UNDERSTANDING, FORTIFYING AND DEMOCRATIZING AI SECURITY

A Dissertation
Presented to
The Academic Faculty

By

Nilaksh Das

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology

May 2022

UNDERSTANDING, FORTIFYING AND DEMOCRATIZING AI SECURITY

Thesis committee:

Dr. Duen Horng Chau
School of Computational Science & Engg.
Georgia Institute of Technology

Dr. Srijan Kumar
School of Computational Science & Engg.
Georgia Institute of Technology

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

Dr. Ponnurangam Kumaraguru
Dept. of Computer Science & Engg.
IIITH - International Institute of Information Technology, Hyderabad

Dr. Xu Chu
School of Computer Science
Georgia Institute of Technology

Dr. Oliver Brdiczka
Principal ML Scientist, Director
Adobe

Date approved: April 13, 2022

Absence of evidence is not evidence of absence.

Carl Sagan

To my parents, Ila and Asim,
for supporting me no matter what.

ACKNOWLEDGMENTS

Throughout my journey as a PhD student, I have met many incredible people who have helped shape my motivations, my goals, my way of scientific thinking and even my approach to life. This dissertation is dedicated to every such person. There is no possible way I could have finished this endeavour without any single one of you.

Before all else, I want to thank my amazing PhD advisor, Polo, who has treated me like family away from family throughout graduate school. I am thankful to him for reinvigorating my sense of scientific pursuit while I was still a Master's student back in 2016. When I was unsure if I was ready for such a commitment of long-term PhD research, Polo had immense confidence in me and constantly reassured me I could be successful as a PhD student. Now I can positively say that all of my success today is largely because of Polo's expert guidance and his relentless support for his students' research ideas. He has not only taught me how to solve difficult problems, but also shown me how to ask the right questions. Because of Polo's unique eye to identify such hidden talent in people, I am also very fortunate to have collaborated with some of the exceptionally brilliant students of Georgia Tech through the Polo Club of Data Science. Shang-Tse Chen and Fred Hohman were one of the first students to enthusiastically welcome me to the group and guide me in the initial years to shape my research aptitude. Here, I also want to thank my academic siblings Haekyu Park, Anthony Peng, Jay Wang, Scott Freitas, Rahul Duggal, Minsuk Kahng and Austin Wright for all your support in generating and working on some awesome research ideas with me. It was also an absolute pleasure to have worked with extraordinary students: Madhuri Shanbhogue, Evan Downing, Sivapriya Vellaichamy, Matthew Hull, Zhiyan Zhou, Jon Saad-Falcon, Siwei Li, Chanil Jeon, Jinho Jung, Carter Yagemann, Robert Turko, Omar Shaikh, Robert Firstman, Renzhi Wu, Pruthvi Perumalla and many more; all of whom had a strong role to play in the success of my research works.

I also greatly appreciate the sage academic advisement I have received over the years from Dr. Ponnurangam Kumaraguru, Dr Bistra Dilkina and Dr. Xu Chu, who have truly strengthened my research approach. I am also thankful to the members of my thesis committee, Dr. Wenke Lee, Dr. Xu Chu, Dr. Srijan Kumar, Dr. Ponnurangam Kumaraguru and Dr. Oliver Brdiczka, for providing your invaluable feedback in shaping my dissertation. I have also been very lucky to have worked with some remarkable people from the industry: Li Chen, Michael E. Kounavis, Anindya Paul, Cory Cornelius and Jason Martin from Intel Labs, and Monica Sunkara, Sundararajan Srinivasan, Sravan Bodapati and Katrin Kirchhoff from Amazon Science; thank you all for offering me the opportunity to work on some really impactful projects.

Now, I would be remiss without thanking my dearest friends, Chirag Dua, Aayush Kalra, Dr. Uttara Koul, Srishti Sharma, Madhulika Mukherjee, Siddharth Sharma, Anjana Pradeep, Sakshi Gandhi, Anisha Bhakri, Rishabh Bhardwaj and Meghna Natraj; all of whom have supported me at different points in my life in maintaining my sanity – all of you make my life fun! I am also very grateful to all my roommates over 7 years at Georgia Tech: Prithvijit Chattopadhyay, Samyak Datta, Ramakrishna Vedantam, Rohan Katyal, Anuj Sobti, Bhanu Verma, Satyajeet Gawas and Sarthak Ghosh; thank you all for putting up with my eccentric habits within the threshold of your domicile.

Speaking of a home, only one name comes to my mind now, Sanya, my companion for life. There are not enough words to express how grateful I am to you for taking this arduous journey with me. Our paths crossed at Georgia Tech right in the first semester of my PhD and your MS studies. Even after collaborating on a hectic research paper together, I cannot believe my luck that you still decided to date me. You had the courage to trust in my capabilities when I myself thought I was failing, and never let me take my eyes off the ball. No matter how much pressure you were under from your own work, you were ever ready to discuss research ideas with me. Through all the crazy paper deadlines, long nights of coding and the constant uncertainty, you always had an inspiriting smile and supportive words for me. This PhD is as much yours as is mine.

Finally, none of this would have been possible without the ardent support and the relentless encouragement of my parents, Ila Das and Dr. Asim Das, and my brother, Niloy Das. They have shown unwavering faith in my endeavors, and have made many personal sacrifices only to enable my dreams. I distinctly remember as a kid, telling my mother that I wanted to become an astronaut when I grew up. Instead of fancifully playing into my fantasy, she calmly told me that I would then need to work hard and get a PhD. That stuck with me over the years, and I may not have become an astronaut, but I took the plunge to explore something equally fascinating as space, and now I am finally here because of your wise words — this dissertation is dedicated to you, Ma.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This dissertation focuses on democratizing security analysis of AI systems by improving how people interpret attacks, quantify vulnerabilities, and protect AI from harm. Through developing a foundational security framework for AI, our work accelerates research innovations and increases education effectiveness by lowering the barriers to entry for people to learn, design, develop, and test AI security techniques. Our interactive visualization systems have significantly expanded the intuitive understanding of AI vulnerabilities. Our research has produced novel defenses that have been tech-transferred to industry. Our scalable AI security framework and research tools, becoming available to thousands of students, is transforming AI security education at scale.

AI is now increasingly powering high-stakes applications, such as self-driving vehicles, financial credit risk scoring and healthcare, which directly impact the safety, security and well-being of humans. However, an expanding body of adversarial ML research has revealed that AI models are highly vulnerable to malicious inputs, raising serious concerns about deploying AI in such critical applications. Nevertheless, most adversarial ML techniques often lack methods for interpretation which severely impairs people's understanding of AI vulnerabilities. Furthermore, adversarial ML research is still developing theoretically, with a dearth of strong, practical solutions to defend real-world AI cyber-systems. Additionally, the firehose of rapidly emerging adversarial ML research is creating huge barriers to entry for newcomers, practitioners and researchers in this field.

This thesis addresses these fundamental challenges through creating holistic interpretation techniques for better understanding of attacks and defenses, developing effective and principled defenses for protecting AI across input modalities, and building tools that enable scalable interactive experimentation with AI security and adversarial ML research. This thesis has a vision of enhancing trust in AI by making AI security more accessible and adversarial ML education more equitable, while focusing on three complementary research thrusts:

1. **Exposing AI Vulnerabilities through Visualization & Interpretable Representations.** We develop intuitive interpretation techniques for deciphering adversarial attacks.
2. **Mitigating Adversarial Examples Across Modalities & Tasks.** We develop robust defenses which are generalizable across diverse AI tasks and input modalities.
3. **Democratizing AI Security Research & Pedagogy with Scalable Interactive Experimentation.** We enable researchers, practitioners and students to perform in-depth security testing of AI models through interactive experimentation.

CHAPTER 1
INTRODUCTION

In our modern-day society, Artificial Intelligence (AI) powered by Machine Learning (ML) is rapidly permeating multiple facets of our daily lives. We now have AI-powered assistants in the palm of our hands, AI controlling and moving our stock markets, and even most of our modern cars are now equipped with some form of AI-assistive technologies. As we steadily move towards the AI-powered utopia that could only be imagined in lofty fiction in the recent past, a formidable threat is emerging that endangers such an acute capitalization of AI in everyday life.

A growing body of *adversarial ML* research has revealed that Deep Neural Networks (DNNs) — the workhorse of modern AI applications — are extremely vulnerable to *adversarial examples*. Adversarial examples are malicious inputs crafted by an attacker by adding small perturbations to benign inputs. These minuscule perturbations are imperceptible to humans; but they can completely confuse DNNs into making incorrect predictions. For example, an attacker can place a small adversarial sticker on a “Stop” sign on the roadside which may be ignored by human drivers, but it can make a self-driving car think there is a “Speed Limit” sign instead! Such revelations from adversarial ML research raise very serious concerns about the increasing use of AI for safety-critical applications.

A majority of people, those who use AI in their lives and even those who study ML, consider AI and DNN models to be “black-boxes”. Hence, for people to have completely restored faith in using AI applications, there is not only an urgent need to develop effective mitigation techniques to **defend real-world AI cyber-systems**; there is also an equally pressing necessity to enable people to **interpret AI vulnerabilities** and *understand* how and why the attacks and defenses work. As these solutions emerge from academia, it is extremely important that they are not walled off to be decoded only by expert researchers. After all, AI impacts people from all walks of life. Therefore, it is also critical that the technologies for **AI security be brought to the masses**, and AI security research be as accessible and as pervasive as AI itself.

This thesis aims to address these substantial challenges by developing new tools, techniques and paradigms that enhance people’s understanding of AI vulnerabilities, mitigate the threat of adversarial examples, and democratize AI security by making it accessible and available to everyone — researchers and non-researchers alike.

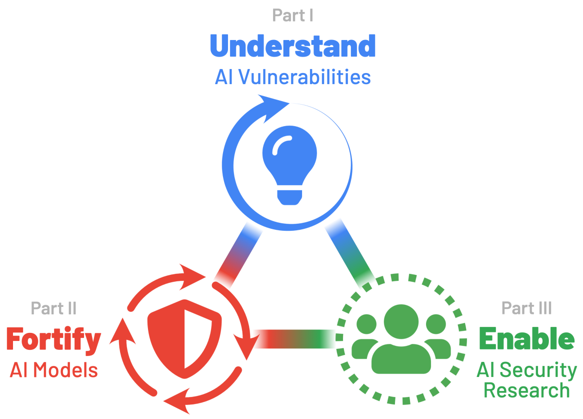


Figure 1.1: My research mission is to **enable** everyone to **understand** and **fortify** AI security.

1.1 THESIS VISION AND GOALS

The fundamental challenge of overcoming the threat posed by adversarial ML and enhancing people's trust in AI calls for (1) intuitive interpretation methods that expose AI vulnerabilities for everyone to understand; (2) effective attack mitigation techniques that restore a sense of safety and security; and (3) scalable, useable systems for in-depth AI security analysis that allows anyone to stress-test the AI applications used by people in their everyday lives.

In order to accomplish my vision of developing robust solutions that span the above three practical aspects of understanding, fortifying and democratizing AI security, I have identified three complementary research thrusts that I present in this dissertation. I refer to these research thrusts summarily as "**Understand**", "**Fortify**" and "**Enable**" (Figure 1.1). Further, these research thrusts map onto my larger research goals for this thesis:

I **Understand**: **Exposing AI Vulnerabilities through Visualization & Interpretable Representations.** My goal is to extricate meaningful, interpretable, semantic representations that can explain AI vulnerabilities, and develop visualization techniques that leverage these semantic representations for gaining a deeper understanding of how an adversarial attack is able to hijack the predictions made by ML models.

II **Fortify**: **Mitigating Adversarial Examples Across Modalities & Tasks.** Based on the understanding of vulnerabilities using semantic representations, my goal is to identify fundamentally unifying factors that can effectively mitigate the harm from adversarial attacks across a diverse range of AI tasks and input modalities.

III **Enable**: **Democratizing AI Security Research & Pedagogy with Scalable Interactive Experimentation.** My ultimate goal is to take AI security research to the masses in an interactive form factor, thus enabling students, practitioners and researchers to easily learn about adversarial ML techniques.

Table 1.1: The publications presented in this dissertation mapped to the thesis outline. Clicking on the PDF button will open a published PDF of the corresponding work. ∗ = equal contribution.

Part I: Understand **Exposing AI Vulnerabilities through Visualization & Interpretable Representations**
[Chapter 2] GOGGLES: Automatic Image Labeling with Affinity Coding. Nilaksh Das, Sanya Chaba, Renzhi Wu, Sakshi Gandhi, Polo Chau, Xu Chu. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020. PDF **[Chapter 3]** Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. Nilaksh Das∗, Haekyu Park∗, Zijie J. Wang, Fred Hohman, Robert Firstman, Emily Rogers, Polo Chau. *IEEE Visualization Conference*, 2020. PDF
Part II: Fortify **Mitigating Adversarial Examples Across Modalities & Tasks**
[Chapter 4] 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, Polo Chau. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018. PDF **[Chapter 5]** SkeleVision: Towards Adversarial Resiliency of Person Tracking with Multi-Task Learning. Nilaksh Das, Sheng-Yun Peng, Polo Chau. *Under peer review*, 2022. PDF **[Chapter 6]** Hear No Evil: Towards Adversarial Robustness of Automatic Speech Recognition via Multi-Task Learning. Nilaksh Das, Polo Chau. *Under peer review*, 2022. PDF
Part III: Enable **Democratizing AI Security Research & Pedagogy with Scalable Interactive Experimentation**
[Chapter 7] ADAGIO: Interactive Experimentation with Adversarial Attack and Defense for Audio. Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Li Chen, Michael E. Kounavis, Polo Chau. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018. PDF **[Chapter 8]** MLsploit: A Framework for Interactive Experimentation with Adversarial Machine Learning Research. Nilaksh Das, Siwei Li, Chanil Jeon, Jinho Jung∗, Shang-Tse Chen∗, Carter Yagemann∗, Evan Downing∗, Haekyu Park, Evan Yang, Li Chen, Michael Kounavis, Ravi Sahita, David Durham, Scott Buck, Polo Chau, Taesoo Kim, Wenke Lee. *Project Showcase at the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. PDF

1.2 THESIS OVERVIEW

This thesis focuses on three related research thrusts which are outlined in Table 1.1 along with the corresponding publications presented in this dissertation. While I am the principal author of these publications, this research would not have been possible without the support of my PhD advisor, Dr. Duen Horng (Polo) Chau, as well as the collaboration with several other mentors, colleagues and researchers. Hence, I would hereon refer to the first-person pronouns in plural (*e.g.*, “we” instead of “I”) to reflect everyone’s contributions. In this overview section, we will briefly discuss the works included in this dissertation.

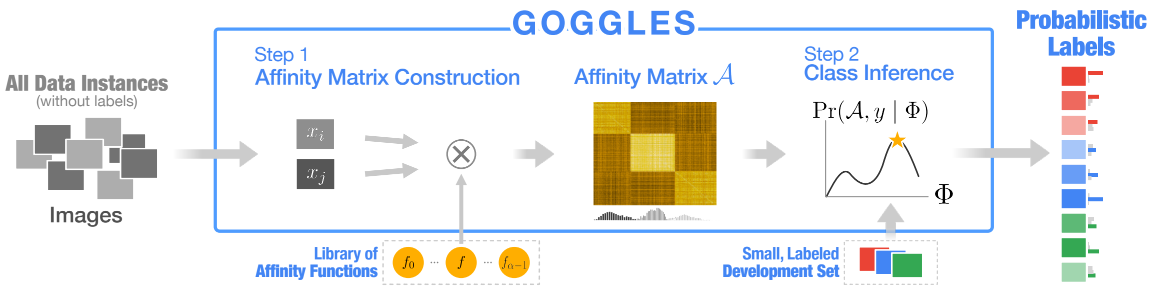

GOGGLES
All Data Instances
(without labels)
Images
Step 1
Affinity Matrix Construction
x_i
x_j
Affinity Matrix $\mathcal{A}$
Step 2
Class Inference
$\Pr(\mathcal{A}, y \mid \Phi)$
Φ
Probabilistic Labels
Library of Affinity Functions
f_0
f
$f_{\alpha-1}$
Small, Labeled Development Set

Figure 1.2: GOGGLES' novel affinity coding paradigm outperforms the state-of-the-art data programming technique Snuba by 23% and is only 7% away from the supervised learning benchmark.

1.2.1 Part I: Exposing AI Vulnerabilities (Understand)

DNNs are incredibly complex and are often thought of as black-box systems. For humans to confidently deploy secure AI systems, there is an urgent need to enable people to truly understand the vulnerabilities of the underlying DNN models and how the adversarial attacks and defenses work the way they do. A powerful notion in this context comes from the deep learning visualization and interpretability domain — humans can understand and trust systems more when they can dig deeper and intuitively visualize and interpret why and how things are working. In this first part of the thesis, we focus on developing new tools and techniques to help people *understand* the underlying vulnerabilities of ML models that are exploited by adversarial examples, expanding the body of visualization and interpretability research to the adversarial ML domain.

GOGGLES: Extracting & Leveraging Interpretable Semantic Prototypes (Chapter 2). The first step of our research plan is to *extract* interpretable units from within an ML model, so that we can leverage them as an explanation basis for the model's decisions. In this chapter on GOGGLES (Figure 1.2), we propose an innovative technique to extract such interpretable "semantic prototypes" from a pre-trained deep neural network. Through developing a novel data programming paradigm for assigning probabilistic labels to unannotated training data, we show that our extracted prototypes are semantic representations that possess significant predictive capacity that can explain how the model operates internally. GOGGLES achieves labeling accuracies ranging from a minimum of 71% to a maximum of 98% without requiring any extensive human annotation.

BLUFF: Understanding and Visualizing AI Vulnerabilities (Chapter 3). It is challenging to decipher an ML model's vulnerabilities since a model typically consist of millions of parameters. In order to understand these vulnerabilities, we need to *summarize* the model's internals in a meaningful way such that it exposes the parts exploited by the attacks. The next step of our research is to build upon the notion of interpretable prototypes from GOGGLES,

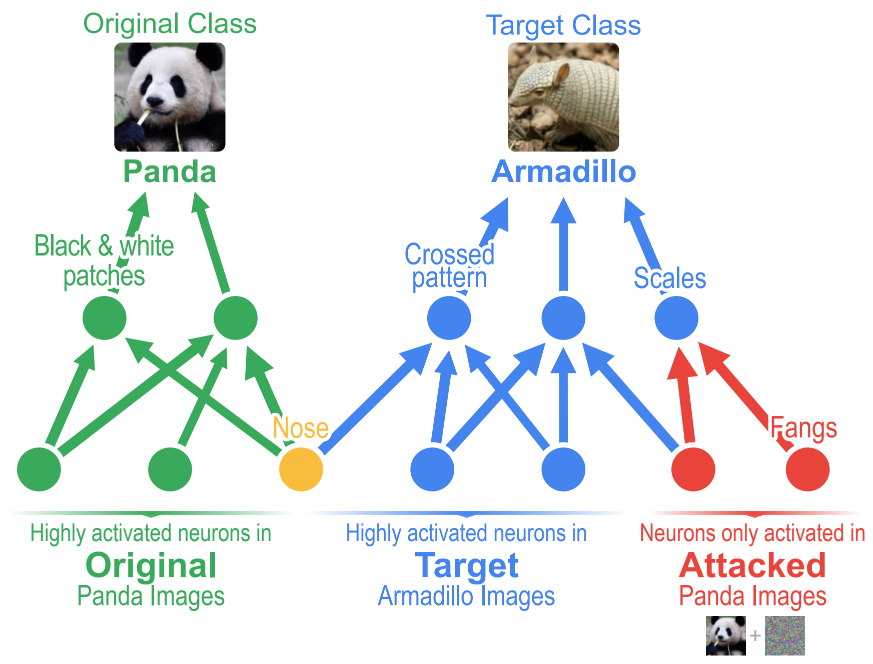

Original Class
Panda
Target Class
Armadillo
Black & white patches
Crossed pattern
Scales
Nose
Fangs
Highly activated neurons in Original Panda Images
Highly activated neurons in Target Armadillo Images
Neurons only activated in Attacked Panda Images
+

Figure 1.3: BLUFF reveals that adversarial attacks leverage confounding features (shown in red).

leveraging them to compute a deep neural network's "attribution graph" (Figure 1.3). The attribution graph constitutes a summary of how the activation values "flow" through the network from input to output. Putting it all together, we develop a scalable visualization tool, BLUFF, that allows researchers to interactively explore this attribution graph of a network under attack in order to discover its vulnerabilities. BLUFF reveals that adversarial attacks leverage "non-robust" and confounding intermediate features to confuse the model.

1.2.2 Part II: Mitigating Adversarial Examples (Fortify)

It is not enough to only expand our understanding of AI vulnerabilities. To truly overcome the threat posed by adversarial ML, we need to leverage this understanding and investigate unifying methods that can effectively mitigate adversarial examples across AI tasks. We do this by first focusing on the input stage of the model, leveraging practical pre-processing techniques to remove adversarial perturbations. Next, capitalizing on the AI security insights afforded by our interpretation and visualization techniques, we move on to the intermediate feature space of the model so as to regularize the model into learning "robust" features. Robust features can be thought of as intermediate representations that can be semantically meaningful to humans. Across a diverse range of AI tasks and input modalities, we uncover that the fundamental technique of **Multi-Task Learning** (MTL) is highly effective in mitigating adversarial examples. MTL stimulates ML models to learn a robust intermediate feature space by jointly training a shared backbone network on different tasks.

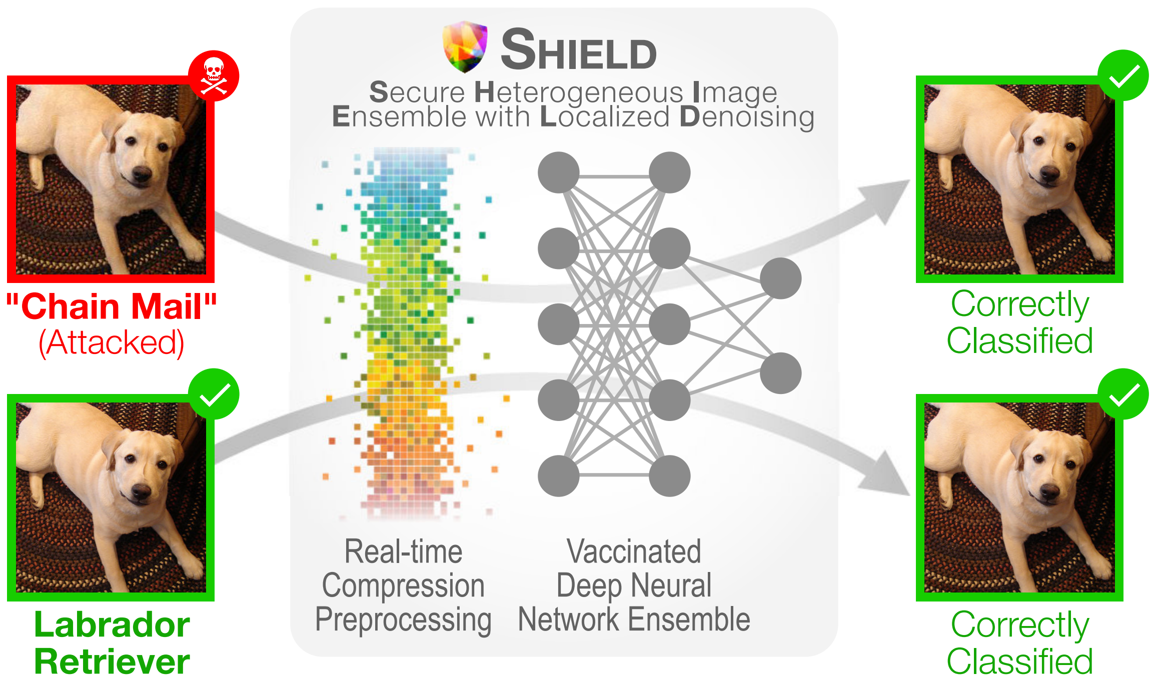

SHIELD
Secure Heterogeneous Image Ensemble with Localized Denoising
"Chain Mail"
(Attacked)
Labrador Retriever
Real-time Compression Preprocessing
Vaccinated Deep Neural Network Ensemble
Correctly Classified
Correctly Classified

Figure 1.4: SHIELD eliminates up to 98% of gray-box attacks delivered by strong adversarial techniques such as *Carlini-Wagner's L2* attack and *DeepFool*.

SHIELD: Fast, Generalizable, Stochastic Defense (Chapter 4). As a first step towards creating operational defenses for real-world AI systems, we develop the SHIELD framework, through which we explore the idea of compression as a fast, practical defense for image classification models. We expand upon the widely used JPEG compression algorithm and propose a novel pre-processing technique that incorporates randomization with compression to develop a multifaceted defense (Figure 1.4). We discover that input compression is powerful in removing imperceptible, high-frequency adversarial perturbations from images, which are introduced by non-adaptive adversarial attacks.

SkeleVision: Adversarially Robust Video Tracking with MTL (Chapter 5). Right at the heels of releasing our SHIELD research, adversarial ML had quickly evolved to propose a new generation of more sophisticated, *adaptive* attacks, that also estimate gradients with respect to the input pre-processing stage. Following this, it is no longer enough to compress away high-frequency perturbations for evading adversarial attacks. Hence, we shift our focus to fortifying the model internals directly by influencing the learning stage of ML models. Our experiments with MTL reveal that it is one such fundamental deep learning approach that has the potential to fortify ML models across AI tasks. As MTL forces a shared backbone network to simultaneously support multiple end-goals, our hypothesis is that the backbone ends up learning more robust features within the intermediate feature

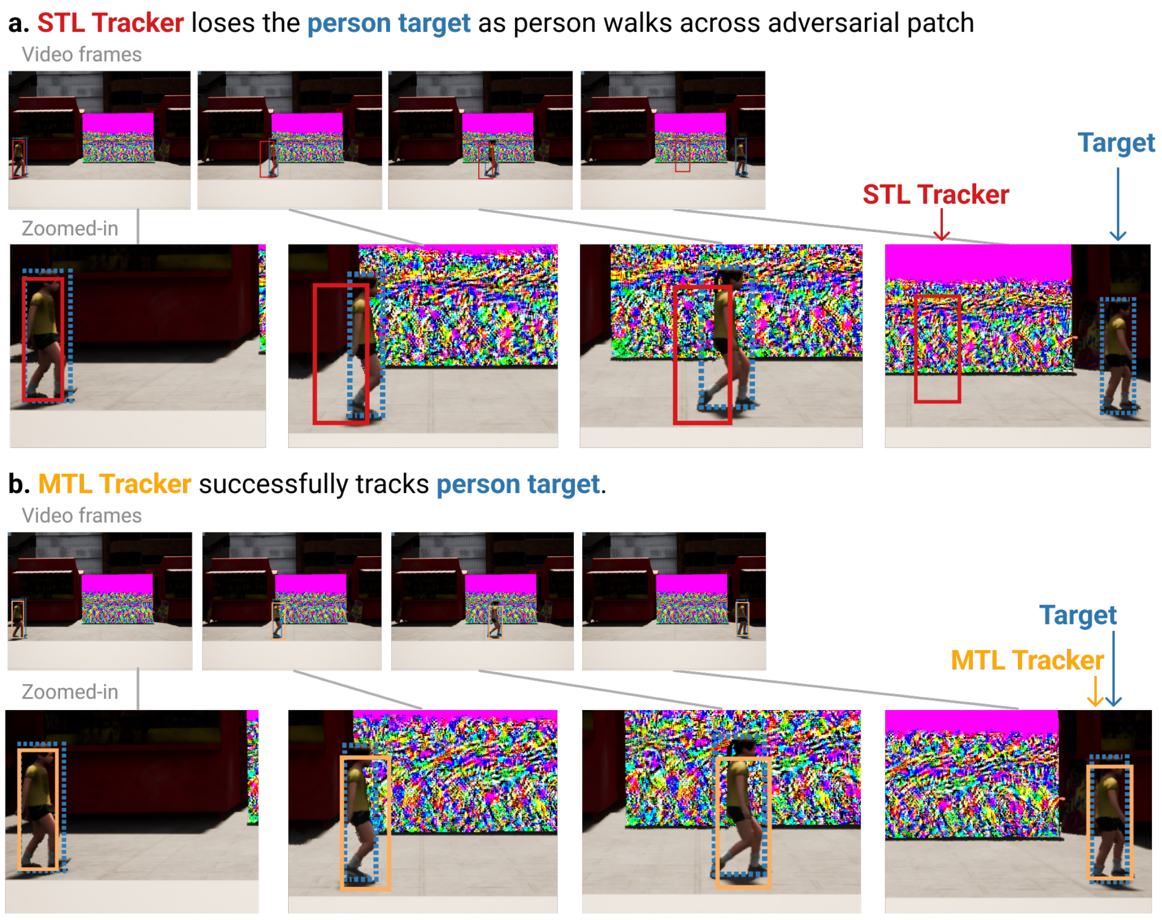

a. STL Tracker loses the person target as person walks across adversarial patch
Video frames
Zoomed-in
STL Tracker
Target
b. MTL Tracker successfully tracks person target.
Video frames
Zoomed-in
Target
MTL Tracker

Figure 1.5: (a) A physically realizable attack is able to confuse the tracker trained with single-task learning (STL) to lock onto the adversarial patch. (b) The tracker trained with multi-task learning (MTL) is consistently able to track the target as it moves across the adaptive adversarial patch.

space. We first demonstrate this ability of MTL to induce a robust feature space in the video domain through the task of person tracking. In our SkeleVision research, we combine semantically analogous tasks of person tracking and human keypoint detection, both of which tasks necessitate some robust form of semantic understanding of the human anatomy. Our experiments reveal that MTL models are consistently more resistant to powerful, adaptive, physically realizable attacks across a high number of attack iterations (Figure 1.5).

Hear No Evil: Adversarially Robust Speech Recognition with MTL (Chapter 6). Following the observations from our SkeleVision research, we shift our attention to an entirely different input modality in order to study the efficacy of MTL robustness across AI tasks. We next experiment with the robustness of automatic speech recognition (ASR) models in the audio domain when trained jointly with MTL (Figure 1.6). We explore semantically equivalent (*e.g.*, CTC and attention) as well as semantically diverse tasks (*e.g.*, ASR with accent classification) for performing MTL. We find that a combination of both types of tasks

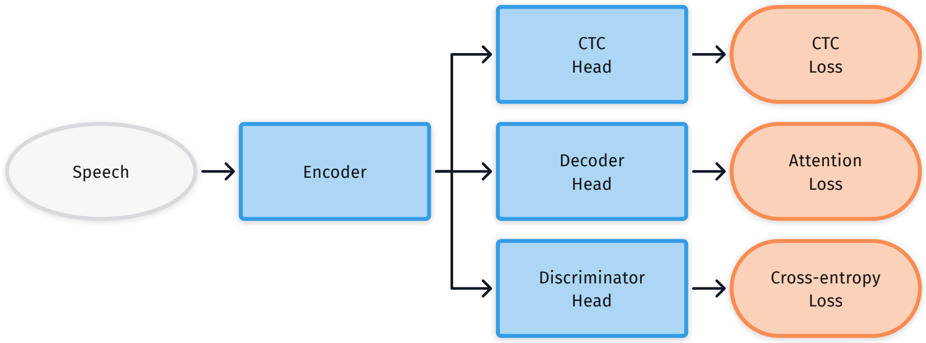


Figure 1.6: Overview of an MTL framework for ASR with accent classification. A shared encoder feeds into multiple task heads with corresponding losses that are jointly optimized.

is necessary to most effectively thwart powerful, adaptive adversarial attacks. Our MTL approach shows considerable absolute improvements in adversarially targeted word error rate. These observations along with our SkeleVision research establishes multi-task learning as a fundamentally unifying deep learning approach across AI tasks and input modalities, which induces models to learn robust features that are resistant to adversarial attacks.

1.2.3 Part III: Democratizing AI Security Research & Pedagogy (Enable)

So far we have explored how we can deeply understand AI vulnerabilities and fortify AI models. However in the coming future, with the radical infusion of AI in everyday life, AI security will not only be restricted to the fancies of expert researchers. While visualization tools and intuitive vulnerability interpretation go a long way in enabling people to understand AI security more deeply, it still requires some level of expertise in the adversarial ML domain to successfully decipher these concepts. In this part of the thesis, we go that last mile to bring AI security research to the masses. Through developing interactive experimentation tools, we democratize AI security for new researchers, practitioners and students, and make adversarial ML research more accessible and more equitable for everyone.

ADAGIO: Interactive Experimentation with Attack and Defense for ASR (Chapter 7). We take the first step towards this goal by developing ADAGIO, a web-based tool that allows real-time interactive experimentation with attacks and defenses on an automatic speech recognition (ASR) model. Through developing ADAGIO, and following the observations from our SHIELD research, we discover that the idea of input compression as a practical defense carries over to the audio domain as well. We see that applying audio compression techniques such as MP3 and AMR compression are very effective in thwarting targeted non-adaptive adversarial attacks.

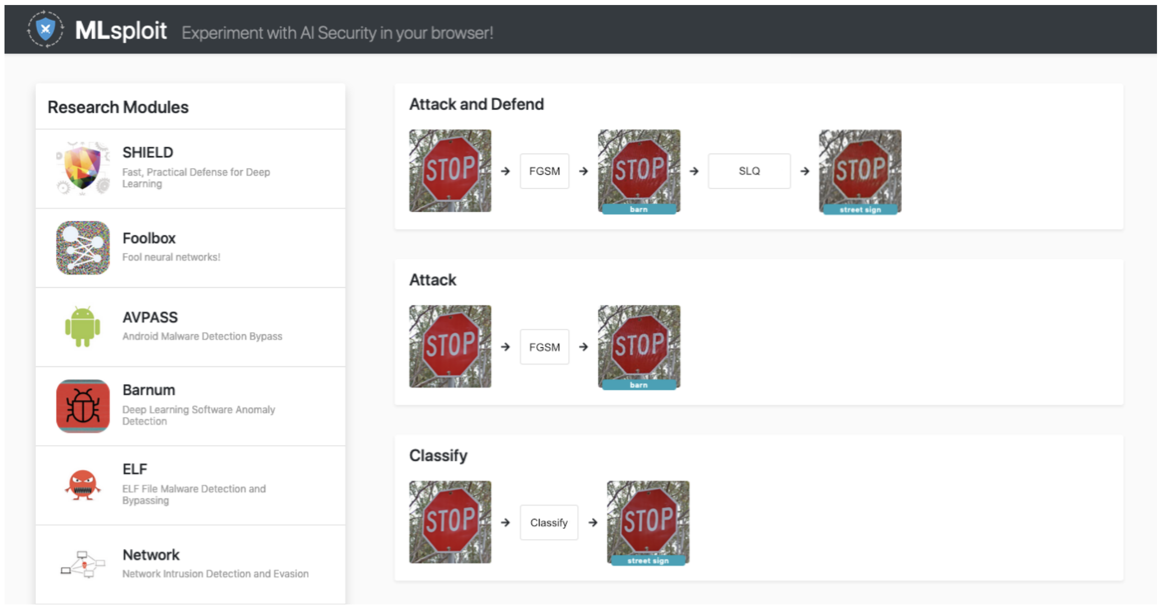

MLsploit Experiment with AI Security in your browser!
Research Modules
SHIELD
Fast, Practical Defense for Deep Learning
Foolbox
Fool neural networks!
AVPASS
Android Malware Detection Bypass
Barnum
Deep Learning Software Anomaly Detection
ELF
ELF File Malware Detection and Bypassing
Network
Network Intrusion Detection and Evasion
Attack and Defend
FGSM
SLQ
barn
street sign
Attack
FGSM
barn
Classify
Classify
street sign

Figure 1.7: Using MLsploit's interactive web interface, users can upload their own samples, apply research functions to them, and compare the results with previous experiments.

MLsploit: AI Security Experimentation at Scale (Chapter 8). Working on ADAGIO gave us the key insight that *interactive experimentation* on the web is an immensely powerful medium for research dissemination. We build upon this notion by developing MLsploit (Figure 1.7), the first open-source, scalable, web-based interactive system that allows seamless experimentation with adversarial ML research. A culmination of the other research directions (Parts I & II), MLsploit provides a modular repository of attacks and defenses, enabling practitioners to interactively study their AI applications under various threat models. Becoming available to thousands of students, MLsploit is already transforming AI security education at scale.

1.3 THESIS STATEMENT

To overcome the significant threat of adversarial attacks on ML models and enhance people's trust in using AI systems by enabling everyone to:
(1) understand AI vulnerabilities through intuitive interpretation;
(2) fortify AI applications with robust mitigation of adversarial examples; and
(3) easily test AI security techniques with interactive experimentation.

1.4 RESEARCH CONTRIBUTIONS

This thesis makes research contributions through multiple major fronts.

- **Novel, principled, interpretable approaches for semantic class inference.**
  Our research contributes novel, theoretically principled approaches to extract interpretable semantic prototypes for data programming (Chapter 2). Our weakly-supervised GOGGLES framework is **only 7% away** from a fully-supervised baseline.

- **Novel visualization technique for deciphering attacks.**
  Our work significantly expands the understanding of model vulnerabilities, which were earlier considered as "black boxes", by proposing attribution graph mining for model activation (Chapter 3). Our novel visualization technique **helps in identifying the vulnerable neurons** in a DNN that are non-performant and lead to misclassifications under attack.

- **Faster, generalizable, practical defenses.**
  This dissertation identifies the need for AI defenses with low computational overhead and develops fast, robust defenses based on input compression that generalize across input modalities. Our SHIELD defense (Chapter 4) is up to **22x faster** than other pre-processing defenses, and the ADAGIO defense (Chapter 7) **removes all non-adaptive targeted adversarial attacks**.

- **Fundamentally unifying approach for training robust models across AI tasks.**
  Our in-depth study of **Multi-Task Learning** (Chapters 5 and 6) establishes it as a fundamentally unifying deep learning approach that induces ML models to learn robust features that are **resistant to adversarial attacks** across AI tasks and input modalities.

- **First scalable system for interactive experimentation with AI security research.**
  This thesis introduces MLsploit (Chapter 8), the **first open-source interactive system** that allows in-depth security testing of AI models, and lowers barriers to entry for everyone — researchers, practitioners and students — to engage in adversarial ML research. MLsploit offers modules of state-of-the-art attacks and defenses.

1.5 IMPACT

Beyond the contributions to the research community, our work has also made a significant impact to broader society and industry.

- Early-stage startups have picked up our GOGGLES framework for it's efficacy in generating training data with low overhead, and even extended the affinity functions to include natural language processing tasks.
- The SHIELD defense framework won the **Audience Appreciation Award, runner-up** at KDD 2018, being in the top 3 among 107 accepted papers, from 983 submissions. SHIELD was also tech-transferred to **Intel Labs** and multiple business units within Intel for integration and testing. The SHIELD tech transfer to Intel also incited a positive shift at the highest levels within the company towards developing hardware accelerators for AI defense.
- Research ideas presented in this dissertation formed a core component of an awarded multi-million dollar **DARPA GARD** (Guaranteeing AI Robustness against Deception) grant for developing next-generation defenses against physically realizable adversarial attacks.
- MLsploit is already transforming adversarial ML education. It is currently being integrated into multiple security and data analytics courses at **Georgia Tech**, becoming available to thousands of students. MLsploit is already being used by graduate students in Dr. Wenke Lee's **"CS 8803 O11: Information Security Lab - System and Network Defenses"** class and was also made available to students of Dr. Polo Chau's **"CSE 6242: Data and Visual Analytics"** class at Georgia Tech. MLsploit won the Institute for Information Security and Privacy's **Cybersecurity Demo Day Finale** held at Georgia Tech in 2019. MLsploit modules also formed the foundation for the DARPA GARD research proposal.

Part I

Exposing AI Vulnerabilities through Visualization & Interpretable Representations

OVERVIEW

For humans to confidently deploy secure AI systems, there is an urgent need to enable people to truly understand the vulnerabilities of the underlying DNN models and how the adversarial attacks and defenses work the way they do. In this first part of the thesis, we focus on developing new tools and techniques to help people *understand* the underlying vulnerabilities of ML models that are exploited by adversarial examples, expanding the body of visualization and interpretability research to the adversarial ML domain.

The first step of our research plan is to *extract* interpretable units from within an ML model, so that we can leverage them as an explanation basis for the model's decisions. In Chapter 2, we present the **GOGGLES** framework. Here, we propose an innovative technique to extract such interpretable "semantic prototypes" from a pre-trained deep neural network. We show that our extracted prototypes are semantic representations that possess significant predictive capacity that can explain how the model operates internally. This chapter is adapted from our published work [1] that appeared at SIGMOD 2020.

GOGGLES: Automatic Image Labeling with Affinity Coding. Nilaksh Das, Sanya Chaba, Renzhi Wu, Sakshi Gandhi, Polo Chau, Xu Chu. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020. PDF

The next step of our research is to build upon the notion of interpretable prototypes from GOGGLES, leveraging them to compute a deep neural network's "attribution graph". The attribution graph constitutes a summary of how the activation values "flow" through the network from input to output. In Chapter 3, we develop a scalable visualization tool, **BLUFF**, that allows researchers to interactively explore this attribution graph of a network under attack in order to discover its vulnerabilities. BLUFF reveals that adversarial attacks leverage "non-robust" and confounding intermediate features to confuse the model. This chapter is adapted from our published work [2] that appeared at IEEE VIS 2020.

Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. Nilaksh Das*, Haekyu Park*, Zijie J. Wang, Fred Hohman, Robert Firstman, Emily Rogers, Polo Chau. *IEEE Visualization Conference*, 2020. PDF

*authors contributed equally

CHAPTER 2
GOGGLES: AUTOMATIC IMAGE LABELING WITH AFFINITY CODING

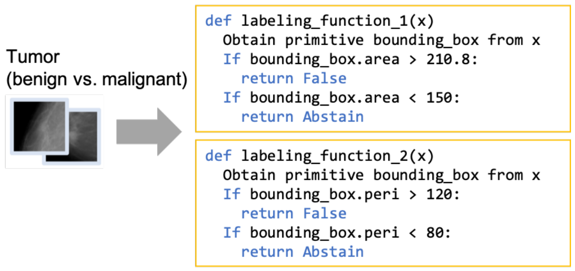
Generating large labeled training data is becoming the biggest bottleneck in building and deploying supervised machine learning models. Recently, the data programming paradigm has been proposed to reduce the human cost in labeling training data. However, data programming relies on designing labeling functions which still requires significant domain expertise. Also, it is prohibitively difficult to write labeling functions for image datasets as it is hard to express domain knowledge using raw features for images (pixels).

We propose *affinity coding*, a new domain-agnostic paradigm for automated training data labeling. The core premise of affinity coding is that the affinity scores of instance pairs belonging to the same class on average should be higher than those of pairs belonging to different classes, according to some affinity functions. We build the GOGGLES system that implements affinity coding for labeling image datasets by designing a novel set of reusable affinity functions for images, and propose a novel hierarchical generative model for class inference using a small development set.

We compare GOGGLES with existing data programming systems on 5 image labeling tasks from diverse domains. GOGGLES achieves labeling accuracies ranging from a minimum of 71% to a maximum of 98% without requiring any extensive human annotation. In terms of end-to-end performance, GOGGLES outperforms the state-of-the-art data programming system Snuba by 21% and a state-of-the-art few-shot learning technique by 5%, and is only 7% away from the fully supervised upper bound.

2.1 INTRODUCTION

Machine learning (ML) is being increasingly used by organizations to gain insights from data and to solve a diverse set of important problems, such as fraud detection on structured tabular data, identifying product defects on images, and sentiment analysis on texts. A fundamental necessity for the success of ML algorithms is the existence of sufficient high-quality labeled training data. For example, the current ConvNet revolution would not be possible without big labeled datasets such as the 1M labeled images from ImageNet [3]. Modern deep learning methods often need tens of thousands to millions of training examples to reach peak predictive performance [4]. However, for many real-world applications, large hand-labeled training datasets either do not exist, or is extremely expensive to create as manually labeling data usually requires domain experts [5].



```
def labeling_function_1(x)
  Obtain primitive bounding_box from x
  If bounding_box.area > 210.8:
    return False
  If bounding_box.area < 150:
    return Abstain
```

```
def labeling_function_2(x)
  Obtain primitive bounding_box from x
  If bounding_box.peri > 120:
    return False
  If bounding_box.peri < 80:
    return Abstain
```

Figure 2.1: Example LFs in data programming [14].

Existing Work. We are not the first to recognize the need for addressing the challenges arising from the lack of sufficient training data. The ML community has made significant progress in designing different model training paradigms to cope with limited labeled examples, such as semi-supervised learning techniques [6], transfer learning techniques [7] and few-shot learning techniques [8, 9, 10, 11]. In particular, the most related learning paradigm that shares a similar setup to us, few-shot learning techniques, usually require users to preselect a source dataset or pre-trained model that is in the same domain of the target classification task to achieve best performance. In contrast, our proposal can incorporate as many available sources of information as affinity functions.

Only recently, the *data programming* paradigm [12] and the Snorkel [13] and Snuba system [14] that implement the paradigm were proposed in the data management community. Data programming focuses on reducing the human effort in training data labeling, particularly in unstructured data classification tasks (images, text). Instead of asking humans to label each instance, data programming ingests domain knowledge in the form of labeling functions (LFs). Each LF takes an unlabeled instance as input and outputs a label with better-than-random accuracy (or abstain). Based on the agreements and disagreements of labels provided by a set of LFs, Snorkel/Snuba then infer the accuracy of different LFs as well as the final probabilistic label for every instance. The primary difference between Snorkel and Snuba is that while Snorkel requires human experts to write LFs, Snuba learns a set of LFs using a small set of labeled examples.

While data programming alleviates human efforts significantly, it still requires the construction of a new set of LFs for every new labeling task. In addition, we find that it is extremely challenging to design LFs for image labeling tasks primarily because raw pixels values are not informative enough for expressing LFs using either Snorkel or Snuba. After consulting with data programming authors, we confirmed that Snorkel/Snuba require images to have associated metadata, which are either text annotations (e.g., medical notes associated with X-Ray images) or primitives (e.g., bounding boxes for X-Ray images). These associated text annotations or primitives are usually difficult to come by in practice.

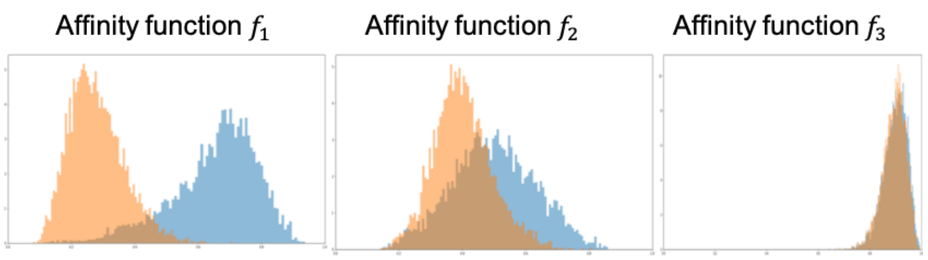


Figure 2.2: Affinity score distributions. Blue and yellow denote the affinity scores of instance pairs from the same class and different classes, respectively.

Example 1 *Figure 2.1 shows two example labeling functions for labeling an X-Ray image as either benign or malignant [14]. As we can see, these two functions rely on the bounding box primitive for each image and use the two properties (area and perimeter) of the primitive for labeling. We observe that these domain-specific primitives are difficult to obtain. Indeed, [14] states, in this particular example, radiologists have pre-extracted these bounding boxes for all images.*

Our Proposal. We propose *affinity coding*, a new domain-agnostic paradigm for automated training data labeling without requiring any domain specific functions. The core premise of the proposed affinity coding paradigm is that the *affinity scores of instance pairs belonging to the same class on average should be higher than those of instance pairs belonging to different classes, according to some affinity functions*. Note that this is quite a natural assumption — if two instances belong to the same class, then by definition, they should be similar to each other in some sense.

Example 2 *Figure 2.2 shows the affinity score distributions of a real dataset we use in our experiments (CUB) using three of the* 50 *affinity functions discussed in Section 2.3. In this particular case, affinity function f_1 is able to distinguish pairs in the same class from pairs in different classes very well; affinity function f_2 also has limited power in separating the two cases; and affinity function f_3 is not useful at all in separating the classes.*

We build the GOGGLES system that implements the affinity coding paradigm for labeling image datasets (Figure 2.3). First, GOGGLES includes a novel set of affinity functions that can capture various kinds of image affinities. Given a new unlabeled dataset and the set of affinity functions, we construct an affinity matrix. Second, using a very small set of labeled examples (development set), we can assign classes to unlabeled images based on the affinity score distributions we can learn from the affinity matrix. Compared with the state-of-the-art data programming systems, our affinity coding system GOGGLES has the following distinctive features.

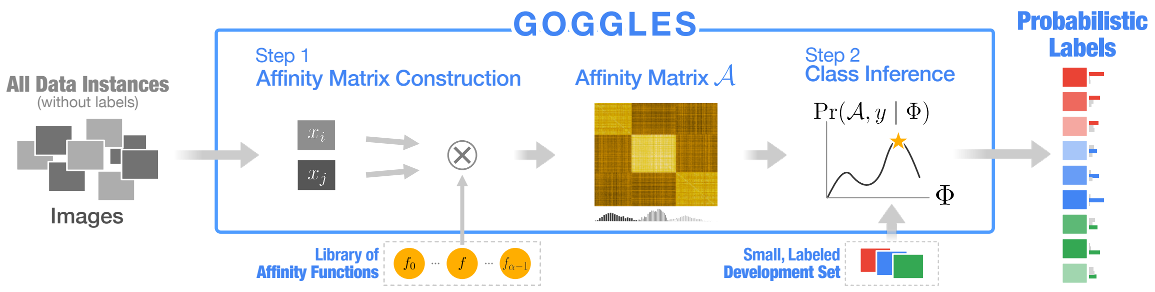

GOGGLES
All Data Instances
(without labels)
Images
Step 1
Affinity Matrix Construction
x_i
x_j
Affinity Matrix $\mathcal{A}$
Step 2
Class Inference
$\Pr(\mathcal{A}, y \mid \Phi)$
Φ
Probabilistic Labels
Library of Affinity Functions
f_0 ··· f ··· $f_{\alpha-1}$
Small, Labeled Development Set

Figure 2.3: Overview of the GOGGLES framework.

- Data programming systems need some kinds of metadata (text annotations or domain-specific primitives) associated with each image to express LFs, while GOGGLES makes no such assumptions.
- Assuming the existence of metadata, data programming still requires a new set of LFs for every new dataset. In contrast, GOGGLES is a domain-agnostic system that leverages affinity functions, which are populated once and can be reused for any new dataset.
- Both Snorkel/Snuba and GOGGLES can be seen as systems that leverage many sources of weak supervision to infer labels. Intuitively, the more weak supervision sources a system has, the better labeling accuracy a system can potentially achieve. In data programming, the number of sources is the number of LFs. In contrast, affinity coding uses affinity scores between instance pairs under many affinity functions. Therefore, the number of sources GOGGLES has is essentially the number of instances multiplied by the number of affinity functions, a significantly bigger set of weak supervision sources.

Challenges. We address the following major challenges with GOGGLES:

- The success of affinity coding depends on a set of affinity functions that can capture similarities of images in the same class. However, without knowing which classes and labeling task we may have in the future, we do not even know what are the potential distinctive features for each class. Even if we have the knowledge of the particular distinctive features, they might be spatially located in different regions of images in the same class, which makes it more difficult to design domain-agnostic affinity functions.
- Given an affinity matrix constructed using the set of affinity functions, we need to design a robust class inference module that can infer class membership for all unlabeled instances. This is quite challenging for multiple reasons. First, some of the affinity functions are indicative for the current labeling, while many others are just noise, as shown in Example 2. Our class inference module needs to identify which affinity functions are

useful given a labeling task. Second, the affinity matrix is high-dimensional with the number of dimension equals to the number of instances multiplied by the number of affinity functions. In this high-dimensional space, distance between any two rows in the affinity matrix becomes extremely small, and thus making it even more challenging to infer class assignments. Third, while we can infer from the affinity matrix which instances belong to the same class by essentially performing clustering, we still need to figure out which cluster corresponds to which class, relying only on a small development set.

Contributions. We make the following contributions:

- **The affinity coding paradigm**. We propose *affinity coding*, a new domain-agnostic paradigm for automatic generation of training data. Affinity coding paradigms consists of two main components: a set of affinity functions and a class inference algorithm. To the best of our knowledge, we are the first to propose a domain-agnostic approach for automated training data labeling.

- **Designing affinity functions.** GOGGLES features a novel approach that defines affinity functions based on a pre-trained VGG-16 model [15]. VGG-16 is a commonly used network for representation learning. Our intuition is that different layers of the VGG-16 network capture different high-level semantic concepts. Each layer may show different activation patterns depending on where a high-level concept is located in an image. We thus leverage all 5 max-pooling layers of the network, extracting 10 affinity functions per layer, for a total of 50 affinity functions.

- **Class inference using hierarchical generative model.** GOGGLES proposes a novel hierarchical model to identify instances of the same class by maximizing the data likelihood under the generative model. The hierarchical generative model consists of two layers: the base layer consists of multiple Gaussian Mixture Models (GMMs), each modeling an affinity function; and the ensemble layer takes the predictions from each GMM and uses another generative model based on multivariate Bernoulli distribution to obtain the final labels. We show that our hierarchical generative model addresses both the curse of dimensionality problem and the affinity function selection problem. We also give theoretical justifications on the size of development set needed to get correct cluster-to-class assignment.

GOGGLES achieves labeling accuracy ranging from a minimum of 71% to a maximum of 98%. In terms of end-to-end performance, GOGGLES outperforms the state-of-the-art data programming system Snuba by 21% and a state-of-the-art few-shot learning technique by 5%, and is only 7% away from the fully supervised upper bound. We also make our implementation of GOGGLES open-source at https://github.com/chu-data-lab/GOGGLES.

2.2 PRELIMINARY

We formally state the problem of automatic training data labeling in Section 2.2.1. We then introduce *affinity coding*, a new paradigm for addressing the problem in Section 2.2.2.

2.2.1 Problem Setup

In traditional supervised classification applications, the goal is to learn a classifier h_θ based on a labeled training set (x_i, y_i), where $x_i \in \mathcal{X}_{train}$ and $y_i \in \mathcal{Y}_{train}$. The classifier is then used to make predictions on a test set.

In our setting, we only have $\mathcal{X}_{train}$ and no $\mathcal{Y}_{train}$. Let n denote the total number of unlabeled data points, and let y_i^* denote the unknown true label for x_i. Our goal is to assign a *probabilistic label* $\tilde{y}_i^k$ for every $x_i \in \mathcal{X}_{train}$, where $\tilde{y}_i^k = \Pr\big([y_i^* = k]\big) \in [0, 1]$, where $k \in \big\{1, 2, \ldots, K\big\}$ with K being the number of classes in the labeling task, and $\sum_k \tilde{y}_i^k = 1$.

These probabilistic labels can then be used to train downstream ML models. For example, we can generate a discrete label according to the highest $\tilde{y}_i^q$ for every instance x_i. Another more principled approach is to use the probabilistic labels directly in the loss function $l(h_\theta(x_i), y)$, i.e., the expected loss with respect to $\tilde{y}$: $\hat{\theta} = \text{argmin}_\theta \sum_{i=1}^n E_{y\sim\tilde{y}_i}[l(h_\theta(x_i), y)]$. It has been shown that as the amount of unlabeled data increases, the generalization error of the model trained with probabilistic labels will decrease at the same asymptotic rate as supervised models do with additional labeled data [12].

2.2.2 The Affinity Coding Paradigm

We propose *affinity coding*, a domain-agnostic paradigm for automatic labeling of training data. Figure 2.3 depicts an overview of GOGGLES, an implementation of the paradigm.

Step 1: Affinity Matrix Construction. An affinity function takes two instances and output a real value representing their similarity. Given a library of α affinity functions $\mathcal{F} = \{f_0, f_1, \ldots, f_{\alpha-1}\}$, a set of n unlabeled instances $\{x_0, \ldots, x_{n-1}\}$, and a small m labeled examples $\{(x_n, y_n), \ldots, (x_{n+m-1}, y_{n+m-1})\}$ as the development set, we construct an affinity matrix $\mathcal{A} \in \mathbb{R}^{(n+m)\times\alpha(n+m)}$ that encodes all affinity scores between all pairs of instances under all affinity functions. Specifically, the i^{th} row of $\mathcal{A}$ corresponds to instance x_i and every j^{th} column of $\mathcal{A}$ corresponds to the affinity function $f_{j/(n+m)}$ and the instance $x_{j\%(n+m)}$, namely, $\mathcal{A}[i, j] = f_{j/(n+m)}(x_i, x_{j\%(n+m)})$.

Step 2: Class Inference. Given $\mathcal{A}$, we would like to infer the class membership for all unlabeled instances. For every unlabeled instance $x_i, i \in [0, n-1]$, we associate a hidden variable $\tilde{y}_i$ representing its unknown class. We aim to maximize the data likelihood $\Pr\big(\mathcal{A}, \tilde{y}|\Phi\big)$, where Φ denotes the parameters of the generative model used to generate $\mathcal{A}$.

Discussion. The affinity coding paradigm offers a domain-agnostic paradigm for training data labeling. Our assumption is that, for a new dataset, there exists one or multiple affinity functions in our library $\mathcal{F}$ that can capture some kinds of similarities between instances in the same class. We verify that our assumption holds on all five datasets we tested. It is particularly worth noting that, out of the five datasets, three of them are in completely different domains than the ImageNet dataset the VGG-16 model is trained on. This suggests that our current $\mathcal{F}$ is quite comprehensive. We acknowledge that there certainly exists potential new labeling tasks that our current set of affinity functions $\mathcal{F}$ would fail.

2.3 DESIGNING AFFINITY FUNCTIONS

Our affinity coding paradigm is based on the proposition that examples belonging to the same class should have certain similarities. For image datasets, this proposition translates to *images from the same class would share certain visually discriminative high-level semantic features.* However, it is nontrivial to design affinity functions that capture these high-level semantic features due to two challenges: (1) without knowing which classes and labeling task we may have in the future, we do not even know what those features are. and (2) even assuming we know the particular features that are useful for a given class, they might be spatially located in different regions of images in the same class.

To address these challenges, GOGGLES leverages pre-trained convolutional neural networks (VGG-16 network [15] in our current implementation) to transplant the data representation from the raw pixel space to semantic space. It has been shown that intermediate layers of a trained neural network are able to encode different levels of semantic features, such as edges and corners in initial layers; and textures, objects and complex patterns in final layers [16].

Algorithm 1 gives the overall procedure of leveraging the VGG-16 network for coding multiple affinity functions. Specifically, to address the issue of not knowing which high-level features might be needed in the future, we use different layers of the VGG-16 network to capture different high-level features that might be useful for different future labeling tasks (Line 1). We call each such high-level feature a *prototype* (Line 2). As not all prototypes are actually informative features, we keep the top-Z most "activated" prototypes, which we treat as informative high-level semantic features (Line 3). For every one of the informative prototype v_j^k extracted from an image x_j, we need to design an affinity function that checks whether another image x_i has a similar prototype (Line 5). Since these prototypes might be located in different regions, our affinity function is defined to be the maximum similarity between all prototypes of x_i and v_j^k (Line 6).

We discuss prototype extraction and selection in Section 2.3.1, and the computation of affinity functions based on prototypes in Section 2.3.2.

Algorithm 1 Coding affinity functions $f_0, f_1, \ldots f_{\alpha-1}$ based on the pre-trained VGG model

```
Input: Two unlabeled images $x_i$ and $x_j$
Output: Affinity scores between $x_i$ and $x_j$ under $f_0, f_1, \ldots f_{\alpha-1}$.
1: for all each max-pooling layer $L$ in VGG-16 do
2:     For image $x_j$, extract all of its prototypes $\boldsymbol{\rho}_j = \{v_j^{(1,1)}, v_j^{(1,2)}, \ldots, v_j^{H \times W}\}$ by passing
       it through the pre-trained VGG until layer $L$ to obtain a filter map of size $C \times H \times W$,
       where $C$, $H$ and $W$ are the number of channels, height and width of the filter map
       respectively and each prototype is vector of length $C$.
3:     Selecting $Z$ most activated prototypes of $x_j$, denoted as $\{v_j^1, v_j^2, \ldots, v_j^Z\}$
4:     Similarly, for image $x_i$, extract all of its prototypes $\boldsymbol{\rho}_i = \{v_i^{(1,1)}, v_i^{(1,2)}, \ldots, v_i^{H \times W}\}$
5:     for all $v_j^k \in \{v_j^1, v_j^2, \ldots, v_j^Z\}$, where $z \in [1, Z]$ do
6:         $f_L^z(x_i, x_j) \leftarrow \max_{h,w} sim(v_j^Z, v_i^{(h,w)})$
7:     end for
8: end for
```

2.3.1 Extracting Prototypes

In this subsection, we discuss (1) how to extract all prototypes from a given image x_i using a particular layer L of the VGG-16 network; and (2) how to select top Z most informative prototypes amongst all the extracted ones.

Extracting all prototypes. To begin, we pass an image x_i through a series of layers until reaching a max-pooling layer L of a CNN to obtain the $F_i = L(x_i)$, known as a *filter map*. We choose max-pooling layers as they condense the previous convolutional operations to provide compact feature representations. The filter map F_i has dimensions $C \times H \times W$, where C, H and W are the number of channels, height and width of the filter map respectively. Let us also denote indexes over the height and width dimensions of F_i with h and w respectively. Each vector $v_i^{(h,w)} \in \mathbb{R}^C$ (spanning the channel axis) in the filter map F_i can be backtracked to a rectangular patch in the input image x_i, formally known as the *receptive field* of $v_i^{(h,w)}$. The location of the corresponding patch of a vector $v_i^{(h,w)}$ can be determined via gradient computation. Since any change in this patch will induce a change in the vector $v_i^{(h,w)}$, we say that $v_i^{(h,w)}$ encodes the semantic concept present in the

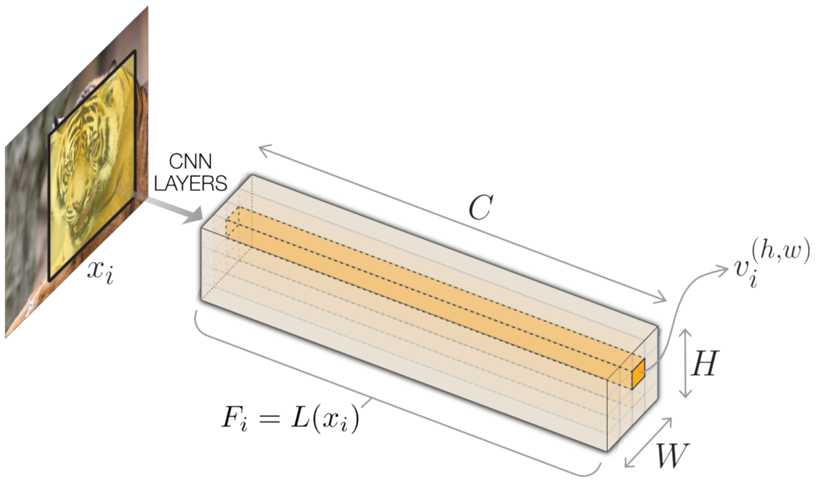


Figure 2.4: Extracting all prototypes in a layer of the VGG-16 network. Each prototype corresponds to a patch in the input image's raw pixel space.

patch. Formally, all prototypes we extract for x_i are as follows:

$$\boldsymbol{\rho}_i = \{v_i^{(1,1)}, v_i^{(1,2)}, \dots, v_i^{(H,W)}\}$$

Example 3 *Figure 2.4 shows the representation of an image patch in semantic space using a tiger image. An image x_i is passed through VGG-16 until a max-pooling layer to obtain the filter map F_i that has dimensions $C \times H \times W$. In this particular example, the yellow rectangular patch highlighted in the image is the receptive field of the orange prototype $v_i^{(h,w)}$, which as we can see, captures the "tiger's head" concept.*

Selecting top Z informative prototypes. In an image x_i, obviously not every patch and the corresponding prototype $v_i^{(h,w)}$ is a good signal. In fact, many patches in an image correspond to background noise that are uninformative for determining its class. Therefore, we need a way to intelligently select the top Z most informative semantic prototypes from all the $H \times W$ possible ones.

In this regard, we first select top-Z channels that have the highest magnitudes of activation. Note that each channel is a matrix $\mathbb{R}^{H \times W}$, and the activation of a channel is defined to be the maximum value of its matrix (typically known as the 2D Global Max Pooling operation in CNNs). We denote the indexes of these top-Z channels as c_z, where $z \in \{1, \dots, Z\}$. Based on the top-Z channels, we can thus define the top-Z prototypes as follow:

$$v_i^z = v_i^{(h,w)}, \text{where } \ h, w = \operatorname*{argmax}_{h,w} F_i[c_z; h; w]. \tag{2.1}$$

The top-Z prototypes we extract for image x_i are:

$$\boldsymbol{\rho}_i = \{v_i^1, v_i^2, \ldots, v_i^Z\}$$

The pair (h, w) may not be unique across the channels, yielding the same concept prototypes. Hence, we drop the duplicate $v_i^{(h,w)}$'s and only keep the unique prototypes.

Example 4 *We illustrate our approach for selecting top-Z prototypes by an example. Suppose we would like to select top-2 prototypes in a layer that produces the following filter map of dimension $C \times H \times W = 3 \times 2 \times 2$. The three channels are:*

$$C_1 = \begin{bmatrix} 1 & 0.5 \\ 0.3 & 0.6 \end{bmatrix} C_2 = \begin{bmatrix} 0.1 & 0.7 \\ 0.4 & 0.3 \end{bmatrix} C_3 = \begin{bmatrix} 0.2 & 0.9 \\ 0.5 & 0.1 \end{bmatrix}$$

First, we sort the three channels by the maximum activation in descent order i.e. the maximum element in the matrix: C_1, C_3, C_2. Then, we select the first Z=2 channels: C_1, C_3. Next, for each of the selected channels we identify the index of its maximum element on the H and W axis: $(h_1, w_1) = (0, 0), (h_2, w_2) = (0, 1)$. Finally, we obtain the Z=2 prototypes by stacking the values over all channels that share the same H and W axis index identified in the last step:
$v^1 = \{C_1[h_1, w_1], C_2[h_1, w_1], C_3[h_1, w_1]\} = \{1, 0.1, 0.2\}$*, and*
$v^2 = \{C_1[h_2, w_2], C_2[h_2, w_2], C_3[h_2, w_2]\} = \{0.5, 0.7, 0.9\}$.

2.3.2 Computing Affinity

Having extracted prototypes for each image, we are ready to define affinity functions and compute affinity scores for a pair of images (x_i, x_j). Affinity functions are supposed to capture various types of similarity between a pair of images. Intuitively, two images are similar if they share some high-level semantic concepts that are captured by our extracted prototypes. Based on this observation, we define multiple affinity functions, each corresponding to a particular type of semantic concept (prototype). Therefore, the number of affinity functions we can define is equal to the number of max-pooling layers (5) of the VGG-16 network multiplied by the number of top-Z prototypes extracted per layer.

Let us consider a particular prototype v_j^z, that is, the z^{th} most informative prototype of x_j extracted from layer L, we define an affinity function as follows:

$$f_L^z(x_i, x_j) = \max_{h,w} sim(v_j^z, v_i^{(h,w)}) \tag{2.2}$$

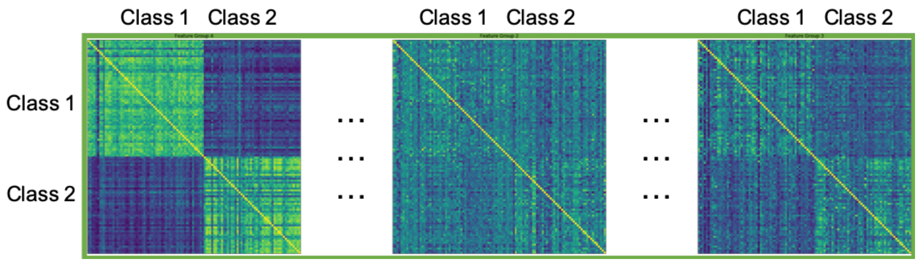


Figure 2.5: An affinity matrix visualized as a heatmap.

As we can see, we calculate the similarity between a prototype v_j^z of x_j and the vector $v_i^{(h,w)} \forall (h,w) \in \{(1,1),\ldots,(H,W)\}$ contained in $F_i = f(x_i)$ using a similarity function $sim(\cdot)$, and pick the highest value as the affinity score. In other words, our approach tries to find the "most similar patch" in each image x_i with respect to a given patch corresponding to one of the top-Z prototypes of image x_j. We use the cosine similarity metric as the similarity function $sim(\cdot)$ defined over two vectors a and b as follows:

$$sim(a,b) = \frac{a^T b}{\|a\|_2 \|b\|_2}. \tag{2.3}$$

Example 5 *Figure 2.5 shows an example affinity matrix $\mathcal{A}$ for the CUB dataset we use in the experiments. It only shows three of the* 50 *affinity functions, which we also used in Example 2. The rows and columns are sorted by class only for visual intuition. As we can see, some affinity functions are more informative than others in this labeling task.*

Discussion. We use all 5 max-pooling layers from the VGG-16. For each max-pooling layer, we use the top-10 prototypes, which we empirically find to be sufficient. Note that while we choose VGG-16 to define affinity functions in the current GOGGLES implementation, GOGGLES can be easily extended to use any other representation learning techniques.

In summary, our approach automatically identifies semantically meaningful prototypes from the dataset, and leverages these prototypes for defining affinity functions to produce an affinity matrix.

2.4 CLASS INFERENCE

In this section, we describe GOGGLES' class inference module: given the affinity matrix $\mathcal{A} \in \mathbb{R}^{N \times \alpha N}$ constructed on $N = n + m$ examples using s affinity functions, where n is the number of unlabeled examples and m is a very small development set (e.g., 10 labeled examples), we would like to assign a class label for every examples $x_i, \forall i \in [1, n]$. In other words, our goal is to predict $P(y_i = k|\mathbf{s}_i)$, where $\mathbf{s}_i$ denote the feature vector for x_i, namely, the i^{th} row in $\mathcal{A}$, and $y_i = k \in \{1, 2 \ldots, K\}$ is a hidden variable representing the class assignment of x_i.

Generative Modelling of the Inference Problem. Recall that our main assumption of affinity coding is that the affinity scores of instance pairs belonging to the same class should be different than affinity scores of instance pairs belonging to different classes. In other words, the feature vector $\mathbf{s}_i$ of one class should look different than that of another class. This suggests a *generative approach* to model how $\mathbf{s}_i$ is generated according to different classes. Generative models obtain $P(y_i = k|\mathbf{s}_i)$ by invoking the Bayes rules:

$$P(y_i = k|\mathbf{s}_i) = \frac{P(y_i = k)P(\mathbf{s}_i|y_i = k)}{P(\mathbf{s}_i)} = \frac{\pi_k \times P(\mathbf{s}_i|y_i = k)}{\sum_{k'=1}^{K} \pi_{k'} \times P(\mathbf{s}_i|y_i = k')} \tag{2.4}$$

where $\pi_k = P(y_i = k)$ is known as the *prior probability* with $\sum_{k'=1}^{K} \pi_{k'} = 1$, which is the probability that a randomly chosen instance is in class k, and $P(y_i = k|\mathbf{s}_i)$ is known as the *posterior probability*. To use Equation (2.4) for labeling, we need to learn π_k and $P(\mathbf{s}_i|y_i = k)$ for every class k. $P(\mathbf{s}_i|y_i = k)$ is commonly assumed to follow a known distribution family parameterized by θ_k, and is written as $P(\mathbf{s}_i|\theta_k)$. Therefore, the entire set of parameter we need to have to compute Equation (2.4) is $\Theta = \{\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K\}$.

A common way to estimate Θ is by maximizing the log likelihood function:

$$\begin{aligned} L(\mathcal{A}, Y|\Theta) &= \log \prod_{i=1}^{N} P(\mathbf{s}_i, y_i|\Theta) = \sum_{i=1}^{N} \log \big(P(y_i|\Theta)P(\mathbf{s}_i|\Theta, y_i)\big) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}_{y_i=k} \log \big(\pi_k P(\mathbf{s}_i|\theta_k)\big) \end{aligned} \tag{2.5}$$

where $\mathbb{1}_{.}$ is the identity function that evaluates to 1 if the condition is true and 0 otherwise.

Therefore, the main questions we need to address include (i) what are the generative models to use, namely, the paramterized distributions $P(\mathbf{s}_i|\theta_k)$; and (ii) how do we maximize Equation (2.5).

Limitations of Existing Models. A commonly used distribution is multi-variate Gaussian distribution, where $\theta_k = \{\mu_k, \Sigma_k\}$, where μ_k is the mean vector and Σ_k is the covariance matrix, and $P(\mathbf{s}_i|\theta_k)$ is the Gaussian PDF:

$$P(\mathbf{s}_i|y_i = k) = P(\mathbf{s}_i|\theta_k) = \frac{\exp\{-\frac{1}{2}(\mathbf{s}_i - \mu_k)^T \Sigma_k^{-1} (\mathbf{s}_i - \mu_k)\}}{(2\pi)^{sn/2} \text{det}(\Sigma_k)^{1/2}} \tag{2.6}$$

This yields the popular Gaussian Mixture Model (GMM), and there are known algorithm for maximizing the likelihood function under GMM. However, a naive invocation of GMM on our affinity matrix $\mathcal{A}$ is problematic:

- **High dimensionality.** The number of feature in the affinity matrix $\mathcal{A}$ is αN. In the naive GMM, the mean vectors and covariance matrices for all classes (components) have $K(\binom{\alpha N}{2} + \alpha N)$ number of parameters, which is much larger than the number of examples N. It is widely known that the eigenstructure in the estimated covariance matrix Σ_j will be significantly and systematically distorted when the number of features exceeds the number of examples [17, 18].
- **Affinity function selection.** Not all affinity functions are useful, as shown in Figure 2.5. If the number of noisy functions is small, GMM naturally handles feature selection as the components will not be well separated by noisy functions and will be well separated by "good" functions. However, under such high dimensionality, there could exist too many noisy features that could form false correlations among them and eventually undermine the accuracy of GMM or other generic clustering methods.

2.4.1 A Hierarchical Generative Model

A fundamental reason for the above two challenges when using GMM is that GMM needs to model correlations between all pairs of columns, which creates a huge number of parameters and makes it difficult for GMM to determine which affinity functions are more informative. In light of this observation, we propose a *hierarchical generative model* which consists of a set of *base models* and an *ensemble model*, as shown in Figure 2.6. Each base model captures the correlations of a subset of columns in $\mathcal{A}$ that originate from the same affinity function f, and we denote this "subset" matrix as $\mathcal{A}_f \in \mathbb{R}^{N \times N}$. The output of each base model is a *label prediction matrix* $LP_f \in \mathbb{R}^{N \times K}$, where the i^{th} row stores the probabilistic class assignments of x_i using affinity function f. All label prediction matrices are concatenated together to form the *concatenated label prediction matrix* $LP \in \mathbb{R}^{N \times \alpha K}$. The ensemble model takes LP and models the correlations of all affinity functions, and produces the final probabilistic labels for each unlabeled instance.

The Base Models. Given the part of the affinity matrix $\mathcal{A}_f \in \mathbb{R}^{N \times N}$ generated by a particular affinity function f, the base model aims to predict $P(y_i = k|\mathbf{s}_i^f)$, where $\mathbf{s}_i^f$ denotes the subset of the feature vector $\mathbf{s}_i$ corresponding to f.

We design a base generative model for computing $P(y_i = k|\mathbf{s}_i^f)$. As discussed before, a generative model requires specifying the class generative distributions $P(\mathbf{s}_i^f|\theta_k)$, parame-

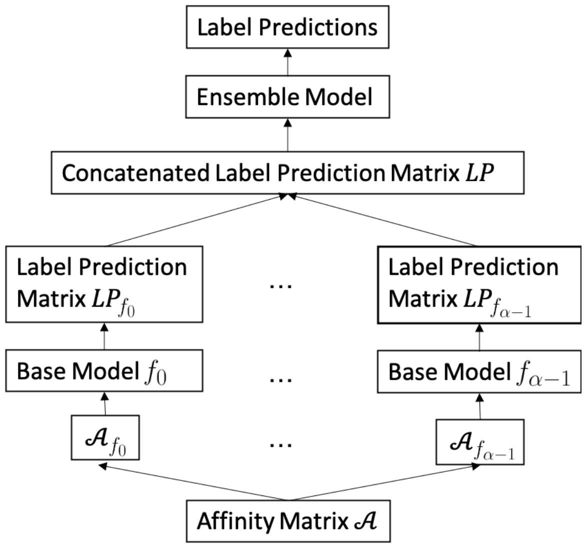

Label Predictions
Ensemble Model
Concatenated Label Prediction Matrix LP
Label Prediction Matrix LP_{f_0}
...
Label Prediction Matrix $LP_{f_{\alpha-1}}$
Base Model f_0
...
Base Model $f_{\alpha-1}$
$\mathcal{A}_{f_0}$
...
$\mathcal{A}_{f_{\alpha-1}}$
Affinity Matrix $\mathcal{A}$

Figure 2.6: The hierarchical generative model.

terized by θ_k. We use the popular GMM for this purpose with an important modification. Instead of using the full covariance matrix Σ_k that models the correlations between all pairs of columns in $\mathcal{A}_f$, we use the diagonal covariance matrix, which reduces the number of parameters significantly from $\binom{N}{2}$ to N. Note that this simplification is only possible under the base generative model, as each column of $\mathcal{A}_f$ corresponds to an independent example.

The output of the base model for affinity function f is a label prediction matrix $LP_f \in \mathbb{R}^{N\times K}$, where $LP_f[i,k] = P(y_i = k|\mathbf{s}_i^f)$, namely, the probability that affinity function f believes example x_i is in class k.

The Ensemble Model. We concatenate α label prediction matrices $LP_{f_0}, \ldots, LP_{f_{\alpha-1}}$ from α base models to obtain the concatenated label prediction matrix $LP \in \mathbb{R}^{N\times \alpha K}$. Let $\mathbf{s}'_i$ denote the new feature vector of the i^{th} row in LP. The goal of the ensemble model is to predict $P(y_i = k|\mathbf{s}'_i)$.

We again design a generative model for performing the final prediction. As before, we need to decide on a class generative distribution $P(\mathbf{s}'_i|\theta'_k)$ parameterized by θ'_k. The Gaussian distribution used for the base models is not appropriate for the ensemble mode. This is because the values in the concatenated label prediction matrix LP are very close to either 0 or 1. Indeed, in an ideal scenario when all base models work perfectly, all values in LP will be 0 or 1 that correspond to the ground truth. This kind of discrete or close to discrete data is problematic for Gaussian distribution which is designed for continuous data. Fitting a Gaussian distribution on this kind of data typically incurs the singularity problem and provides poor predictions [19]. In light of this observation, we convert LP to a one-hot encoded matrix by converting the highest class prediction to 1 and the rest predictions to 0

for each instance and each affinity function, and we propose to use a categorical distribution to model LP.

After converting LP into a true discrete matrix, Multivariate Bernoulli distribution is a natural fit for modeling $P(\mathbf{s}'_i|\theta'_k)$, which is parameterized by $\theta'_k = \{b_{k,1}, \ldots, b_{k,\alpha K}\}$:

$$P(\mathbf{s}'_i|\theta'_k) = \prod_{l=1}^{\alpha K} b_{k,l}^{\mathbf{s}'_i[l]}(1 - b_{k,l})^{1-\mathbf{s}'_i[l]} \tag{2.7}$$

where $\mathbf{s}'_i[l]$ is the l^{th} dimension of the binary vector $\mathbf{s}'_i$, and we have a total of αK dimensions. The output of the ensemble model is the final label predictions $L \in \mathbb{R}^{N \times K}$, where $L[i, k] = P(y_i = k|\mathbf{s}'_i)$, namely, the probability that the ensemble model f believes example x_i is in class k.

Hierarchical Model Address the Two Challenges. First, the total number of parameters in the hierarchical model is $2\alpha KN + \alpha K$, much smaller than the number of parameters in the naive GMM $K(\binom{\alpha N}{2} + \alpha N)$, effectively addressing the high-dimensionality problem. Second, by consolidating the affinity scores produced by each affinity function to produce a binary LP, the ensemble model can only need to model the accuracy of the α affinity functions better instead of the original αN features, and thus can better distinguish the good affinity functions from the bad ones.

2.4.2 Parameter Learning

We need to learn the parameters of the base models and the ensemble model under their respective generative distributions. Expectation-maximization algorithm is the canonical algorithm for maximizing the log likelihood function in the presence of hidden variables [20]. We first show the EM algorithm for maximizing the general data log likelihood function in Equation (2.5), and then discuss how it needs to modified to learn the base models and the ensemble model.

EM for Maximizing Equation (2.5) Each iteration of the EM algorithm consists of two steps: an Expectation (E)-step and a Maximization (M)-step. Intuitively, the E-step determines what is the (soft) class assignment y_i for every instance x_i based on the parameter estimates from last iteration Θ^{t-1}. In other words, E-step computes the posterior probability. The M-step takes the new class assignments and re-estimates all parameters Θ^t by maximizing Equation (2.5). More precisely, the M-step maximizes the expected value of Equation (2.5), since the E-step produces soft assignments.

1. **E Step.** Given the parameter estimates from the previous iteration Θ^{t-1}, compute the

posterior probabilities:

$$\gamma_{i,k} = P(y_i = k|\mathbf{s}_i) = \frac{\pi_k \times P(\mathbf{s}_i|\theta_k)}{\sum_{k'=1}^{K} \pi_{k'} \times P(\mathbf{s}_i|\theta_{k'})} \tag{2.8}$$

2. **M Step.** Given the new class assignments as defined by $\gamma_{i,k}$, re-estimate Θ_t by maximizing the following expected log likelihood function:

$$\mathbb{E}\{L(\mathcal{A}, Y|\Theta)\} = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{i,k} \log\left(\pi_k p(\mathbf{s}_i|\theta_k)\right) \tag{2.9}$$

EM for Maximizing the Base Model. For each base model associated with affinity function f, $P(\mathbf{s}_i|\theta_k)$ in the EM algorithm is replaced with $P(\mathbf{s}_i^f|\theta_k^f)$, which is a multivariate Gaussian distribution as shown in Equation (2.6), but with a diagonal covariance matrix. The entire set of parameters is $\Theta = \{\pi_k^f, \mu_k^f, \Sigma_k^f\}$, where $k = 0, \ldots, K-1$, which update in each M-step as follows:

$$\begin{aligned} N_k &= \sum_{i=1}^{N} \gamma_{i,k}; \pi_k^f = N_k/N; \mu_k^f = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{i,k} \mathbf{s}_i^f \\ \Sigma_k^f &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{k,i} (\mathbf{s}_i^f - \mu_k^f)(\mathbf{s}_i^f - \mu_k^f)^T \end{aligned} \tag{2.10}$$

EM for Maximizing the Ensemble Model. For the ensemble model, $P(\mathbf{s}_i|\theta_k)$ in the EM algorithm is replaced with $P(\mathbf{s}_i'|\theta_k')$, which is a multivariate Bernoulli distribution, as shown in Equation (2.7). The entire set of parameters is $\{\pi_k, b_{k,1},$
$\ldots, b_{k,\alpha K}\}$, where $k = 0, \ldots, K-1$, which we update in each M-step as follows:

$$\begin{aligned} N_k &= \sum_{i=1}^{N} \gamma_{i,k}; \pi_k = N_k/N \\ b_{k,l} &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{i,k} \mathbf{s}_i'[l], \text{ where } l \in \{1, 2, \ldots, \alpha K\} \end{aligned} \tag{2.11}$$

2.4.3 Exploiting Development Set

Consider a scenario without any labeled development set, in this case, the hierarchical model essentially clusters all unlabeled examples into K clusters without knowing which cluster corresponds to which class. Following the data programming system [14], we assume we have access to a small development set that is typically too small to train any machine learning models, but is powerful enough to determine the correct “cluster-to-class” assignment. Note that the theory developed here can also be used to provide theoretical guarantees on the mapping feasibility in the “cluster-then-label” category of semi-supervised learning approaches [6, 21].

Let $LS_{k'}$ denote the set of labeled examples for class k'. To make our analysis easier, we assume the size of $LS_{k'}$ is the same for all classes. Intuitively, we want to map cluster k to class k' if most examples from $LS_{k'}$ are in cluster k. However, this simple cluster-to-class mapping strategy may create conflicting assignments, namely, the same cluster is mapping to multiple classes. We propose a more principled way to obtain the one-to-one mapping $g : k \mapsto k'$. We first define the "goodness" of the mapping L_g as:

$$L_g = \sum_{k=1}^{K} \sum_{l \in LS_{g(k)}} \gamma_{l,k} \tag{2.12}$$

To see why L_g can represent the "goodness" of a mapping. We represent development sets with a one-hot encoded ground truth matrix T where each element $t_{i,k'}$ is obtained by:

$$t_{i,k'} = \begin{cases} 1, \text{ if } i \in LS_{k'} \\ 0, \text{ otherwise} \end{cases} \quad i = 1, \ldots, N;\ k' = 1, \ldots, K \tag{2.13}$$

L_g is essentially the summation of the element-wise multiplication between the ground truth matrix T and label prediction matrix LP under a column mapping defined by g on the development set. Therefore, L_g is expected to be maximized when a mapping g makes the two matrices the most similar under cosine distance. Given L_g, the final mapping g is obtained by:

$$g = \arg\max_g \{L_g\}, \text{ and } g \text{ is a one-to-one mapping} \tag{2.14}$$

In other words, the final mapping is a one-to-one mappings that maximizes L_g. When $K = 2$, Equation (2.14) becomes

$$g(k) = \begin{cases} k, \text{ if } \sum_{l \in LS_1} \gamma_{l,1} \geq \sum_{l \in LS_0} \gamma_{l,1} \\ 1 - k, \text{ otherwise} \end{cases} \tag{2.15}$$

Algorithm for Solving Equation (2.14). Instead of enumerating all possible mappings with a complexity of $O(K!)$ (which is actually feasible for a small K), we convert it to the *assignment problem* which can be solved in $O(K^3)$. Let $w_{k,k'}$ denote $w_{k,k'} =$

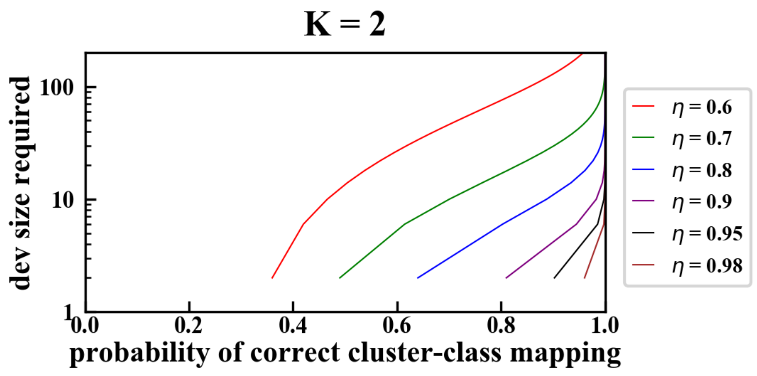


Figure 2.7: Size of the Development Set Needed.

$\sum l \in LS_{k'}\gamma_{l,k}$, then Equation (2.12) becomes:

$$L_g = \sum_{k=1}^{K} w_{k,g(k)} \tag{2.16}$$

Finding a g that maximizing Equation (2.16) is essentially the *Assignment problem*, and there are known algorithms [22] that solve it with a worst case time complexity of $O(K^3)$.

This "cluster-to-class" mapping is performed after obtaining base model predictions and the ensemble model predictions. After the mapping is obtained, we rearrange the the columns in the label prediction matrix LP_f produced by each base model, and the final label matrix L produced by the ensemble according to the mapping g, so that the true classes are aligned with the clusters.

2.4.4 The Size of Development Set Needed

In this section, we give an analysis about the size of the development set needed for GOGGLES to produce the correct "cluster-to-class" mapping, where the correct mapping is defined to be the mapping that achieves the highest labeling accuracy, which we denote as η. Intuitively, the higher η is, the less size we need. Consider an extreme scenario with $K = 2$ classes and our hierarchical generative model produces two clusters that perfectly separate the two classes. In this case, we only need one labeled example to determine which cluster corresponds to which class with 100% confidence. Figure 2.7 shows the size of development set required when $K = 2$ based on our theory to be discussed in the following, we can see when $\eta = 0.8$, only about 20 examples are required to produces the correct cluster-class mapping with a probability close to 1. However, as we will shown in the experiment section, the number of required development set size is actually much smaller in practice. This is because the theoretical lower-bound we will provide is a rather loose one, for ease of derivation.

A Theory on the Size of the Development Set. let us first assume the mapping of each class is independent, so the probability of a completely correct mapping is $P_{\text{ind}} = \prod_{k'=1}^{K} P_{k'}$, where $P_{k'}$ denote the probability that class k' is correctly mapped to its corresponding cluster.

To simplify derivation, we further assume "hard" assignment of classes labels: an example is only assigned to one cluster, in other words, γ only contains 0 and 1. This is a natural assumption because values in γ will be converted to be binary anyway when evaluating the accuracy of the algorithm. In the development set, we have a labeled set $LS_{k'}$ with a size of $d = m/K$ for every class k'. Let $d_{k',j},\ j = 1, \dots K$, denote the number of examples in the development set $LS_{k'}$ that are in the jth cluster, so $\sum_{j=1}^{K} d_{k',j} = d$. Under the independence assumption, Equation (2.14) becomes

$$g^{-1}(k') = \text{arg} \max_{1 \leq j \leq K} \sum_{i \in LS_{k'}} \gamma_{i,j} \tag{2.17}$$

where g^{-1} denote the inverse mapping of g, that is $k' \mapsto k$. Equation (2.17) means that each each class is mapped to the cluster in which its majority lies, so class k' is mapped to its correct cluster only when the majority of $LS_{k'}$ are in that cluster. Assume the kth cluster is the correct cluster for class k', so the probability of the k' class mapped correctly is:

$$\begin{aligned} P_{k'} > Pl_{k'} &= \sum_{d_{k',k}=0}^{d} P(d_{k',k} > \max_{1 \leq j \leq K, j \neq k} d_{k',j}) \\ &= \sum_{d1, \dots, d_K} P(d_{k',1}, \dots, d_{k',K}) \text{ s.t. } d_{k',k} > \max_{1 \leq j \leq K, j \neq k} d_{k',j} \end{aligned} \tag{2.18}$$

The first $>$ sign is because on the right side we don't consider the situations with ties in majority vote (the second $>$ sign), where we break the ties randomly and a correct mapping is also possible. The P_{ind} is then lower bounded by:

$$P_{\text{ind}} > \prod_{k'=1}^{K} Pl_{k'} \tag{2.19}$$

Suppose the accuracy of our algorithm η is known, so the probability of an example being predicted to be its true label equals to η. An example in the development set $LS_{k'}$ is predicted to be its true label by the algorithm only when it is in the correct cluster, so the probability of it being in the correct cluster equals to η. In case of incorrect assignment, we assume the probability of assigning to every possible incorrect classes is equal, being

$\rho = \frac{\eta}{K-1}$. $d_{k',1}, \dots, d_{k',K}$ follow a multinomial distribution:

$$\begin{aligned} P(d_{k'1}, \dots, d_{k',K}) =& \frac{d!}{\prod_{j=1}^{K} d_{k',j}!} \eta^{d_{k',1}} \rho^{d_{k',2}} \rho^{d_{k',3}} \dots \rho^{d_{k',K}} \\ =& \frac{d!}{\prod_{j=1}^{K} d_{k',j}!} \eta^{d_{k',k}} \rho^{d-d_{k',k}} \end{aligned} \tag{2.20}$$

The correct mapping under independent assumption requires the mapping of every class to be correct on their own. This is a rather strict assumption. Without assuming independence, Equation (2.14) is able to produce a completely correct mapping when some classes would otherwise fail to be mapped correctly on their own. In other words, the probability of a completely correct mapping is:

$$P_{\text{correct}} \geq P_{\text{ind}} \tag{2.21}$$

Combining Equation (2.21) and Equation (2.19), we get the following theorem.

Theorem 6 *The probability that Equation* (2.14) *gives the optimal mapping is lower bounded by $P_{correct} > \prod_{k'=1}^{K} Pl_{k'}$, where $Pl_{k'}$ is obtained by Equation* (2.18).

Therefore, the size of development set m^* that produces an optimal mapping with a probability of as least p is given by $m^* = Kd^*$, where d^* is the smallest value of d that makes $\prod_{k'=1}^{K} Pl_{k'} \geq p$.

The time complexity of solving the right hand side in Equation (2.18) by a brute-force iteration over all combinations of d_j is $O(d!)$, but it can be solved in $O(Kd^2)$ using a dynamic programming based approach.

For ease of of notation, we assume the 1st cluster is the correct cluster for class k'. Let $S(j, D_j)$ denote the following:

$$\begin{aligned} S(j, D_j) = & \sum_{d_{k',j},\dots,d_{k',K}} P(d_{k',1}, ..., d_{k',K}) \\ \text{s.t.} & \max_{j \leq l \leq K, l \neq k} d_{k',l} < d_{k',1} \text{ and } \sum_{l=j}^{K} d_{k',l} = D_j \end{aligned} \tag{2.22}$$

so $Pl_{k'} = S(1, d)$, and for each j:

$$S(j, D_j) = \sum_{d_{k',j}=0}^{d} S(j+1, D_j - d_j) \tag{2.23}$$

The time complexity of obtaining Pl_k by dynamic programming using Equation (2.23) is $O(Kd^2)$.

2.5 EXPERIMENTS

We conduct extensive experiments to evaluate the accuracy of labels generated by GOGGLES. Specifically, we focus on the following three dimensions:

- *Feasibility and Performance of GOGGLES (Section 2.5.2).* Is it possible to automatically label image datasets using a domain-agnostic approach? How does GOGGLES compare with existing data programming systems?
- *Ablation Study (Section 2.5.3).* How do the two primary innovations in GOGGLES (namely, affinity matrix construction and class inference) compare against other techniques?
- *Sensitivity Analysis (Section 2.5.4).* Is GOGGLES sensitive to the set of affinity functions? What is the size of the development set needed for GOGGLES to correctly determine the correct "cluster-to-class" mapping?

2.5.1 Setup

Datasets

We consider real-world image datasets with varying domains to evaluate the versatility and robustness of GOGGLES. Since our approach internally uses a pre-trained VGG-16 model for defining affinity functions, we select datasets which have minimal or no overlap with classes of images from the ImageNet dataset [3], on which the VGG-16 model was originally trained. Robust performance across these datasets show that GOGGLES is domain-agnostic with respect to the underlying pre-trained model. We perform our experiments on the following datasets, which are roughly ordered by domain overlap with ImageNet:

- **CUB**: The Caltech-UCSD Birds-200-2011 dataset [23] comprises of 11,788 images of 200 bird species. The dataset also provides 312 binary image-level attribute annotations that help explain the visual characteristics of the bird in the image, e.g., white head, grey wing etc. We use this metadata information for designing binary labeling functions which are used by a data programming system. To evaluate the task of generating binary labels, we randomly sample 10 class-pairs from the 200 classes in the dataset and report the average performance across these 10 pairs for each experiment. These sampled class-pairs are not present in the ImageNet dataset. However, since ImageNet and CUB contain common images of other bird species, this dataset may have a higher degree of domain overlap with the images that VGG-16 was trained on.

- **GTSRB**: The German Traffic Sign Recognition Benchmark dataset [24] contains 51,839 images for 43 classes of traffic signs. Again, for testing the performance of binary label generation, we sample 10 random class-pairs from the dataset and use them for all the experiments. Although this dataset contains images labeled by specific traffic signs, ImageNet contains a generic "street sign" class, and hence this dataset may also have some degree of domain overlap.
- **Surface**: The surface finish dataset [25] contains 1280 images of industrial metallic parts which are classified as having "good" (smooth) or "bad" (rough) metallic surface finish. This is a more challenging dataset since the metallic components look very similar to the untrained eye, and has minimal degree of domain overlap with ImageNet.
- **TB-Xray**: The Shenzhen Hospital X-ray set [26] has 662 images belonging to 2 classes, normal lung X-ray and abnormal X-ray showing various manifestations of tuberculosis. These images are of the medical imaging domain and have absolutely no domain overlap with ImageNet.
- **PN-Xray**: The pneumonia chest X-ray dataset [27] consists of 5,856 chest X-ray images classified by trained radiologists as being normal or showing different types of pneumonia. These images are also of the medical imaging domain and have no domain overlap with ImageNet.

Development Set. GOGGLES uses a small development set to determine the optimal class mapping for a given label assignment, the same assumption in Snuba [14]. By default, we use only 5 label annotations arbitrarily chosen from each class for this. Hence, for the task of generating binary labels, we use a development set having a size of 10 images for all the experiments. We report the performance of GOGGLES on the remaining images from each dataset.

Data Programming Systems

We compare GOGGLES with existing systems: Snorkel [13] and Snuba [14].

Snorkel is the first system that implements the data programming paradigm [12]. Snorkel requires humans to design several *labeling functions* that output a noisy label (or abstain) for each instance in the dataset. Snorkel then models the high-level interdependencies between the possibly conflicting labeling functions to produce probabilistic labels, which are then used to train an end model. For image datasets, these labeling functions typically work on metadata or extraneous annotations rather than image-based features since it is very hard to hand design functions based on these features.

Since CUB is the only dataset having such metadata available, we report the mean performance of Snorkel on the 10 class-pairs sampled from the dataset by using the attribute annotations as labeling functions. More specifically, we combine CUB's image-level attribute annotations (which describe visual characteristics present in an image, such as white head, grey wing etc.) with the class-level attribute information provided (e.g., class A has white head, class B has grey wing etc.) in order to design labeling functions. Hence, each attribute annotation in the union of the class-specific attributes acts as a labeling function which outputs a binary label corresponding to the class that the attribute belongs to. If an attribute belongs to both classes from the class-pair, the labeling function abstains. We used the open-source implementation provided by the authors with our labeling functions for generating the probabilistic labels for the CUB dataset.

Snuba extends Snorkel by further reducing human efforts in writing labeling functions. However, Snuba requires users to provide per-instance primitives for a dataset (c.f. Example 1), and the system automatically generates a set of labeling functions using a labeled small development set.

Since all 6 datasets do not come with user-provided primitives, to ensure a fair comparison with Snuba, we consulted with Snuba's authors multiple times. They suggested that we use a rich feature representation extracted from images as their primitives, which would allow Snuba to learn labeling functions. As such, we use the *logits layer* of the pre-trained VGG-16 model for this purpose, as it has been well documented in the domain of computer vision that such feature representations encode meaningful higher order semantics for images [28, 29]. For the VGG-16 model trained on ImageNet, this yields us feature vectors having 1000 dimensions for each image. To obtain densely rich primitives which are more tractable for Snuba, we project the logits output onto a feature space of the top-10 principal components of the entire data determined using principal component analysis [30]. We use these projected features having 10 dimensions as primitives for Snuba. Empirical testing revealed that providing more components does not change the results significantly. We also use the same development set size for Snuba and GOGGLES. We used the open-source implementation provided by the authors for learning labeling functions with automatically extracted primitives and for generating the final probabilistic labels.

Few-shot Learning (FSL)

Our affinity coding setup which uses 5 development set labels from each class is comparable to the 2-way 5-shot setup for few-shot learning from the computer vision domain. Hence, we compare GOGGLES's end-to-end performance with a recent FSL approach [11] that achieves state-of-the-art performance on domain adaptation. We use the same development set used by GOGGLES as the few-shot labeled examples for training the FSL model.

The original FSL Baseline implementation uses a model trained on mini-ImageNet for domain adaptation to the CUB dataset, and achieves better performance than other state-of-the-art FSL methods. For a more comparable analysis, we use a VGG-16 model trained on ImageNet, which is the same pre-trained model GOGGLES uses for affinity coding. Note that our adaptation of the FSL Baseline method achieved a much better performance for domain adaptation on CUB than the original results reported in [11]. The FSL models as well as all end models are trained with the Adam optimizer with a learning rate of 10^{-3}, same as in [11].

Empirical upper-bound (supervised approach).

We also would like to compare GOGGLES' performance with an empirical upper-bound, which is obtained via a typical supervised transfer learning approach for image classification. Specifically, we freeze the convolutional layers of the VGG-16 model and only update the weights of the fully connected layers in the VGG-16 architecture while training. We also modify the last fully connected "logits" layer of the architecture to our corresponding number of classes.

Ablation Study: Other image representation techniques for computing affinity

GOGGLES computes affinity scores by extracting prototype representations from intermediate layers of a pre-trained model. We compare the efficacy of this representation technique with two other typical methods of image representation used in the computer vision domain. We compare the predictive capacity of each representation technique by constructing an affinity matrix from each candidate feature representation using pair-wise cosine similarity, and then using our class inference approach for labeling.

HOG. We compare with the histogram of oriented gradients HOG descriptor, which is a very popular feature representation technique used for recognition tasks in classical computer vision literature [31, 32]. The HOG descriptor [33] represents an image by counting the number of occurrences of gradient orientation in localized portions of the image.

Logits. We also compare with a modern deep learning-based approach, leveraged by recent works in computer vision [34, 35], that uses an intermediate output from a convolutional neural network as an image's feature representation. We use the logits layer from the trained VGG-16 model in our comparison, which is the output of the last fully connected layer, before it is fed to the softmax operation.

Ablation Study: Baseline methods for class inference

The class inference method in GOGGLES consists of a clustering step followed by class mapping. We compare our proposed hierarchical model for clustering with other baseline methods, including **K-means** clustering, Gaussian mixture modeling with expectation maximization (**GMM**) and spectral co-clustering (**Spectral**). Since these clustering methods are incognizant of the structural semantics of our affinity-based features which are derived from multiple affinity functions, we simply concatenate all affinity functions to create the feature set for each dataset, and then feed these features to the baseline methods. As we would like to see the absolute best performance of the baseline clustering approaches, we use the optimal "cluster-class" mapping for all baselines.

Evaluation Metrics.

We use the train/test split as originally defined in each dataset. We report the *labeling accuracy* on the training set for comparing different data labeling systems, Snorkel, Snuba, and GOGGLES. We follow the same approach used in [13, 14] to train an end discriminative model by using the probabilistic labels generated from each data labeling system as training data and report the *end-to-end accuracy* as the end model's performance on the held-out test set. For labeling tasks, all experiments, including baselines, are conducted 10 times, and we report the average.

2.5.2 Feasibility and Performance

Table 2.1 shows the end-to-end system labeling accuracy for GOGGLES, Snorkel, Snuba, and a supervised approach that serves as an upper bound reference for comparison. (1) GOGGLES achieves labeling accuracies ranging from a minimum of 71% to a maximum of 98%. GOGGLES shows an average of 21% improvement over the state-of-the-art data programming system Snuba. (2) To ensure a fair comparison, we consulted with authors of Snuba and took their suggested approach of automatically extracting the required primitives. As we can see, the performances of Snuba on all datasets are just slightly better than random guessing. This is primarily because Snuba is really designed to operate on human annotated

Table 2.1: Evaluation of GOGGLES labeling accuracy on training set. The '-' symbol represents cases where evaluation was not possible. GOGGLES shows on average an improvement of 23% over the state-of-the-art data programming system Snuba.

Dataset	GOGGLES	Data Programming		Representation		Class Inference Baselines		
	(our results)	Snorkel	Snuba	HoG	Logits	K-Means	GMM	Spectral
CUB	97.83	89.17	58.83	62.93	96.35	98.67	97.62	72.08
GTSRB	70.51	-	62.74	75.48	64.77	70.74	69.64	62.40
Surface	89.18	-	57.86	85.82	54.08	69.08	69.14	60.82
TB-Xray	76.89	-	59.47	69.13	67.16	76.33	76.70	75.00
PN-Xray	74.39	-	55.50	53.11	71.18	50.66	68.66	75.90
Average	**81.76**	-	58.88	69.30	70.71	73.09	76.35	69.24

primitives (c.f. Example 1). Furthermore, Snuba's performance degrades if the size of the development set is not sufficiently high. Our experiments showed that indeed, if we increase the development set size for Snuba from 10 to 100 (10x increase) for the PN-Xray dataset, the performance jumps from 55.50% to 67.84%. In comparison, GOGGLES still performs better with a development set size of only 10 images. (3) We can only use Snorkel on CUB, as CUB is the only dataset that comes with annotations that we can leverage as labeling functions. These labeling functions may be considered perfect in terms of coverage and accuracy since they are essentially human annotations. GOGGLES uses minimal human supervision and still outperforms Snorkel on CUB.

2.5.3 Ablation Study

We conduct a series of experiments to understand the goodness of different components in GOGGLES, including the proposed affinity functions and the proposed class inference method. Results are shown in Table 2.1.

Goodness of Proposed Affinity Functions. We compare GOGGLES affinity functions with the two common methods of obtaining the distance between two images: HOG and Logits. We use the two baseline methods to generate affinity matrices and run GOGGLES' inference module on them. GOGGLES' affinity functions outperform the other two on almost all datasets. This is because GOGGLES's affinity functions covers features at different scales and locations.

Goodness of Proposed Class Inference. We compare GOGGLES' inference module with three representative clustering methods: K-means, GMM, and Spectral co-clustering. All methods use the GOGGLES affinity matrix as input data. Note that the three clustering methods are not able to map the clusters to the classes automatically. As we would like to see the absolute best performance of the baseline clustering approaches, we use the

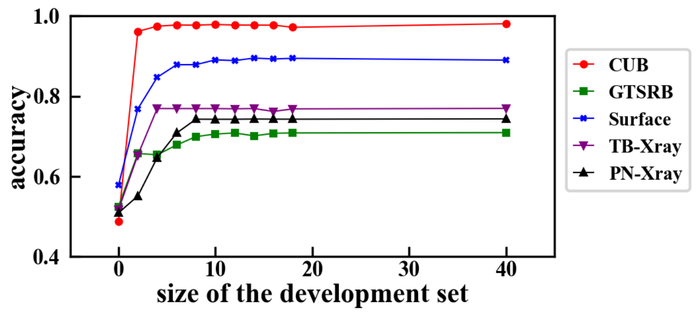


Figure 2.8: Accuracy w.r.t. development set size.

optimal “cluster-class” mapping for all baselines. GOGGLES’s inference module has the best average performance. The primary reason for our improvement over generic clustering methods is that our generative model adapts to the design of our affinity matrix. Specifically, our generative model is better at (1) handling the high-dimensionality through using the hierarchical structure and reducing the parameters in the base model by using diagonal covariance matrices; and (2) selecting affinity functions through the ensemble model (c.f. Section 2.4.1).

In terms of running time, without parallelization, our generative model is α (the number of base models) slower than the GMM model (the best baseline method). However, in practice (and in our experiments), we can parallelize all of the base models using different slices of the affinity matrix.

2.5.4 Sensitivity Analysis

Varying Size of the Development Set. We vary the size of the development set from 0 to 40 to understand how it affects performance (Figure 2.8). As the development set size increases, the accuracy increases initially, but finally converges. This is expected as when the development set is small, the mapping obtained by Equation (2.14) has a low probability being the optimal as predicted in Figure 2.7. When the development set size is large enough, the mapping given by Equation (2.14) converges to the optimal mapping, so the accuracy converges. Another observation is that datasets with higher accuracy converge at a smaller development set size. For example, the CUB dataset has an accuracy of 97.63% and its accuracy converges at a development set size of 2, while the GTSRB dataset requires a development set size of 8 to converge as it achieves an lower accuracy of 70.75%. Finally, the empirical size of the development set required to converge is much smaller than the theory predicted in Figure 2.7. A development set with 5 examples per class enough for all datasets.

Varying Number of Affinity Functions. We vary the number of affinity functions to study

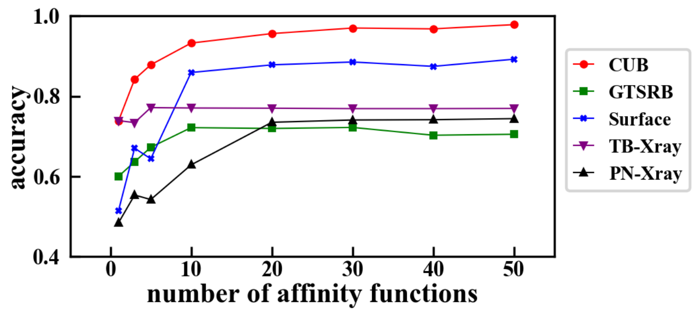


Figure 2.9: Accuracy w.r.t. varying number of affinity functions.

its effects on the results (Figure 2.9). Accuracy increases as the number of affinity functions increases for all datasets. This is understandable as more affinity functions brings more information that the inference module can exploit.

2.5.5 End-to-End Performance Comparison.

We also use the probabilistic labels generated by Snorkel, Snuba and GOGGLES to train downstream discriminative models following the similar approach taken in [13, 14]. Specifically, we use the VGG-16 as the downstream ML model architecture, and tune the weights of the last fully connected layers using cross-entropy loss. For training the FSL model, we use the same development set used by Snuba and GOGGLES for labeling. For training the upper bound model, we use the entire training set labels. The performance of each approach on hold-out test sets is reported in Table 2.2.

First, GOGGLES outperforms Snuba by an average of 21%, a similar number to the labeling performance improvement of 23% GOGGLES has over Snuba (c.f. Table 2.1), and the end model performance of Snuba is worse than FSL. This is because the labels generated by Snuba (59%) are only slightly better than random guessing, and having many extremely

Table 2.2: Comparison of end model accuracy on held-out test set. GOGGLES uses only 5 labeled instances per class but is only 7% away from the supervised upper bound (in gray) which uses the ground-truth labels of the training set.

Dataset	FSL	Snorkel	Snuba	GOGGLES	Upper Bound
CUB	84.74	87.85	56.32	95.30	98.44
GTSRB	90.72	-	70.11	91.54	98.94
Surface	76.00	-	51.67	83.33	92.00
TB-Xray	66.42	-	62.71	70.90	82.09
PN-Xray	68.28	-	62.19	69.06	74.22
Average	77.23	-	60.60	**82.03**	89.14

noisy labels can be more harmful than having fewer labels in training an end model. Second, GOGGLES outperforms the fine-tuned state-of-the-art FSL method (c.f. Section 2.5.1) by an average of 5%, which is significant considering GOGGLES is only 7% away from the upper bound. Third, not surprisingly, the more accurate the generated labels are, the more performance gain GOGGLES has over FSL (e.g., the improvements are more significant on CUB and Surface, which have higher labeling accuracies compared with other datasets).

This experiment demonstrates the advantage GOGGLES has over FSL and data programming systems — GOGGLES has the exact same inputs compared with FSL (both only have access to the pre-trained VGG-16 and the development set), and does not require dataset-specific labeling functions needed by data programming systems.

2.6 RELATED WORK

ML Model Training with Insufficient Data. Semi supervised learning techniques [6] combine labeled examples and unlabeled examples for model training; and active learning techniques aim at involving human labelers in a judicious way to minimize labeling cost [36]. Though semi-supervised learning and active learning can reduce the number of labeled examples required to obtain a competent model, they still need many labeled examples to start. Transfer learning [7] and few-shot learning techniques [8, 9, 10, 11] often use models trained on source tasks with many labeled examples to help with training models on new target tasks with limited labeled examples. Not surprisingly, these techniques often require users to select a source dataset or pre-trained model that is in a similar domain as the target task to achieve the best performance. In contrast, our proposal can incorporate several sources of information as affinity functions.

Data Programming. Data programming [12], and Snuba [14] and Snorkel [13] systems that implement the paradigm were recently proposed in the data management community. Data programming focuses on reducing the human effort in training data labeling, and is the most relevant work to ours. Data programming ingests domain knowledge in the form of labeling functions. Each labeling function takes an unlabeled instance as input and outputs a label with better-than-random accuracy (or abstain). As we show in this work, using data programming for image labeling tasks is particularly challenging, as it requires images to have associated metadata (e.g., text annotations or primitives), and a different set of labeling functions is required for every new dataset. In contrast, affinity coding and GOGGLES offer a domain-agnostic and automated approach for image labeling.

Other Related Work in Database Community. Many problems in database community share similar challenges to our work. In particular, data fusion/truth discovery [37, 38], crowdsourcing [39], and data cleaning [40], in one form or another, all need to reconcile information from multiple sources to reach one answer. While the information sources are assumed as input in these problems, labeling training data faces the challenge of lacking enough information sources. In fact, one primary contribution of GOGGLES is the affinity coding paradigm, where each unlabeled instance becomes an information source.

2.7 CONCLUSION

We proposed affinity coding, a new paradigm that offers a domain-agnostic way of automated training data labeling. Affinity coding is based on the proposition that affinity scores of instance pairs belonging to the same class on average should be higher than those of instance pairs belonging to different classes, according to some affinity functions. We build the GOGGLES system that implements the affinity coding paradigm for labeling image datasets. GOGGLES includes a novel set of affinity functions defined using the VGG-16 network, and a hierarchical generative model for class inference. GOGGLES is able to label images with high accuracy without any domain-specific input from users, except a very small development set, which is economical to obtain.

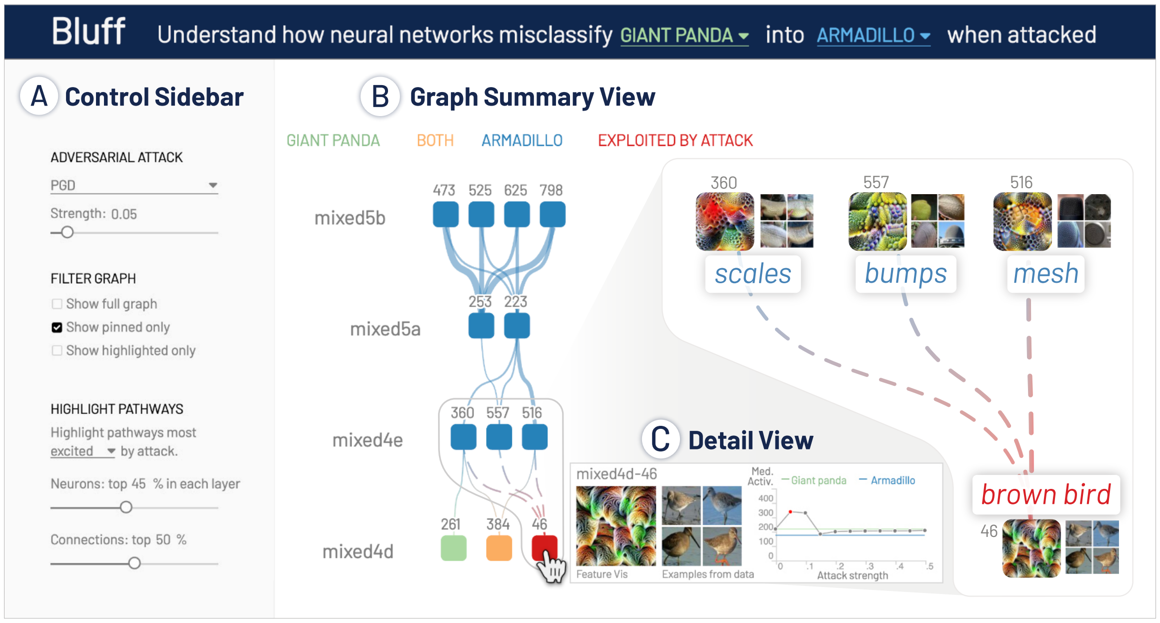
CHAPTER 3
BLUFF: INTERACTIVELY DECIPHERING ADVERSARIAL ATTACKS ON DEEP NEURAL NETWORKS

Deep neural networks (DNNs) are now commonly used in many domains. However, they are vulnerable to *adversarial attacks*: carefully-crafted perturbations on data inputs that can easily fool a model into making incorrect predictions. Despite significant research on developing DNN attack and defense techniques, people still lack an understanding of how such attacks penetrate a model's internals. For example, which neurons are exploited by an attack to fool a model into misclassifying an *ambulance* as a *street sign*? Is a stronger attack harming the same neurons as a weaker attack, or are they completely different? We present BLUFF, an interactive system for visualizing, characterizing, and deciphering adversarial attacks on vision-based neural networks. BLUFF allows people to flexibly visualize and compare the activation pathways for benign and attacked images, revealing specific mechanisms that adversarial attacks employ to inflict harm on a model. We present neural network exploration scenarios where BLUFF helps us discover multiple surprising insights into the vulnerability of a prevalent, large-scale image classifier, such as how atypical neuron activation pathways are exploited by attacks, and how class similarity correlates with exploitation intensity. Our findings help inform future research on designing models that are more robust against attacks. BLUFF is open-sourced and runs in modern web browsers.

3.1 INTRODUCTION

Deep neural networks (DNNs) have been a major driving force behind recent technological breakthroughs in a wide spectrum of applications [41, 42, 43, 44, 45, 46]. However, they have been found to be highly vulnerable to *adversarial attacks*: typically small, human-imperceptible noise injected into inputs can easily fool DNNs into arriving at incorrect predictions [47, 48, 49, 50]. This jeopardizes many DNN-based technologies, especially in security and safety-critical applications such as autonomous driving and data-driven healthcare. To make deep learning more robust against such attacks, it is essential to understand how the attacks permeate DNN models [51, 52].

Interpreting and ultimately defending against adversarial attacks remain fundamental research challenges. Due to DNNs' complex model architectures and their huge number of parameters, they are often considered to be "unintelligible." When such a model is


Bluff Understand how neural networks misclassify GIANT PANDA into ARMADILLO when attacked

A Control Sidebar

ADVERSARIAL ATTACK
PGD
Strength: 0.05

FILTER GRAPH
☐ Show full graph
☑ Show pinned only
☐ Show highlighted only

HIGHLIGHT PATHWAYS
Highlight pathways most excited by attack.
Neurons: top 45 % in each layer
Connections: top 50 %

Figure 3.1: With BLUFF, users interactively visualize and interpret how adversarial attacks penetrate a deep neural network (DNN) to induce incorrect outcomes. Here, a user inspects why INCEPTIONV1 misclassifies adversarial **giant panda** images, crafted by the *Projected Gradient Descent* (PGD) attack, as **armadillo**. BLUFF reveals that PGD has successfully perturbed pixels to induce the "*brown bird*" feature, an appearance more likely shared by an armadillo (small, roundish, brown body) than a panda. Such a feature helps activate more features in subsequent layers that contribute to the (mis)classification of armadillo (e.g., "*scales*," "*bumps*," "*mesh*"). Altogether, these neurons and their connections form adversarial pathways that overwhelm the benign panda pathways, leading to the ultimate incorrect prediction. **(A) Control Sidebar** allows users to specify what data to be included, filtered, and highlighted. **(B) Graph Summary View** visualizes pathways most activated or changed by an attack as a graph, where nodes are neurons and edges are their connections. When hovering over a neuron, **(C) Detail View** displays its feature visualization, representative dataset examples, and activation patterns over attack strengths.

attacked, it is difficult to pinpoint the parts of the model that the attack is exploiting, let alone to understand how such exploitations lead to incorrect outcomes [53]. Also, there is a lack of research in understanding how varying attack "strengths" influence neurons' activation patterns, despite attack strength being a user-specified hyperparameter [54]. For example, does a stronger attack activate the same neurons as a weaker attack, or are these sets completely different?

To address the above challenges, we develop BLUFF, an interactive visualization tool for discovering and interpreting how adversarial attacks mislead DNNs into making incorrect decisions. Our main idea is to visualize **activation pathways** within a DNN traversed by the signals of benign and adversarial inputs. More technically, an *activation pathway* consists of neurons (i.e., *channels* or *features*) that are highly activated or changed by the input images, and the connections among the neurons. BLUFF finds and visualizes where a model is exploited by an attack, how they are used, and what impact the exploitation has on the final prediction, across multiple attack strengths. As demonstrated in Figure 3.1, BLUFF visualizes the activation pathways commonly taken by 911 **adversarial panda** images (from ImageNet ILSVRC 2012 [3]) as they pass through INCEPTIONV1, a prevalent, large-scale image classifier. Crafted by the *Projected Gradient Descent* (PGD) attack [54], such images contain human-imperceptible pixel perturbations that would cause INCEPTIONV1 to confidently misclassify them as **armadillos**. BLUFF reveals that the perturbation has successfully induced the "*brown bird*" feature, an appearance more likely to be shared by an armadillo (small, roundish, brown body) than a panda. Such a feature in turn help activate more features in subsequent layers that contribute to the (mis)classification of armadillo (e.g., "*scales*," "*bumps*," "*mesh*"). Altogether, these neurons and their connections form *adversarial* pathways that overwhelm the benign panda pathways, leading to the ultimate incorrect prediction.

Contributions. In this work, we contribute:

- **BLUFF, an interactive system for summarization and interpretation** of how adversarial perturbations penetrate DNNs to induce incorrect outcomes in INCEPTIONV1 [55], a large-scale, prevalent image-classification model. BLUFF interactively visualizes and compares the activation pathways of benign and attacked inputs in a graph representation (Section 3.7.2), which allows users to freely track, explore, and interpret the pathways exploited by an attack.
- **Visual characterization of activation pathway dynamics.** Adversarial perturbations stimulate changes in the activation pathways typically used for benign inputs. For example, an attack can inhibit neurons detecting important features for the benign class to suppress

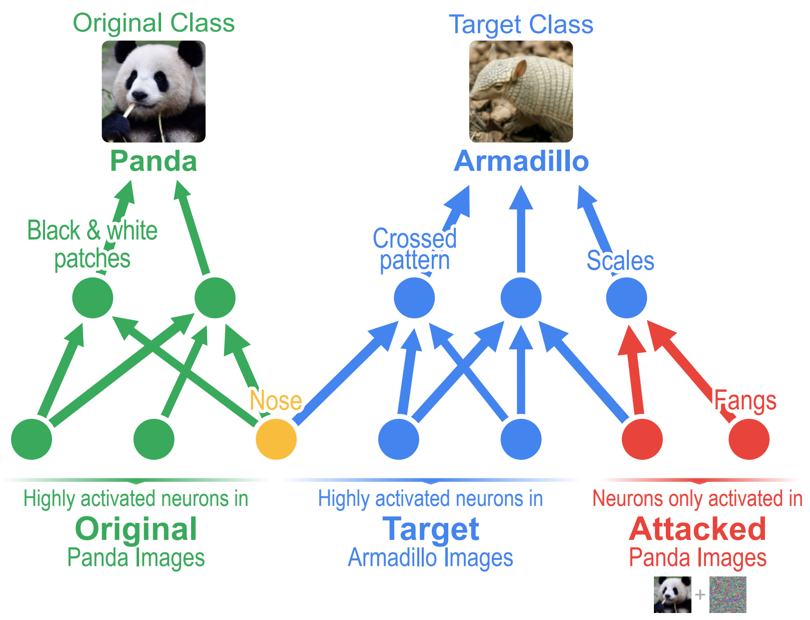


Figure 3.2: Adversarial attacks confuse DNNs to make incorrect predictions, e.g., attacking benign *panda* images so they are misclassified as *armadillo*. We aim to understand where such attacks occur inside the model and what features are used.

activation signals from traversing the network. Such changes can induce cascading neuron exploitations that ripple through the whole network. BLUFF visualizes and highlights activation pathways exploited by an attack (Figure 3.2) and shows how they mutate and transmit throughout a network (Section 3.7.2).

- **Interactive comparison of attack escalation.** BLUFF allows users to understand how activation pathways would morph over increasing attack strengths (Section 3.7.2). With BLUFF's interactive comparison over different attack strengths, users can understand the essence of an attack (e.g., common trends of an attack across all strengths) and its multi-faceted characteristics (e.g., various strategies that different strengths may employ).
- **An open-source, web based implementation.** BLUFF is open-sourced to broaden people's access to interpretability of adversarial attacks on deep neural networks. The demo and code is available at the public link: https://poloclub.github.io/bluff.
- **Discovery usage scenarios.** Using BLUFF, we investigate how BLUFF can help discover multiple surprising insights into the vulnerability of deep neural networks, such as how unusual activation pathways may be exploited by attacks, and how class similarity and attack exploitation intensity may be correlated. Our findings help inform future research on designing models that are more robust against attacks (Section 3.8).

3.2 BACKGROUND: ADVERSARIAL ATTACKS ON DNNS

Adversarial attacks aim to confuse a given DNN model into making incorrect predictions by adding carefully crafted, but seemingly imperceptible perturbations to the input. One of the first gradient-based attacks proposed was the Fast Gradient Sign Method (FGSM) [48], which perturbs the input by adding the product of a small constant and the direction of the gradient of the output with respect to the input. DeepFool [56] computes the minimal perturbation required to flip the classification label by assuming the DNN decision boundary to be hyperplanar. Carlini-Wagner's L_2 attack [57] was another highly effective optimization-based attack that introduced a relaxation term to the perturbation minimization problem based on a differentiable surrogate of the model.

In this work, we focus on *targeted* adversarial attacks, since they pose a more severe threat to practical applications of deep learning [47]. A targeted adversarial attack tries to induce a model into making a prediction of the attacker's choosing. More specifically, given a benign input instance x, a targeted adversarial attack aims to find a small perturbation δ that changes the prediction of model $\mathcal{M}$ to a target class t different from the true class y, i.e., $\mathcal{M}(x+\delta)=t$, where $t \neq y, ||\delta|| \leq \epsilon$. We call ϵ the perturbation strength with respect to a specific norm $|| \cdot ||$.

Projected Gradient Descent (PGD) [54] is one of the *strongest* first-order targeted attacks in adversarial machine learning literature [58]. Hence, in this work, we examine the PGD attack for our visualizations. PGD iteratively minimizes a loss function $L(x,t)$, such as cross-entropy, which computes the distance between the softmax of the logit layer and the one-hot representation of the target class t. Let us denote the logit layer of model $\mathcal{M}$ as $f(x,\mathcal{M})$. In each iteration i, PGD computes the direction of perturbation by taking the sign of the gradient of loss function L with respect to the current perturbed instance x^i. It then performs a projection step back to the feasible set, i.e., within ϵ hypersphere of the original instance, while remaining a valid image:

$$x^{i+1} = Proj\left(x^i - \alpha \cdot \text{sign}(\nabla_{x^i} L(x^i, t))\right)$$

3.3 RELATED WORKS: NEURAL NETWORK INTERPRETABILITY

Deep neural networks have often been used as "black boxes" due to their complex and "unintelligible" internal structure. This is problematic when a model is attacked, or when it makes incorrect predictions, since people lack the understanding to correct and protect them. For example, DNNs are known to be highly vulnerable to human-imperceptible adversarial perturbations on input data [48, 54, 59]. To make models robust against attacks, it is critical to understand how the models are affected, so as to inform the development of effective defenses. While there has been increasing demands for interpretability for DNNs [60], less work has focused on interpreting and characterizing and adversarial attacks on DNNs. Below, we provide an overview of existing methods for interpreting DNNs and adversarial attacks.

3.3.1 Neural Network Interpretability

A natural approach to understand how neural networks represent data inside their internal structure is to study a network's constituent neurons and their activations. Neurons, sometimes referred to as *channels* in convolutional neural networks, are known to selectively activate in response to particular learned features. To glean insight into what concepts a neuron is detecting, the popular approach called *feature visualization* [61, 62, 63, 64] algorithmically generates visualizations that maximize a particular neuron. Inspecting multiple feature visualizations helps provide evidence for what concepts a model has learned more generally [65]. Beyond qualitative interpretations for neuron activations, several recent techniques have been proposed to quantitatively understand neurons and their features. For example, TCAV [66] quantifies the degree to which a concept (e.g., striped pattern) would be important for a class (e.g., zebra) by measuring how sensitive the prediction for the class can be to the concept based on neuron activations. Network Dissection [67] and Net2Vec [68] propose to quantify interpretability by measuring alignment between the neuron activations and concept features.

Motivated by the fact that neural networks build up increasingly complex features through their multiple layers and connections among neurons, several approaches [69, 65] propose visualization techniques to interpret the connections between neurons. Circuits [65] visually explains how the connections between neurons can be extracted by model weights and how higher-level concepts can be constructed and calculated through these connections. Summit [69] visualizes what a model has learned for a class using a graph representation, where nodes are neurons and edges connect neurons based on their influence. Inspired by previous work, we also leverage feature visualization, and expand on previous graph-based approaches to interpretability by allowing users to summarize and interactively compare both benign and attack inputs across varying strengths.

3.3.2 Interpretability for Adversarial Attacks

Research on machine learning security and adversarial attacks has received great attention over recent years [47, 70, 54, 48, 59]. In response, there has been a growing need for understanding how adversarial attacks permeate neural networks to protect them. Goodfellow et al. [48] ascribe the primary cause of neural network vulnerability to adversarial attacks to the linear behavior of DNN models in high-dimensional space. They discovered that adversarial perturbations strongly align with the model's weight vectors. Papernot et al. [71] propose a distillation approach to reduce the explanation space of DNNs by training a surrogate model on top of the primary model to evade attacks.

Recent work has developed interactive visualization tools to help interpret how adversarial attacks affect models [53, 72, 73]. AEVis [53] and its extension [72] propose to extract critical neurons and their connections, coined as a *data path*, for benign and adversarial examples. They formulate the extraction problem as a subset selection optimization problem, and demonstrate the method on small sets of images; it is unclear how the approach may scale to larger datasets that BLUFF operate on (e.g., 900+ adversarial images for a single class). BLUFF also provides new techniques for discovering, summarizing, visualizing, and comparing activation pathways, enabling novel analysis (e.g., based on neuron inhibition and excitation) and discoveries (e.g., correlation between class similarity and neuron exploitation intensity).

3.4 DESIGN CHALLENGES FOR DECIPHERING ATTACKS

Our goal is to build an interactive visualization tool to help users discover and interpret how adversarial attacks penetrate the internals of DNNs to induce incorrect predictions. To develop BLUFF, we identified the following four key challenges:

C1 Entangled activation pathways. Due to the complex structure of DNNs (e.g., numerous layers, neurons, and connections), *benign* activation pathways could significantly overlap with adversarial pathways, as some neurons are "*multi-purpose*" (also referred to as *polysemantic* [65]), i.e., they detect multiple concepts at the same time [67]. In addition, two classes with strong resemblance (e.g., *diamondback snake* and *vine snake*) could share similar pathways too. Effective visualization of these entangled pathways calls for new strategies different from those designed for explaining single inputs [53] (e.g., a single image) or inputs of the same kind [69] (e.g., all images from the same class).

C2 Comparing multiple activation pathways. Comparison is core to detecting and defending against adversarial attacks [48, 59, 54]. Beyond a single activation pathway, how do we visualize and interpret what multiple pathways detect? Can visualization explain the differences and similarities between how benign and attacked inputs are represented within neural networks? Interactive systems should visualize high-level overviews of activation pathways while supporting drilling-down into specific subpaths to identify what they detect and how they contribute to the prediction. Interactive comparison for activation pathways across benign and attacked inputs can help people understand how adversarial attacks permeate within DNNs, shedding light on how specific groups of neurons are exploited to inflict harm on a model.

C3 Summarizing attack characteristics. Existing work for interpreting adversarial attacks on deep neural networks often focus on visualizing the activation patterns for a single adversarial input [63, 73]. While this can be useful, it can be difficulty to generalize to help understand how attacks behave on other inputs (e.g., images within the same class or different classes). To better understand the characteristics of an attack, it is necessary to observe the behavior of attacks over multiple inputs, classes, and attack hyperparameters. Specifically, adversarial attacks typically have a specified strength hyperparameter. Visualizing how an attack changes the activation pathway as we vary the attack strength provides stronger insight into how the attack works generally.

C4 Barrier of entry for interpreting and deciphering adversarial attacks. The visualization community is contributing a variety of methods and tools to help people more easily interpret different kinds of DNNs [67, 53, 69, 63, 61, 74, 60]. Efforts that aim to support deciphering adversarial attacks, however, are relatively nascent [67, 53, 73]. Can we make interpreting adversarial attacks more accessible to everyone, perhaps by following the footsteps of prior success from the community?

3.5 DESIGN GOALS

Based on the identified design challenges, we distill the following main design goals for BLUFF, an interactive visualization tool for deciphering adversarial attacks on deep neural networks.

G1. Visualization of exploited activation pathways. We aim to design and develop an interactive interface to support users to freely explore and interpret activation pathways, helping decipher how adversarial attacks permeate models. Identifying both excited and inhibited neurons that response differently between benign and attacked inputs, helps discover where and how a model is exploited by an attack to induce incorrect predictions (Item **C1**). To better understand what each neuron detects, we pair every neuron with its feature visualization (synthetic examples that maximize a particular neuron), and dataset examples that together provide strong evidence for what particular neurons and their connections are detecting.

G2. Interactive comparison and drill-down for multiple activation pathways. To unravel complex activation pathways of benign and attacked inputs, BLUFF provides an overview that visualizes multiple activation pathways simultaneously, giving users a quick sense of the similarity between the internal representations of different inputs. BLUFF's interactivity enables users to highlight, filter, and drill-down on multiple activation pathways. These complementary views let people compare how attacks deviate from benign inputs and pinpoint specific branching layers within a network (Item **C2**).

G3. Understanding attacks with varying strengths. Understanding model vulnerability under different attack strategies helps inform how to create more general and robust defenses [70, 71, 56]. To achieve this, BLUFF provides a *comparison mode* that helps people summarize and visualize the differing attack strategies under varying strengths (Item **C3**). For example, each neuron may be visually encoded based on which attack strengths they correspond to, characterizing the neuron's overall vulnerability.

G4. Cross-platform deployment with standard web technologies. BLUFF uses standard web technologies so that it is broadly accessible to people without specialized computational resources. To support reproducible research, BLUFF is also open-sourced (Item **C4**).

3.6 UNIFYING MULTIPLE GRAPH SUMMARIES

BLUFF builds upon the *attribution graph* concept introduced in [69]. The original attribution graph visualizes the neurons and their connections within a DNN that respond strongly to images from a *single* class. We adapt this idea to scalably aggregate influential *activation pathways* across *multiple* contexts. Specifically, BLUFF identifies the activation pathways with the most "important" neurons for:

1. All benign images belonging to the **original** class, on which the targeted attacks are performed.
2. All benign images belonging to the **target** class, which the attacks try to flip the label to.
3. All successfully **attacked** images; further, this set of images can be different for different attack strengths, as we compute the activation pathways for multiple attack strengths.

In this section, we define what "important" neurons constitute influential activation pathways in a DNN, and how we compute them. We also provide insights regarding the various computational and conceptual decisions we make when designing BLUFF.

3.6.1 Dataset and Model

In this work, we demonstrate the effect of adversarial attacks for images from the training split of the ILSVRC2012 dataset [3], also known as ImageNet. The dataset consists of over 1.2 millions images across 1000 classes. BLUFF visualizes the internals of the InceptionV1 model [55], a prevalent, large-scale convolutional neural network that achieves top-5 accuracy of 89.5% on ImageNet. InceptionV1 is composed of multiple inception modules which are distinct groups of parallel convolutional layers. The output of each inception module is labeled as "mixed{number}{letter}," where the {number} and {letter} denote the hierarchical location of the corresponding layer in the network; for example, mixed3a (the first inception module) or mixed5b (the last inception module). In InceptionV1, there are 9 such layers: mixed3{a,b}, mixed4{a,b,c,d,e}, and mixed5{a,b}. Similar to [69], and following existing interpretability literature [61, 75], we consider the 9 mixed layers for our visualization. In our implementation, we use the InceptionV1 model provided by Tensorflow Lucid[1], which has also been used in several other neural network interpretability works [76, 63, 77].

3.6.2 Adversarial Attack

To understand the effects of an adversarial attack on a DNN, BLUFF aims to provide a holistic summarization of an attack across different levels of perturbation budget provided to an attacker. We do this by visualizing the impact of an attack across varying attack strengths. BLUFF visualizes the influence of the PGD attack [54], one of the strongest and most effective first-order attacks from the adversarial ML literature [58]. The PGD attack computes adversarial perturbations which are bound by a given l_p norm, thus allowing us to modulate the attack strength. We use the l_2 norm attack for our visualization, and vary the attack strength from 0.0 (no attack) to 0.5 (strong attack) in increments of 0.05 in the normalized color space. This corresponds to a maximum change of approximately 64 units in the pixel space, which ranges from 0 to 255. We leverage the CleverHans[2] python library, a popular adversarial ML toolkit, for performing the adversarial attacks. Since there are multiple attacks that need to be performed for various attack strengths, we efficiently parallelize all of the attacks using Apache Airflow[3], a library for performing distributed computing.

Even though we discuss specific choices for the dataset, model and attack, the methods presented hereon developed for BLUFF are easily extensible to other scenarios with minor

[1]Tensorflow Lucid: https://github.com/tensorflow/lucid

[2]CleverHans: https://github.com/tensorflow/cleverhans

[3]Apache Airflow: https://airflow.apache.org/

changes in the implementation.

3.6.3 Computing Neuron Importance

It has been shown that neurons at successive layers of a DNN learn to detect semantic concepts with increasing levels of complexity [61]. Our aim with BLUFF is to not only identify the most "interesting" neurons that are essential for distinguishing the **original** class of images, but also compare and contrast them in relation to the neurons that are correspondingly important for a **target** class. Finally, we also want to identify the neurons at each layer that are exploited by an adversarial **attack** that flip the final prediction from the original to the target class. To do so, we follow a similar approach to [69], but adapt our implementation to scale to the multiple sets of images.

To begin, we consider the DNN model $\mathcal{M}$ (InceptionV1 in our case), having Q layers, where $\mathcal{L}^q$ is the q'th layer of $\mathcal{M}$. Let $Z^q \in \mathbb{R}^{H_q, W_q, D_q}$ be the output tensor of layer $\mathcal{L}^q$, where H_q, W_q and D_q are the height, width and depth dimensions respectively. This implies that the layer has D_q neurons. We further denote the d'th neuron and corresponding output channel in the layer as $\mathcal{N}_q^d$ and $\mathcal{C}_q^d$ respectively, where $d \in \{1, ..., D_q\}$ and $\mathcal{C}_q^d \in \mathbb{R}^{H_q, W_q}$. Also, we index the values in the channel $\mathcal{C}_q^d$ as $\mathcal{C}_q^d[h, w]$, where $h \in \{1, ..., H_q\}$ and $w \in \{1, ..., W_q\}$.

Now, given an image x_i that is passed through the model for inference, we find the maximum activation of each neuron induced by the image using the global max-pooling operation:

$$a_q^d[i] = \max_{h,w} \mathcal{C}_q^d[h, w]$$

This represents the relative induced magnitude by which the d'th neuron in the q'th layer maximally detects the corresponding semantic feature for image x_i. This technique of extracting maximal activation as a proxy for semantic features has also demonstrated tremendous predictive power in the data programming domain [1]. Finally, we pass all images from each of **original**, **target** and **attacked** datasets, where the **attacked** datasets are further differentiated by the value of attack strengths. For each set, we aggregate $a_q^d[i]$ values for all images and quantify the "importance" of each neuron according to each set of images, by the median value of induced maximal activation values, and rank the neurons by this *median-of-max* activation values. In BLUFF, we filter the neurons by top-k based on these values, as we will outline in Section 3.6.5.

Since this step involves performing the same operations multiple times for different sets of images (**original**, **target** and **attacked** for different attack strengths), we efficiently parallelize these steps using the Apache Airflow distributed computing library.

3.6.4 Computing Propagation of Influence

The neuron importances characterize the semantics detected at a local level for each layer. We further want to characterize how these semantic features propagate through the network, leading up to the final class label. In BLUFF, we visualize the influence of important neurons for different sets of images in a single view. We follow similar steps as [69] to compute the influence from each neuron in a given layer to each neuron in a successive layer. At a high-level, the influence of each neuron-neuron connection is proportional to the maximum convolution value of the channel activation corresponding to the source neuron, multiplied with its learned kernel tensor. We refer the reader to [69] for a more in-depth treatment of this approach.

Our implementation deviates from [69] in the last step, when aggregating influence values across several images. Instead of counting how many images have a given neuron-neuron connection in the top-k influences, we characterize the connection by taking the median influence across all images from a given set. We take this approach since we want to summarize the influence characteristics across multiple datasets (**original**, **target** and **attacked** for different attack strengths), and each dataset may have a different size. Simple counting may skew the results towards a particular dataset while the median value provides a characteristic aggregation of the influence scores.

3.6.5 Summarizing Pathways across Multiple Contexts

The InceptionV1 model consists of $\sim 5,500$ neurons (nodes) and ~ 3.1 million connections (edges) across all inception modules, which is beyond the realm of human cognition to comprehend concurrently. However, a significantly smaller number of them play a pivotal role in the final inference [69]. Hence, BLUFF summarizes these neurons and their connections as a graph, where the nodes are neurons and the edges are connections among them, inspired by the attribution graph proposed in [69]. A crucial distinction between [69] and this work is that the former only visualizes the model for a *single* class, whereas BLUFF analyzes *multiple* contexts. Now we discuss how BLUFF tackles the compounded task of summarizing the attribution graph across **original**, **target** and **attacked** contexts, including summarization of multiple attack strengths in a unified view.

BLUFF screens neurons that are important for both benign (*original* and *target* classes) or adversarial (*attacked* across all strengths) cases. For the benign case, we filter the top 10 neurons at each layer by neuron importance induced by the benign *original* and *target* class images. We show the union set of these neurons. We visualize the neurons *only important* for the *original* class as **green** nodes, and the neurons *only important* for the *target* class as **blue** nodes. Neurons which are important in *both* classes are shown as **orange** nodes. Correspondingly for the benign case, there could be a maximum of 20 *green*, *blue*, or *orange* nodes (all important neurons for *original* and *target* classes are separate), and a minimum of 10 such nodes (all important neurons are the same), for each layer in the BLUFF's Graph Summary View.

For the adversarial case, we consider 10 attack strengths in total, as outlined in Section 3.6.2. Since there are several attack strengths to summarize, for *each* attack strength, we filter the top 5 neurons from every layer that are *only important* for the successfully **attacked** images, and visualize the union set of these neurons as **red** nodes. Consequently, there can be a maximum of 50 *red* nodes (all 5 important neurons are different across all 10 attack strengths), and a minimum of 5 *red* nodes (all important neurons are same across all attack strengths), for each layer in the BLUFF Graph Summary View.

BLUFF allows users to further filter the attribution graph nodes and edges to their liking, which is further discussed in Section 3.7. We also engineered our implementation of summarizing the activation pathways in a modular fashion, such that it easily facilitates further tweaking of our design parameters for generating more complex views.

3.7 USER INTERFACE

From our design goals (Section 3.5), we present BLUFF, an interactive visualization system for deciphering adversarial attacks on deep neural networks. Figure 3.1 shows BLUFF's interface, consisting of the main views: the Control Sidebar, the Graph Summary View, and the Detail View. The following sections detail each view's features and how the views tightly interact with one another.

3.7.1 Control Sidebar

Included within the header and the control sidebar (Figure 3.1A) are user-specified controls for selecting which data are included, filtered, highlighted, and compared in the Graph Summary View.

Specifying a Benign-Attacked Class Pair

In the header (Figure 3.1, top), users can select a pair of the **original** and **target** class. BLUFF then generates the main visualization in the Graph Summary View for how neural networks misclassify **original** class images into the **target** class when attacked, by displaying the activation pathways of adversarial inputs.

Specifying an Attack

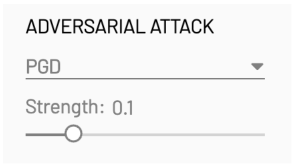


Figure 3.3: Specifying an attack.

The attack control panel allows users to manipulate the input images of the selected classes. Users can first select the adversarial attack method with a drop-down menu and then select the strength of the attack with a slider. This is shown in Figure 3.3.

Highlighting Activation Pathways

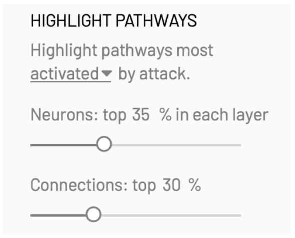
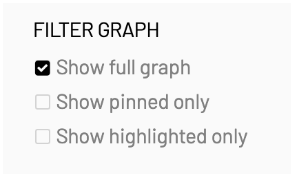

HIGHLIGHT PATHWAYS
Highlight pathways most activated by attack.
Neurons: top 35 % in each layer
Connections: top 30 %

Figure 3.4: Highlighting activation pathways.

The highlight control panel allows users to select which type of activation pathways to visualize: pathways that are most *activated*, most *changed*, most *inhibited*, or most *excited* by an attack. Users can also control how many neurons and connections of the pathways will be highlighted by two sliders. The first slider controls how many neurons will be highlighted. This is shown in Figure 3.4. For example, if a user selects "35%" in the neuron slider and "the most activated pathways" option, BLUFF highlights 35% of most activated neurons in each layer from all neuron groups: neurons of the original class, the target class, both classes, and those exploited by the attack—these groups are further described in Section 3.7.2. The second slider controls how many connections will be highlighted. For example, if a user selects "30%" in the connection slider and "the most activated pathways" option, BLUFF highlights 30% of the most activated connections out of all positive connections among the highlighted neurons between two adjacent layers. We select a certain percentage of neurons and edges from each layer because activation levels can vary in magnitude by layers. For example, in the INCEPTIONV1 model, neurons and their connections in the lower layers generally activate significantly more strongly than, and thus not directly comparable to, those in the higher layers.

Filtering Activation Pathways

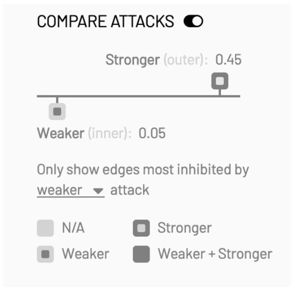


FILTER GRAPH
- [✓] Show full graph
- [ ] Show pinned only
- [ ] Show highlighted only

Figure 3.5: Filtering activation pathways.

Sometimes there may be too many neurons and edges to visualize at once. To avoid over-plotting and visual complexity, the filter control panel allows users to filter the activation pathways. BLUFF supports three filtering options: show the full graph, show only user-pinned neurons, and show only highlighted neurons. This is shown in Figure 3.5. Pinning a neuron displays its neuron number on its top edge.

Selecting Multiple Adversarial Attacks



COMPARE ATTACKS

Stronger (outer): 0.45

Weaker (inner): 0.05

Only show edges most inhibited by weaker attack

Figure 3.6: Selecting multiple adversarial attacks

The comparison control panel allows user to select two different attack strengths for comparison. After toggling on the comparison mode, the range slider controls two attack strengths: the left knob for the weaker attack, and the right knob for the stronger attack. Users can also select which edges to visualize: those inhibited by the weaker or stronger attack. This is shown in Figure 3.6.

3.7.2 Graph Summary View

The Graph Summary View is the main view of BLUFF (Figure 3.1B). It summarizes and visualizes *activation pathways*: the pathways of signal from input to output layer passing through highly activated neurons (i.e., channels) and their connections. We represent the pathways as a graph, where nodes are neurons and edges are their connections.

Spatially Grouping Neurons by Their Roles

Existing works [69, 61] have found that neurons are selectively activated to particular features and classes. Since we aim to understand why a DNN model misclassifies, we seek to group neurons spatially according to which class they are typically important to (i.e., highly activated). There are four groups of neuron in the BLUFF user interface (see Figure 3.1B): neurons that are highly activated by only the **original** class, by only the **target** class, by **both** classes, or those highly activated when only adversarial images are given, meaning that they are **exploited by attack**. We visually differentiate the groups with colors: the original class is green, the target class is blue, both classes are orange, and those exploited by attack are red.

Visualizing Activation Pathways

In the Graph Summary View, activation pathways are visualized in a zoomable and panable canvas. Neurons are positioned horizontally according to which groups they belong to, in the order of **original**, **both**, **target**, and **exploited by attack**, from left to right. Vertically, they are positioned based on the layers of the neural network. The topmost row corresponds to the last mixed network layer (i.e., mixed5b) which is closest to the output layer, whereas the bottom layer corresponds to the first mixed network layer (i.e., mixed3a) which is closest to the input layer. Each neuron is visualized as a rounded rectangle and colored by its corresponding group. Each connection between two neurons is visualized as a curved line, where the width is linearly scaled by the influence values computed as in Section 3.6.4.

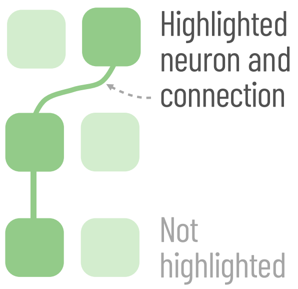


Figure 3.7: Neurons and their connections highlighted by BLUFF.

Out of all neurons, BLUFF highlights those that are most activated or changed based on the user-specified settings from the highlight control panel. BLUFF highlights neurons by coloring them darker and connections by making others invisible. This highlighting approach naturally visualizes the pathways of activation signal passing through neurons and their connections, while preserving the context of all potentially relevant neurons nearby. This approach can also help compare the neurons that are highly activated by benign against

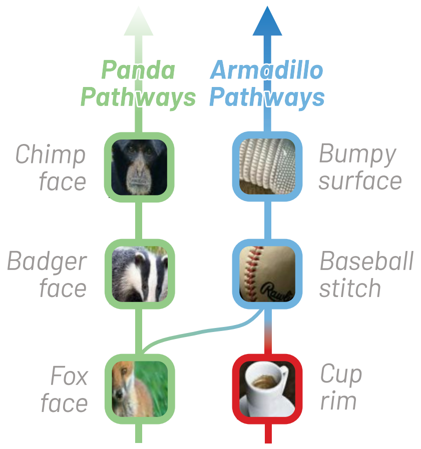


Figure 3.8: BLUFF visualizes neurons and connections that are **most activated** given adversarial **panda** images misclassified as **armadillo**. These activation pathways can be more interpretable with visual explanations such as *dataset examples*, patches of images that highly activate the corresponding neurons. For example, we can understand that the attack exploits a neuron detecting *cup rim* to fire *baseball stitch* neuron, so that the activation can keep flow towards the **armadillo** outcome.

adversarial images. All neurons displayed in BLUFF are already the most activated neurons in benign images, regardless of whether they are highlighted or not. On top of the neurons that are important for benign images, BLUFF highlights neurons that are highly activated by adversarial images. This helps distinguish neurons that are activated by benign as well as adversarial images, as compared to neurons only activated by benign images.

Activation Pathway Interpretability

To help users more easily interpret the concepts that a neuron is detecting, alongside each neuron, BLUFF shows (1) a *feature visualization*, an algorithmically generated image that maximizes the neuron's activation, and (2) *dataset examples*, cropped from real images in the dataset, that also highly activate the neuron [61]. Hovering on a neuron shows the corresponding feature visualization and dataset examples as seen in Figure 3.1C. These visual explanations help translate abstract activation pathways into the composition and flow of learned concepts—this is a primary mechanism that BLUFF uses to identify how a particular class is misclassified. For example, in Figure 3.8, BLUFF visualizes the activation pathways of adversarial **panda** images misclassified as **armadillo**. The adversarial images induce the model to detect a "*cup rim*" feature, which successively fires the "*baseball stitch*" neuron and "*bumpy surface*" neuron, both of which are typically important to **armadillo**. This explains how the model misrecognizes features for armadillo, ultimately predicting the adversarial panda images as armadillos. In the intermediate layer, the "*fox face*" neuron slightly fires a "*baseball stitch*" neuron, causing armadillo to be detected.

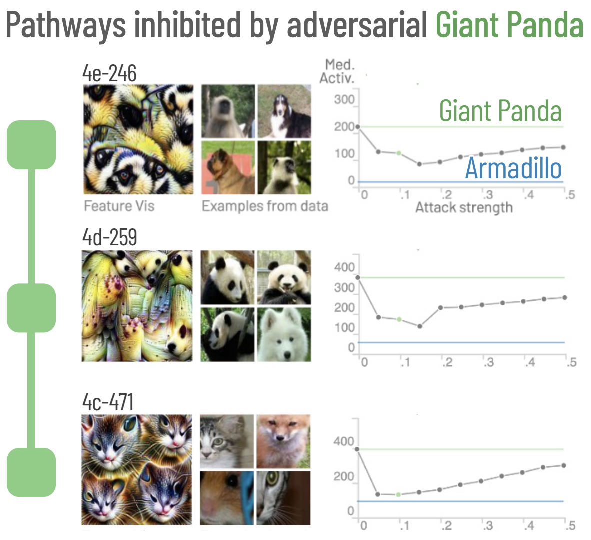


Figure 3.9: Neurons and connections most inhibited by an attack to fool a model into misclassifying panda as armadillo. These vulnerable neurons that detect typically important features for panda such as "*pug face*," "*panda face*," and "*cat face*" become less detected, preventing the model finding panda-ness from the inputs.

Investigating Vulnerable Activation Pathways

An adversarial input is often a slightly perturbed version of a benign input. Thus, the activation pathways of an benign image and those of that image's corresponding attacked version would be very similar at the input layer [53]. However, the two pathways at the output layer are decidedly different — the *benign* pathways lead to the prediction of the **original** class, and the *adversarial* pathways lead to the prediction of the **target** class. Given the similar starting points but different outcomes, the adversarial activation pathways must have veered off the benign pathways somewhere inside the model. Which neurons and connections are the most vulnerable and contributing to such deviations? And how do we interpret the meaning of such vulnerable pathways?

To answer these research questions, we focus on detecting and characterizing the changes in the activation pathways between benign and attacked data, which could help us better understand the attributing factors for the resulting (mis)classification. In BLUFF, users can highlight the neurons and connections they are changed the most by an attack, by using the highlight control panel. There are two main types of pathway changes: **inhibition** and **excitation**. If a pathway for the original class is *inhibited* by an attack, this means the pathway's constituent neurons (features) are activated *much less* than they are supposed to (i.e., fewer original features). Figure 3.9 shows such as an example, where adversarial **panda** images inhibits neurons that detect "*black fur*," "*panda face*," and "*cat face*," and also their connections across the DNN. Conversely, if a pathway of the target class is *excited*

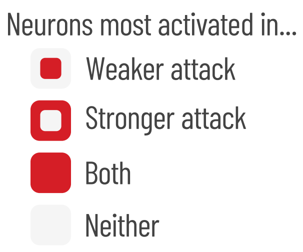


Figure 3.10: Neurons of different attack strengths in the comparison mode.

by an attack, this means the constituent neurons are activated *much more* than expected (i.e., more target features). Figure 3.1 shows such an example, where features and connections important for the prediction of armadillo, such as "*scales*," "*bumps*," "*mesh*," and "*brown bird*" (thanks to its similarity to armadillos' roundish, brown body).

The amount of changes in the activation of neurons and connections, whether inhibited or excited, is computed by comparing the activations of benign and adversarial inputs. Let z_{benign} be a neuron's activation for some given benign images, and $z_{attacked}$ be that for some attacked images, as described in Section 3.6.3. Then, the activation *inhibition* of the neuron is computed as $z_{benign} - z_{attacked}$, and the activation *excitation* is computed as $z_{attacked} - z_{benign}$. BLUFF also computes the absolute activation difference $|z_{attacked} - z_{benign}|$, to help users quickly identify both *inhibited* and *excited* neurons at the same time (we collectively call these *vulnerable* neurons). The changes in neuron connections are computed similarly. Let i_{benign} be a connection's influence for some given benign images, and $i_{attacked}$ be that for some given attacked images, as described in Section 3.6.4. Then, the amount of the activation inhibition is computed as $i_{benign} - i_{attacked}$, the activation excitation as $i_{attacked} - i_{benign}$, and the difference as $|i_{attacked} - i_{benign}|$.

Comparing Adversarial Attacks of Different Strengths

Beyond a single attack, does a stronger or weaker attack have the same or different impact on a neuron? BLUFF offers the *comparison mode*, to help visualize and compare the changes in the resulting pathways of a weaker attack and those of a stronger attack. We visually encode the neurons based on which attack strengths they corresponded to, drawing inspiration from Alper et al [78]. Each neuron consists of an inner and an outer rectangle: the *inner* rectangle is colored when the neuron is in the activation pathways of the *weaker* attack; whereas the *outer* rectangle is colored when the neuron is in the activation pathways of *stronger* attack. Thus, our design can visually encode all four possible neuron responses — a neuron is activated by: (1) weaker attacks only; (2) stronger attacks only; (3) both attack strengths; and (4) neither attack strengths. This is shown in Figure 3.10.

Pinning User-specified Neurons and Connections

When exploring the activation pathways, users can freely pin neurons that are interested in by simply clicking them. When pinned, the neuron number will be displayed on the top of the neuron. Clicking a pinned neuron again unpins it. Users can filter the graph to show only the pinned neurons and their connections using the “Show pinned only” option in the filter control panel.

3.7.3 Detail View

When hovering on each neuron, a detail view pops up to provide more information about the neuron (as seen in Figure 3.1C), display the feature visualization for the neuron and four example image patches to represent the detected concept the neuron detects (as described in Section 3.7.2). The detail view also presents a line plot that displays the *median activation values* for: (1) images of the original class (green); (2) images of the target class (blue); and (3) adversarial images (gray). The horizontal axis represents the attack strength. Thus, each data point in the line plot represents a median activation value at a particular attack strength.

3.7.4 System Design

BLUFF uses the standard HTML/CSS/JavaScript stacks, and D3.js[4] for SVGs rendering. We ran all the backend code that computes neuron importance (Section 3.6.3) and connection scores (Section 3.6.4) on an NVIDIA DGX-1 workstation equipped with 8 GPUs (each with 32GB memory), 80 CPU cores, and 504GB RAM. It takes about 30 minutes to generate the full backing graph summary for one class pair (about 2500 images in total) for each attack strength; a majority of that time is spent on running the attack algorithm. All our source code is available at https://github.com/poloclub/bluff.

3.8 DISCOVERY USAGE SCENARIOS

We provide four example usage scenarios for BLUFF, demonstrating how it can enhance the understanding of adversarial attacks, and support discoveries that reveal the attacks' strategies to confuse a DNN model.

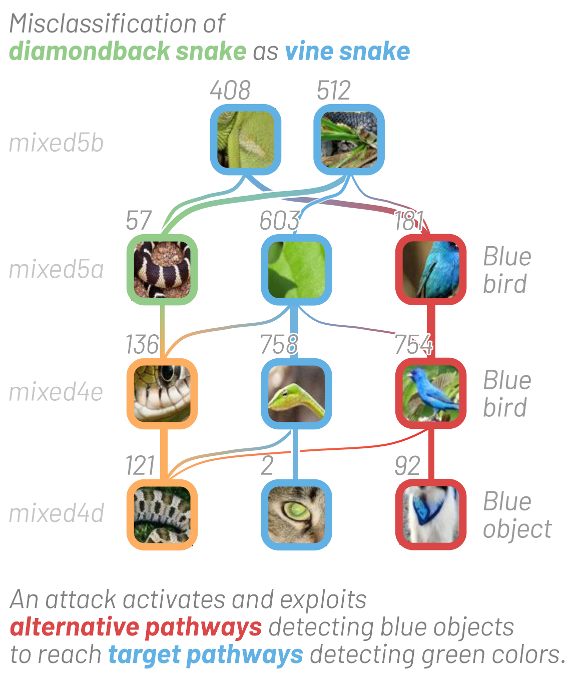

Misclassification of
diamondback snake as vine snake
408
512
mixed5b
57
603
181
mixed5a
Blue bird
136
758
754
mixed4e
Blue bird
121
2
92
mixed4d
Blue object
An attack activates and exploits
alternative pathways detecting blue objects
to reach target pathways detecting green colors.

Figure 3.11: BLUFF helps users understand how an attack infiltrates a model, by visualizing activation pathways that are additionally exploited by the attack. In this example, BLUFF highlights the neurons and connections that PGD attack exploits (red) to make InceptionV1 model would get confused adversarial **diamondback snake** images with **vine snake**.

3.8.1 Understanding How Attacks Penetrate DNNs

The lack of interpretation techniques for adversarial attacks on DNNs severely hinders machine learning experts from defending their models from malicious actors. This difficulty in explaining the model's behaviour under attack is further exacerbated by the complicated structure of DNNs. BLUFF addresses these challenges by visualizing how adversarial attacks penetrate the DNN's internals, pinpointing the neurons and their connections that are exploited by the attacks.

Consider a DNN classifier designed for identifying multiple types of snakes. *Diamondback snake* is one of the deadliest venomous snakes, and *vine snake* is a green slender snake with moderately potent venom that only causes mild swelling. Misclassifying diamondback snakes as vine snakes can be disastrous. Understanding how an adversarial attack exploits the model's internals — whether intentional or not — helps prevent such critical failures. In Figure 3.11, BLUFF reveals the activation pathways (neurons and their connections) that are *most activated* (exploited) by adversarial diamondback images attacked by PGD. While exploring the Graph Summary View, we noticed that there are several exploited neurons that look for *"blue color"* (e.g., *"blue birds"*), as shown in Figure 3.11. This is surprising

[4]D3.js: https://d3js.org/

because the *vine snake* (the attack's target class) have a green body, not blue. What we can infer from this finding is that PGD exploits the pathway for *"blue color"* as a bypassing alternative route to reach the pathways for vine snake, which look for *"green leaves"* and *"green bumpy skin"* (mixed5a-603 and mixed5b-408, blue in Figure 3.11). We also noticed that PGD leverages *"snake-like"* pathways that are typically important for both classes (Figure 3.11, orange neurons), which seems reasonable, given that both the original and target classes are snakes. PGD also uses a neuron for diamondback that detects *"striped snake"* (mixed5a-57, green in Figure 3.11), to fire *"green bumps"* and *"snake"* neurons. Finding such alternative routes exploited by an attack can give us a fundamental insight that could inform future defenses.

3.8.2 Correlating Class Similarity with Exploitation Intensity

An adversarial attack changes a data instance's predictions from its original class to a target class of the attacker's choosing. However, some class changes may be "easier" (e.g., from one type of dog to another), and some "harder" (e.g., from an animal to a vehicle). How does class similarity correlate with the magnitude of changes that an attack needs to induce inside a model?

Dissimilar class pairs → strong neuron inhibition. While attacks generally inhibit features from the original class (further explored in Section 3.8.4), so that features of the target adversarial class are relatively detected more, we found that the magnitude of neuron inhibition is much higher when the original and target classes are very different. For example, Figure 3.12-left shows that adversarial *ambulance* images need to *strongly* inhibit car-related neurons (i.e., big drop in their activation) to misclassify such images as *street signs* due to strong class dissimilarities. Conversely, as shown in Figure 3.12-right, adversarial *brown bear* images only need to *mildly* inhibit brown-fur neurons to induce misclassification of black bear due to close class resemblance. We made similar observations for other dissimilar (e.g., *giant panda* vs. *armadillo*), and similar class pairs (e.g., *vine snake* vs. *green snake*).

Dissimilar classes → more neurons exploited. BLUFF helps reveal *unusual* pathways that an attack exploits to induce incorrect predictions. Such unusual pathways (colored red in Graph Summary View) typically consist of neurons and connections that are *neither* commonly traversed by *benign original* images, nor *benign target* images; rather they are exploited unexpectedly by the adversarial images. We found that these unusual pathways are more heavily used — and they consist of more neurons and connections — when the original and target classes are dissimilar (e.g., *ambulance* vs. *street sign*; *giant panda* vs. *armadillo*). The unusual pathway for the *ambulance* class pair contains 136 neurons, and the

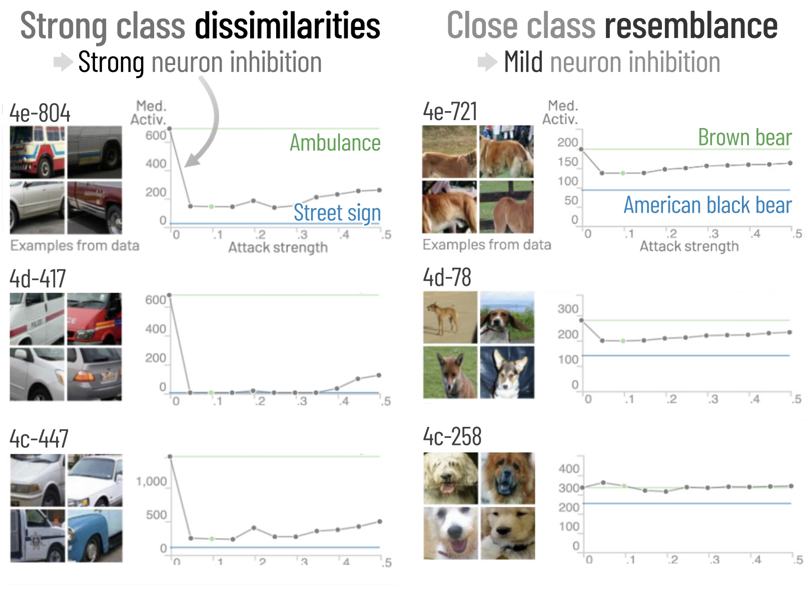


Figure 3.12: The most inhibited neurons for the *dissimilar* class pair (ambulance, street sign), and a *similar* class pair (brown bear, black bear). **Left:** adversarial *ambulance* images need to *strongly* inhibit car-related neurons (i.e., big drop in their activation) to misclassify such images as *street signs* due to strong class dissimilarities. **Right:** adversarial *brown bear* images only need to *mildly* inhibit brown-fur neurons to induce misclassification of *black bear* due to close resemblances.

giant panda pair contains 85, across all attack strengths. For similar class pairs (e.g., *brown bear* vs. *black bear*; *vine snake* vs. *green snake*, both with green bodies), their unusual pathways are dramatically more compact, 28 neurons for the *bear*s, and 16 for the *snake* for all attack strengths.

3.8.3 Shifting Crosshairs across Attack Strengths

As an adversary can perform an attack on the model at various levels of attack strengths — starting from imperceptible noise, all the way up to high intensity perturbation — we wonder whether an attack's strategy evolves as the attack strength escalates, or remains the same. BLUFF enables such a comparative analysis through its "Compare Attacks" mode. Consider the example of attacking the *diamondback* images to induce the misclassification of *vine snake*. Setting the weaker and stronger attack strength to 0.1 and 0.5 respectively in the Control Sidebar, BLUFF visualizes their differing attack strategies (Figure 3.13).

"Death by a thousand cuts" from weaker attacks. Figure 3.13-right reveals the surprising finding that the weaker attack targets mostly red neurons — neurons that are activated by the weaker attack, and are associated with *neither* snake class. This is represented by the large number of red neurons (28 in total) which only have their inner parts colored (see Figure 3.13-right). Just 9 red neurons are activated only for the stronger attack, and only

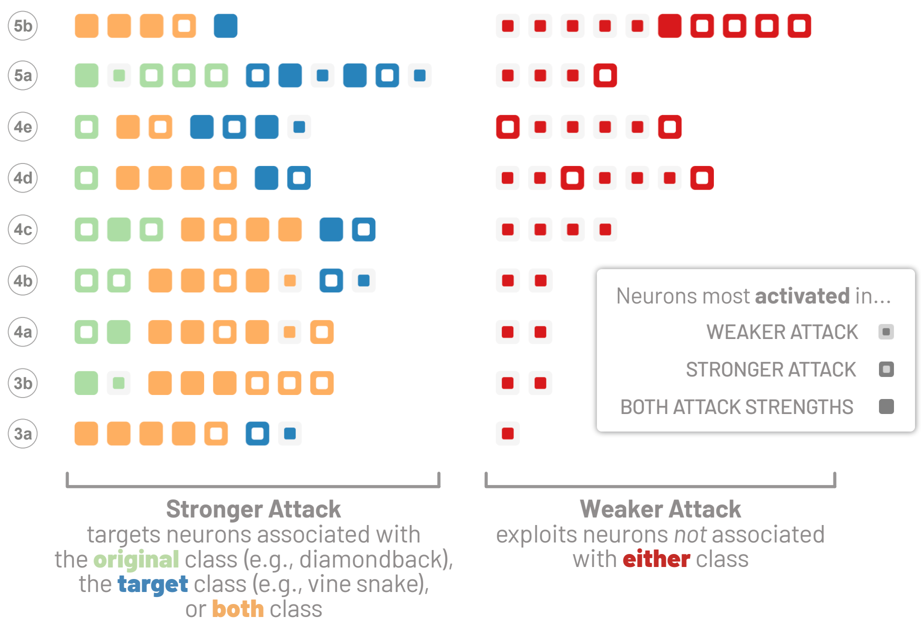


Figure 3.13: Using BLUFF's unified comparison view, we examine the attack's strategy for misclassifying *diamondback* images into *vine snake* class for 2 attack strengths (0.1 vs 0.5). **Left**: stronger attack generally targets **green**, **blue** and **orange** neurons, i.e., it is more precise in attacking semantically meaningful neurons. **Right**: weaker attack generally targets **red** neurons, i.e., it requires perturbing several disassociated neurons to successfully confuse the model.

one neuron activated for both attack strengths. On further examining the example image patches for these neurons, we noticed the images consist an assortment of semantic features such as *spider legs*, *blue bird* and *car hood*, seemingly unrelated to snakes. This leads us to believe that a weak attack relies on leveraging a large number of disassociated semantic features to induce misclassification, i.e., "death by a thousand cuts".

Surgical strike from stronger attacks. Through BLUFF's visualization, we also observed that stronger attacks target neurons that are commonly associated with the original class, the target class, or both classes (Figure 3.13-left). This is represented by the relatively large number (28 in total) of neurons from these three groups that only have the outer borders colored (see Figure 3.13-left). 34 neurons are activated by both classes, and just 9 neurons are activated only for the weaker attack. As we discussed in Section 3.8.1, the green, blue and orange neurons are likely induced by input features that have high semantic correlation with the original *diamondback* and the target *vine snake* classes. This lends to the notion that a stronger attack mainly targets semantically-relevant neurons with a precision much higher than a weaker attack would, and causes minimal collateral damage to other non-relevant neurons.

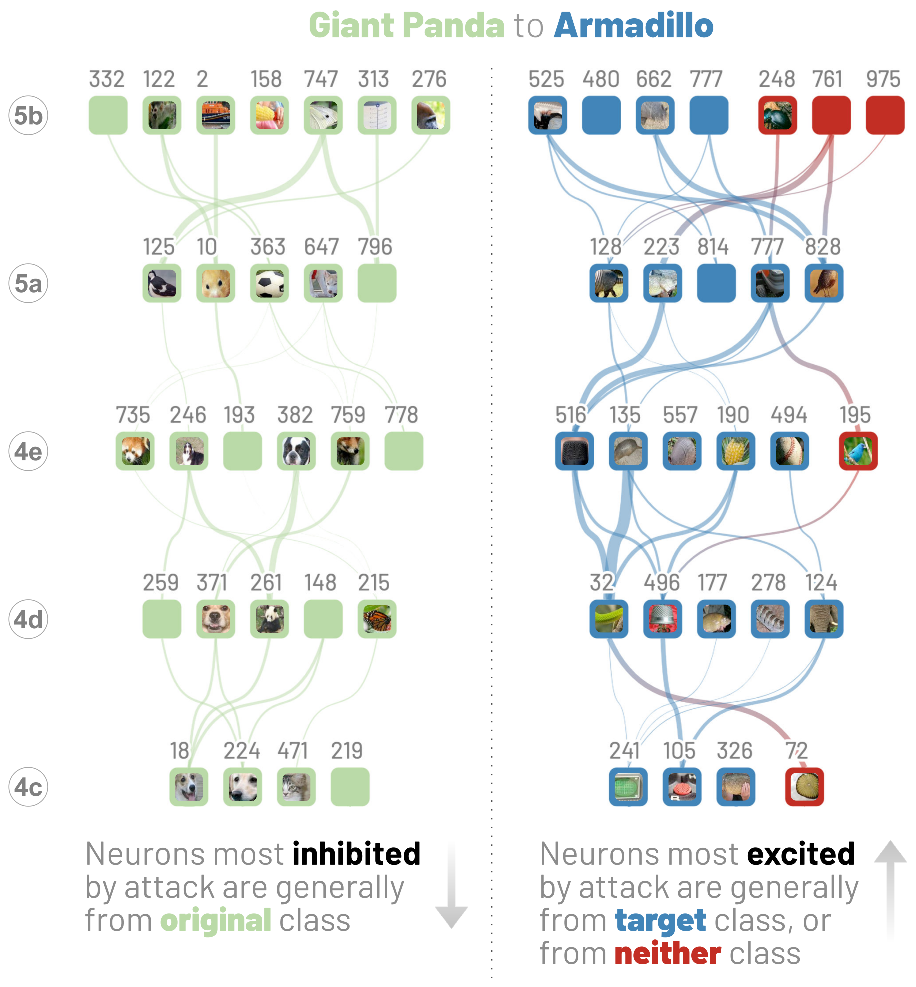


Figure 3.14: Neurons whose activation is most increased (excited) or decreased (inhibited) by adversarial attack. Here, we highlight the most excited/inhibited neurons using BLUFF's view filtering feature. The attack tries to misclassify *giant panda* images into *armadillo* class with attack strength of 0.5. We also show example images from data for all neurons which are consistently excited/inhibited across multiple attack strengths. **Left**: generally, **green** neurons are most **inhibited** by the attack, implying that the attack suppresses features associated with *giant panda* class. **Right**: generally, **blue** neurons are most **excited** by the attack, suggesting that the attack increasingly induces *armadillo* class features.

3.8.4 Multi-pronged Attack Strategy

Besides revealing the scope of attack (e.g., number of neurons and connections), as described in the previous scenario in Section 3.8.3, BLUFF can also enable research exploration of exciting research questions currently not adaquately supported by existing tools, such as whether the relative activation changes in an attacked neurons would align with semantic adjustments within the model, or are these changes seemingly arbitrary. Such discovery sheds light on the possible vectors that an adversarial attack could utilize to confuse a model. Consider the example of misclassifying adversarial **giant panda** images as **armadillo**. Figure 3.14 shows the top 20% neurons that are most **inhibited** (left) or most **excited** (right) for the strong attack strength of 0.5. In the figure, we also show data examples for the neurons that are consistently inhibited or excited across multiple attack strengths. Enabled by BLUFF, such focused examination on neuron inhibition and excitation allow us to discover and study a highly intriguing, multi-pronged approach employed by adversarial attacks, which we will describe below.

Attack depletes the original semantic features. Figure 3.14 (left) shows that the most inhibited neurons are generally **green** neurons, suggesting that the adversarial attack precisely targets the **original** class neurons and inhibits their activation to, in this case, decrease the "*panda*-ness" of the image. Moreso, many such neurons are consistently inhibited across *multiple* attack strengths. We observe several features semantically-relevant to a panda are inhibited, such as "*panda body*" (mixed4d-259, mixed4d-261), "*dog face with black/white patches*" (mixed4e-382), "*black/white bird*" (mixed5a-125) and "*black/white soccer balls*" (mixed5a-363). This summarization from BLUFF provides strong support for our hypothesis that an adversarial attack would diminish features that are semantically correlated with the **original** class.

Attack imbues the target semantic features. Figure 3.14 (right) reveals that the neurons most consistently excited by an attack across multiple strengths are generally **blue** or **red** neurons. This reinforces the belief that adversarial perturbation attempts to increase the "*armadillo*-ness." The Detail View provides further support, showing that the median neuron activation increases monotonically (i.e., increasingly excited) across attack strengths. Correspondingly, we also observe several features that would be considered semantically relevant for armadillo, such as "*grid pattern*" (mixed4c-241), "*dotted pattern*" (mixed4c-105), "*rugged surface*" (mixed4d-496), "*dotted mic*" (mixed4e-516) and "*armadillo body*" (mixed4e-135, mixed5a-128, mixed5b-662). Thus, BLUFF enables us to substantiate that an adversarial attack would inject features that are semantically associated with the **target** class.

3.9 LIMITATIONS AND FUTURE WORK

Interactive neuron editing for model robustness. BLUFF currently visualizes neurons that are highly excited or inhibited by an adversarial attack under varying attack strengths. This is an important first step to identifying vulnerable pathways of a network; however, the ultimate goal is to use BLUFF to inform and construct robust defenses against such attacks. One approach is to use interactive experimentation techniques to allow real-time neuron editing, e.g., deletion. Here, a user could actively identify vulnerable neurons using BLUFF and interactively remove them from the network to observe the effect on the resulting pathway and prediction in real-time. Masking the activations of a particular neuron could potentially prevent targeted attacks to propagate deeper into the network. For example, a user could preemptively edit a DNN to enhance its robustness by deleting neurons that only feed into exploited pathways, preventing adversarially activated neurons from affecting subsequent layers.

Visualizing activation pathways in adversarially trained models. In the adversarial machine learning literature, one common technique to improve the robustness of deep learning models is to train them with an augmented dataset that additionally includes adversarially attacked inputs [48, 59, 54]. The motivation is that during training models will learn to ignore the noise from the attacks and instead focus on the common signal between both benign and attacked inputs. However, it is still not understood how the internal representations of these adversarially trained robust models can be compared to standard networks. Using our summarization techniques and activation pathways could provide a mechanism to compare standard and robust models beyond simple summary statistics, such as accuracy, but enable inspection of specific subpaths and neurons.

Comparing activation pathways across other model architectures. In Section 3.6.1 we justify our data and model selection, but another natural next step for future work is generating and comparing activation pathways across different model architectures. Recent work [65] has provided evidence that different types of neural network architectures learn similar concepts (e.g., curve detectors and high-low frequency detectors) for vision tasks. Our approach using activation pathways could be adopted for comparing what features multiple networks have learned and how their vulnerability could be different.

Mining activation pathways for motifs. Since activation pathways are extracted from neural networks, we can leverage data mining and graph analysis methods to find the most common motifs across all pathways. For example, perhaps all snake classes share one specific vulnerable subpath that is always attacked, or maybe all bird classes share multiple subpaths that are never effected, even from strong attacks. Extracting these smaller subpath motifs could give further insight into how neural networks arrange and prioritize hierarchical concepts.

3.10 CONCLUSION

As deep neural networks (DNNs) become increasingly used in many domains, it is important for us to understand their vulnerabilities. In this work, we present BLUFF, an interactive system for visualizing, characterizing, and deciphering adversarial attacks on vision-based neural networks. The BLUFF visualization runs in modern web browsers and is open-sourced. We believe our visualization, summarization, and comparison approaches will help promote user understanding of adversarial attacks, and support discoveries that reveal the mechanisms that attacks employ to inflict harm on models. Ultimately, we hope our findings will help inform future research on designing models that are more robust against attacks.

Part II

Mitigating Adversarial Examples Across Modalities & Tasks

OVERVIEW

While we develop methods to enhance our intuitive interpretation of attacks in Part I, it is not enough to only expand our understanding of AI vulnerabilities. To truly overcome the threat posed by adversarial ML, we need to leverage this understanding and investigate unifying methods that can effectively mitigate adversarial examples across AI tasks.

We do this by first focusing on the input stage of the model, leveraging practical pre-processing techniques to remove adversarial perturbations. In Chapter 4, we present the **SHIELD** framework, through which we explore the idea of compression as a fast, practical defense for image classification models. We expand upon the widely used JPEG compression algorithm and propose a novel pre-processing technique that incorporates randomization with compression to develop a multifaceted defense. This chapter is adapted from our published work [70] that appeared at KDD 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, Polo Chau. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018. PDF

Right at the heels of releasing our SHIELD research, adversarial ML had quickly evolved to propose a new generation of more sophisticated, *adaptive* attacks. Following this, it is no longer enough to compress away high-frequency perturbations for evading adversarial attacks. Hence, we capitalize on the AI security insights afforded by our interpretation and visualization techniques, and shift our focus to fortifying the model internals directly by influencing the learning stage of ML models. Our experiments with **Multi-Task Learning** (MTL) reveal that it is one such fundamental deep learning approach that has the potential to fortify ML models across AI tasks.

We first demonstrate this ability of MTL to induce a robust feature space in the video domain through the task of person tracking. In Chapter 5, we present our **SkeleVision** research, where we combine semantically analogous tasks of person tracking and human keypoint detection. Our experiments reveal that MTL models are consistently more resistant to powerful, adaptive, physically realizable attacks across a high number of attack iterations. The work from this chapter is submitted for peer review.

> SkeleVision: Towards Adversarial Resiliency of Person Tracking with Multi-Task Learning. Nilaksh Das, Sheng-Yun Peng, Polo Chau. *Under peer review*, 2022. PDF

Following the observations from SkeleVision, we shift our attention to an entirely different input modality in order to study the efficacy of MTL robustness across AI tasks. We next experiment with the robustness of **automatic speech recognition** (ASR) models in the audio domain when trained jointly with MTL in Chapter 6. We explore semantically equivalent as well as semantically diverse tasks for performing MTL. We find that a combination of both types of tasks is necessary to most effectively thwart powerful, adaptive adversarial attacks. Our MTL approach shows considerable absolute improvements in adversarially targeted word error rate. The work from this chapter is submitted for peer review.

> Hear No Evil: Towards Adversarial Robustness of Automatic Speech Recognition via Multi-Task Learning. Nilaksh Das, Polo Chau. *Under peer review*, 2022. PDF

Our in-depth research in Chapters 5 and 6 establishes multi-task learning as a fundamentally unifying deep learning approach across AI tasks (*e.g.,* person tracking and ASR) and input modalities (*e.g.,* video and audio), which induces models to learn robust features that are resistant to adversarial attacks.

CHAPTER 4
SHIELD: FAST, PRACTICAL DEFENSE AND VACCINATION FOR DEEP LEARNING USING JPEG COMPRESSION

The rapidly growing body of research in adversarial machine learning has demonstrated 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 place 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

Deep neural networks (DNNs), while enjoying tremendous success in recent years, suffer from serious vulnerabilities to adversarial attacks [79]. For example, 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 the gradient information of the model, which guides the adversary in discovering vulnerable regions of the input space that would most drastically affect the model output [48, 80]. 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 [81]. To make this even more troubling, it is possible to print out physical 2D or 3D

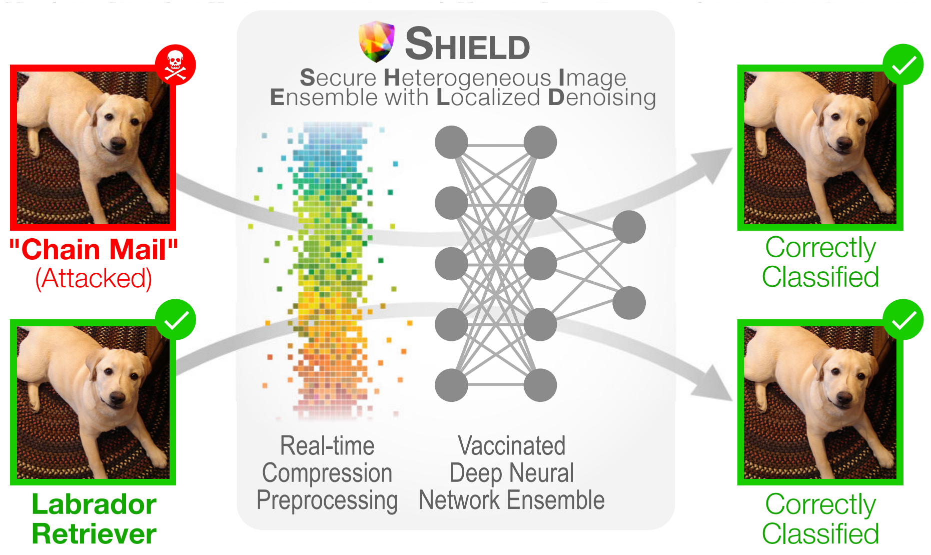

SHIELD
Secure Heterogeneous Image Ensemble with Localized Denoising
"Chain Mail"
(Attacked)
Labrador Retriever
Real-time Compression Preprocessing
Vaccinated Deep Neural Network Ensemble
Correctly Classified
Correctly Classified

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*.

objects to fool recognition systems in realistic settings [82, 83].

The threat of adversarial attack casts a shadow over deploying DNNs in security and safety-critical applications like self-driving cars. To better understand and fix the vulnerabilities, there is a growing body of research on defending against various attacks and making DNN models more robust [71, 84, 85]. However, the progress of defense research has been lagging behind the attack side so far. Moreover, research on defense rarely focuses on practicality and scalability, both essential for real-world deployment. For example, total variation denoising and image quilting are image preprocessing techniques that have potential in mitigating adversarial perturbations to some extent [86], but they incur significant computational overhead, calling into question how feasibly they can be used in practical applications, which often require fast, real-time defense [87, 88].

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 powerful, scalable, and practical protection for deep learning models against adversarial image perturbations in real-time. 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 is particularly designed to reducing 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 show that a model can be “vaccinated” by training 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 recovers significant amount of model accuracy lost to adversarial instances, with little effect on the accuracy for benign instances. SHIELD stands for **S**ecure **H**eterogeneous **I**mage **E**nsemble with **L**ocalized **D**enoising. (Sections 4.3.2 and 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 [57] and *DeepFool* [56]. (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

Our work focuses on defending against adversarial attacks on deep learning models. This section provides background information for readers new to the adversarial attack literature.

Given a trained classifier C and an instance $x \in \mathcal{X}$, the objective of an adversarial *untargeted* attack is to compute a perturbed instance x' such that $C(x') \neq C(x)$ and $d(x, x') \leq \rho$ for some distance function $d(\cdot, \cdot)$ and $\rho \geq 0$. Popular choices of $d(\cdot, \cdot)$ are Euclidean distance $d(x, x') = \|x - x'\|_2$, and Chebychev distance $d(x, x') = \|x - x'\|_\infty$. A *targeted* attack is similar, but is required to induce a classification for a specific target class t, i.e., $C(x') = t$. In both cases, depending on whether the attacker has full knowledge of C or not, the attack can be further categorized into *white-box* attack and *black-box* attack. The latter is obviously harder for the attacker since less information is known about the model, but has been shown to be possible in practice by relying on the property of transferability from a substitute model to the target model when both of them are DNNs trained using gradient backpropagation [79, 81].

The seminal work by Szegedy et al. [79] proposed the first effective adversarial attack on DNN image classifiers by solving a box-constrained L-BFGS optimization problem and showed that the computed perturbations to the images were indistinguishable to the human eye — a rather troublesome property for people trying to identify adversarial images. This discovery has gained tremendous interest, and many new attack algorithms have been invented [48, 56, 89, 80] and applied to other domains such as malware detection [90, 91], sentiment analysis [92], and reinforcement learning [93, 94]. Below, we describe the major, well-studied attacks in the literature, against which we will evaluate our approach.

Carlini-Wagner's L_2 (*CW-L2*) [57] 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:

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

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

DeepFool (*DF*) [56] 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*) [49] is the iterative version of the **Fast Gradient Sign Method** (*FGSM*) [48], which is a fast algorithm that computes perturbations subject to an L_∞ constraint. *FGSM* simply takes the sign of the gradient of loss function J w.r.t. the input x,

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

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

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

4.3 PROPOSED METHOD: COMPRESSION AS DEFENSE

In this section, 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 perturbation. As compression would modify 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, which increases its robustness towards 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, which, to the best of our knowledge, has yet been challenged.

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*** 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 8×8 blocks of the YC_bC_r channels to a frequency domain representation using Discrete Cosine Transform (DCT).
4. Perform quantization of 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 more since these coefficients contribute less to the human perception of the image. As adversarial attacks do not optimize for maintaining the spectral signature of the image, they tend to introduce more high frequency components which can be removed at this step. This step also renders the preprocessing stage non-differentiable, which makes it non-trivial for an adversary to optimize against, allowing only estimations to be made of the transformation [95]. We show in our evaluation (Section 4.4.2) that JPEG compression effectively removes adversarial perturbation across a wide range of compression levels.

4.3.2 Vaccinating Models with Compressed Images

As DNNs are typically trained on high quality images (with little or compression), they are often invariant to the artifacts introduced by the preprocessing of JPEG at high-quality settings. This is especially useful in an adversarial setting as our pilot study has shown that applying even mild compression removes the perturbations introduced by some attacks [96]. However, applying too much compression could reduce the model accuracy on benign images.

We propose to "vaccinate" the model by training it with compressed images, especially those at lower JPEG qualities, which increases the model's robustness towards 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 significant advantage that our vaccination strategy provides, recovering more than 7 *absolute* percentage points in model accuracy for high-perturbation attacks.

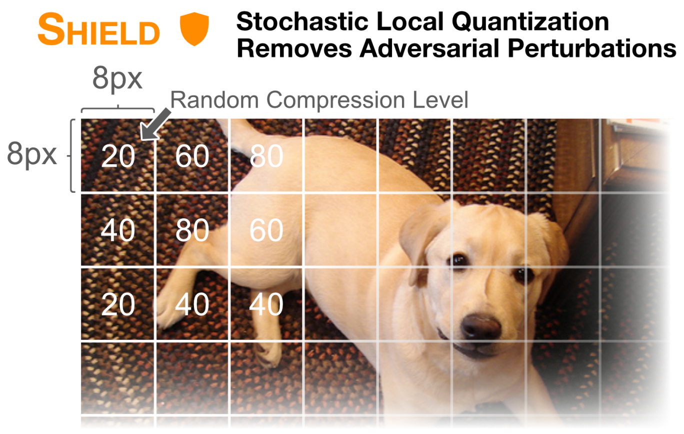

SHIELD
Stochastic Local Quantization
Removes Adversarial Perturbations
8px
Random Compression Level
8px
20
60
80
40
80
60
20
40
40

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

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, making it harder for the adversaries to estimate the preprocessing transformation.

Figure 4.2 illustrates our proposed strategy, where we vary the quantization table for each 8×8 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 8×8 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. Thus, 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 a simple average over JPEG compressed images, our method allows for maintaining the original semantics of the image in the blocks compressed to higher qualities, while performing more localized denoising in the blocks compressed to lower qualities. In the case of simple average, all perturbations may not be removed at higher qualities and they might simply dominate the other components participating in the average, still posing to be adversarial. Introducing localized stochasticity reduces this expectation.

In our evaluation (Section 4.4.3), we will show that by using the spectrum of JPEG compression levels with our stochastic approach, our model can simultaneously attain a high accuracy on benign images, while being more robust to adversarial perturbations — a strong benefit that using a single JPEG quality cannot provide. Our method is further fortified by using an ensemble of vaccinated models individually trained on the set of qualities picked for randomization. We show in Section 4.4.3 how our method can achieve high model accuracies, comparable to those of much larger ensembles, but is significantly faster.

4.4 EVALUATION

In this section, we show that our approach is scalable, effective and practical in removing adversarial image perturbations. For 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 [97], 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[1], 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 [86], 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 ϵ 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 [98, 86]. Median filter (MF) collapses

[1] https://github.com/tensorflow/cleverhans

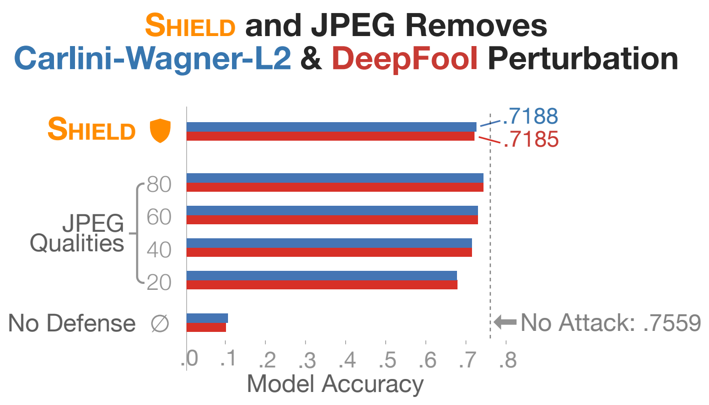


Figure 4.3: Carlini-Wagner-L2 (CW-L2) and DeepFool, two recent strong attacks, introduce perturbations that lowers model accuracy to around 10% (∅). JPEG compression recovers up to 98% of the original accuracy (with DeepFool), while SHIELD achieves similar performance, recovering up to 95% of the original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

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 vary the parameters of each method to evaluate how their values affect defense performance.

- For JPEG compression, we vary the compression level from quality 100 (least compressed) to 20 (greatly compressed), in decrements of 10.
- For *median filter* (MF), we use window sizes of 3 (smallest possible) and 5. We tested larger window sizes (e.g., 7), which led to extremely poor model accuracies, thus were ruled out as parameter candidates.
- For *total variation denoising* (TVD), we vary its weight parameter from 10 through 40, in increments of 10. Reducing the weight of TVD further (e.g., 0.3) produces blurry images that lead to poor model accuracy.

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.

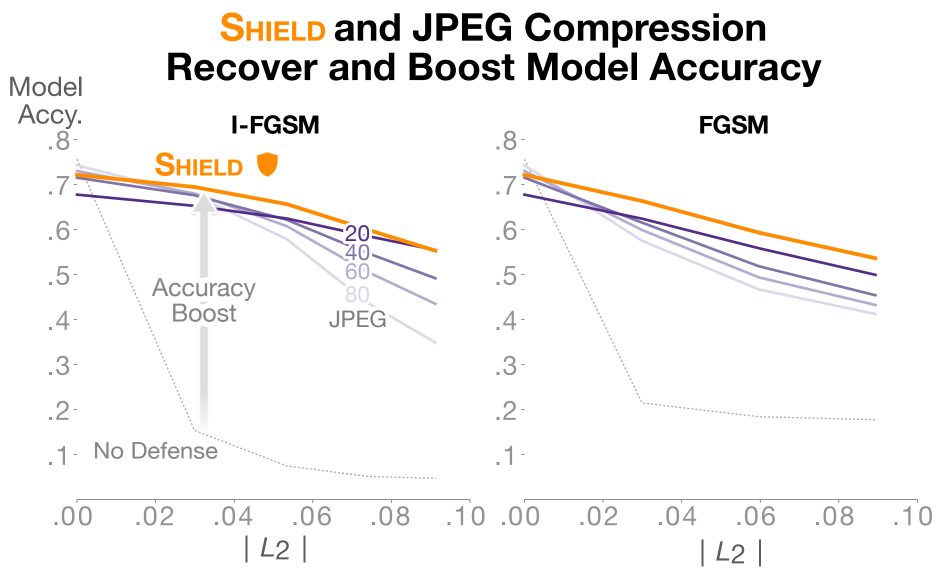


Figure 4.4: SHIELD recovers the accuracy of the model when attacked with I-FGSM (left) and FGSM (right). Both charts show the accuracy of the model when undefended (gray dotted curve). Applying varying JPEG compression qualities (purple curves) helps recover accuracy significantly, and SHIELD (orange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

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 [86], 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.

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

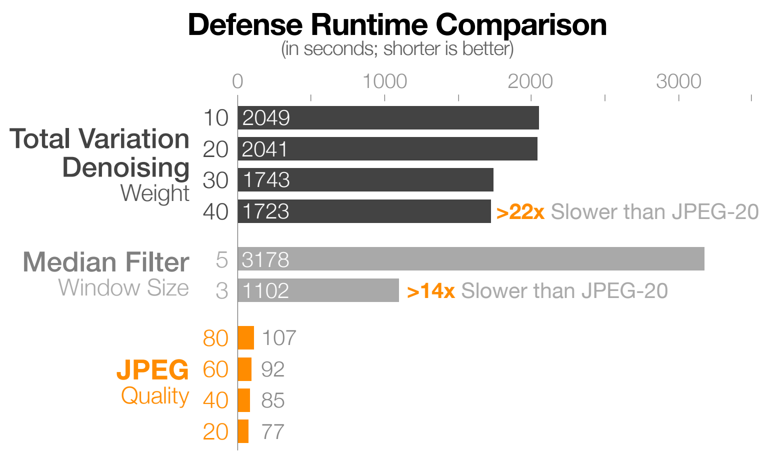


Figure 4.5: Runtime comparison for three defenses: (1) total variation denoising (TVD), (2) median filter (MF), and (3) JPEG compression, timed using the full 50k ImageNet validation images, averaged over 3 runs. JPEG is at least 22x faster than TVD, and 14x faster than MF.

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 [98, 86]. 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 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 fastest, most popular Python implementations of the image processing techniques. We used JPEG and MF from Pillow 5.0, and TVD from scikit-image.

Table 4.1: Summary of model accuracies (in %) for all defenses: SHIELD, 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, both strongly optimized attacks with lower perturbation strength, the best performing settings are from JPEG (in bold font). SHIELD benefits from the combination of Stochastic Local Quantization, vaccination and ensembling, outperforming all other techniques when facing high perturbation delivered by I-FGSM and FGSM.

Defense	No Attack	**CW-L2** ($\kappa = 0$)	**DF**	**I-FGSM** ($\epsilon = 4$)	**FGSM** ($\epsilon = 4$)
No Defense	*75.59*	*10.29*	*9.78*	*7.49*	*18.40*
SHIELD [20, 40, 60, 80]	72.11	71.85	71.88	**65.63**	**59.29**
JPEG [quality=100]	**74.95**	74.37	**74.41**	52.52	44.00
JPEG [quality=90]	74.83	**74.43**	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

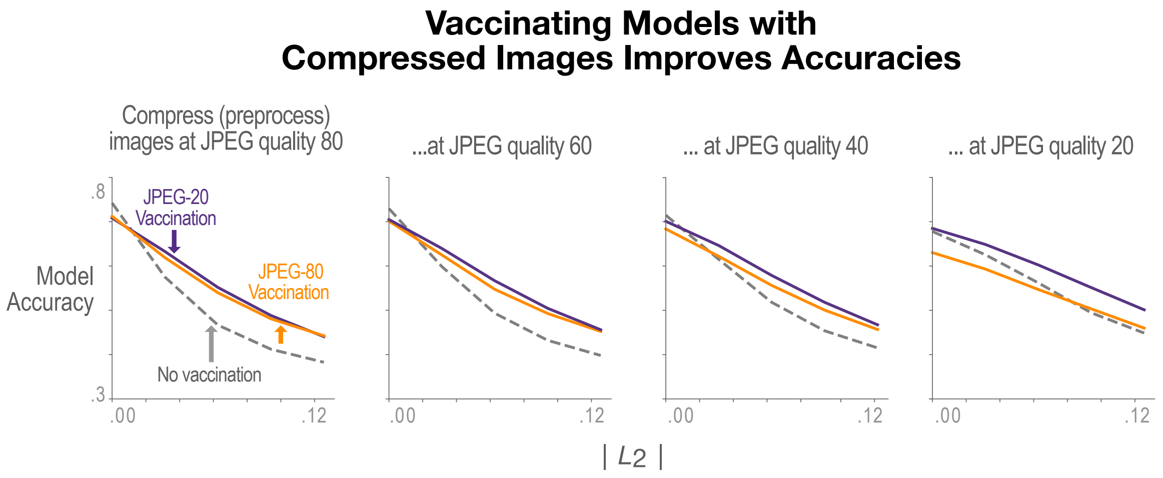


Figure 4.6: Vaccinating a model by retraining it with compressed images helps recover its accuracy. Each plot shows the model accuracies when preprocessing with different JPEG qualities with the FGSM attack. Each curve in the plot corresponds to a different model. The gray dotted curve corresponds to 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.

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.

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 5×10^{-3}, with a decay of 94% over 25×10^4 iterations. We conducted the retraining on a GPU cluster with 12 NVIDIA Tesla K80 GPUs. In this manner, we obtain 8 models from quality 20 through quality 90 in increments of 10 ($\mathcal{M}_{20}, \mathcal{M}_{30}, \mathcal{M}_{40}...\mathcal{M}_{90}$), to cover a wide spectrum of JPEG qualities. Figure 4.6 shows

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 $q \in \{20, 30, 40, ..., 90\}$. 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.

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

the results of model vaccination against FGSM attacks, whose parameter ϵ ranges from 0 (no perturbation) to 8 (severe perturbation), in steps of 2. The plots show that retraining the model helps recover even 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, which was expected.

We test 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 $q \in \{20, 30, 40, ..., 90\}$. This has a total cost of 64 votes, from which we derive 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.

4.4.4 Transferability in Black-Box Setting

In this setup, we evaluate the transferability of attacked images generated using ResNet-v2 50 on ResNet-v2 101 and Inception-v4. The attacked images are 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%.

		Inc-v4 (80.2%)		**RN-v2 101** (77.0%)	
Attack	Defense	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 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, 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 not only educates industry regarding concepts and defenses of adversarial machine learning, but also 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 [86, 99], using multiple models for model ensembles [100], the introduction of adversarial perturbation detectors [85, 98], or the increase in training time for adversarial training [48]. However, while hardware and system improvement for fast deep learning training and inference remains an active area of 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, and we do not expect or intend for it 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 [101, 102]. 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 [85, 103], modifying network architecture and the underlying primitives used [104, 105, 106], modifying the training process [48, 71], and using preprocessing techniques to remove adversarial perturbations [107, 84, 99, 86]. 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 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 [86, 98].

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 [86].

4.6.2 Attacks against Preprocessing Techniques

One of the reasons 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 [95] 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. However, the paper only reports preliminary results on 1000 images. It is also unclear whether their attack is effective against our more advanced SHIELD method, which introduces more randomization to combat against adversarial noise.

Backward Pass Differentiable Approximation [108] 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 (https://github.com/poloclub/jpeg-defense).

CHAPTER 5
SKELEVISION: TOWARDS ADVERSARIAL RESILIENCY OF PERSON TRACKING WITH MULTI-TASK LEARNING

Person tracking using computer vision techniques has wide ranging applications such as autonomous driving, home security and sports analytics. However, the growing threat of adversarial attacks raises serious concerns regarding the security and reliability of such techniques. In this work, we study the impact of multi-task learning (MTL) on the adversarial robustness of the widely used SiamRPN tracker, in the context of person tracking. Specifically, we investigate the effect of jointly learning with semantically analogous tasks of person tracking and human keypoint detection. We conduct extensive experiments with more powerful adversarial attacks that can be physically realizable, demonstrating the practical value of our approach. Our empirical study with simulated as well as real-world datasets reveals that training with MTL consistently makes it harder to attack the SiamRPN tracker, compared to typically training only on the single task of person tracking.

5.1 INTRODUCTION

Person tracking is extensively used in various real-world use cases such as autonomous driving [109, 110, 111], intelligent video surveillance [112, 113, 114] and sports analytics [115, 116, 117]. However, vulnerabilities in the underlying techniques revealed by a growing body of adversarial ML research [79, 48, 54, 47, 118, 119, 120, 121, 122] seriously calls into question the trustworthiness of these techniques in critical use cases. While several methods have been proposed to mitigate threats from adversarial attacks in general [123, 54, 59, 124, 125], defense research in the tracking domain remains sparse [126]. This is especially true for the new generation of physically realizable attacks [47, 118, 119] that pose a greater threat to real-world applications.

In this work, we aim to investigate the robustness characteristics of the SiamRPN tracker [127], which is widely used in the tracking community. Specifically, our goal is to improve the tracking robustness to a physically realizable patch attack [119]. Such attacks are unbounded in the perceptual space and can be deployed in realistic scenarios, making them more harmful than imperceptible digital perturbation attacks. Figure 5.1 shows an example of such a physically realizable patch attack that blends in the background.

Multi-task learning (MTL) has recently been touted to improve adversarial robustness to imperceptible digital perturbation attacks for certain computer vision tasks [128, 129].

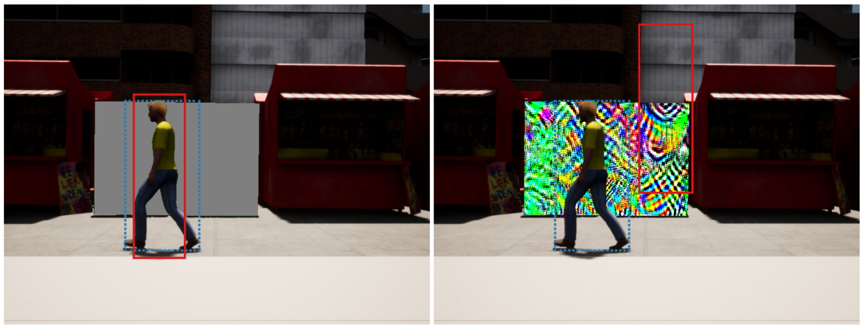


Figure 5.1: Example of a physically realizable patch attack. The dashed blue box shows the ground-truth bounding box and the solid red box shows the bounding box predicted by SiamRPN. In the benign case (left), the tracker is able to correctly track the person whereas in the adversarial case (right) the tracker is fooled by the adversarial patch.

However, it is unclear if these proposed methods translate to physically realizable attacks. Moreover, these methods have primarily been studied in the context of a single backbone branch with one-shot inference, whereas the Siamese architecture of the SiamRPN tracker involves multiple branched stages, posing interesting design considerations. In this work, we aim to address these research gaps by focusing on improving single-person tracking robustness.

As physically realizable attacks are unbounded in the perceptual space, they can create easily perceptible, but inconspicuous perturbations that fools a deep neural network into making incorrect predictions. However humans can ignore such perturbations by processing semantic knowledge of the real world. This calls for implicitly incorporating some inductive biases that supervise the neural network to learn semantic constraints that humans so instinctively interpret. To this effect, in this work we study the impact of MTL on robustness of person tracking with a semantically analogous task such as human keypoint detection.

Contributions

- **First Study of Tracking Robustness with MTL.** To the best of our knowledge, our work is the first to uncover the robustness gains from MTL in the context of person tracking for physically realizable attacks. Our code is made available at: https://github.com/nilakshdas/SkeleVision.
- **Novel MTL Formulation for Tracking.** We augment the SiamRPN tracker for MTL by attaching a keypoint detection head to the template branch of the shared backbone while jointly training.
- **Extensive Evaluation.** We conduct extensive experiments to empirically evaluate the effectiveness of our MTL approach by varying attack parameters, network architecture, and MTL hyperparameters.

- **Discovery.** Our experiments with simulated and real-world datasets reveal that training with MTL consistently makes it harder to attack the SiamRPN tracker as compared to training only on the single task of person tracking.

5.2 RELATED WORK

Since its inception with SiamFC [130], the Siamese architecture has been leveraged by multiple real-time object trackers including DSiam [131], SiamRPN [127], DaSiamRPN [132], SiamRPN++ [133], SiamAttn [134] and SiamMOT [135]. In this work, we experiment with SiamRPN as the target tracker since many other trackers share a similar network architecture as SiamRPN, and the properties of SiamRPN can be generalized to other such state-of-the-art trackers.

5.2.1 Multi-task Learning

MTL aims to learn multiple related tasks jointly to improve the generalization performance of all the tasks [136]. It has been applied to various computer vision tasks including image classification [137], image segmentation [128], depth estimation [138], and human keypoint detection [139].

MTL has also been introduced for the video object tracking task [140, 141, 142]. Zhang *et al.* [143, 144, 145] formulate the particle filter tracking as a structured MTL problem, where learning the representation of each particle is treated as as an individual task. Wang *et al.* [146] show that joint training of natural language processing and object tracking can link the local and global search together, and lead to a better tracking accuracy. Multi-modal RGB-depth and RGB-infrared tracking also demonstrate that including the depth or infrared information in the tracking training process can improve the overall performances [147, 148, 149, 150].

5.2.2 Adversarial Attacks

Machine learning model are easily fooled by adversarial attacks [151]. Adversarial attacks can be classified as digital perturbation attacks [79, 48, 54] and physically realizable attacks [47, 118, 119, 121]. In the tracking community, multiple attacks have been proposed to fool the object tracker [120, 122]. Fast attack network [152] attacks the Siamese network based trackers using a drift loss and embedded features. The attack proposed by Jia *et al.* [153] degrades the tracking accuracy through an IoU attack, which sequentially generates perturbations based on the predicted IoU scores. The attack requires ground-truth when performing the attack. Wiyatno and Xu [119] propose a method to generate an adversarial

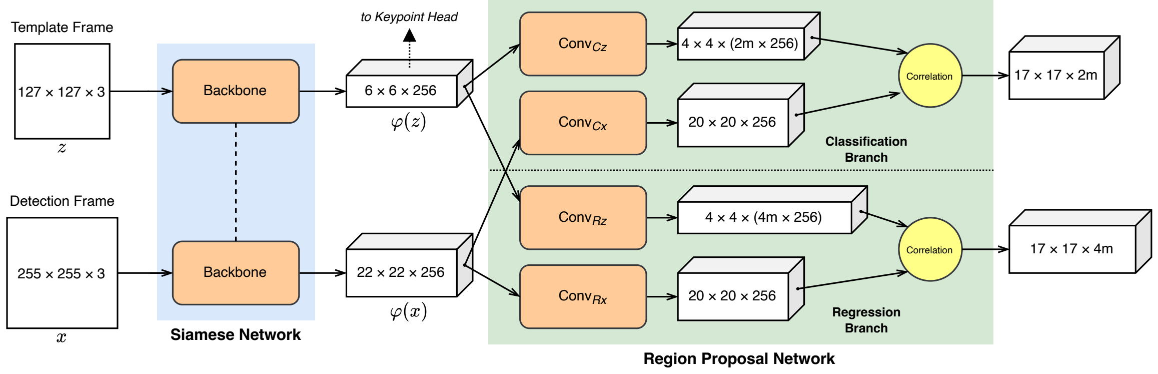


Figure 5.2: Overview of the SiamRPN architecture for tracking. For multi-task learning, the output of the template branch is passed to a keypoint head for keypoint detection.

texture. The texture can lock the GOTURN tracker [154] when a tracking target moves in front of it.

5.2.3 Adversarial Defenses in Tracking

General defense methods for computer vision tasks include adversarial training [59], increasing labeled and unlabeled training data [124], decreasing the input dimensionality [125], and robust optimization procedures [155, 156]. However, not many defense methods have been proposed to improve the tracking robustness under attack. Jia *et al.* [126] attempt to eliminate the effect of the adversarial perturbations via learning the patterns from the attacked images. Recently, MTL has been shown to improve the overall network robustness [129], especially in image segmentation [128] and text classification [157]. Our work is the first that studies MTL for person tracking with a physically realizable attack.

5.3 PRELIMINARIES

The input to the tracker can be denoted as $\{x, z, \bar{y}_x, \bar{y}_z\}$, where x is the detection frame in which the subject is to be tracked, z is the template frame containing an exemplar representation of the subject, and respectively, $\bar{y}_x$ and $\bar{y}_z$ are the ground-truth bounding box coordinates within the corresponding frames.

5.3.1 Tracking with SiamRPN

In this work, we focus on the Siamese-RPN model (SiamRPN) [127], which is a widely used tracking framework based on the Siamese architecture. An overview of the SiamRPN architecture is shown in Figure 5.2. SiamRPN consists of a Siamese network for extracting features and a region proposal network (RPN), also referred to as the RPN head, for predicting bounding boxes.

The Siamese network has two branches: (1) the *template branch* which receives a template patch $z' = \Gamma(z, \bar{y}_z, s_z)$ as input; and (2) the *detection branch* which receives a detection patch $x' = \Gamma(x, \bar{y}_x, s_x)$ as input. Here, $\Gamma(\cdot)$ is simply a crop operation that ensures only a limited context of size s centered on the bounding box y is passed to the network [127]. The corresponding sizes s_z and s_x are shown in Figure 5.2. For notational convenience, we use z for z' and x for x' hereon. The two branches of the Siamese network use a shared backbone model such that inputs to both branches undergo the same transformation $\varphi(\cdot)$. Hence, we can denote the output feature maps of the Siamese network as $\varphi(z)$ and $\varphi(x)$ for the template and detection branches, respectively. In this work, we use the SiamRPN model with AlexNet backbone [97].

The RPN head can also be separated into two branches as shown in Figure 5.2. Considering m anchors distributed across the detection frame, the classification branch predicts whether each respective anchor is a background or foreground anchor. Hence, the classification branch has $2m$ output channels corresponding to m anchors. The regression branch on the other hand predicts 4 box coordinate regression deltas [158] for each anchor, and therefore has $4m$ output channels.

While training, the classification and regression branches of the RPN head yield $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$ respectively, where $\mathcal{L}_{cls}$ is the cross-entropy loss and $\mathcal{L}_{reg}$ is a smooth L_1 loss [127]. Finally, the total weighted loss optimized for is as follows:

$$\mathcal{L}_{TRK}(x, z, \bar{y}_x) = \lambda_C \mathcal{L}_{cls}(x, z, \bar{y}_x) + \lambda_R \mathcal{L}_{reg}(x, z, \bar{y}_x) \tag{5.1}$$

During inference, the network acts as a single-shot detector. Typically, a sequence of frames $\mathbf{x} = \{x_1, \dots x_n\}$ is provided with the ground-truth bounding box coordinates $\bar{y}_{x_1}$ of the first frame as input. Hence, the first frame x_1 becomes the template frame z used to compute the feature map $\varphi(z)$ once, which can be considered as detector parameters for predicting bounding box coordinates for input frames from the same sequence. We denote the predicted bounding box for an input frame as $\hat{y}_x$. As mentioned previously, SiamRPN crops the context centered on the ground-truth bounding box. For inference, the context region is determined by the predicted bounding box of the previous frame. Finally, the tracking performance is evaluated using a mean intersection-over-union (mIoU) metric of the predicted and ground-truth bounding boxes across all frames from all input sequences.

5.3.2 Multi-task Learning with Shared Backbone

To provide semantic regularization for tracking, we perform joint multi-task training by attaching a fully convolutional keypoint prediction head to the template branch of SiamRPN. Our hypothesis is that joint training with an additional task head attached to the shared backbone would encourage the backbone to learn more robust features [159] for facilitating multiple tasks. Since the shared backbone is also used during tracking inference, the learned robust features can make it harder for adversarial perturbations to fool the model. We select the task of human keypoint prediction for this purpose as it is more semantically analogous to the task of person tracking.

The keypoint head is attached to the template branch as it has a more focused context [127]. Therefore, the keypoint head receives $\varphi(z)$ as input. The keypoint head network consists of convolutional blocks followed by a transpose convolution operation that "up-samples" the intermediate feature map to an expanded size with number of output channels equaling the number of keypoints being predicted. Finally, bilinear interpolation is performed to match the size of the input frame. The resulting feature volume has a shape of $H \times W \times K$, where H and W are the height and width of the input frame respectively, and K is the number of keypoints. Hence, each position in the K-channel dimension corresponds to a keypoint logit score. Given the ground-truth keypoints $\bar{k}_z$, the binary cross-entropy loss is computed with respect to each position in the channel dimension. We denote this as $\mathcal{L}_{KPT}$. For multi-task training, the total loss is a weighted sum:

$$\mathcal{L}_{MTL}(x, z, \bar{y}_x, \bar{k}_z) = \mathcal{L}_{TRK}(x, z, \bar{y}_x) + \lambda_K \mathcal{L}_{KPT}(z, \bar{k}_z) \tag{5.2}$$

The ground-truth keypoint annotation also consists of a visibility flag that allows us to suppress spurious loss from being backpropagated for keypoints that are occluded or not annotated.

5.3.3 Adversarial Attacks

Adversarial attacks introduce malicious perturbations to the input samples in order to confuse the tracker into making incorrect predictions. In this work, we use white-box untargeted attacks that aim to reduce the tracking performance by minimizing the mIoU metric. Adversarial attacks target a task loss, whereby the objective is to increase the loss by performing gradient ascent. Given the predicted and ground-truth bounding boxes $\hat{y}_x$ and $\bar{y}_x$ respectively, we use the L1-loss as the task loss as proposed in [119] for attacking an object tracker:

$$\mathcal{L}_{ADV}(\hat{y}_x, \bar{y}_x) = \|\hat{y}_x - \bar{y}_x\|_1 \tag{5.3}$$

Based on means of application of the adversarial perturbation and additional constraints placed on the perturbation strength, attacks can be further classified into two distinct types:

Digital Perturbation Attacks. These attacks introduce fine-grained pixel perturbations that are imperceptible to humans [79, 48, 54]. Digital perturbation attacks can manipulate any pixel of the input, but place imperceptibility constraints such that the adversarial output x_{adv} is within an l_p-ball of the benign input x_{ben}, *i.e.*, $\|x_{adv} - x_{ben}\|_p \leq \epsilon$. Such attacks, although having high efficacy, are considered to be physically non-realizable. This is due to the spatially unbounded granular pixel manipulation of the attack as well as the fact that a different perturbation is typically applied to each frame of a video sequence.

Physically Realizable Attacks. These attacks place constraints on the input space that can be manipulated by the attack [47, 118, 119, 121]. In doing so, the adversarial perturbations can be contained within realistic objects in the physical world, such as a printed traffic sign [47] or a T-shirt [121]. As an attacker can completely control the form of the physical adversarial artifact, physically realizable attacks are unbounded in the perceptual space and place no constraints on the perturbation strength. In this work, we consider a physically realizable attack based on [119] that produces a background patch perturbation to fool an object tracker (Figure 5.1). It is an iterative attack that follows gradient ascent for the task loss described in Equation (5.3) by adding a perturbation to the input that is a product of the input gradient and a step size δ:

$$x^{(i)} = x^{(i-1)} + \delta \nabla_{x^{(i-1)}} \mathcal{L}_{ADV} \tag{5.4}$$

5.4 EXPERIMENT SETUP

We perform extensive experiments and demonstrate that models trained with MTL are more resilient to adversarial attacks. The multi-task setting consists of jointly training a shared backbone for semantically related tasks such as person tracking and human keypoint detection (Section 5.4.1). We evaluate the tracking robustness on a state-of-the-art physically realizable adversarial attack for object trackers. We test our models on a photo-realistic simulated dataset as well as a real-world video dataset for person tracking (Section 5.4.4).

5.4.1 Architecture

For tracking, we leverage a SiamRPN model (Figure 5.2) with an AlexNet backbone and an RPN head as described in [127]. The inputs to the model are a template frame (127×127) and a detection frame (255×255), fed to the backbone network. Finally, the RPN head of the model produces classification and localization artifacts corresponding to $m = 5$ anchors for each spatial position. The anchors have aspect ratios of $\{0.33, 0.5, 1, 2, 3\}$ respectively.

A keypoint head is also attached to the template branch of the network, *i.e.*, the keypoint head receives the activation map with dimensions $6 \times 6 \times 256$ as input. We attach the keypoint head to the template branch as the template frame has a more focused context, and typically has only one subject in the frame, leading to more stable keypoint training. The base keypoint head has 2 convolutional blocks with $\{128, 64\}$ channels respectively. We also perform ablation experiments by increasing the depth of the keypoint head to 4 blocks with $\{128, 128, 64, 64\}$ channels respectively (Section 5.5.2). The convolutional blocks are followed by a transpose convolution block with 17 output channels, which is the same as the number of human keypoints represented in the MS COCO format [160]. Bilinear interpolation is performed on the output of the transpose convolution block to expand the spatial output dimensions, yielding an output with dimensions $127 \times 127 \times 17$. Hence each of the 17 channels correspond to spatial logit scores for the 17 keypoints.

5.4.2 Training Data

We found that there is a dearth of publicly available tracking datasets that support ad-hoc tasks for enabling multi-task learning. Hence, for our MTL training, we create a hybrid dataset that enables jointly training with person tracking and human keypoint detection. For human keypoint annotations, we leverage the MS COCO dataset [160] which contains more than $200k$ images and $250k$ person instances, each labeled with 17 human keypoints. The MS COCO dataset also annotates person bounding boxes that we use for the tracking scenario. As the MS COCO dataset consists of single images, there is no notion of temporal sequences in the input. Hence, for person tracking, we leverage data augmentation to differentiate the template and detection frames for the person instance annotation from the same image. Therefore, the MS COCO dataset allows us to train both the RPN head and keypoint head jointly for person tracking and human keypoint detection. We use the defined train and val splits for training and validation. Additionally, we merge this data with the Large-scale Single Object Tracking (LaSOT) dataset [161]. Specifically, we extract all videos for the "*person*" class for training the person tracking network. This gives us 20 video sequences, of which we use the first 800 frames from each sequence for training and

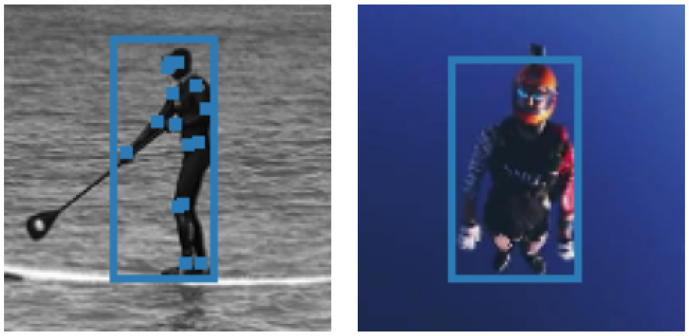


Figure 5.3: Annotated training examples from the MS COCO (left) and LaSOT (right) datasets for person tracking. MS COCO has additional human keypoint annotations.

the subsequent 100 frames for the validation set. Hence, the combined hybrid dataset from MS COCO and LaSOT enables our multi-task training. Figure 5.3 shows 2 example frames from MS COCO and LaSOT datasets.

5.4.3 Multi-Task Training

For the multi-task training, we fine-tune a generally pre-trained SiamRPN object tracker jointly for the tasks of person tracking and human keypoint detection. As we are specifically interested in the impact of multi-task training, we use the same loss weights λ_C and λ_R as proposed in [127] for the tracking loss $\mathcal{L}_{TRK}$. We perform an extensive sweep of the MTL loss weight λ_K associated with the keypoint loss $\mathcal{L}_{KPT}$ (Section 5.5.1). For the baseline, we perform single-task learning (STL) for person tracking by dropping the keypoint head and only fine-tuning the RPN head with the backbone, *i.e.*, the STL baseline has $\lambda_K = 0$. All STL and MTL models are trained with a learning rate of 8×10^{-4} that yields the best baseline tracking results as verified using a separate validation set. We also study the impact of pre-training the keypoint head separately before performing MTL (Section 5.5.3). For pre-training the keypoint head, we drop the RPN head and freeze the parameters of the backbone network. This ensures that the RPN head parameters are still compatible with the backbone after pre-training the keypoint head. The keypoint head is pre-trained with a learning rate of 10^{-3}. We train all models for 50 epochs and select the models with best validation performance over the epochs.

5.4.4 Evaluation

We evaluate our trained STL and MTL models for the tracking scenario using the mIoU metric between ground-truth and predicted bounding boxes, which is first averaged over all frames for a sequence, and finally averaged over all sequences.

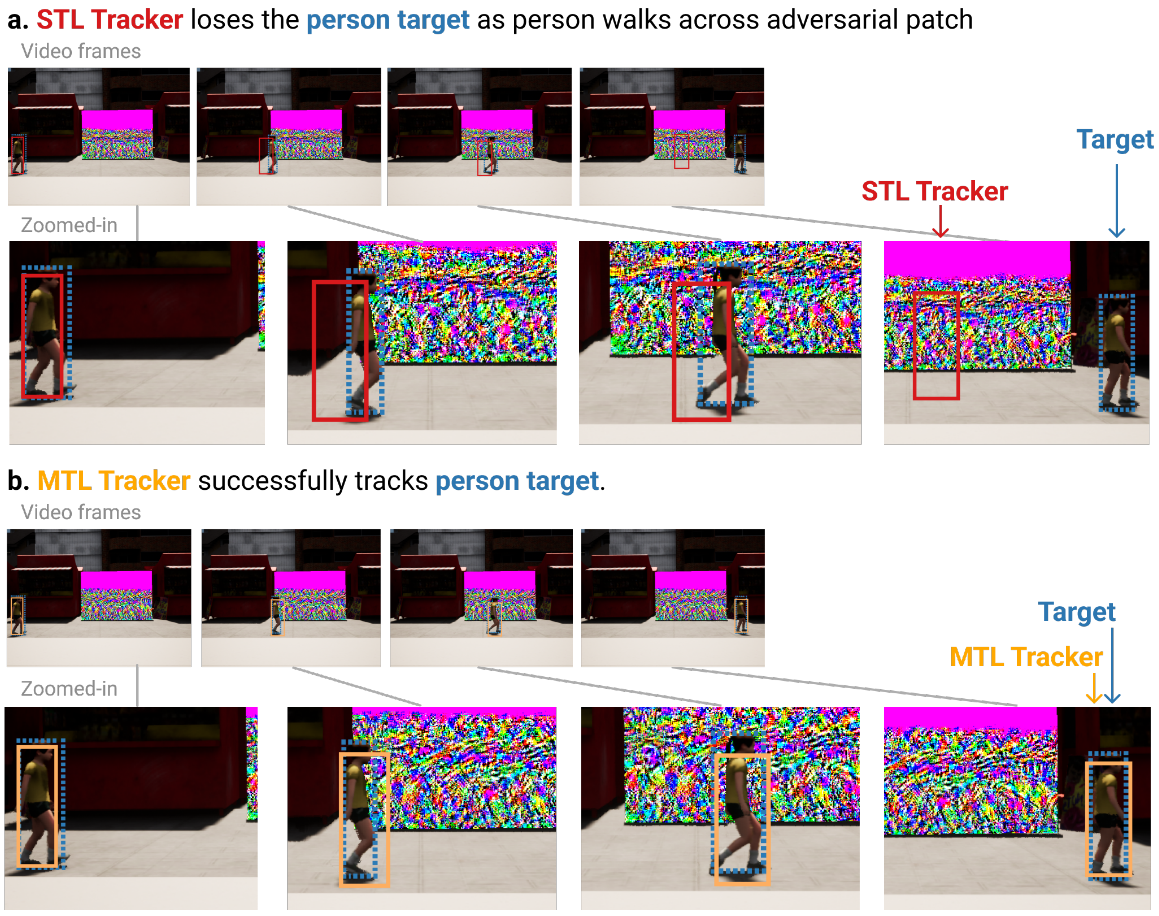

a. STL Tracker loses the person target as person walks across adversarial patch
Video frames
Zoomed-in
STL Tracker
Target
b. MTL Tracker successfully tracks person target.
Video frames
Zoomed-in
Target
MTL Tracker

Figure 5.4: Example video frames from the ARMORY-CARLA dataset showing static adversarial patches for (a) STL and (b) MTL for an attack with $\delta = 0.1$ and 10 steps. The patch is able to lock onto the STL tracker prediction (top), whereas the MTL tracker is consistently able to track the target (bottom).

For testing the adversarial robustness of person tracking in a practical scenario, we leverage a state-of-the-art physically realizable adversarial attack for object trackers [119]. The attack adds a static adversarial background patch to a given video sequence that targets the tracking task loss $\mathcal{L}_{ADV}$. At each iteration of the attack, gradient ascent is performed on the task loss as per Equation (5.4) with a step size δ. In order to observe the effect of varying the step size and attack iterations, we experiment with multiple values and report results for $\delta = \{0.1, 0.2\}$, which we found to have stronger adversarial effect on the tracking performance. The attack proposed in [119] has no imperceptibility constraints and is unbounded in the perceptual space, and can thus be considered an extremely effective adversarial attack. As the attack relies on the gradients of the task loss, we implement an end-to-end differentiable inference pipeline for the SiamRPN network using the Adversarial Robustness Toolbox (ART) framework [162].

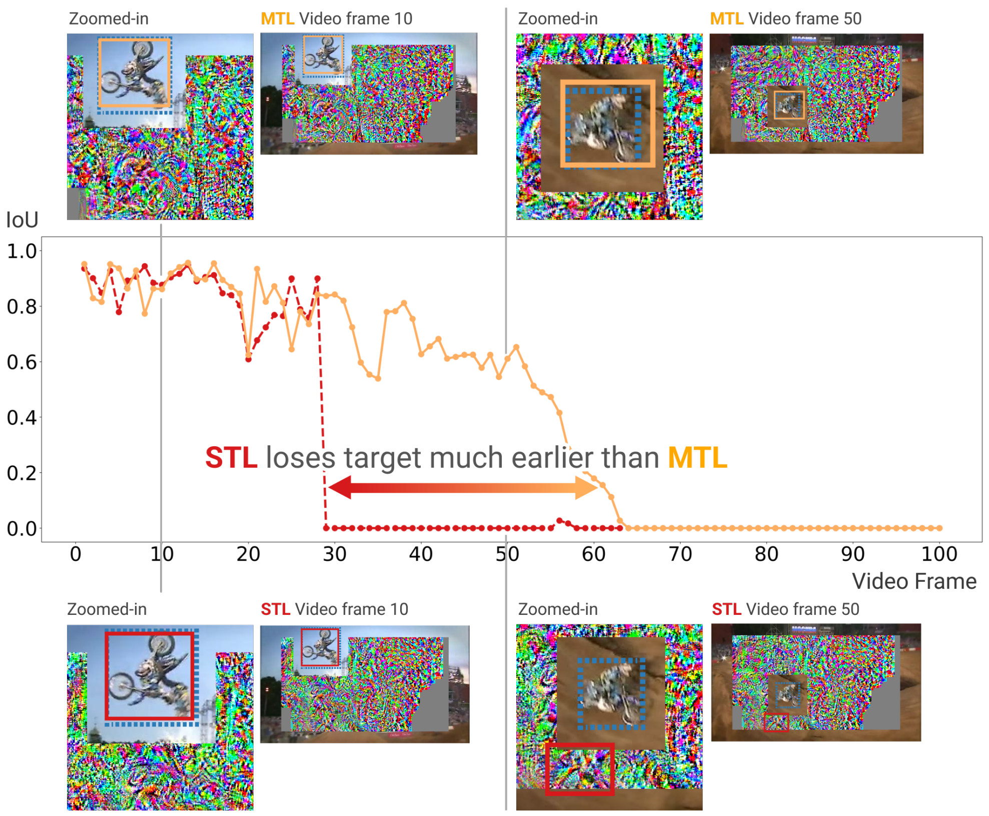


Figure 5.5: Example video frames and the corresponding adversarial IoU charts for the video from the OTB2015-Person dataset showing the constructed static adversarial patches for STL (red) and MTL (orange) for an attack with $\delta = 0.1$ and 10 steps. The dashed blue box shows the ground-truth target. The attack misleads the STL tracker early, but struggles to mislead the MTL tracker until much later. The unperturbed gray regions in the patch are locations which are never predicted by the tracker.

We evaluate the adversarial robustness of STL and MTL models on 2 datasets:

ARMORY-CARLA. This is a simulated photo-realistic person tracking dataset created using the CARLA simulator [163], provided by the ARMORY test bed [164] for adversarial ML research. We use the "*dev*" dataset split. The dataset consists of 20 videos of separate human sprites walking across the scene with various background locations. Each video has an allocated patch in the background that can be adversarially perturbed to mimic a physically realizable attack for person tracking. The dataset also provides semantic segmentation annotations to ensure that only the patch pixels in the background are perturbed when a human sprite passes in front of the patch. Figure 5.4 shows example video frames from the dataset where this can be seen. We find that the SiamRPN person tracker, having been trained on real-world datasets, has a reasonably high mIoU for tracking the human sprites when there is no attack performed; thus qualifying the photo-realism of the simulated scenario.

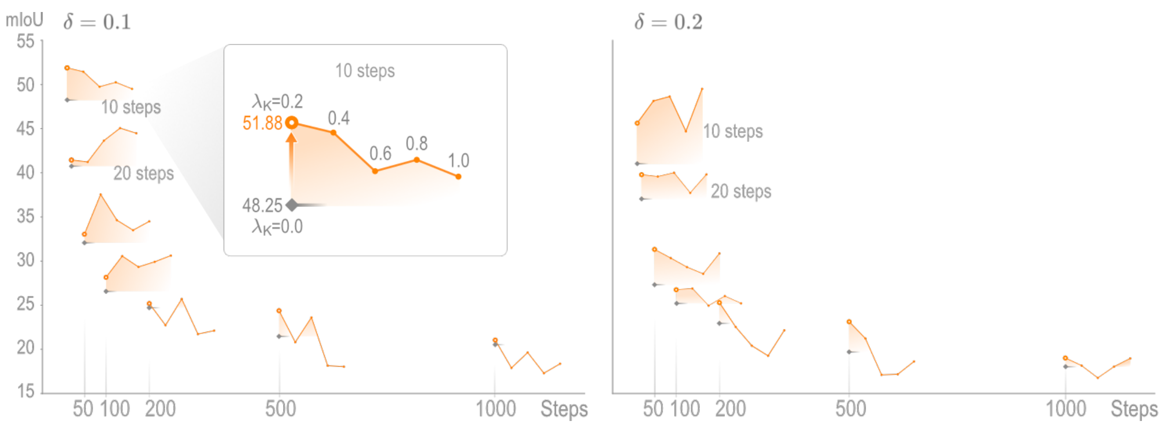


Figure 5.6: A unified visualization of the adversarial mIoU results from Table 5.1 for the ARMORY-CARLA dataset with $\delta = 0.1$ (left) and $\delta = 0.2$ (right). The orange dots represent the MTL mIoU's and the gray flat lines represent the STL baseline mIoU's. We see that the hollow orange dots ($\lambda_K = 0.2$) are consistently above the STL baseline.

OTB2015-Person. We use the Object Tracking Benchmark (OTB2015) [165] to test the robustness of MTL for person tracking on a real-world dataset. We extract all videos that correspond to the task of person tracking, which yields 38 videos that we call the OTB2015-Person split. As the dataset is intended for real-world tracking and is not readily amenable to implement physically realizable attacks, we digitally modify the videos for our attack to work. For each video, we overlay a static background patch that has a margin of 10% from each edge, covering 64% of the total area that can be perturbed by the attack. Finally, for each frame of a video, we only uncover the region annotated by the ground-truth bounding box with a padding of 25 pixels on each side. Hence, the annotated subject is always completely visible to the tracker with a digitally perturbed adversarial patch boundary. Figure 5.5 shows example video frames with the static patch attack as described here. To ensure that the tracker gets a clean ground-truth template, we do not perturb the first frame. Since this implements an unbounded digital attack on inputs from the real-world perceptual space, the attack is much stronger than real-world physically realizable attacks. For computational tractability, we only attack the first 100 frames.

5.5 RESULTS

Our experiments reveal that models trained with MTL consistently make it harder for an adversarial attack to succeed by making the shared backbone network learn robust MTL features. Given an iterative attack, higher number of iterations corresponds to increased attack difficulty and higher attacker cost. We report the mIoU for increasing attack steps from {10, 20, 50, 100, 200, 500, 1000} for the ARMORY-CARLA dataset in Table 5.1. We

Table 5.1: Adversarial mIoU results for ARMORY-CARLA dataset (↑ is better). Values highlighted in orange show the cases in which MTL is more robust than STL. MTL model with $\lambda_K = 0.2$ is consistently harder to attack than the STL model, and most often has the best performance. This table is also visualized in Figure 5.6 for clarity.

		STL	MTL				
	Steps	$\lambda_K = 0.0$	$\lambda_K = 0.2$	$\lambda_K = 0.4$	$\lambda_K = 0.6$	$\lambda_K = 0.8$	$\lambda_K = 1.0$
benign	0	69.45	69.59	69.46	69.70	72.20	**72.08**
$\delta = 0.1$	10	48.25	**51.88**	51.44	49.74	50.24	49.50
	20	40.70	41.44	41.22	43.63	**45.05**	44.47
	50	32.07	33.04	**37.54**	34.63	33.49	34.49
	100	26.57	28.16	30.56	29.33	29.91	**30.62**
	200	24.72	25.19	22.73	**25.70**	21.73	22.12
	500	21.47	**24.38**	20.81	23.61	18.15	18.04
	1000	20.54	**21.05**	17.90	19.64	17.30	18.37
$\delta = 0.2$	10	41.03	45.62	48.13	48.63	44.68	**49.50**
	20	37.04	39.78	39.57	**40.00**	37.72	39.81
	50	27.32	**31.32**	30.33	29.31	28.55	30.86
	100	25.24	26.76	**26.89**	24.95	26.03	25.21
	200	22.95	**25.29**	22.54	20.41	19.27	22.16
	500	19.71	**23.13**	21.23	17.11	17.18	18.63
	1000	18.04	**19.02**	18.16	16.77	18.04	18.97

orange = MTL $>$ STL; gray = MTL $\leq$ STL; **bold** = highest in row

also visually summarize these gains from the MTL approach for the ARMORY-CARLA dataset in Figure 5.6. For $\lambda_K = 1.0$, which is the base MTL setting, the MTL model improves upon the benign mIoU from 69.45 to 72.08. Additionally, the MTL model is more robust than the STL baseline up to 100 attack steps for $\delta = 0.1$ and 50 attack steps for $\delta = 0.2$. This implies the attack cost is higher for attacking an MTL model compared to its STL counterpart. The mIoU for increasing attack steps for the OTB2015-Person dataset are shown in Table 5.2. We observe a degradation in the benign MTL performance in this case, which may partly be attributed to the resolution mismatch between the high resolution training examples [160, 161] from MS COCO and LaSOT datasets, compared to lower resolution evaluation videos samples from the OTB2015 dataset. In the adversarial case, the base MTL model is more robust than STL baseline for up to 200 steps for $\delta = 0.1$. For $\delta = 0.2$, the base MTL model fails to show robustness for 20 steps, and slightly better robustness for other attack steps. We see further improvements in the adversarial resiliency for varying λ_K, discussed in Section 5.5.1.

Table 5.2: Adversarial mIoU results for OTB2015-Person dataset (↑ is better). In most cases, MTL models are harder to attack compared to STL model, with $\lambda_K = 0.2$ being most robust to the attack across several attack steps and step sizes.

		STL	MTL				
	Steps	$\lambda_K = 0.0$	$\lambda_K = 0.2$	$\lambda_K = 0.4$	$\lambda_K = 0.6$	$\lambda_K = 0.8$	$\lambda_K = 1.0$
benign	0	69.42	68.62	67.84	67.89	65.97	68.50
$\delta = 0.1$	10	54.29	57.29	57.78	**57.95**	57.09	56.24
	20	52.62	55.45	**55.68**	55.56	53.01	52.68
	50	48.54	**52.84**	52.33	50.54	50.94	50.67
	100	44.92	**52.45**	48.54	48.63	48.77	48.25
	200	45.40	47.73	**49.18**	47.26	46.50	47.34
$\delta = 0.2$	10	53.93	**57.65**	56.60	55.70	56.46	54.34
	20	53.57	55.21	**56.16**	56.08	54.46	52.86
	50	49.15	**52.81**	51.12	49.48	50.71	49.65
	100	47.27	**52.72**	49.74	48.13	49.87	47.81
	200	46.19	**51.05**	48.83	47.24	48.26	47.04

orange = MTL > STL; gray = MTL ≤ STL; **bold** = highest in row

5.5.1 Varying MTL Weight

We study the effect of varying the MTL weight λ_K, which controls the amount of keypoint loss $\mathcal{L}_{KPT}$ that is backpropagated. We train separate models by enumerating $\lambda_K = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and perform adversarial patch attack on each model for multiple adversarial settings. The results for ARMORY-CARLA and OTB2015-Person are shown in Table 5.1 and Table 5.2 respectively. We find that for the given shallow keypoint head architecture ($\{128, 64\}$ channels), a lower value of λ_K is more optimal under adversarial attack. For both datasets, the MTL model with $\lambda_K = 0.2$ is consistently harder to attack than the STL model, and most often has the best performance for the corresponding adversarial setting. Since a shallow keypoint head has relatively lower learning capacity, a higher λ_K value will force the shared backbone to focus excessively on the keypoint detection task, causing deterioration in the robust MTL features learned for the tracking task. From Table 5.1, although we observe better generalization for the MTL model with $\lambda_K = 1.0$ in the benign case (mIoU = 72.08), the adversarial robustness quickly gives away (at 100 steps for $\delta = 0.2$). Conversely, a lower value of $\lambda_K = 0.2$ offers the best trade-off for generalization and robustness.

5.5.2 Increasing Depth of Keypoint Head

Following the observations with a shallow keypoint head architecture, we also experiment with increasing the depth of the keypoint head from $\{128, 64\}$ channels to $\{128, 128, 64, 64\}$

Table 5.3: Ablation study with the ARMORY-CARLA dataset for attack step size $\delta = 0.1$. We report the adversarial mIoU results ($\uparrow$ is better).

	$\lambda_K = 0.0$	$\lambda_K = 0.2$				$\lambda_K = 1.0$			
		not pre-trained		pre-trained		not pre-trained		pre-trained	
Steps	(STL)	shallow	deep	shallow	deep	shallow	deep	shallow	deep
0	69.45	69.59	66.85	69.62	69.36	**72.08**	67.28	64.14	69.40
10	48.25	51.88	45.28	47.05	48.70	49.50	**55.46**	49.91	49.77
20	40.70	41.44	38.54	39.94	40.47	44.47	**47.44**	43.10	42.55
50	32.07	33.04	31.71	34.11	35.28	34.49	**36.52**	36.31	32.96
100	26.57	28.16	27.21	31.33	32.04	30.62	30.07	**32.14**	31.60
200	24.72	25.19	24.15	25.67	25.34	22.12	**26.53**	25.16	24.77

orange = MTL $>$ STL; gray = MTL $\leq$ STL; **bold** = highest in row

channels, doubling the parameters of the keypoint head network. Table 5.3 shows the ablation results for the shallow and deep keypoint heads with the ARMORY-CARLA dataset for $\lambda_K = \{0.2, 1.0\}$ and $\delta = 0.1$. In this section we will focus on the "not pre-trained" results. The robustness of the MTL model degrades for $\lambda_K = 0.2$ when the model depth is increased, and is easier to attack compared to the STL model. However, the MTL model with deeper keypoint head has the best adversarial robustness for a higher $\lambda_K = 1.0$, even outperforming the MTL model with shallow keypoint head for $\lambda_K = 0.2$. As the deeper keypoint head has a relatively higher learning capacity, it can learn to detect keypoints with smaller changes to the feature space of the backbone network. Hence, a higher λ_K is required to adequately supervise the backbone in learning robust MTL features. Although we see a decline in the benign mIoU for increasing depth, the deep MTL model with $\lambda_K = 1.0$ has overall best robustness. On the other hand, the shallow MTL model with $\lambda_K = 0.2$ has better adversarial robustness than the STL model as well as better benign performance.

5.5.3 Pre-training the Keypoint Head

As we start with a pre-trained SiamRPN model and an untrained keypoint head, we also study the impact of pre-training the keypoint head before performing MTL fine-tuning. Table 5.3 shows the results of this ablation study. We report the MTL performance with and without pre-training the keypoint head with the ARMORY-CARLA dataset for $\lambda_K = \{0.2, 1.0\}$ and $\delta = 0.1$. For the shallow keypoint head architecture, we see minor improvements in the MTL performance for a higher value of $\lambda_K = 1.0$, especially at higher number of attack steps. However, there is a sharp decrease in the benign performance (benign mIoU = 64.14). On the other hand, the deep keypoint head architecture shows relative improvement with pre-training for a lower value of $\lambda_K = 0.2$ (benign mIoU = 69.36). Overall, there is no significant advantage observed from pre-training the keypoint head. A pre-trained keypoint head would have lower potential to significantly modify the learned feature space of the shared backbone as it is already near the optima for the keypoint loss space.

5.6 CONCLUSION

We perform an extensive set of experiments with adversarial attacks for the task of person tracking to study the impact of multi-task learning. Our experiments on simulated and real-world datasets reveal that models trained with multi-task learning for the semantically analogous tasks of person tracking and human keypoint detection are more resilient to physically realizable adversarial attacks. Our work is the first to uncover the robustness gains from multi-task learning in the context of person tracking for physically realizable attacks. As the tracking use case has widely ranging real-world applications, the threat of adversarial attacks has equally severe implications. We hope our work triggers new research in this direction to further secure tracking models from adversarial attacks.

CHAPTER 6
HEAR NO EVIL: TOWARDS ADVERSARIAL ROBUSTNESS OF AUTOMATIC SPEECH RECOGNITION VIA MULTI-TASK LEARNING

As automatic speech recognition (ASR) systems are now being widely deployed in the wild, the increasing threat of adversarial attacks raises serious questions about the security and reliability of using such systems. On the other hand, multi-task learning (MTL) has shown success in training models that can resist adversarial attacks in the computer vision domain. In this work, we investigate the impact of performing such multi-task learning on the adversarial robustness of ASR models in the speech domain. We conduct extensive MTL experimentation by combining semantically diverse tasks such as accent classification and ASR, and evaluate a wide range of adversarial settings. Our thorough analysis reveals that performing MTL with semantically diverse tasks consistently makes it harder for an adversarial attack to succeed. We also discuss in detail the serious pitfalls and their related remedies that have a significant impact on the robustness of MTL models. Our proposed MTL approach shows considerable absolute improvements in adversarially targeted WER ranging from 17.25 up to 59.90 compared to single-task learning baselines (attention decoder and CTC respectively). Ours is the first in-depth study that uncovers adversarial robustness gains from multi-task learning for ASR.

6.1 INTRODUCTION

Automatic speech recognition (ASR) systems have penetrated our daily lives with real-world applications such as digital voice assistants, IVR and news transcription. These are increasingly relying on deep learning methods for their superior performance. At the same time, a mature body of adversarial machine learning research has exposed serious vulnerabilities in these underlying methods [166, 167, 168, 169, 170, 171, 172], raising grave concerns regarding the trustworthiness of ASR applications. There is an urgent need to address these vulnerabilities for restoring faith in using ASR for safety-critical functions. Our study aims to push the envelope in this direction with a meticulous approach.

An adversarial attack on an ASR model allows the attacker to introduce faint noise to a speech sample that can influence the model into making an exact transcription of the attacker's choosing. Since this targeted adversarial scenario is considered more threatening for an ASR system as compared to adding noise that leads to some arbitrary prediction [168, 171], we focus on studying the characteristics of ASR models that can thwart *targeted* adversarial attacks. Research has shown that adversarial examples are manifestations of non-robust features learned by a deep learning model [159]. Hence, our objective is to regularize the ASR model training paradigm so as to learn robust features that can resist such attacks.

Multi-task learning (MTL) is one such approach that has shown some success in this aspect for computer vision tasks by making the underlying models more resilient to adversarial attacks [128, 129]. However, it is unclear whether such robustness would transfer to the audio modality. Moreover, hybrid ASR models are often trained with semantically equivalent tasks like CTC and attention decoding [173]. This raises interesting questions about their inherent adversarial robustness. In this work, we aim to study the impact of MTL on the adversarial robustness of ASR models, and compare it to robustness of single-task learning (STL). Here, we consider MTL as jointly training a shared feature encoder with multiple losses from diverse task heads. Our expectation is that MTL would induce the encoder to learn a robust feature space that is harder to attack [159]. We consider semantically equivalent as well as semantically diverse tasks for performing MTL. We find that a combination of both types of tasks is necessary to most effectively thwart adversarial attacks. We also discuss serious pitfalls related to inference for MTL models that have an adverse impact on robustness. Finally, we demonstrate remedies for such pitfalls that significantly make it harder for an attacker to succeed.

Contributions

- **First MTL Study of Adversarial Robustness for ASR.** To the best of our knowledge, this is the first work to uncover adversarial robustness gains from MTL for ASR models.
- **Extensive Evaluation.** We perform extensive experimentation with one of the most powerful adversarial attacks, and evaluate models trained with semantically equivalent as well as semantically diverse tasks across a wide range of training hyperparameters and strong adversarial settings.
- **Robustness of Hybrid ASR Inference.** Our study exposes the extreme vulnerability of using CTC head for inference in hybrid CTC/attention models; while showing that MTL training with CTC and attention loss improves resiliency to adversarial attacks if CTC head is dropped during inference.
- **Robust ASR with Semantically Diverse Tasks.** Our thorough analysis reveals that performing MTL with semantically diverse tasks such as combining ASR with accent classification makes it most difficult for an attacker to induce a maliciously targeted prediction. Our MTL approach shows considerable absolute improvements in adversarially targeted WER ranging from 17.25 up to 59.90 compared to STL baselines (attention decoder and CTC respectively).

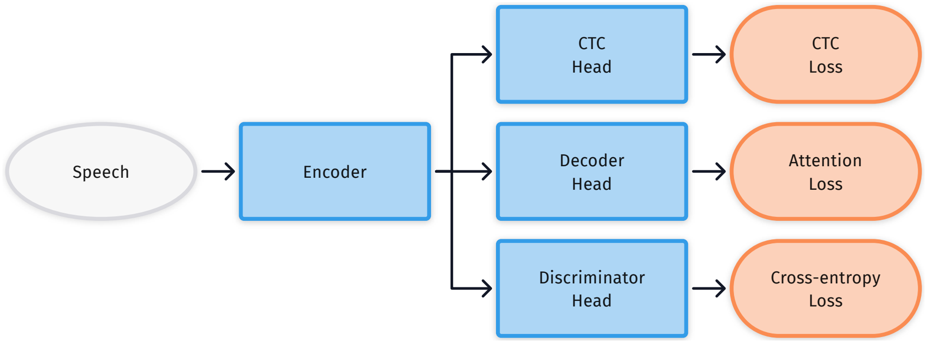


Figure 6.1: Overview of an MTL framework for ASR with accent classification. A shared encoder feeds into multiple task heads with corresponding losses that are jointly optimized.

6.2 RELATED WORKS

Several adversarial attacks have been proposed for maliciously influencing ASR models by leveraging model gradients to optimize a faint perturbation to the input speech [166, 167, 169, 170, 172]. Many such gradient-based adversarial techniques can be formulated as variations to the projected gradient descent (PGD) method [54], which is one of the strongest digital perturbation attacks proposed in the adversarial ML literature. In this work, we experiment with the *targeted* PGD attack, as this attack scenario is considered more threatening for ASR systems [168, 171].

Defenses proposed to evade adversarial attacks on ASR models mostly employ input preprocessing [174, 175, 176, 177] that places an undue burden on inference-time computation. Adversarial training has also shown some success in improving adversarial robustness [54, 178]. However, these methods are extremely computationally expensive. MTL with a shared backbone has the potential to provide a reasonable middle ground as it is much less computationally expensive than adversarial training, and does not introduce any inference-time load. Many studies have indeed looked at MTL in the context of ASR robustness [173, 179, 180, 181, 182, 183]. However, the bulk of such works focus mostly on improvement in benign performance (when no attack is performed) or robustness to arbitrary background noise. Ours is the first work to study the impact of MTL on ASR models specifically in the context of adversarial attacks.

6.3 APPROACH

In this work, we study the impact of MTL on adversarial robustness of ASR models by jointly training a shared feature encoder ϕ. The encoder takes a speech sample x as input, and outputs $\phi(x)$, which can be considered as a latent sequence embedding in a shared feature space. The feature embedding is then passed to various task heads with corresponding losses. This approach is depicted in Figure 6.1. In Section 6.3.1, we will discuss this MTL setup in more detail. We will then outline the adversarial setting evaluated in this work in Section 6.3.2.

6.3.1 ASR with Multi-Task Learning

We consider semantically diverse tasks in addition to semantically equivalent tasks for performing MTL. The semantically equivalent task heads for ASR are: (1) CTC and (2) attention decoder. We also jointly train a discriminator for a semantically diverse task such as accent classification.

For the ASR task, we denote the ground-truth transcription as $\bar{y}$, and the CTC loss [184] and decoder attention loss [185] as $\mathcal{L}_{\text{CTC}}$ and $\mathcal{L}_{\text{DEC}}$ respectively. We compute the loss for ASR as:

$$\mathcal{L}_{\text{ASR}}(x, \bar{y}) = \lambda_C^{(t)} \mathcal{L}_{\text{CTC}}(x, \bar{y}) + (1 - \lambda_C^{(t)}) \mathcal{L}_{\text{DEC}}(x, \bar{y}) \tag{6.1}$$

Consequently, for performing joint ASR inference, we denote the CTC and decoder scoring functions [173] as f_{CTC} and f_{DEC} respectively. Finally, we determine the predicted output transcription $\hat{y}$ as follows:

$$\hat{y} = \lambda_C^{(i)} f_{\text{CTC}}\big(\phi(x)\big) + (1 - \lambda_C^{(i)}) f_{\text{DEC}}\big(\phi(x)\big) \tag{6.2}$$

Note here that $\lambda_C^{(t)}$ and $\lambda_C^{(i)}$ are training and inference weights respectively. Generally, we follow that $\lambda_C^{(i)} = \lambda_C^{(t)}$ in our experiments while performing inference, unless otherwise specified. Correspondingly, setting $\lambda_C^{(i)}$=1.0 allows us to use only the trained CTC head for inference, and vice-versa for the trained decoder head by setting $\lambda_C^{(i)}$=0.0.

For accent classification, we are given an accent label $\bar{z}$ that is to be predicted for a speech sample x. Besides being semantically diverse, accent classification is also functionally distinct as it is a *sequence-to-label* task, compared to the *sequence-to-sequence* task for ASR. Hence, denoting the cross-entropy loss for the discriminator head as $\mathcal{L}_{\text{DIS}}$, we compute the full MTL loss for jointly training the model as:

$$\mathcal{L}_{\text{MTL}}(x, \bar{y}, \bar{z}) = \lambda_A^{(t)} \mathcal{L}_{\text{ASR}}(x, \bar{y}) + (1 - \lambda_A^{(t)}) \mathcal{L}_{\text{DIS}}(x, \bar{z}) \tag{6.3}$$

From Equations (6.1) and (6.3), we can see that modulating the $\lambda_A^{(t)}$ and $\lambda_C^{(t)}$ weights allows us to independently modulate the effect of various heads during training, *e.g.*, setting $\lambda_A^{(t)}$=1.0 and $\lambda_C^{(t)}$=1.0 allows us to train using only the CTC head. Conversely, we can train only the decoder head by setting $\lambda_A^{(t)}$=1.0 and $\lambda_C^{(t)}$=0.0. These can also be considered as the single-task learning (STL) baselines. With $\lambda_A^{(t)} < 1.0$, we can train the model with the discriminator head included. We perform extensive experiments across a range of these hyperparameters to study the impact of MTL on ASR robustness to attacks.

6.3.2 Adversarial Attack on ASR with PGD

An adversarial attack on ASR introduces an inconspicuous and negligible perturbation δ to a speech sample that confuses the ASR model into making an incorrect prediction. In this work, we focus on the projected gradient descent (PGD) attack [54]. Specifically, we consider the *targeted* PGD attack, as it is considered more malicious and threatening for ASR systems [168, 171]. Given, a target transcription $\tilde{y}$, the targeted attack aims to minimize the following inference loss function so as to force the ASR model into making a prediction of the attacker's choosing:

$$\mathcal{L}_{\text{ADV}}(x, \tilde{y}) = \lambda_C^{(i)} \mathcal{L}_{\text{CTC}}(x, \tilde{y}) + (1 - \lambda_C^{(i)}) \mathcal{L}_{\text{DEC}}(x, \tilde{y}) \tag{6.4}$$

The PGD attack is an iterative attack that consists of two main stages. The first stage is the perturbation stage, wherein a small perturbation of step size α is computed in the direction of the gradient of $\mathcal{L}_{\text{ADV}}$ with respect to the sample from the previous iteration. This perturbation having a magnitude of α is added to the sample. The second stage is the projection stage that ensures that the perturbed sample remains within an ϵ-ball of the original input. This is also called the $L2$ threat model, as it uses the $L2$ norm for limiting the perturbation. Hence, the targeted PGD attack optimizes the perturbation δ as:

$$x_{\text{ADV}} = \operatorname*{argmin}_{\delta} \mathcal{L}_{\text{ADV}}(x + \delta, \tilde{y}), \text{ s.t. } \|\delta\|_2 \leq \epsilon \tag{6.5}$$

The perturbation and projection stages are performed iteratively, and the computational cost to the attacker increases with increasing number of iterations. In order to isolate and study the impact of our MTL training on adversarial robustness, we perform greedy ASR inference while implementing the attack. Since the adversarial objective is to induce a maliciously targeted prediction as opposed to untargeted arbitrary predictions, we analyze the MTL robustness to specifically evade such targeted attacks. Hence, we examine the adversarially targeted word error rate (abbreviated as **AdvTWER** hereon) in this work, which reports the word error rate of the prediction $\hat{y}$ with respect to the adversarial target transcription $\tilde{y}$, *i.e.,* a higher AdvTWER implies that the model is more robust in evading the attack. Conversely, a lower AdvTWER means that the attack was more successful.

We report the adversarially targeted word error rate as we find it to be more compelling in studying the MTL robustness. Correspondingly, we also observe equivalent robustness trends with respect to the benign word error rate.

6.4 EXPERIMENT SETUP

6.4.1 Data

We use annotated speech data from Mozilla's Common Voice dataset [186] for our experiments. Common Voice consists of naturally spoken human speech in a variety of languages. The dataset also includes demographic metadata like age, sex and accent that is self-reported by speakers, and the speech is validated by annotators. Specifically, we use the English language speech and extract accent-labeled speech samples for US and Indian accents, which are among the most abundantly available accented speech in the dataset. Using the splits as defined by the dataset, we get $\sim$260K samples for training and $\sim$1.2K samples for validation. Finally, we report the performance metrics on $\sim$1K samples obtained from the test split.

6.4.2 Model Architecture and Training

We leverage a pre-trained hybrid CTC-attention model that is publicly available [187], and fine-tune it using multi-task learning. The model consists of a conformer-based encoder [188] that is shared by multiple task heads. The encoder has 12 conformer blocks, each with 8 attention heads and a hidden unit size of 2048. The output size of the encoder is 512, which is consequently the size of the sequence embeddings from the shared feature space. The MTL model employs a CTC head and a decoder head for ASR. The decoder has 6 transformer blocks, with 8 attention heads and a hidden unit size of 2048. The ASR heads output scores for 5000 subword units. For accent classification, we use a dense feed-forward discriminator that passes the mean sequence embedding through 5 fully connected layers, and finally outputs class labels corresponding to the accents. The MTL models are implemented using the ESPnet2 toolkit [189]. We train several MTL models by extensively modulating the $\lambda_A^{(t)}$ and $\lambda_C^{(t)}$ weights. Each MTL model is trained with a learning rate of 10^{-3} for 30 epochs, and we pick the model with the lowest validation loss from all the epochs.

6.4.3 Adversarial Setting

For the targeted PGD attack, we generate a fixed set of adversarial transcriptions having varying lengths. For each sample, the target transcription $\tilde{y}$ is chosen in a deterministic manner by finding the adversarial transcription from the fixed set having the closest length to the original transcription $\bar{y}$. To ensure there is minimal word overlap between the original transcriptions and the adversarial transcriptions, we use a lorem-ipsum generator for generating the adversarial transcriptions. We leverage the targeted PGD attack implemented using the Adversarial Robustness Toolbox [162] for our experiments. While attacking the ASR model, we focus solely on the MTL robustness by performing greedy inference for determining the gradients, and no additional LMs are used. The targeted PGD attack is performed using an $L2$ threat model in a white-box attack for inference. We use an extremely strong perturbation limit of ϵ=2.0, beyond which we observed that the perturbed samples would no longer remain within natural constraints. As the attack's computational cost increases with increasing number of steps, we report the AdvTWER metric for performing multiple attack steps, up to as high as 2000 steps. We use a step size α=0.05, allowing the attack to make fine-grained perturbations at each step.

6.5 RESULTS

Our extensive experimentation reveals that increasing the proportion of multi-task learning (MTL) while training significantly improves the model's resiliency to adversarial attacks. Before analysing the adversarial performance of the models, we first briefly discuss the benign performance, *i.e.*, when no attack is performed. On the test set, the single-task learning (STL) baselines have a benign word error rate (WER) of 20.85 and 16.62 for the CTC and the decoder heads respectively. As has been documented by previous studies [179, 190, 181], models trained with MTL also show better benign performance compared to STL. For example, we find that training an MTL model with a decoder and discriminator ($\lambda_A^{(t)}$=0.8; $\lambda_C^{(t)}$=0.0) yields a benign WER of 15.86 on the test set. For the accent classification task as well, we observe a reasonable bening accuracy of $\sim$90% for the MTL models with $\lambda_A^{(t)} < 1.0$.

We now shift our focus to the adversarial performance of ASR. First, we discuss the adversarial robustness of MTL models with semantically equivalent tasks (CTC and decoder heads) in Section 6.5.1. Next, we study the combination of semantically diverse tasks for MTL in Section 6.5.2. An overview of the MTL robustness trends is also depicted in Figure 6.2, comparing different MTL combinations with STL baselines. We see that training with MTL makes it consistently harder for an attacker to succeed.

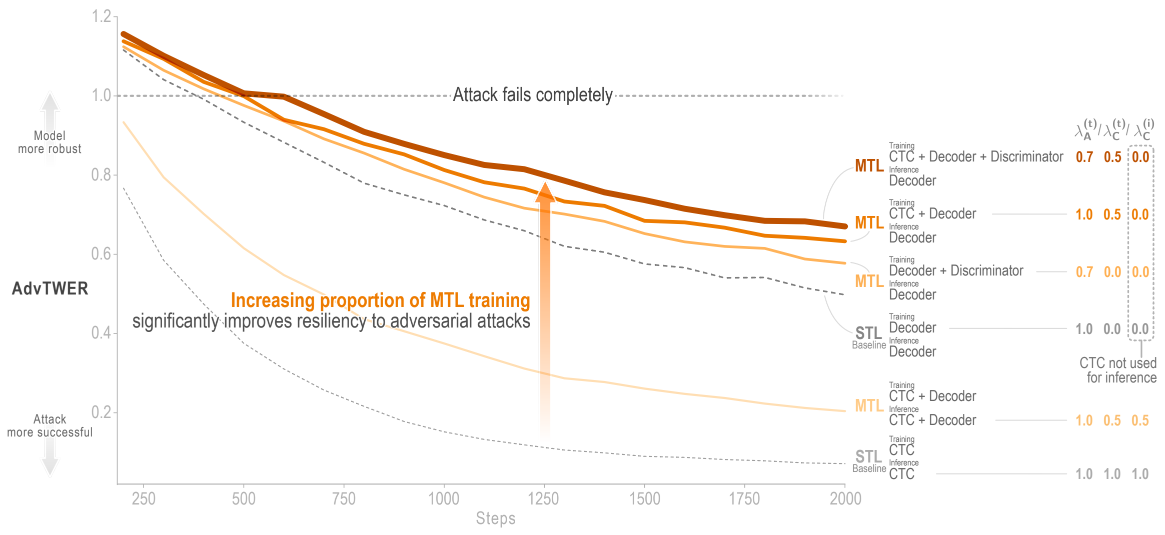


Figure 6.2: Adversarial performance for various MTL combinations. MTL outperforms STL baselines when CTC is dropped for inference.

6.5.1 MTL with CTC and Attention Decoder

We first study the adversarial performance of the semantically equivalent task heads by training jointly using the CTC and decoder heads. We train multiple models by setting $\lambda_A^{(t)}$=1.0 and modulating $\lambda_C^{(t)}$ from values $\{0.0, 0.3, 0.5, 0.7, 1.0\}$. We perform the PGD attack on each of these models as decribed in Section 6.4.3. For performing the attack, we follow two inference modes: (1) using the CTC head with same inference weights as training, *i.e.*, $\lambda_C^{(i)} = \lambda_C^{(t)}$; and (2) drop the CTC head for inference, *i.e.*, $\lambda_C^{(i)}$=0.0. It will become clear in the following discussion why we follow this. The adversarial performance for these inference modes is reported in Table 6.1 using the AdvTWER metric for various attack steps.

When the trained CTC head is included for inference, *i.e.*, $\lambda_C^{(i)} = \lambda_C^{(t)}$, we see from Table 6.1 that performing MTL with a combination of CTC and decoder heads has no robustness benefit compared to the STL decoder baseline. In fact, the STL CTC baseline itself is significantly more vulnerable to the adversarial attacks. Performing MTL moderately improves the robustness compared to the STL CTC baseline but it is still not better than the STL decoder baseline. This implies that the attacker is able to easily influence the CTC head into hijacking the overall joint prediction. This can be seen in Figure 6.2 with the clear gap between the gray STL baselines and the MTL model with CTC included in inference ($\lambda_A^{(t)}$=1.0; $\lambda_C^{(t)}$=$\lambda_C^{(i)}$=0.5). Decreasing $\lambda_C^{(t)}$ from 0.7 to 0.3 clearly shows worsening robustness in Table 6.1. This means that attacking a less trained CTC head (with lower $\lambda_C^{(t)}$) having a lower impact on the overall prediction (due to correspondingly

Table 6.1: AdvTWER (↑ is more robust) with $\lambda_A^{(t)}$=1.0. Training with CTC and dropping CTC for inference ($\lambda_C^{(i)}$=0.0) is more robust.

	$\lambda_C^{(i)} = \lambda_C^{(t)}$					$\lambda_C^{(i)} = 0.0$		
	STL (Decoder)	MTL with joint inference			STL (CTC)	MTL with Decoder inference		
$\lambda_C^{(t)} \rightarrow$	0.0	0.3	0.5	0.7	1.0	0.3	0.5	0.7
PGD-500	93.35	59.87	61.55	76.64	37.53	96.80	**99.95**	97.06
PGD-1000	72.31	35.21	37.49	53.91	15.19	79.36	81.29	**82.45**
PGD-1500	57.57	23.94	26.09	42.44	8.99	67.16	68.45	**69.57**
PGD-2000	49.79	19.22	20.40	36.99	7.14	60.15	**63.29**	60.56

gray = STL; highlighted = MTL $>$ STL; **bold** = highest in row

lower $\lambda_C^{(i)}$) is still able to overpower the more trained decoder head scores. This evident vulnerability of the CTC head may be attributed to the well studied peaky behavior of the CTC loss [191, 192] that would allow the attacker to induce overconfident prediction scores through the CTC head.

Next we study the effect of performing MTL by training the CTC head, but dropping the CTC head during inference, *i.e.*, $\lambda_C^{(i)}$=0.0. This means that the attacker can no longer manipulate the CTC head during inference, but the shared encoder is still jointly trained using the CTC loss. Immediately, we see the adversarial performance of the MTL models beat both the STL baselines in Table 6.1. It is consistently harder for the attacker to attack the MTL models across many attack steps. This is also visualized in Figure 6.2 for $\lambda_A^{(t)}$=1.0, $\lambda_C^{(t)}$=0.5 and $\lambda_C^{(i)}$=0.0.

6.5.2 Robust ASR with Semantically Diverse Tasks

We now study the impact of performing MTL with semantically diverse tasks. As we saw in Section 6.5.1 that the CTC head is extremely vulnerable to adversarial attacks, we first examine MTL with the decoder and discriminator combination, *i.e.*, we set $\lambda_C^{(t)}$=0.0. Table 6.2 shows the adversarial performance for training the decoder and discriminator heads by modulating $\lambda_A^{(t)}$ from values $\{1.0, 0.9, 0.8, 0.7, 0.6, 0.5\}$. We see that AdvTWER for $\lambda_A^{(t)}$=0.8, 0.7, 0.6 consistently outperforms the STL decoder baseline. This implies that the $\lambda_A^{(t)}$ should not be too high (less MTL) or too low (less ASR training) for optimal MTL robustness. We can also see this robustness in Figure 6.2 by comparing the STL decoder baseline with the MTL model for $\lambda_A^{(t)}$=0.7 and $\lambda_C^{(t)}$=$\lambda_C^{(i)}$=0.0. However, we see that decoder/discriminator MTL is still not able to beat CTC/decoder MTL when the CTC head is dropped for inference.

Table 6.2: AdvTWER (↑ is more robust) with $\lambda_C^{(t)}$=0.0. Training the decoder with a discriminator shows better robustness.

$\lambda_A^{(t)} \rightarrow$	1.0	0.9	0.8	0.7	0.6	0.5
PGD-500	93.35	89.60	95.48	**97.60**	97.04	90.94
PGD-1000	72.31	68.67	73.02	78.07	**78.57**	70.25
PGD-1500	57.57	54.88	60.88	**65.24**	63.15	57.88
PGD-2000	49.79	47.28	51.23	**57.75**	55.61	50.43

gray = STL (Decoder); highlighted = MTL > STL; **bold** = highest in row

Table 6.3: AdvTWER (↑ is more robust) with $\lambda_A^{(t)}$=0.7, $\lambda_C^{(i)}$=0.0. Training all heads combined shows most superior robustness.

$\lambda_C^{(t)} \rightarrow$	0.0	0.3	0.5	0.7
PGD-500	97.60	97.02	100.65	**101.04**
PGD-1000	78.07	76.83	**85.06**	83.69
PGD-1500	65.24	62.75	**73.70**	70.67
PGD-2000	57.75	56.57	**67.04**	65.79

gray = Baseline; highlighted = MTL > Baseline; **bold** = highest in row

Therefore, we next study the combination of all three heads (CTC, decoder and discriminator) for performing MTL while dropping the CTC head during inference ($\lambda_C^{(i)}$=0.0). For these experiments, we first set $\lambda_A^{(t)}$=0.7 which shows superior adversarial robustness in Table 6.2. We then modulate $\lambda_C^{(t)}$ from values $\{0.0, 0.3, 0.5, 0.7\}$. Table 6.3 shows the adversarial performance results for this setting. Similar to Table 6.1, we can see that increasing MTL training weight for the CTC head but dropping the CTC head during inference improves the overall robustness, with $\lambda_C^{(t)}$=0.5 showing the best robustness of all MTL models. Hence, combining all three heads and performing MTL with semantically diverse tasks makes the model most resilient to the adversarial attacks consistently across multiple attack steps. This can also be clearly seen in Figure 6.2 where the MTL model corresponding to $\lambda_A^{(t)}$=0.7, $\lambda_C^{(t)}$=0.5 and $\lambda_C^{(i)}$=0.0 outperforms all other MTL combinations and baselines.

6.6 CONCLUSION

In this work, we study the impact of multi-task learning (MTL) on the adversarial robustness of ASR models. We perform extensive experimentation with multiple training and inference hyperparameters as well as wide-ranging adversarial settings. Our thorough empirical testing reveals that performing MTL with semantically diverse tasks consistently makes it harder for an attacker to succeed across several attack steps. We also expose the extreme vulnerability of the CTC loss, and discuss related pitfalls and remedies for using the CTC head during training with MTL. In future work, we aim to investigate regularization methods that can reduce the peaky behavior of CTC so as to induce more adversarially robust hybrid inference.

Part III

Democratizing AI Security Research & Pedagogy with Scalable Interactive Experimentation

OVERVIEW

So far we have explored how we can deeply understand AI vulnerabilities and fortify AI models. However in the coming future, with the radical infusion of AI in everyday life, AI security will not only be restricted to the fancies of expert researchers. While visualization tools and intuitive vulnerability interpretation go a long way in enabling people to understand AI security more deeply, it still requires some level of expertise in the adversarial ML domain to successfully decipher these concepts. In this part of the thesis, we go that last mile to bring AI security research to the masses.

In Chapter 7, we take the first step towards this goal by developing **ADAGIO**, a web-based tool that allows real-time interactive experimentation with attacks and defenses on an automatic speech recognition (ASR) model. Through developing ADAGIO, and following the observations from our SHIELD research, we discover that the idea of input compression as a practical defense carries over to the audio domain as well. This chapter is adapted from our published work [174] that appeared at ECML PKDD 2018.

ADAGIO: Interactive Experimentation with Adversarial Attack and Defense for Audio. Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Li Chen, Michael E. Kounavis, Polo Chau. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018. PDF

Working on ADAGIO gave us the key insight that *interactive experimentation* on the web is an immensely powerful medium for research dissemination. In Chapter 8, we build upon this notion by developing **MLsploit**, the first open-source, scalable, web-based interactive system that allows seamless experimentation with adversarial ML research. MLsploit provides a modular repository of attacks and defenses, enabling practitioners to interactively study their AI applications under various threat models. Becoming available to thousands of students, MLsploit is already transforming AI security education at scale. This chapter is adapted from our project [193] that was presented at the KDD 2019 Project Showcase.

MLsploit: A Framework for Interactive Experimentation with Adversarial Machine Learning Research. Nilaksh Das, Siwei Li, Chanil Jeon, Jinho Jung*, Shang-Tse Chen*, Carter Yagemann*, Evan Downing*, Haekyu Park, Evan Yang, Li Chen, Michael Kounavis, Ravi Sahita, David Durham, Scott Buck, Polo Chau, Taesoo Kim, Wenke Lee. *Project Showcase at the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. PDF

*authors contributed equally

Through developing such interactive experimentation tools, we democratize AI security for new researchers, practitioners and students, and make adversarial ML research more accessible and more equitable for everyone.

CHAPTER 7
ADAGIO: INTERACTIVE EXPERIMENTATION WITH ADVERSARIAL ATTACK AND DEFENSE FOR AUDIO

Adversarial machine learning research has recently demonstrated the feasibility to confuse automatic speech recognition (ASR) models by introducing acoustically imperceptible perturbations to audio samples. To help researchers and practitioners gain better understanding of the impact of such attacks, and to provide them with tools to help them more easily evaluate and craft strong defenses for their models, we present ADAGIO, the first tool designed to allow interactive experimentation with adversarial attacks and defenses on an ASR model in real time, both visually and aurally. ADAGIO incorporates AMR and MP3 audio compression techniques as defenses, which users can interactively apply to attacked audio samples. We show that these techniques, which are based on psychoacoustic principles, effectively eliminate targeted attacks, reducing the attack success rate from 92.5% to 0%.

7.1 INTRODUCTION

Deep neural networks (DNNs) are highly vulnerable to adversarial instances in the image domain [48]. Such instances are crafted by adding small imperceptible perturbations to benign instances to confuse the model into making wrong predictions. Recent work has shown that this vulnerability extends to the audio domain [169], undermining the robustness of state-of-the-art models that leverage DNNs for the task of automatic speech recognition (ASR). The attack manipulates an audio sample by carefully introducing faint "noise" in the background that humans easily dismiss. Such perturbation causes the ASR model to transcribe the manipulated audio sample as a target phrase of the attacker's choosing. Through this research demonstration, we make two major contributions:

1. Interactive exploration of audio attack and defense. We present ADAGIO, the first tool designed to enable researchers and practitioners to interactively experiment with adversarial attack and defenses on an ASR model in real time[1]. ADAGIO incorporates AMR and MP3 audio compression techniques as defenses for mitigating perturbations introduced by the attack. Figure 7.1 presents a brief usage scenario showing how users can experiment with their own audio samples. ADAGIO stands for **A**dversarial **D**efense for **A**udio in a **G**adget with **I**nteractive **O**perations.

[1]see demo: https://youtu.be/0W2BKMwSfVQ

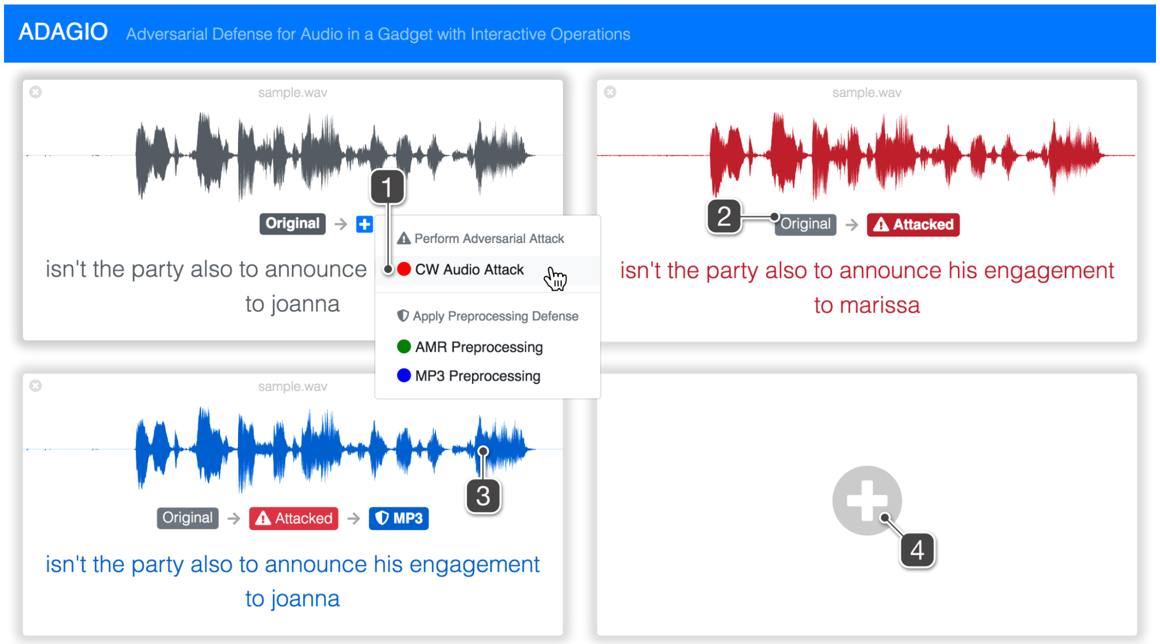

ADAGIO Adversarial Defense for Audio in a Gadget with Interactive Operations
sample.wav
Original
isn't the party also to announce to joanna
1
Perform Adversarial Attack
CW Audio Attack
Apply Preprocessing Defense
AMR Preprocessing
MP3 Preprocessing
sample.wav
2
Original
Attacked
isn't the party also to announce his engagement to marissa
sample.wav
Original
Attacked
MP3
3
isn't the party also to announce his engagement to joanna
4

Figure 7.1: ADAGIO usage scenario. (1) Jane uploads an audio file that is transcribed by DeepSpeech [194]; then she performs an adversarial attack on the audio in real time by entering a target transcription after selecting the attack option from the dropdown menu, e.g., the state-of-the-art Carlini-Wagner Audio Attack [169]. (2) Jane decides to perturb the audio to change the last word of the sentence from "joanna" to "marissa"; she can listen to the original audio and see the transcription by clicking on the "Original" badge. (3) Jane applies MP3 compression to recover the original, correct transcription from the manipulated audio; clicking on a waveform plays back the audio from the selected position. (4) Jane can experiment with multiple audio samples by adding more cards. For presentation, operations 1, 2 and 3 are shown as separate cards.

2. Compression as an effective defense. We demonstrate that non-adaptive adversarial perturbations are extremely fragile, and can be eliminated to a large extent by using audio processing techniques like Adaptive Multi-Rate (AMR) encoding and MP3 compression. We assume a non-adaptive threat model since an adaptive version of the attack is prohibitively slow and often does not converge.

7.2 ADAGIO: EXPERIMENTING WITH AUDIO ATTACK & DEFENSE

We first provide a system overview of ADAGIO, then we describe its primary building blocks and functionality. ADAGIO consists of four major components: (1) an interactive UI (Figure 7.1); (2) a speech recognition module; (3) a targeted attack generator module; and (4) an audio preprocessing (defense) module. The three latter components reside on a back-end server that performs the computation. The UI communicates the user intent with the back-end modules through a websocket messaging service, and uses HTTP to upload/download audio files for processing. When the messaging service receives an action to be performed from the front-end, it leverages a custom redis-based job queue to activate the correct back-end module. When the back-end module finishes its job, the server pings back the UI through the websocket messaging service to update the UI with the latest results. Below, we describe the other three components in ADAGIO.

7.2.1 Speech Recognition

In speech recognition, state-of-the-art systems leverage Recurrent Neural Networks (RNNs) to model audio input. The audio sample is broken up into frames $\{x^{(1)}, ..., x^{(T)}\}$ and fed sequentially to the RNN function $f(\cdot)$ which outputs another sequence $\{y^{(1)}, ..., y^{(T')}\}$, where each $y^{(t)}$ is a probability distribution over a set of characters. The RNN maintains a hidden state $h^{(t)}$ which is used to characterize the sequence up until the current input $x^{(t)}$, such that, $(y^{(t)}, h^{(t)}) = f(x^{(t-1)}, h^{(t-1)})$. The most likely sequence based on the output probability distributions then becomes the transcription for the audio input. The performance of speech-to-text models is commonly measured in Word Error Rate (WER), which corresponds to the minimum number of word edits required to change the transcription to the ground truth phrase.

ADAGIO uses Mozilla's implementation [195] of DeepSpeech [194], a state-of-the-art speech-to-text DNN model, to transcribe the audio in real time.

7.2.2 Targeted Audio Adversarial Attacks

Given a model function $m(\cdot)$ that transcribes an audio input x as a sequence of characters y, i.e., $m(x) = y$, the objective of the targeted adversarial attack is to introduce a perturbation δ such that the transcription is now a specific sequence of characters y' of the attacker's choosing, i.e., $m(x + \delta) = y'$. The attack is only considered successful if there is no error in the transcription.

ADAGIO allows users to compute adversarial samples using a state-of-the-art iterative attack [169]. After uploading an audio sample to ADAGIO, the user can click the attack button and enter the target transcription for the audio (see Figure 7.1.1). The system then runs 100 iterations of the attack and updates the transcription displayed on the screen at each step to show progress of the attack.

7.2.3 Compression as Defense

In the image domain, compression techniques based on psychovisual theory have been shown to mitigate adversarial perturbations of small magnitude [96]. We extend that

Table 7.1: Word Error Rate (WER) and the targeted attack success rate on the DeepSpeech model (lower is better for both). AMR and MP3 eliminate all targeted attacks, significantly improving WER.

Defense	WER (no attack)	WER (with attack)	Targeted attack success rate
None	0.369	1.287	92.45%
AMR	0.488	0.666	**0.00%**
MP3	0.400	0.780	**0.00%**

hypothesis to the audio domain and let users experiment with AMR encoding and MP3 compression on adversarially manipulated audio samples. Since these techniques are based on psychoacoustic principles (AMR was specially developed to encode speech), we posit that these techniques could effectively remove the adversarial components from the audio which are imperceptible to humans, but would confuse the model.

To determine the efficacy of these compression techniques in defending the ASR model, we created targeted adversarial instances from the first 100 test samples of the Mozilla's Common Voice dataset [186] using the attack as described in [169]. We constructed five adversarial audio instances for every sample, each transcribing to a phrase randomly picked from the dataset, yielding a total of 500 adversarial samples. We then preprocessed these samples before feeding it to the DeepSpeech model. Table 7.1 shows the results from this experiment. We see that the preprocessing defenses are able to completely eliminate the targeted success rate of the attack.

7.3 CONCLUSION

We present ADAGIO, an interactive tool that empowers users to experiment with adversarial audio attacks and defenses. We demonstrate and highlight ADAGIO's features using a usage scenario on the Mozilla Common Voice dataset. Using ADAGIO, users can also discover how MP3 and AMR compression can effectively resist targeted adversarial attacks.

CHAPTER 8
MLSPLOIT: A FRAMEWORK FOR INTERACTIVE EXPERIMENTATION WITH ADVERSARIAL MACHINE LEARNING RESEARCH

We present MLsploit, the first user-friendly, cloud-based system that enables researchers and practitioners to rapidly evaluate and compare state-of-the-art adversarial attacks and defenses for machine learning (ML) models. As recent advances in adversarial ML have revealed that many ML techniques are highly vulnerable to adversarial attacks, MLsploit meets the urgent need for practical tools that facilitate interactive security testing of ML models. MLsploit is jointly developed by researchers at Georgia Tech and Intel, and is open-source (https://mlsploit.github.io). Designed for extensibility, MLsploit accelerates the study and development of secure ML systems for safety-critical applications. In this work, we highlight the versatility of MLsploit in performing fast-paced experimentation with adversarial ML research that spans a diverse set of modalities, such as bypassing Android and Linux malware, or attacking and defending deep learning models for image classification. MLsploit enables users to perform experiments interactively in real time by varying different parameters of the experiments or using their own samples, and finally compare and evaluate the effects of such changes on the performance of the ML models through an intuitive user interface, all without writing any code.

8.1 INTRODUCTION

As AI is increasingly being used in safety-critical applications, it is crucial to extensively evaluate the security of these systems before deployment. However, current adversarial ML research efforts are often disparate and fragmented, with different works reporting evaluation results on different models, parameters and datasets, making it hard to directly compare these approaches, and thus weakening overall security analysis. Although many upcoming works [162, 164] attempt to address this issue by providing a unified software interface for performing attacks and defense on a variety of ML models, these solutions lack any interactivity and require the end-users to have deep knowledge of the software package itself in order to effectively use it. Often, users are only interested in knowing the high-level security risks of using off-the-shelf ML models without having to get into the minutiae of implementing the attack and defense infrastructure (an analogy can be drawn with ready-to-use anti-virus/anti-malware solutions). Several solutions [196, 2, 174] also exist that support interactive experimentation to explore vulnerabilities and understand the internals of ML models, but they are often limited to models that can only fit in the web browser memory, or don't allow for users to experiment with their own samples, or these solutions don't readily scale to a large number of users in real-time — such tools are primarily built for academic demonstration.

To address these challenges of making adversarial ML more accessible to researchers, practitioners and students, we developed **MLsploit** — the first user-friendly, cloud-based system that enables researchers and practitioners to rapidly evaluate and compare state-of-the-art adversarial attacks and defenses for ML models. MLsploit is a machine learning evaluation and fortification framework designed for education and research of adversarial ML. It is the first cloud-based tool that allows real-time interactive experimentation with adversarial ML research through an intuitive web-based interface. Designed for extensibility, MLsploit accelerates the study and development of secure ML systems.

Through MLsploit, we highlight the following contributions to the research community:

- **Interactive experimentation with adversarial ML research.** MLsploit enables ML researchers and practitioners to perform experiments on their own data interactively, without writing any code. This is done by combining plug-able components and services that implement various ML and security-related functions.
- **Enabling comparison of attacks and defenses across modalities.** MLsploit currently integrates adversarial ML research modules that include bypassing the detection of Android, Linux and Windows malware, and attacks and defenses on image classification models. Users can compare the effect of varying different parameters for these techniques seamlessly through MLsploit.
- **Facilitating easy integration of novel research.** The modularity of the MLsploit framework allows researchers to easily integrate their own research modules into the toolchain. We have also developed a python library[1] that abstracts away the orchestration of the input files and experiment parameters from the web backend, and provides a simple API to the researcher for accessing these artifacts in their own python module. Hence, it becomes very easy to write and integrate new research functions for MLsploit, and keep the tool updated with state-of-the-art techniques.

To enable more and more people to try out MLsploit, develop research modules to be integrated into MLsploit, and host MLsploit on their own infrastructure, the entire MLsploit framework is open-sourced on GitHub[2]. Here, we also provide a one-click deployment solution to the Microsoft Azure cloud, making MLsploit a limitlessly scalable application by allowing anyone to easily and independently host MLsploit.

[1]https://github.com/mlsploit/mlsploit-py

[2]https://github.com/mlsploit/mlsploit-system

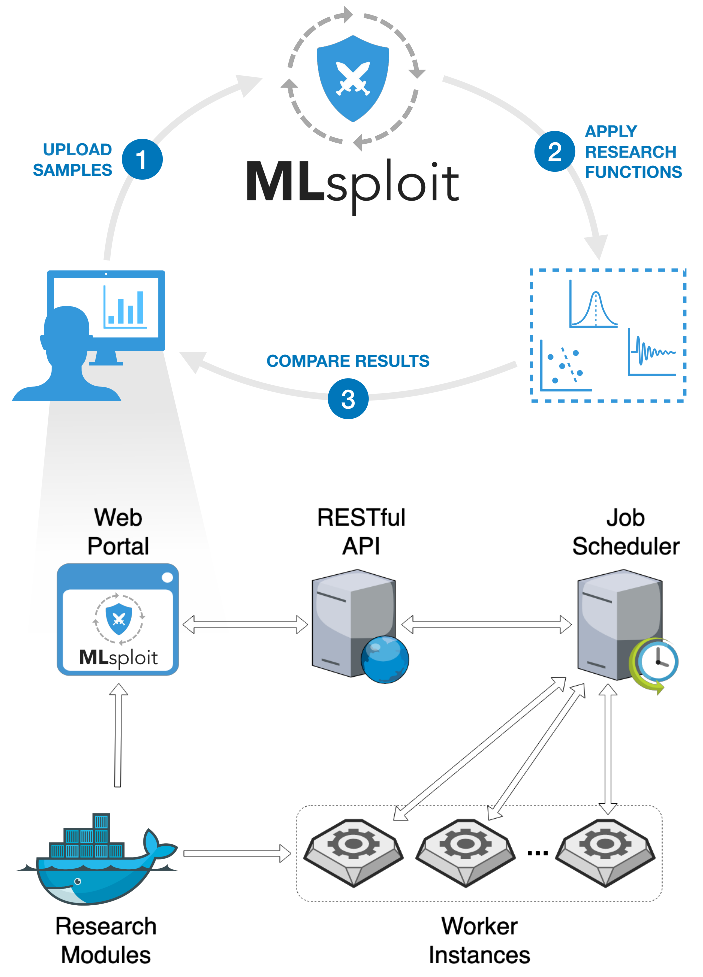


Figure 8.1: Top: Overview of MLsploit. Using MLsploit's interactive web interface, users can upload their own samples (e.g., APK files, images etc.), apply research functions to them (e.g., modifying APK files to bypass malware detection), and compare the results with previous experiments. Bottom: MLsploit system architecture. The front-end web portal gets data from a back-end REST API. The experiments are run on separate instances by the job scheduler. The research modules are independently developed and deployed using Docker containerization.

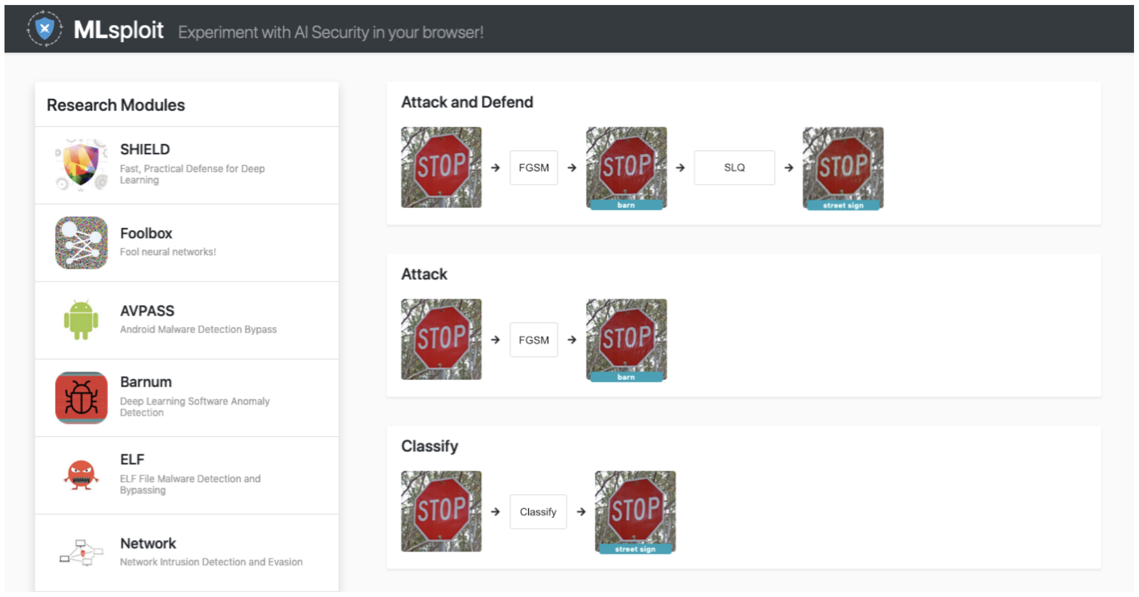

MLsploit Experiment with AI Security in your browser!
Research Modules
SHIELD
Fast, Practical Defense for Deep Learning
Foolbox
Fool neural networks!
AVPASS
Android Malware Detection Bypass
Barnum
Deep Learning Software Anomaly Detection
ELF
ELF File Malware Detection and Bypassing
Network
Network Intrusion Detection and Evasion
Attack and Defend
FGSM
barn
SLQ
street sign
Attack
FGSM
barn
Classify
Classify
street sign

Figure 8.2: The MLsploit UI. The Research Modules view (on the left) enlists different adversarial ML research modules that have been developed independently by researchers. The user can construct research pipelines using research functions provided by the research modules. The Research Pipelines view (on the right) shows 3 such research pipelines. The user uploads an image of a street sign and runs it through: (1) the "Classify" function (bottom) from the Foolbox module that classifies the image as a "street sign", (2) the "FGSM" attack function (middle) from the Foolbox module that leads to a misclassification of the image as a "barn", and (3) a combination (top) of the "FGSM" attack function from the Foolbox module and "SLQ" defense function from the SHIELD module that recovers the correct classification of "street sign".

8.2 SYSTEM DESIGN

The MLsploit system has a service-oriented architecture that includes a web portal for users to interact with, and a RESTful API to further automate experiments. An overview of the system design is shown in Figure 8.1. The research functions integrated in MLsploit can be thought of as modular components which can be combined in different arrangements to create the desired experiment pipeline using an intuitive interface. Since MLsploit leverages Docker containerization in the backend, each component can be implemented in any language and on any platform, and MLsploit glues everything together through well-defined APIs. This flexible component design is agnostic to the underlying implementation of the ML function, and hence allows quick development for module developers as well.

Figure 8.2 shows the MLsploit user interface along with an example usage scenario. MLsploit consists of (1) a Research Modules view that displays adversarial ML modules developed by researchers; and (2) a Research Pipelines view that shows all the research functions executed by the end-user. A demo of the MLsploit workflow can also be seen at https://youtu.be/U3rTnagWXFQ.

8.2.1 Adversarial ML Modules

MLsploit supports a host of adversarial ML modules spanning a diverse set of modalities:

- **AVPass**: leak and bypass Android malware detection [197].
- **Barnum**: defend models for document malware detection [198].
- **ELF**: bypass Linux malware detection with API perturbation.
- **PE**: create and attack models for Windows PE malware detection.
- **Foolbox & SHIELD**: attack and defend state-of-the-art image classifiers [199, 70].

8.2.2 One-click Cloud Deployment

From our extensive experience in hosting MLsploit to enable new researchers to learn about AI security through classes at Georgia Tech, we identified certain bottlenecks in the pedagogy involving adversarial ML research. The main bottleneck arises from the limited computational resources available to modest academic research labs and most teaching staff, restricting the number of students that can be concurrently served by a self-hosted MLsploit system. Moreover, since the MLsploit software is a highly sophisticated confluence of several sub-services (Figure 8.1), the one-time end-to-end infrastructure setup itself can be very daunting and a gargantuan task for researchers and practitioners who want to self-host MLsploit. To address these issues, we developed a solution for one-click end-to-end deployment of MLsploit to cloud infrastructure[3], including integration with the Microsoft Azure cloud. MLsploit can now, all at the click of a few buttons, connect and authenticate with the user's cloud provider, automatically orchestrate the computational infrastructure for running the system, and finally, setup and configure all MLsploit sub-services end-to-end so that it is ready to go. This will enable researchers and practitioners to swiftly setup their own MLsploit service and scale the infrastructure according to their own requirements, without having to execute a single line of setup code manually.

Additionally, cloud providers such as Microsoft Azure also provide education credits to educators, specifically targeted to allow students to get hands-on experience with their cloud services. Thus, a one-click end-to-end deployment solution for MLsploit integrated with such cloud services opens up new avenues for AI security pedagogy. Now, educators can leverage these education credits so that each student can setup their own MLsploit service on the cloud without any overhead, thus reducing a huge burden from the teaching staff to setup and maintain a monolithic computational infrastructure.

[3]https://github.com/mlsploit/mlsploit-system#quickstart-deploying-mlsploit-one-click-setup

8.3 IMPACT

MLsploit is already transforming AI security education at scale. Touted as the capstone project coming out of the $1.5 million collaboration between Intel Corporation and Georgia Tech through the Intel Science & Technology Center for Adversary-Resilient Security Analytics (ISTC-ARSA) program, MLsploit has actively been used at Georgia Tech in Dr. Wenke Lee's "CS 8803 O11: Information Security Lab - System and Network Defenses" class to teach hundreds of students cumulatively over the semesters on how to train and defend deep neural networks for malware detection using system call sequence data. MLsploit was also made available to $>$1,000 students of Dr. Polo Chau's "CSE 6242: Data and Visual Analytics" class across its campus and online sections at Georgia Tech in Spring 2021, enabling them to experiment with adversarial attacks and defenses for image classification models. For such potential in transforming AI security education and experimentation, MLsploit was also declared the winner[4] of the Institute for Information Security and Privacy's Cybersecurity Demo Day Finale (Commercialization Track) held at Georgia Tech in 2019.

8.4 CONCLUSION

We present MLsploit, a one-stop shop for AI security research and pedagogy. MLsploit is a machine learning evaluation and fortification framework designed for education and research of adversarial ML. It is the first open-source interactive system that allows in-depth security testing of AI models, and lowers barriers to entry for everyone — researchers, practitioners and students — to engage in adversarial ML research. MLsploit provides everyone a simple user interface for interactively studying new AI security techniques, and our hope is that researchers would continue to leverage our framework for disseminating their state-of-the-art adversarial ML research.

[4]https://www.cc.gatech.edu/news/621227/demo-day-shows-future-cybersecurity-machine-learning

Part IV

Conclusions

CHAPTER 9
CONCLUSION AND FUTURE DIRECTIONS

This dissertation takes a step forward in expanding people's understanding of AI vulnerabilities through extracting and intuitively visualizing an ML model's internal semantic representations. Our interpretation research reveals that adversarial attacks leverage semantically non-robust features to confuse a deep neural network. Based on this understanding, this dissertation proposes fundamentally unifying approaches such as multi-task learning, for making ML models more resilient to adversarial attacks across AI tasks and input modalities. Finally, this dissertation develops a scalable framework for enabling everyone to interactively experiment with AI security research in order to stress-test the AI applications used in everyday life.

9.1 RESEARCH CONTRIBUTIONS

This thesis makes research contributions through multiple major fronts.

- **Novel, principled, interpretable approaches for semantic class inference.**
  Our research contributes novel, theoretically principled approaches to extract interpretable semantic prototypes for data programming (Chapter 2). Our weakly-supervised GOGGLES framework is **only 7% away** from a fully-supervised baseline.

- **Novel visualization technique for deciphering attacks.**
  Our work significantly expands the understanding of model vulnerabilities, which were earlier considered as "black boxes", by proposing attribution graph mining for model activation (Chapter 3). Our novel visualization technique **helps in identifying the vulnerable neurons** in a DNN that are non-performant and lead to misclassifications under attack.

- **Faster, generalizable, practical defenses.**
  This dissertation identifies the need for AI defenses with low computational overhead and develops fast, robust defenses based on input compression that generalize across input modalities. Our SHIELD defense (Chapter 4) is up to **22x faster** than other pre-processing defenses, and the ADAGIO defense (Chapter 7) **removes all non-adaptive targeted adversarial attacks**.

- **Fundamentally unifying approach for training robust models across AI tasks.** Our in-depth study of **Multi-Task Learning** (Chapters 5 and 6) establishes it as a fundamentally unifying deep learning approach that induces ML models to learn robust features that are **resistant to adversarial attacks** across AI tasks and input modalities.

- **First scalable system for interactive experimentation with AI security research.** This thesis introduces MLsploit (Chapter 8), the **first open-source interactive system** that allows in-depth security testing of AI models, and lowers barriers to entry for everyone — researchers, practitioners and students — to engage in adversarial ML research. MLsploit offers modules of state-of-the-art attacks and defenses.

9.2 IMPACT

Beyond the contributions to the research community, our work has also made a significant impact to broader society and industry.

- Early-stage startups have picked up our GOGGLES framework for it's efficacy in generating training data with low overhead, and even extended the affinity functions to include natural language processing tasks.

- The SHIELD defense framework won the **Audience Appreciation Award, runner-up** at KDD 2018, being in the top 3 among 107 accepted papers, from 983 submissions. SHIELD was also tech-transferred to **Intel Labs** and multiple business units within Intel for integration and testing. The SHIELD tech transfer to Intel also incited a positive shift at the highest levels within the company towards developing hardware accelerators for AI defense.

- Research ideas presented in this dissertation formed a core component of an awarded multi-million dollar **DARPA GARD** (Guaranteeing AI Robustness against Deception) grant for developing next-generation defenses against physically realizable adversarial attacks.

- MLsploit is already transforming adversarial ML education. It is currently being integrated into multiple security and data analytics courses at **Georgia Tech**, becoming available to thousands of students. MLsploit is already being used by graduate students in Dr. Wenke Lee's **"CS 8803 O11: Information Security Lab - System and Network Defenses"** class and was also made available to students of Dr. Polo Chau's **"CSE 6242: Data and Visual Analytics"** class at Georgia Tech. MLsploit won the Institute for Information Security and Privacy's **Cybersecurity Demo Day Finale** held at Georgia Tech in 2019. MLsploit modules also formed the foundation for the DARPA GARD research proposal.

9.3 FUTURE DIRECTIONS

While this dissertation expands people's understanding of AI vulnerabilities and how they can be mitigated, it also unlocks new research directions for the data programming, visualization and adversarial ML research communities.

Affinity coding for diverse range of AI tasks.
In Chapter 2, we propose the affinity coding paradigm and develop the GOGGLES framework for the task of image labeling. The affinity coding paradigm can be extended to cover more AI tasks such as accent classification from the speech domain or sentiment analysis from the natural language processing (NLP) domain. Several strong modeling techniques exist in these respective domains that can be leveraged for forming interpretable prototypes. Doing so will extend our understanding of how different input modalities are semantically represented in the intermediate feature space of deep neural networks. This would allow us to conceive of attack mitigation techniques that can leverage semantically robust features.

Comparing activation pathways across other model architectures.
In Chapter 3, the BLUFF tool exposes vulnerable activation pathways from the INCEPTIONV1 model. A natural next step is generating and comparing such activation pathways across different model architectures. Recent work [65] has provided evidence that different types of neural network architectures learn similar concepts (e.g., curve detectors and high-low frequency detectors) for vision tasks. Our approach using activation pathways could be adopted for comparing what features multiple networks have learned and how their vulnerability could be unified or differentiated.

Mining activation pathways for motifs.
As BLUFF extracts activation pathways, we can leverage data mining and graph analysis methods to aggregate such pathways for different classes and find the most common motifs across all pathways. Extracting these smaller subpath motifs could give further insight into how neural networks arrange and prioritize hierarchical semantic concepts.

Enforcing semantic constraints with Multi-Task Learning.
In Chapters 5 and 6, we study multi-task learning (MTL) to perform adversarially robust inference for person tracking and automatic speech recognition respectively. The MTL

models are supplemented with human keypoint detection and accent classification tasks respectively during training. However, these secondary task heads are discarded during inference. The computational cost for an adaptive attacker can be significantly increased by performing joint inference while enforcing semantic constraints. For example, in person tracking, the keypoint detection head can be jointly used to ensure that all detected keypoints are indeed within the predicted bounding box. Performing such explicitly constrained joint inference would force an adaptive attacker to consider gradients from the keypoint head, and additionally ensure that the adversarial perturbation satisfies the semantic constraints.

Multi-task auditing for single-task inference.
Following in the footsteps of the previous proposed research direction, the secondary task heads obtained from performing multi-task learning can also be used for post-hoc auditing of suspected model failures. As these secondary task heads are not exposed to the inference pipeline, an adaptive attacker may not optimize the adversarial perturbations to produce a reasonable output for these ad-hoc tasks. As the secondary task heads were trained jointly, they are compatible with the shared backbone network, and can be used to determine if commonsense semantic constraints are not preserved while privately auditing the model failures for certain inputs at a later time. For example, in the case of person tracking, if we are given a suspected adversarial video, we can use the jointly trained keypoint detection head separately using visualization tools [200] to determine if the video was attacked or not. This may not prevent adversarial attacks in real-time, but can be used to investigate model failures at a later time.

Interactive neuron editing for model robustness.
In Chapter 8, we present an interactive experimentation framework, MLsploit, that allows security testing of AI models. At the same time, our BLUFF tool (Chapter 3) currently supports visualizing neurons that are highly excited or inhibited by an adversarial attack. Combining these two technologies could lead to a new defense paradigm. Using our interactive experimentation framework, we can enable real-time neuron editing. A user could actively identify vulnerable neurons using BLUFF and interactively remove them from the network to observe the effect on the resulting pathway and prediction in real-time. Masking the activations of a particular neuron could potentially prevent targeted attacks to propagate deeper into the network. For example, a user could preemptively edit a model to enhance its robustness by deleting neurons that only feed into exploited pathways, preventing adversarially activated neurons from affecting subsequent layers.

9.4 CONCLUSION

Through my extensive research work in the domain of adversarial ML and countless interactions with students, practitioners and new researchers in AI, I have discovered that mass adoption of AI in safety-critical applications would be possible only when people trust in the security of ML models through a broad understanding of the underlying vulnerabilities. Thus, my continued research mission is to enable everyone to understand and fortify AI security. Just like AI has real-world applications, AI security needs real-world solutions. This dissertation takes an initial step towards addressing this challenge by developing practical solutions for AI security that everyone can understand and use.

REFERENCES

[1] N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu, “GOGGLES: Automatic image labeling with affinity coding,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1717–1732.

[2] N. Das*, H. Park*, Z. J. Wang, F. Hohman, R. Firstman, E. Rogers, and D. H. P. Chau, “Bluff: Interactively deciphering adversarial attacks on deep neural networks,” in *2020 IEEE Visualization Conference (VIS)*, 2020, pp. 271–275.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[4] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 843–852.

[5] A. P. Davis, T. C. Wiegers, P. M. Roberts, B. L. King, J. M. Lay, K. Lennon-Hopkins, D. Sciaky, R. Johnson, H. Keating, N. Greene, *et al.*, “A CTD–Pfizer collaboration: Manual curation of 88 000 scientific articles text mined for drug–disease and drug–phenotype interactions,” *Database-the Magazine of Electronic Database Reviews*, vol. 2013, 2013.

[6] X. J. Zhu, “Semi-supervised learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[7] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[8] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[9] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[10] Y. Wang and Q. Yao, “Few-shot learning: A survey,” *CoRR*, vol. abs/1904.05046, 2019, http://arxiv.org/abs/1904.05046. arXiv: 1904.05046.

[11] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, “A closer look at few-shot classification,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[12] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, “Data programming: Creating large training sets, quickly,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3567–3575.

[13] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: Rapid training data creation with weak supervision,” *Proceedings of the VLDB Endowment*, vol. 11, no. 3, pp. 269–282, 2017.

[14] P. Varma and C. Ré, “Snuba: Automating weak supervision to label training data,” *Proceedings of the VLDB Endowment*, vol. 12, no. 3, pp. 223–236, 2018.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014. arXiv: 1409.1556.

[16] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision*, Springer, 2014, pp. 818–833.

[17] A. P. Dempster, “Covariance selection,” *Biometrics. Journal of the International Biometric Society*, vol. 28, no. 1, pp. 157–175, Mar. 1972.

[18] S. Velasco-Forero, M. Chen, A. Goh, and S. K. Pang, “Comparative analysis of covariance matrix estimation for anomaly detection in hyperspectral images,” *IEEE J. Sel. Top. Signal Process.*, vol. 9, no. 6, pp. 1061–1073, Sep. 2015.

[19] C. M. Bishop, *Pattern Recognition and Machine Learning*. springer, 2006.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[21] M. Peikari, S. Salama, S. Nofech-Mozes, and A. L. Martel, “A cluster-then-label semi-supervised learning approach for pathology image classification,” *Scientific reports*, vol. 8, no. 1, p. 7193, 2018.

[22] R. Jonker and A. Volgenant, “A shortest augmenting path algorithm for dense and sparse linear assignment problems,” *Computing. Archives for Scientific Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[23] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The caltech-ucsd birds-200-2011 dataset,” 2011.

[24] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, 2012.

[25] W. Louhichi, "Automated surface finish inspection using convolutional neural networks," Ph.D. dissertation, Georgia Institute of Technology, 2019.

[26] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R. K. Singh, S. Antani, *et al.*, "Automatic tuberculosis screening using chest radiographs," *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 233–245, 2013.

[27] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[28] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning*, 2014, pp. 647–655.

[29] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1717–1724.

[30] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[31] D. Weinland, M. Özuysal, and P. Fua, "Making action recognition robust to occlusions and viewpoint changes," in *European Conference on Computer Vision*, Springer, 2010, pp. 635–648.

[32] X. Yang, C. Zhang, and Y. Tian, "Recognizing actions using depth motion maps-based histograms of oriented gradients," in *Proceedings of the 20th ACM International Conference on Multimedia*, ACM, 2012, pp. 1057–1060.

[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005.

[34] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[35] T. Akilan, Q. J. Wu, A. Safaei, and W. Jiang, “A late fusion approach for harnessing multi-CNN model high-level features,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2017, pp. 566–571.

[36] B. Settles, “Active learning: Synthesis lectures on artificial intelligence and machine learning,” *Long Island, NY: Morgan & Clay Pool*, 2012.

[37] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava, “Fusing data with correlations,” in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ACM, 2014, pp. 433–444.

[38] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré, “Slimfast: Guaranteed results for data fusion and source reliability,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, ACM, 2017, pp. 1399–1414.

[39] A. Das Sarma, A. Parameswaran, and J. Widom, “Towards globally optimal crowdsourcing quality management: The uniform worker setting,” in *Proceedings of the 2016 International Conference on Management of Data*, ACM, 2016, pp. 47–62.

[40] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, “Holoclean: Holistic data repairs with probabilistic inference,” *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1190–1201, 2017.

[41] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, “A guide to deep learning in healthcare,” *Nature medicine*, vol. 25, no. 1, p. 24, 2019.

[42] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *Journal of Field Robotics*, 2019.

[43] G. Guo and N. Zhang, “A survey on deep learning based face recognition,” *Computer Vision and Image Understanding*, vol. 189, p. 102 805, 2019.

[44] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, “Speech recognition using deep neural networks: A systematic review,” *IEEE access : practical innovations, open solutions*, vol. 7, pp. 19 143–19 165, 2019.

[45] J. Heaton, N. G. Polson, and J. H. Witte, “Deep learning in finance,” *arXiv preprint arXiv:1602.06561*, 2016. arXiv: 1602.06561.

[46] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[47] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on Faster R-CNN object detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018, pp. 52–68.

[48] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[49] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016. arXiv: 1607.02533.

[50] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," *arXiv preprint arXiv:1903.10346*, 2019. arXiv: 1903.10346.

[51] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[52] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: Attribute-steered detection of adversarial samples," in *Advances in Neural Information Processing Systems*, 2018, pp. 7717–7728.

[53] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu, "Analyzing the noise robustness of deep neural networks," in *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2018, pp. 60–71.

[54] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[56] 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.

[57] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[58] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training," in *International Conference on Machine Learning*, 2019, pp. 6586–6595.

[59] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Apr. 2018.

[60] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 8, pp. 2674–2693, 2018.

[61] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[62] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[63] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas," *Distill*, vol. 4, no. 3, e15, 2019.

[64] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[65] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits," *Distill*, vol. 5, no. 3, e00024–001, 2020.

[66] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," *ICML*, 2018.

[67] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6541–6549.

[68] R. Fong and A. Vedaldi, "Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8730–8738.

[69] F. Hohman, H. Park, C. Robinson, and D. H. Chau, "Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations," *IEEE VIS*, 2019.

[70] 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," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018, pp. 196–204.

[71] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 582–597.

[72] K. Cao, M. Liu, H. Su, W. Jing, and S. Liu, “Analyzing the noise robustness of deep neural networks,” *arXiv preprint arXiv:2001.09395*, 2020. arXiv: 2001.09395.

[73] A. P. Norton and Y. Qi, “Adversarial-Playground: A visualization suite showing how adversarial examples fool deep learning,” in *Visualization for Cyber Security (VizSec), 2017 IEEE Symposium On*, IEEE, 2017, pp. 1–4.

[74] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg, “Gan lab: Understanding complex deep generative models using interactive visual experimentation,” *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 1–11, 2018.

[75] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, “The building blocks of interpretability,” *Distill*, vol. 3, no. 3, e10, 2018.

[76] N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, and L. Schubert, “Thread: Circuits,” *Distill*, 2020, https://distill.pub/2020/circuits.

[77] S. Sietzen and M. Waldner, “Interactive feature visualization in the browser,” *2nd Workshop on Visualization for AI Explainability at IEEE VIS*, 2019, http://visxai.stefansietzen.at/.

[78] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, “Weighted graph comparison techniques for brain connectivity analysis,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 483–492.

[79] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[80] 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.

[81] 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 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’17, Abu Dhabi, United Arab Emirates, 2017, pp. 506–519.

[82] 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 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 1528–1540.

[83] A. Athalye and I. Sutskever, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017. arXiv: 1707.07397.

[84] 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. arXiv: 1704.02654.

[85] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *ICLR*, 2017.

[86] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," *International Conference on Learning Representations*, 2018.

[87] 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," *arXiv preprint arXiv:1707.08945*, 2017. arXiv: 1707.08945.

[88] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, D. Song, T. Kohno, A. Rahmati, A. Prakash, and F. Tramer, "Note on attacking object detectors with adversarial stickers," *arXiv preprint arXiv:1712.08062*, 2017. arXiv: 1712.08062.

[89] S. M. Moosavi Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, 2017.

[90] 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. arXiv: 1606.04435.

[91] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," *arXiv preprint arXiv:1702.05983*, 2017. arXiv: 1702.05983.

[92] 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.

[93] 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. arXiv: 1703.06748.

[94] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017. arXiv: 1702.02284.

[95] R. Shin and D. Song, "JPEG-resistant adversarial images," *NIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

[96] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression," *arXiv:1705.02900*, 2017. arXiv: 1705.02900.

[97] 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.

[98] 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.

[99] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," *arXiv preprint arXiv:1511.06292*, 2015. arXiv: 1511.06292.

[100] 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. arXiv: 1709.03423.

[101] 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.

[102] 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.

[103] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017. arXiv: 1703.00410.

[104] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014. arXiv: 1412.5068.

[105] D. Krotov and J. J. Hopfield, "Dense associative memory is robust to adversarial inputs," *arXiv preprint arXiv:1701.00939*, 2017. arXiv: 1701.00939.

[106] 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. arXiv: 1712.00699.

[107] 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. arXiv: 1608.00853.

[108] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” *arXiv preprint arXiv:1802.00420*, 2018. arXiv: 1802.00420.

[109] A. Bhattacharyya, M. Fritz, and B. Schiele, “Long-term on-board prediction of people in traffic scenes under uncertainty,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4194–4202.

[110] T. Yagi, K. Mangalam, R. Yonetani, and Y. Sato, “Future person localization in first-person videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7593–7602.

[111] Y. Yao, M. Xu, Y. Wang, D. J. Crandall, and E. M. Atkins, “Unsupervised traffic accident detection in first-person videos,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 273–280.

[112] R. Bohush and I. Zakharava, “Robust person tracking algorithm based on convolutional neural network for indoor video surveillance systems,” in *International Conference on Pattern Recognition and Information Processing*, Springer, 2019, pp. 289–300.

[113] S. Ye, R. Bohush, H. Chen, I. Y. Zakharava, and S. Ablameyko, “Person tracking and reidentification for multicamera indoor video surveillance systems,” *Pattern Recognition and Image Analysis*, vol. 30, no. 4, pp. 827–837, 2020.

[114] I. Ahmed and G. Jeon, “A real-time person tracking system based on SiamMask network for intelligent video surveillance,” *Journal of Real-Time Image Processing*, vol. 18, no. 5, pp. 1803–1814, 2021.

[115] L. Bridgeman, M. Volino, J.-Y. Guillemaut, and A. Hilton, “Multi-person 3d pose estimation and tracking in sports,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[116] L. Kong, D. Huang, and Y. Wang, “Long-term action dependence-based hierarchical deep association for multi-athlete tracking in sports videos,” *IEEE Transactions on Image Processing*, vol. 29, pp. 7957–7969, 2020.

[117] Q. Liang, W. Wu, Y. Yang, R. Zhang, Y. Peng, and M. Xu, "Multi-player tracking for multi-view sports videos with improved k-shortest path algorithm," *Applied Sciences*, vol. 10, no. 3, p. 864, 2020.

[118] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[119] R. R. Wiyatno and A. Xu, "Physical adversarial textures that fool visual object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4822–4831.

[120] Y. J. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. W. Wei, "Fooling detection alone is not enough: Adversarial attack against multiple object tracking," in *International Conference on Learning Representations (ICLR'20)*, 2020.

[121] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial T-shirt! Evading person detectors in a physical world," in *European Conference on Computer Vision*, Springer, 2020, pp. 665–681.

[122] X. Chen, C. Fu, F. Zheng, Y. Zhao, H. Li, P. Luo, and G.-J. Qi, "A unified multi-scenario attacking network for visual object tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 1097–1104.

[123] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *International Conference on Machine Learning*, PMLR, 2017, pp. 854–863.

[124] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," *Advances in neural information processing systems*, vol. 31, 2018.

[125] C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz, "First-order adversarial vulnerability of neural networks and input dimension," in *International Conference on Machine Learning*, PMLR, 2019, pp. 5809–5817.

[126] S. Jia, C. Ma, Y. Song, and X. Yang, "Robust tracking against adversarial attacks," in *European Conference on Computer Vision*, Springer, 2020, pp. 69–84.

[127] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[128] C. Mao, A. Gupta, V. Nitin, B. Ray, S. Song, J. Yang, and C. Vondrick, "Multitask learning strengthens adversarial robustness," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 12347, Springer, 2020, pp. 158–174.

[129] S. Ghamizi, M. Cordy, M. Papadakis, and Y. L. Traon, "Adversarial robustness in multi-task learning: Promises and illusions," *arXiv preprint arXiv:2110.15053*, 2021. arXiv: 2110.15053.

[130] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision*, Springer, 2016, pp. 850–865.

[131] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1763–1771.

[132] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 101–117.

[133] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[134] Y. Yu, Y. Xiong, W. Huang, and M. R. Scott, "Deformable siamese attention networks for visual object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6728–6737.

[135] B. Shuai, A. Berneshawi, X. Li, D. Modolo, and J. Tighe, "Siammot: Siamese multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 372–12 382.

[136] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[137] Y. Luo, D. Tao, B. Geng, C. Xu, and S. J. Maybank, "Manifold regularized multitask learning for semi-supervised multilabel image classification," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 523–536, 2013.

[138] L. Liebel and M. Körner, "Multidepth: Single-image depth estimation via multi-task regression and classification," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1440–1447.

[139] M. Kocabas, S. Karagoz, and E. Akbas, "Multiposenet: Fast multi-person pose estimation using pose residual network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 417–433.

[140] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, *et al.*, "The seventh visual object tracking vot2019 challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[141] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav, *et al.*, "The eighth visual object tracking VOT2020 challenge results," in *European Conference on Computer Vision*, Springer, 2020, pp. 547–601.

[142] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, H. J. Chang, M. Danelljan, L. Cehovin, A. Lukežič, *et al.*, "The ninth visual object tracking vot2021 challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2711–2738.

[143] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via structured multi-task sparse learning," *International journal of computer vision*, vol. 101, no. 2, pp. 367–383, 2013.

[144] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4335–4343.

[145] ——, "Learning multi-task correlation particle filters for visual tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 2, pp. 365–378, 2018.

[146] X. Wang, X. Shu, Z. Zhang, B. Jiang, Y. Wang, Y. Tian, and F. Wu, "Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 763–13 773.

[147] S. Yan, J. Yang, J. Käpylä, F. Zheng, A. Leonardis, and J.-K. Kämäräinen, "Depthtrack: Unveiling the power of rgbd tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 725–10 733.

[148] X.-F. Zhu, T. Xu, and X.-J. Wu, "Visual object tracking on multi-modal RGB-D videos: A review," *arXiv preprint arXiv:2201.09207*, 2022. arXiv: 2201.09207.

[149] X. Zhang, P. Ye, S. Peng, J. Liu, K. Gong, and G. Xiao, “SiamFT: An RGB-infrared fusion tracking method via fully convolutional siamese networks,” *IEEE access : practical innovations, open solutions*, vol. 7, pp. 122 122–122 133, 2019.

[150] X. Zhang, P. Ye, S. Peng, J. Liu, and G. Xiao, “DSiamMFT: An RGB-T fusion tracking method via dynamic Siamese networks using multi-layer feature fusion,” *Signal Processing: Image Communication*, vol. 84, p. 115 756, 2020.

[151] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9185–9193.

[152] S. Liang, X. Wei, S. Yao, and X. Cao, “Efficient adversarial attacks for visual object tracking,” in *European Conference on Computer Vision*, Springer, 2020, pp. 34–50.

[153] S. Jia, Y. Song, C. Ma, and X. Yang, “IoU attack: Towards temporally coherent black-box adversarial attack for visual object tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6709–6718.

[154] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *European Conference on Computer Vision*, Springer, 2016, pp. 749–765.

[155] Z. Yan, Y. Guo, and C. Zhang, “Deep defense: Training dnns with improved adversarial robustness,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[156] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, “Adversarial attacks and defenses in images, graphs and text: A review,” *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[157] P. Liu, X. Qiu, and X. Huang, “Adversarial multi-task learning for text classification,” *arXiv preprint arXiv:1704.05742*, 2017. arXiv: 1704.05742.

[158] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.

[159] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, “Adversarial examples are not bugs, they are features,” *Advances in neural information processing systems*, vol. 32, 2019.

[160] 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 *Computer Vision –*

ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.

[161] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5374–5383.

[162] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018, https://arxiv.org/pdf/1807.01069.

[163] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[164] Two Six Technologies, *ARMORY*, https://github.com/twosixlabs/armory.

[165] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[166] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017. arXiv: 1707.05373.

[167] D. Iter, J. Huang, and M. Jermann, "Generating adversarial examples for speech recognition," *Stanford Technical Report*, 2017.

[168] M. Alzantot, B. Balaji, and M. Srivastava, "Did you hear that? adversarial examples against automatic speech recognition," *arXiv preprint arXiv:1801.00554*, 2018. arXiv: 1801.00554.

[169] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2018, pp. 1–7.

[170] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "CommanderSong: A systematic approach for practical adversarial voice recognition," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 49–64.

[171] R. Taori, A. Kamsetty, B. Chu, and N. Vemuri, "Targeted adversarial examples for black box audio systems," in *2019 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2019, pp. 15–20.

[172] Q. Wang, B. Zheng, Q. Li, C. Shen, and Z. Ba, "Towards query-efficient adversarial attacks against automatic speech recognition systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 896–908, 2020.

[173] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[174] N. Das, M. Shanbhogue, S.-T. Chen, L. Chen, M. E. Kounavis, and D. H. Chau, "Adagio: Interactive experimentation with adversarial attack and defense for audio," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2018, pp. 677–681.

[175] S. Hussain, P. Neekhara, S. Dubnov, J. McAuley, and F. Koushanfar, "WaveGuard: Understanding and mitigating audio adversarial examples," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2273–2290.

[176] A. Sreeram, N. Mehlman, R. Peri, D. Knox, and S. Narayanan, "Perceptual-based deep-learning denoiser as a defense against adversarial attacks on ASR systems," *arXiv preprint arXiv:2107.05222*, 2021. arXiv: 2107.05222.

[177] P. Żelasko, S. Joshi, Y. Shao, J. Villalba, J. Trmal, N. Dehak, and S. Khudanpur, "Adversarial attacks and defenses for speech recognition systems," *arXiv preprint arXiv:2103.17122*, 2021. arXiv: 2103.17122.

[178] C.-H. Yang, J. Qi, P.-Y. Chen, X. Ma, and C.-H. Lee, "Characterizing speech adversarial examples using self-attention u-net enhancement," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 3107–3111.

[179] A. Jain, M. Upreti, and P. Jyothi, "Improved accented speech recognition using accent embeddings and multi-task learning.," in *Interspeech*, 2018, pp. 2454–2458.

[180] Y. Adi, N. Zeghidour, R. Collobert, N. Usunier, V. Liptchinsky, and G. Synnaeve, "To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3742–3746.

[181] N. Das, S. Bodapati, M. Sunkara, S. Srinivasan, and D. H. Chau, "Best of both worlds: Robust accented speech recognition with adversarial transfer learning," in *Proc. Interspeech 2021*, 2021, pp. 1314–1318.

[182] L. Li, Y. Kang, Y. Shi, L. Kürzinger, T. Watzel, and G. Rigoll, "Adversarial joint training with self-attention mechanism for robust end-to-end speech recognition,"

EURASIP Journal on Audio, Speech, and Music Processing, vol. 2021, no. 1, pp. 1–16, 2021.

[183] Y. Tang, J. Pino, C. Wang, X. Ma, and D. Genzel, "A general multi-task learning framework to leverage text data for speech to text tasks," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 6209–6213.

[184] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 369–376.

[185] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. arXiv: 1409.0473.

[186] Mozilla, *Common Voice*, https://commonvoice.mozilla.org.

[187] N. Kamo, *ESPnet2 pretrained model*, https://doi.org/10.5281/zenodo.4604011, 2021.

[188] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020. arXiv: 2005.08100.

[189] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, 2018, pp. 2207–2211.

[190] S. Sun, C.-F. Yeh, M.-Y. Hwang, M. Ostendorf, and L. Xie, "Domain adversarial training for accented speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4854–4858.

[191] H. Liu, S. Jin, and C. Zhang, "Connectionist temporal classification with maximum entropy regularization," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[192] A. Zeyer, R. Schlüter, and H. Ney, "Why does CTC result in peaky behavior?" *arXiv preprint arXiv:2105.14849*, 2021. arXiv: 2105.14849.

[193] N. Das, S. Li, C. Jeon, J. Jung*, S.-T. Chen*, C. Yagemann*, E. Downing*, H. Park, E. Yang, L. Chen, *et al.*, "MLsploit: A framework for interactive experimentation with adversarial machine learning research," in *25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (Project Showcase)*, ACM, 2019.

[194] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, “Deep speech: Scaling up end-to-end speech recognition,” *arXiv:1412.5567*, 2014. arXiv: 1412.5567.

[195] Mozilla, *DeepSpeech*, https://github.com/mozilla/DeepSpeech.

[196] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Chau, “CNN explainer: Learning convolutional neural networks with interactive visualization,” *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[197] J. Jung, C. Jeon, M. Wolotsky, I. Yun, and T. Kim, “AVPASS: Leaking and bypassing antivirus detection model automatically,” *Black Hat USA Briefings (Black Hat USA), Las Vegas, NV*, 2017.

[198] C. Yagemann, S. Sultana, L. Chen, and W. Lee, “Barnum: Detecting document malware via control flow anomalies in hardware traces,” in *Proceedings of the 22nd Information Security Conference*, 2019.

[199] J. Rauber, W. Brendel, and M. Bethge, “Foolbox: A Python toolbox to benchmark the robustness of machine learning models,” in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, http://arxiv.org/abs/1707.04131, 2017.

[200] H. Park, Z. J. Wang, N. Das, A. S. Paul, P. Perumalla, Z. Zhou, and D. H. Chau, “SkeletonVis: Interactive visualization for understanding adversarial attacks on human action recognition models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 16 094–16 096.