobrică, and S. Elena, “Solutions based on gis technology in components of urban management applications,” *U.P.B Scientific Bulletin*, 2010.
  - [187] J. Gonzalez, *Fuzzywuzzy*, <https://github.com/seatgeek/fuzzywuzzy>, 2015.
  - [188] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *JMLR*, vol. 12, pp. 2825–2830, 2011.
  - [189] J. A. Nelder and R. J. Baker, “Generalized linear models,” *Encyclopedia of Statistical Sciences*, 1972.
  - [190] J. H. Friedman, “Stochastic gradient boosting,” *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.
  - [191] J. B. Carr and K. LeRoux, “Which local governments cooperate on public safety?: Lessons from michigan,” *Working Group on Interlocal Services Cooperation*, p. 4, 2005.
  - [192] C. Cornelius, S.-T. Chen, J. Martin, and D. H. Chau, “Talk proposal: Towards the realistic evaluation of evasion attacks using carla,” *Dependable and Secure Machine Learning*, 2019.