Defense Against the Adversarial Arts:
Applying Green Team Evaluations to Harden Machine Learning Algorithms from Adversarial Attacks

by

Josh Kalin

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2022

Keywords: Adversarial Machine Learning, Robustness, Classification

Approved by

Gerry Dozier, Chair, Charles D. McCrary Eminent Professor of Computer Science and Software Engineering
Bo Liu, Associate Professor of Electrical and Computer Engineering
Ahn Nguyen, Associate Professor of Electrical and Computer Engineering
Cheryl Seals, Charles W. Barkley Endowed Professor of Computer Science and Software Engineering

Abstract

Machine Learning permeates all facets of our lives today. Given these models are trusted to make important decisions in our lives, how susceptible are those models to attacks? Adversarial machine learning is the study of vulnerabilities in machine learning models. Common vulnerabilities exist in popular models because the dataset, model weights, and inference code are all in the public domain. Adversarial Actors will probe and discover adversarial weaknesses within these models due to the open nature of the repositories. Once these adversarial gaps are discovered, the literature will typically stop and raise flags about the models. In this work, the focus is to find, evaluate, and fix known vulnerabilities in production machine learning model. This work will present six research concepts on evaluating model vulnerabilities, structured approaches to fixing them, and team constructs that can be used in production machine learning systems to prevent current and new adversarial attack vectors.

The objective of this work is creating a repeatable evaluation system for production machine learning models that focuses on identifying the underlying model vulnerabilities, benchmarking the attack surface, and suggesting solutions to reduce the efficacy of these adversarial issues. Each research concept progressively builds on the previous one. An early result of this research demonstrated the vulnerabilities inherent in semantic classifiers and showed simple mitigation strategies that can be used with these deployments. As our research progressed, we discovered weaknesses in multiple models and multiple disciplines existed without a repeatable methodology for fixing them. Green Team machine learning, one our the core concepts, is an answer to this problem and will be discussed.

Each research concept in this proposal shows viable methodologies for securing a machine learning model in a production environment. Deep image and text classifiers are trained

as example deployments. One final note: each attack surface demonstrated in this work is detectable and preventable.

Acknowledgments

One of the first classes in my PhD program was with Dr. Gerry Dozier on Adversarial Attacks on Machine Learning Models. I want to personally thank Dr. Dozier for inspiring my line of research over the last three years and for eagerly agreeing to be my advisor. His mentorship and encouragement has helped me immensely and I am forever grateful.

I started my PhD journey with Boeing and then transitioned to a small company called PeopleTec. During the most formative years of my research experience, I was incredibly lucky to be mentored by Dr. David Noever. With over 10,000 citations, he has written and authored many popular publications. As a Technical Fellow at PeopleTec, I was fortunate to be allowed to continue my research while contributing to meaningful problems for the commercial and government sectors. I am thankful for PeopleTec and David's guidance throughout my journey to this point.

I started my technical career fourteen years ago at a company named FastMetrix. I interned for a gentleman by the name of Don Fronek. A no frills PhD in Electrical Engineering, Don always had an amazing liveliness to his every day work and would always share stories about teaching in academia and coming back to industry. I had been fortunate to have his mentorship over the last thirteen years. He continued to provide me career and life guidance up until the end. After losing Don last summer, the best thing I can do to honor him is to complete this degree and defend this dissertation.

Throughout eight years of graduate school and three graduate degrees, my family has always been wildly supportive of my academic dreams. I have had many friends who joined, graduated, and left school in favor of jobs. I, however, have continued to go to graduate school and worked for eight years with the support of my family and I appreciate their flexibility and understanding when I had classes, homework, tests, and projects over the

years. Thanks especially to my Pop for always checking in on my academic progress - it always motivated me to continue to reach higher.

Finally, I have to thank my wife Lara. We have been together for almost nineteen years and she has supported every crazy dream I have had over the years. Without fail, she pushes, motivates, and supports every failure and success I have had in this journey. It is hard to express my gratitude to her in any other way than to successfully graduate with this terminal degree and not enter any new graduate programs. We are ready for the next chapter and I am excited to share it with her.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As machine learning permeates all parts of digital life, there is an increasing need to secure each deployed machine learning models in production. There are public vulnerabilities in modern machine learning based systems and these vulnerabilities can be exploited [2]. These vulnerabilities are typically exercised and documented in a subfield of machine learning called Adversarial Machine Learning [3]. Adversarial Machine Learning focuses on understanding the underlying attack surface for a model architecture and ways to protect those models from attacks discovered [4]. There is a gap in this field though: papers focus either on attacking the model or protecting the model [5]. This work presented in this chapter will explore end to end systems for benchmarking, evaluating, and securing machine learning models.

Model Security [6] is a popular topic today in modern literature. Machine Learning models are under constant attack and require protections to be developed to protect them [7]. For the research introduced in this chapter, two popular modes of attacks on models are selected: data poisoning [8] and adversarial examples [3]. Each of these methods can affect the machine learning pipeline from training to deployment [9]. This work covers the basics of each attack and provides practical protections.

Data Poisoning [8] refers to an adversarial technique that allows an attacker to change the decision boundaries of a machine learning model by adding targeted examples into the dataset. By adding specific examples to the dataset, an attacker can reduce the efficacy of a model to detect a particular class or they can make certain classes easier to detect [10]. In this attack mode, an attacker is focused on changing the model performance for their benefit. For image classifiers, data poisoning can be harder to detect without structured techniques

like monitoring data drift within a production environment [11]. Industry datasets [12] are typically too large to allow for manual inspection and have led to increasing concerns that the Data Poisoning is a top concern [13].

Adversarial Examples [3] are inputs that have been altered to provide a targeted outcome for an attacker. Like dataset poisoning, the goal is to provide an attacker a favorable outcome for a particular event - in a classifier, an attacker can avoid detection for a particular class or change the detected class to another target. Adversarial examples are heavily explored in the research community and have inspired development libraries for attacking models in the image and text space [14]. There are two convenience libraries used in this work: FoolBox [15] and TextFooler [16]. Each of these libraries provides access to state of the art adversarial examples for attacking models and changing classification outputs.

Current model robustness solutions will only provide protections for a certain type of machine learning model like an image classifier [17]. There should be a focus instead on addressing the system level vulnerabilities within a deployed machine learning [2]. This dissertation will focus on proposing team based strategies for evaluating, attacking, and protecting machine learning models in a structured, repeatable manner. This research will also build on available literature by demonstrating these strategies how two sample machine learning models. The final research direction is an empirical evaluation of the adversarial model risk based on a modification of the Drake equation.

Each method demonstrated can be filtered and controlled to protect the models in a production environment [18]. Research in this dissertation seeks to expand on available literature to close the loop - this paper is not solely about breaking models or fixing models. This work presents a framework that can go from one end machine learning development to the other end with monitoring and control for adversarial attacks.

Chapter 2

Literature Review

This literature review will cover the basics of adversarial machine learning, cyber teaming structures, and the targeted applications of adversarial machine learning at a high level. Each section will cover the relevant topics for the innovations included in this dissertation.

2.1 Adversarial Machine Learning

Adversarial machine learning (AML) [19] is a broad field of attacks that can manipulate or avoid detection by machine learning models. Goodfellow et.al [3] showed practical adversarial attacks in their paper and proposed structured ways to attack each model in 2014 . Adversarial attacks can be broken into two larger categories: black box [20] and white box attacks [21]. These style of attacks are divided by the amount of incoming knowledge an attacker has about a system [22]. White box attacks allow an attacker to know everything about a model: deployment, training style, inference style, dataset, and architecture. Black box attacks are the opposite: only access to the model is supplied and no additional information [23]. Typically, white box attacks are more successful due to the attacker's ability to tailor the attacks to the particular structural components of the underlying model [24]. Before diving into the specific adversarial attack vectors, we discuss the high level attack paths that are relevant to this work.

First, data poisoning [25] is a method for manipulating the dataset itself to influence the machine learning models built from that dataset. Data poisoning can occur in a number of different ways. One way is for the attacker to introduce structured noise into specific examples in the dataset [26]. This will cause the decision boundary of the models drift in a way advantageous to an attacker [27]. Another type of dataset poisoning attack is to

add incorrect labels to a dataset [10]. This can cause models to reduce their efficacy in identifying classes. As a final example, an attacker can remove or duplicate data points in an effect to influence the underlying model. Data poisoning requires access from an attacker to the dataset itself [28].

A second common adversarial attack is to manipulate data input prior to model inference. When the input is brought to a machine learning model, the input is perturbed in a targeted way to give advantages to an attacker. For instance, the adversarial paper by Goodfellow et.al [3] used structured noise at inference time to move the decision boundary of common ImageNet [29] classes to the wrong class with strikingly high effective rates. This is the classic example of adversarial attacks. As these attacks have developed over the years, black box and white box attacks have evolved into their own subfields within AML. The next sections will go deeper into black box and white box attacks.

2.1.1 Black Box Attacks

A black box attack [20, 30] on a machine learning model does not have access to the model information such as the dataset or the architecture. Black box attacks are the zero day attacks of the machine learning world. Without any underlying knowledge of the models, an attacker is able to avoid detection or manipulate the predictions of the machine learning models [31]. Black box attacks come in a few different types in machine learning - dataset poisoning, manipulation of the prediction to positively effect the attackers, and avoidance of detection by the underlying model [32]. A current limitations OF black box models is that they can require a large number of queries to attack the model [33].

Adversarial attacks [34] in black box models do rely on higher than usual number of queries to establish a credible attack. The black box attacker is looking for the appropriate reaction from the model. Production Systems can protect their models by simply reducing

number of queries available to users for these cruder attack avenues. Recent research demonstrated by Alzantot et.al generated a method called GenAttack that significantly reduced the number of queries needed to effectively attack the models in their experiments. [35]

Black box attacks can manipulate the output of the a model. The simplest of these attacks is to overlay adversarial noise on top of an image. The detected class can be changed with imperceptible noise on top of the image for a human. The model sees the structure it expects for a different class [36]. Recently, a new adversarial attack showed the ability of an attacker to change the class of a model simply by changing image attributes [37]. Manipulating the output of a model can take many forms but are difficult without underlying knowledge about the model itself. White box attacks have this information and show the true vulnerabilities of these models.

2.1.2 White Box Attacks

With white box attacks [21], it is assumed that an attacker will have access to the underlying model architecture, data, and even weights. Adversarial attacks efficacy relies on a baseline knowledge of the underlying model including items like architecture, design, and dataset [38]. A white box model makes it easier on the attacker to custom design attacks to meet their needs. Because the attacker knows the data and/or the architecture, the attacks can be designed to exploit either one of these properties.

Data poisoning can occur with white box attackers. A recent work proposed targeted attacks on a model by dataset poisoning called Property Inference From Poisoning [39]. In this work, the authors demonstrate effective attacks when the adversary has access to the dataset itself, along with the model specifications. Instead of simply trying to avoid detection or reduce detection efficacy, they show throughout the paper that information leakage can be manipulated at a fine tuned level. One of the most surprising outcomes - while manipulating approximately 10% of the data, they achieved 90% attack accuracy against classifiers looking at global features such as sentiment of incoming text.

2.1.3 Taxonomy of an Attack

The study of adversarial machine learning [40] is mature field that has been studying the effects these attacks for 30+ years. As the field matured, authors like Barreno et.al [41] enumerated a taxonomy of attacks that an attacker can carry out. There are three categories and six methods to discuss:

- **Influence**
  - *Causative* attacks manipulate training data for advantages to the attacker
  - *Exploratory* attacks probe a machine learning model to misclassify or avoid detections by the model
- **Security Violation**
  - *Integrity* attacks exploit a model via the use of false negatives
  - *Availability* attacks are similar to denial of service attacks in Cyber. This can be carried out by flooding the production pipeline or creating an abnormal set of false positives as two examples.
- **Specificity**
  - *Targeted* attacks focus on a specific model, dataset, or architecture
  - *Indiscriminate* attacks should be effective against a large number of models. This is commonly called a Universal attack in literature today.

White box and black box models can fall into multiple categories depending on their design [42]. An attack could be designed for a single purpose but as the field has progressed, so have the ability of the attackers to accomplish multiple goals in this taxonomy. Throughout this work, these methods will be referred to as appropriate.

2.2 Protections for Machine Learning Models

Machine learning models can be attacked during development and deployment [43]. There are a diverse set of ways to protect machine learning models. This section will introduce two relevant topics to protecting these models with automated techniques for production systems: Adversarial Training [44] and Dataset Poisoning Protections [45].

2.2.1 Adversarial Training

Transferability of attacks from one model to another is a crucial piece for black box attackers - if an attack can transfer from one model to another, it is likely to be a successful black box attack [46]. Adversarial Training [44] is the idea that adversarial images can be used as a training augmentation for input samples to provide robustness to downstream model [47]. Adversarial Training at Scale [44] was the first paper in 2016 to attempt to do this at Google Scale type problems. Kurakin et.al created a technique for Adversarial Training at Scale for larger models and datasets. Before this paper, Adversarial Training success stories were mainly on smaller datasets or problems [48]. This paper provides suggestions on how to properly successfully train on adversarial examples to improve the robustness of the underlying machine learning model.

The field progressed to introduce black box and white box style attack to improve the performance of the underlying model with a technique called Gray Box Training [49]. This method focuses on using multiple models to understand the effects of using the different style of attacks as training data. The authors are able to demonstrate additional robustness in the new models using this training method. One of the largest problems with adversarial training is the explosive computational cost to training on adversarial examples in addition to training on regular data. The number of training samples can exponentially increase due to optimization of the model against the attack vectors. [50] Vivek continued his research and recently proposed Single Step Adversarial Training Dropout [51]. This method proposed using a dropout mechanic to reduce the number of adversarial examples needed to improve

the robustness of the model. Finally, most recently, Liu et.al. proposed a new process that only requires a single step in the adversarial training process [52]. Instead of focusing on different perturbations, this method proposes iterative improvements to the models that save training time and improves text accuracy when models are attacked.

2.2.2 Dataset Poisoning Protections

Dataset Poisoning [45] is a more sinister attack method; if protections are built into the model and pipelines, they cannot protect against adversarial examples that are contained the dataset. Steinhardt et.al wrote on addressing losses of a diverse set of attacks with two key assumptions . The first assumption is to compare the statistical concentration between the train and test errors - this allowed the authors to understand the relationship between the the models loss function and test set error. This first assumption answers the question about whether the train and test dataset populations are close - if they are, poisoned samples can have a large effect on the model. The second assumption is that outliers in a non-poisoned dataset do not have a large effect on the model itself. These tests demonstrated that MNIST style datasets were robust to attacks but the IMDB dataset could be poisoned with a low amount of poisoned data.

2.3 Cyber Color Teams and Structures

Cyber security is built and tested by adversarial attackers every day. Borrowing from their experience is an integral part of this work. First, this section will cover cyber color teams [53]. These teams are formed around particular goals in protecting the underlying systems. These concepts translate to the machine learning production teams. Second, combining teams into new colors to combine responsibilities. Especially in Machine Learning, how the system is built and maintained will drive what vulnerabilities are available for an attacker to exploit. There are processes that utilize these teams called Build, Attack, Defend that bring structure to attacking and fixing issues found during these phases.

2.3.1 Cyber Color Teams

In cyber security, teams highlight their purpose through the use of colors [53]. A red team will attack a cyber physical system and attempt to find vulnerabilities in the system [54]. A blue team will take open lists of threats and discovered vulnerabilities and propose fixes to these issues [55]. Recently, cyber color teams have been evolving to mix concepts from each of these different teams. The Green team, for instance, will mix the jobs of Blue and Red Teams. The Green Team discovers vulnerabilities and proposes fixes to each of those discovered issues [56]. There is a concept around colored teams for attacking, defending, and designing systems:

1. Red Team: Ethical hacking of a target system
2. Blue Team: A group of people building defenses against the attacks on the model
3. Green Team: A blend of the red and blue teams where the group will simultaneously run both attacks and create defenses for those attacks.

In this work, the green team construct is used to improve the machine learning model development process. By utilizing adversarial design knowledge (red team) and model building knowledge (blue team), it is possible to baseline and attack a system with the goal of proposing fixes to the underlying model design or production pipeline [57].

2.3.2 Build, Attack, Defend for Cyber Systems

Build, Attack, Defend [58] is a construct built around breaking down cyber security problems into actionable problem spaces for teams to digest. There are three distinct steps proposed by this method. First, the build phase is where the system is constructed and designed with protections built into the underlying system. Second, the red team will attack the system identify vulnerabilities and exploits that exist in the current production deployment. Third, the final step is to defend the system by improving defenses or adding

additional protections. This is a continuous process that is designed to evaluate, fix, and monitor for issues with a production cyber system.

2.4 Attacks on Sentiment Analysis Models

Adversarial Attacks on natural language models will rely on exploiting general knowledge about the underlying language model or language characteristics [59]. Language characteristics like slang and synonyms can have the same meaning to a human but drastically different predictions from a machine learning models [60]. Adversarial actors have figured out how to exploit these weaknesses and use them against NLP models in the wild. A classic example of one of these attacks is the HotFlip [61] attack. HotFlip is a set of White-Box Adversarial Examples for Text Classification where the authors successfully demonstrated the ability to drastically change the classification accuracy of the underlying model by changing a single character in the sentence . The authors demonstrate that even word classifiers are vulnerable to these types of attacks.

The field has evolved since the publication of simple attacks like HotFlip [61]. Belinkov et.al [62] built methods for generating synthetic and natural noise to break neural machine translation. The authors in this paper demonstrated simple character attacks that would fool state of the art models. Furthermore, they showed that noisy text like misspellings were especially difficult for the models to understand. Alzanot et. al [63] created a methodology for Generating Natural Language Adversarial Examples for Sentiment Analysis models. By using small perturbations in the sentences, like replacing a single word, the model detected different semantics in the sentence. When these same changes were shown to humans, they saw a 92.3% successful classification to the correct sentiment label by human annotators.

Chapter 3

Datasets

There are two primary problem sets addressed in this work - Sentiment Analysis for Natural Language Understanding [64] and Image Classification in Computer Vision [65]. The methods demonstrated throughout this work can be applied to machine learning pipelines and show vulnerabilities on these datasets.

3.1 Sentiment Analysis Background

Sentiment Analysis [64] is the task of analyzing text to provide a classification of such as positive, negative, or neutral for a given string (paragraph, sentence, sub-sentence). Sentiment Analysis has use cases like polarity at for words, sentences, paragraphs, and documents. Organizations use sentiment analysis to moderate their websites, apps, and comment sections with the vast amounts of textual data [66]. These machine learning systems are susceptible to adversarial attacks [67]. Recently, the threat of bias in the data has led the research community to explore how to combat and understand the inherit qualities in social media data - both positive and neutral that cause machine learning models to incorrectly classify certain phrases [68].

The field of Sentiment Analysis has progressed rapidly due to the expansion of social media platforms [69]. Among the numerous applications across the internet, Sentiment Analysis is a required piece of policing social media and comment sections [70]. With state of the art sentiment analysis systems progressively improving, it became evident that there is unintended bias (like race, gender, sexual orientation, etc.) built into dataset of comments and tweets [71]. This bias has led to research in how to create systems that can find negative text in the wild without overfitting. Last year, a famous example for the Perspective API is

the classification for the input "I am a gay black woman" which carried a 95% toxicity [72]. Sentiment Analysis state of the art papers [73] are dominated by ensembles of transformer technologies for this particular problem set. Each of these technologies inherits weaknesses of the underlying language models used for the classification task.

3.2 Image Classification Background

Image classification [74] is a computer vision task where an image goes into a system and an array of classification scores is provided back to a user [65]. Image Classifiers were based on traditional computer vision algorithms [75]. Recently, a new class of methods relying on deep neural network features has produced state of the art results in image classification [76]. This work presented in this chapter will focus on using deep image classifiers and exercising those algorithms throughout the course of this paper. In this research, the following datasets are explored using multiple experiments. These datasets are as follows:

1. CalTech101 [77]
    (a) Description: A standard image classification benchmark with 101 categoies. The number of image per category ranges from 40 to 800. This dataset was collected in 2003 [78].
2. Icons-50 [79]
    (a) Description: Icons-50 dataset has 50 classes of icons with 10K images included in the dataset. The icons have different styles and subclasses associated with them. [80].
3. indoorCVPR [81]
    (a) Description: Indoor Scene Recognition dataset consists of indoor scenes with 67 categories and 15,000 total images. The images are categorized and each class has at least 100 images. [81].

4. Stanford Dogs Dataset [82]

    (a) Description: A dataset of only dogs containing 120 classes from around the world. This dataset is a subset of the ImageNet dataset and contains approximately 20K images. [29].

5. tiny-imagenet [83]

    (a) Description: A smaller version of the ImageNet challenge that includes 100K images with 200 classes sized to a 64x64 size for faster evaluations. [84].

In this research, seven following pretrained networks are explored using multiple experiments. These pretrained networks are as follows:

1. MobileNetV2 [85]

    (a) Description: MobileNetV2 is the improvement of the state of the art architecture for multiple tasks in the Computer Vision world. This architecture is still widely utilized in the community due to it's favorable trade off between power and performance [86].

2. NASNetMobile [87]

    (a) Description: The unique part of this model architecture is that is performs a model architecture search on a particular dataset. This model focuses on transferablity of the architecture from the base weights to learning new weights on the target dataset.

3. DenseNet121 [88]

    (a) Description: This DenseNet architecture has a the unique features - they eliminated the vanishing gradient problem that plagued networks at this time, they increased the ability of features to propagate through the network, and, finally, reduced the size of the network since the features went through the network.

4. ResNet50 [89]

    (a) Description: The ResNet architecture uses residual net architectures with various numbers of layer depth. In this case, the 50 layer version of the ResNet version is explored in the experiments. In 2016, this architecture was the state of the art architecture for many Computer Vision tasks.

5. DenseNet201 [88]

    (a) Description: DenseNet201 improves the 121 architecture by adding a large network size to increase the number of features used with a small degradation in performance.

6. Xception [90]

    (a) Description: Chollet interpreted the Inception models from convolutional neural networks in a novel way - he proposed a depthwise separable convolution that exists in between the regular convolution and the depthwise separable convolution step. In order to accomplish this, he proposed replacing Inception Modules with depthwise separable convolutions. With the same number of parameters, the XCeption network gained increased performance with better use of those parameters that the corresponding Inception network.

7. InceptionV3 [91]

    (a) Description: Another novel state of the art image recognition model. The InceptionV3 architecture focused on deeper networks while keeping the number of parameters from growing too large.

Each of these networks can be exercised, trained, and evaluated using the Keras framework with a Tensorflow backend. Using these networks provides an overview of current network architectures and how they perform against certain styles of attacks.

3.3 Datasets

There exist an extremely large number of datasets available in machine learning for every topic area. Within the areas of Sentiment Analysis and Image Classification, we use the following datasets in this work: Jigsaw Bias Dataset for Sentiment Analysis and nine standard image classification datasets.

3.3.1 Sentiment Analysis

Sentiment Analysis has been around for decades. The JigSaw Toxic Bias dataset has an explicit goal to combat bias in sentiment analysis. The Conversation AI Team, funded in conjunction with Jigsaw and Google, created a dataset around toxicity, biases, and threats in comment sections [92]. The JigSaw Toxic Bias dataset is a set of publicly released comments augmented with new labels for ML tasks. It has a wide range of different toxicity classifications such as severe toxicity, obscene, identity attack, insult, and threat. In the last year, the Conversation AI team has augmented JigSaw with additional categories including gender, sexual orientation, and religious identity. The new evaluation categories were added to combat inherent biases that are included in the data but do not represent a negative sentiment. This dataset is part of a challenge on the Kaggle competition page for creating the best classification models around toxicity. This is the primary Sentiment Analysis dataset used in this dissertation.

In order to understand the data better, here are a few examples of the dataset in action. First, here is the specification for the JigSaw dataset with possible labels:

1. Comment Input: String
    (a) Toxicity (Label):
        i. toxic (Label Subcategory)
        ii. severe_toxic (Label Subcategory)
        iii. obscene (Label Subcategory)

iv. threat (Label Subcategory)

v. insult (Label Subcategory)

vi. identity_hate (Label Subcategory)

(b) Identity Attributes (Label):

i. Label Subcategories: [male, female, transgender, other_gender, heterosexual, homosexual_gay_or_lesbian, bisexual,other_sexual_orientation, christian, jewish, muslim, hindu, buddhist, atheist, other_religion, black,white, asian,latino, other_race_or_ethnicity, physical_disability, intellectual_or_learning_disability, psychiatric_or_mental_illness, other_disability]

The identity labels were not used in each of the explorations. Instead, the focus is on the broader toxicty categories and what each of the machine learning models atyacked learned about the distribution of the data in relation to the label. Here are three examples of data in the JigSaw dataset:

1. Comment Input: i'm a white woman in my late 60's and believe me, they are not too crazy about me either!!

   (a) Toxicity (Label):

   i. toxic (Label Subcategory) : 0.00

   ii. severe_toxic (Label Subcategory) : 0.00

   iii. obscene (Label Subcategory) : 0.00

   iv. threat (Label Subcategory) : 0.00

   v. insult (Label Subcategory) : 0.00

   vi. identity_hate (Label Subcategory) : 0.00

   (b) Identity Attributes (Label):

   i. female (Label Subcategory) : 1.0

ii. white (Label Subcategory): 1.0

iii. **NOTE:** All other identity subcategories are zero

2. Comment Input: Why would you assume that the nurses in this story were women?

   (a) Toxicity (Label):

      i. toxic (Label Subcategory) : 0.00

      ii. severe_toxic (Label Subcategory) : 0.00

      iii. obscene (Label Subcategory) : 0.00

      iv. threat (Label Subcategory) : 0.00

      v. insult (Label Subcategory) : 0.00

      vi. identity_hate (Label Subcategory) : 0.00

   (b) Identity Attributes (Label):

      i. female (Label Subcategory) : 0.8

      ii. **NOTE:** All other identity subcategories are zero

3. Comment Input: Continue to stand strong LGBT community. Yes, indeed, you'll overcome and you have.

   (a) Toxicity (Label):

      i. toxic (Label Subcategory) : 0.00

      ii. severe_toxic (Label Subcategory) : 0.00

      iii. obscene (Label Subcategory) : 0.00

      iv. threat (Label Subcategory) : 0.00

      v. insult (Label Subcategory) : 0.00

      vi. identity_hate (Label Subcategory) : 0.00

   (b) Identity Attributes (Label):

i. homosexual_gay_or_lesbian (Label Subcategory) : 0.8

ii. bisexual (Label Subcategory) : 0.6

iii. transgender (Label Subcategory) : 0.3

iv. **NOTE:** All other identity subcategories are zero

Because of the toxic nature of the comments, only positive toxic comments (zero toxicity) were documented in this section.

3.3.2 Image Classification

There are nine datasets used in the Image Classification experiments in this work. The following list will show examples and describe the common literature use case for these datasets.

Figure 3.1: Example of the BSD100 dataset

1. BSD100 [93]
   (a) Description: The Berkeley Segmentation Dataset and Benchmark is a dataset used primarily to benchmark image segmentation and boundary detection problems [94].
   (b) Formats:
      i. Image - Label
      ii. Image - Mask
   (c) Number of Images: 100
   (d) Resolution : 481x321 pixels
   (e) Sample Use Cases:
      i. Image Segmentation
      ii. Boundary Detection
      iii. Classification

Figure 3.2: Example of the DIV2X dataset

2. DIV2K [95]

   (a) Description: DIV2K is a super-resolution dataset that contains 1,000 images and is used primarily to benchmark differences in blur, resolution scaling, and other computer vision tasks.

   (b) Formats

      i. Image - Label

      ii. Image - Mask

   (c) Number of Images: 2000

   (d) Resolution: 2K Pixels

   (e) Sample Use Cases

      i. Image Segmentation

      ii. Boundary Detection

      iii. Classification

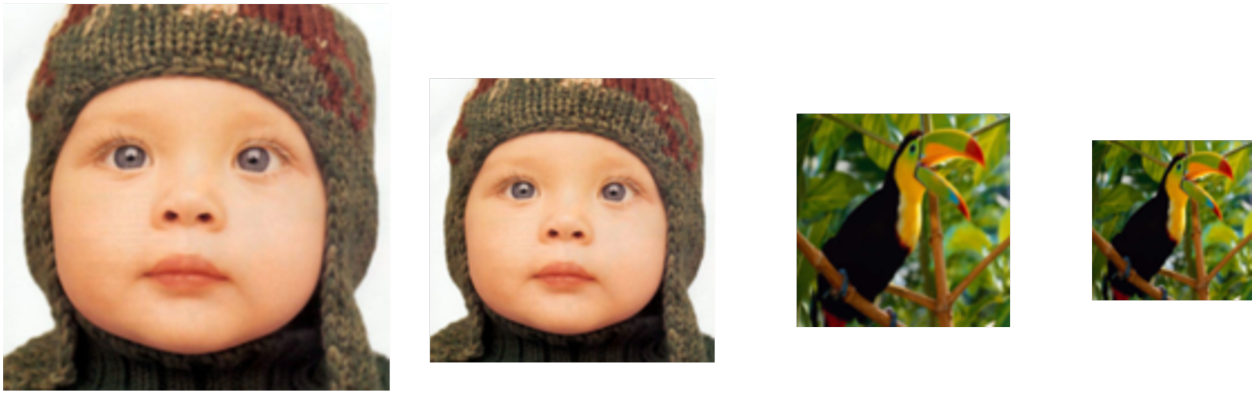

Figure 3.3: Example of the Set5 dataset

3. Set5 [96]

   (a) Description: A super resolution testing dataset which consists of five images: butterfly, baby, head, woman, and bird.

   (b) Formats

      i. Image - Label

      ii. Image - Mask

   (c) Number of Images: 5

   (d) Resolution: Varies by Image - for instance, 512x512, 74x112, etc.

   (e) Sample Use Cases

      i. Image Segmentation

      ii. Boundary Detection

      iii. Classification

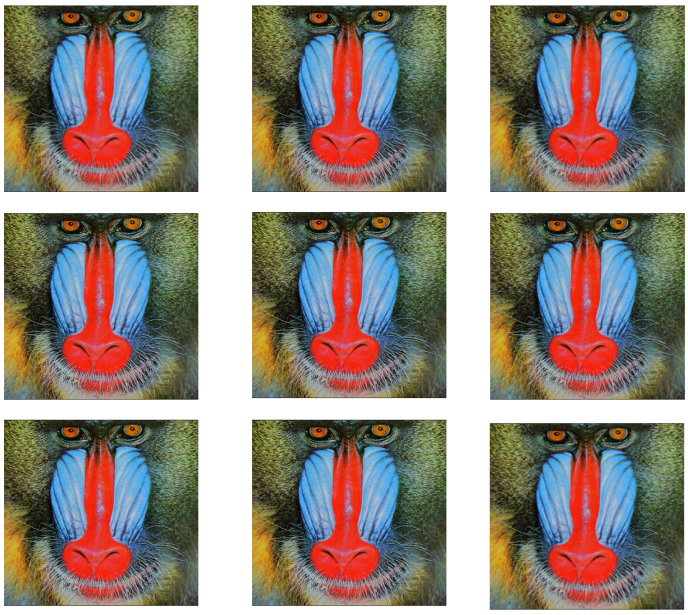

Figure 3.4: Example of the Set14 dataset

4. Set14 [97]

   (a) Description: A super resolution testing datatset which consists of fourteen images

   (b) Formats

      i. Image - Label

      ii. Image - Mask

   (c) Number of Images: 14

   (d) Resolution: Varies by Image - for instance, 512x512, 74x112, etc.

   (e) Sample Use Cases

      i. Image Segmentation

      ii. Boundary Detection

      iii. Classification

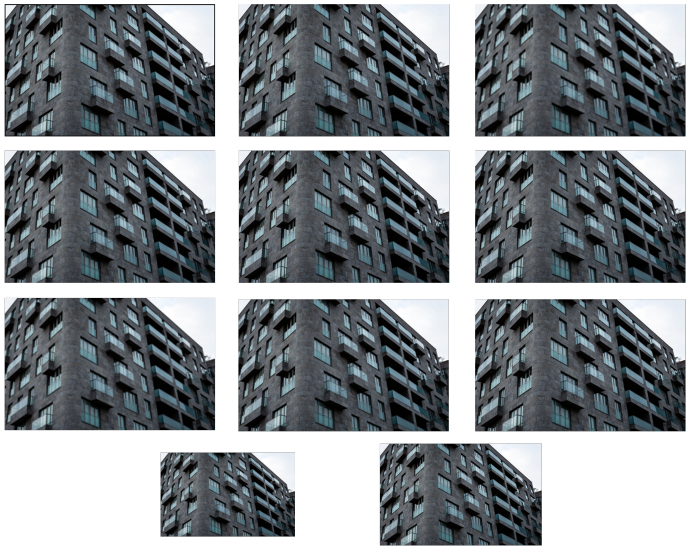

Figure 3.5: Example of the Urban100 dataset

5. Urban100 [98]

   (a) Description: 100 images of urban scenes used with super resolution problem sets.

   (b) Formats

      i. Image - Label

      ii. Image - Mask

   (c) Number of Images: 100

   (d) Resolution: Multiple Resolutions for different problem sets

   (e) Sample Use Cases

      i. Image Segmentation

      ii. Boundary Detection

      iii. Classification

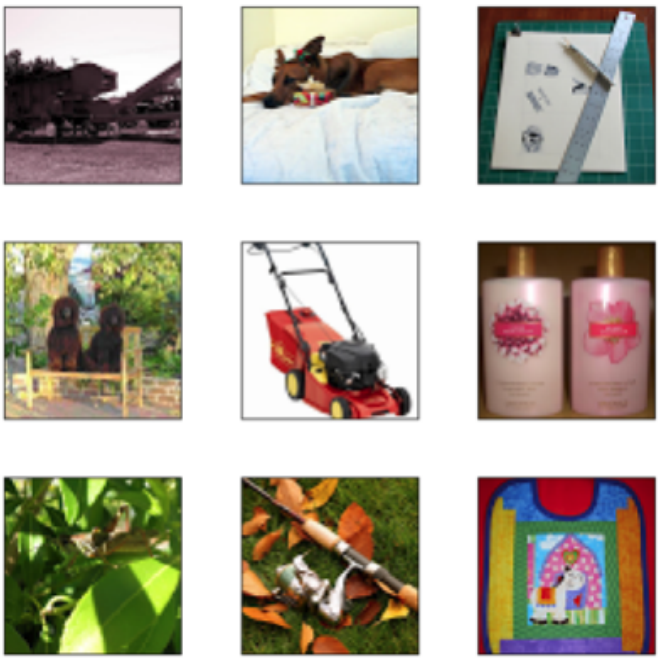

Figure 3.6: Example of the Urban100 dataset

6. ImageNet64 (16k subset) [99]

   (a) Description: A small subset of the ImageNet dataset used for benchmarking models and attacks.The focus of this work is to test adversarial attacks on multiple targets so this provided a reasonable sampling of the ImageNet dataset.

   (b) Format

      i. Image - Label

   (c) Number of Images: 1.2 million

   (d) Resolution: 64x64

   (e) Sample Use Cases

      i. Image Classification

VEDAI Dataset

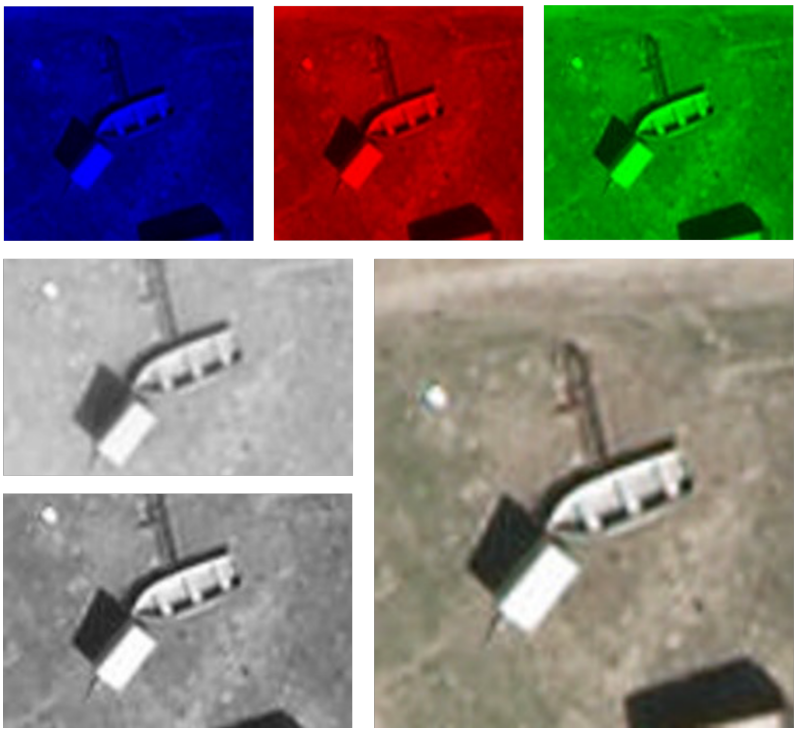

Figure 3.7: Example of the VEDAI dataset

The VEDAI or Vehicle Detection in Aerial Imagery dataset is a benchmarking dataset with multiple channels looking at the same locations. The dataset includes nine vehicle classes that can be challenging to distinguish for computer vision applications. A critical aspect of this data is the rectified Infrared images included inside of this dataset. The original paper by Razakarivony and Jurie highlights the use of common machine learning applications to identify the classes inside of the dataset [100]. In this paper, the goal is to extend previous work by protecting a common image classification machine learning model from adversarial attacks. Three primary channels are included in the dataset: visible, infrared, and gray. For the experiments included in this paper, the Visible channel is also split into its principal components of red, green, and blue channels to explore the color dependence of the modeling.

Chapter 4

Black to White Box - Discovering Model Characteristics with Strategic Probing

4.1 Introduction

White Box Adversarial Machine Learning Attacks rely on knowing underlying knowledge about the model construction and training data. In this chapter, we are able to discover the underlying architecture from a black box model with a structured set of input probes. Once the architecture has been discovered, the underlying dataset it was trained on can also be determined. With imagery, we focus on classification and explore multiple architectures and datasets commonly available in Keras. Text generation with a single transformer architecture is explored by fine tuning off different datasets. Each of those datasets are distinguishable from one another in the output of the transformer. Diversity in text transformer outputs implies further research is needed to successfully classify architecture attribution in text domain.

4.1.1 Background

Adversarial Attacks can be broken into two larger categories: black box and white box attacks. With Black Box methods, the attacker does not have access to the model while in White Box attacks they do have access to the model. Adversarial Attacks rely on baseline knowledge of the underlying model architecture and/or dataset [101]. Black and White Box attacks are divided by the amount of incoming knowledge an attacker has about the system. In the Cyber community, it is considered a huge advantage to know the underlying hardware, software, or network architecture when attacking a system [102]. Similarly, in Machine Learning, understanding the dataset, model architecture, or hardware it's running on can provide a large advantage to an attacker [9].

How can an adversarial agent get access to this information without access to the model? This exploration of image and text classifiers demonstrates the ability to find the underlying model architecture or fine tuned dataset from a set of input probes.

4.1.2 Challenges

We focus on detecting commonly used datasets and architectures applied in the text and image space. If a machine learning development team trains their model on a completely custom dataset, then this method would only detect how similar the given dataset or architecture is to the previously known ones.

The key limitation in this work is using published datasets and architectures to build datasets of knowledge for each of our targeted attacks. For example, in a dogs dataset, we choose a dog from the dataset where the model would produce a predictable output for our detector. Specific attacks to specific model architectures or datasets are more effective for adversarial attacks [103].

4.1.3 Contributions

Adversarial Attack papers are either universal or targeted to a particular model/dataset. There is a prior step that is often overlooked in the adversarial attack process – discovering which attack is effective against a target. This short paper offers a proposed process for discovering underlying model architecture or dataset for use in an adversarial system.

4.2 Approach

To classify the underlying model architecture and dataset, the first step is to explore the different common datasets and model architectures for the image and text space. Our approach will document the experiment's permutations for both imagery and natural language processing as well as a brief overview of the classifier tooling used to predict the underlying architecture.

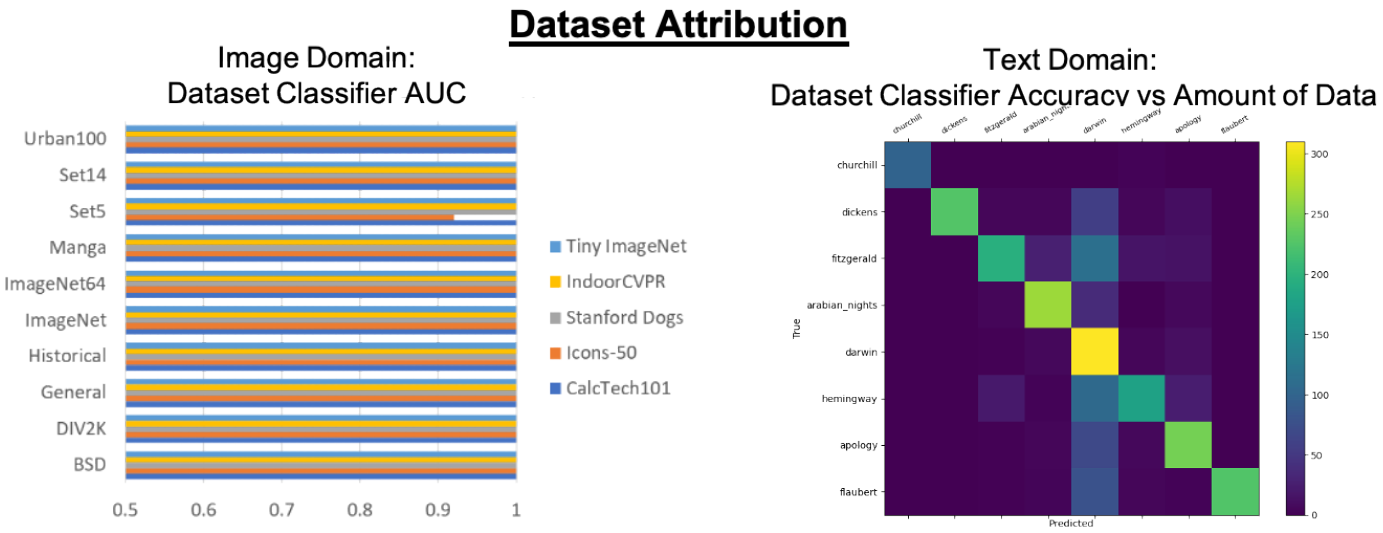

Figure 4.1: Results of Dataset Attribution Experiments

4.2.1 Experiment Design

Each domain in machine learning has common architectures and datasets that are used to create the base weights. Our goal was to identify the model architecture and the data it was trained on strictly from probing the final model.

Architecture Attribution

There are hundreds of model architectures for each problem in a sub-domain. In the adversarial domain, the potency of adversarial techniques relies on knowing the underlying architecture. With our approach of determining the architecture with only access to the model, we attempt to classify the classifier.

Dataset Attribution

Model fine tuning is defined as starting with a set of base weights and updating the model weights based on training on a smaller, targeted dataset. This work seeks to show that the fine tuning step actually leaves model weight nuances than can be detected by a classifier from the smaller dataset.

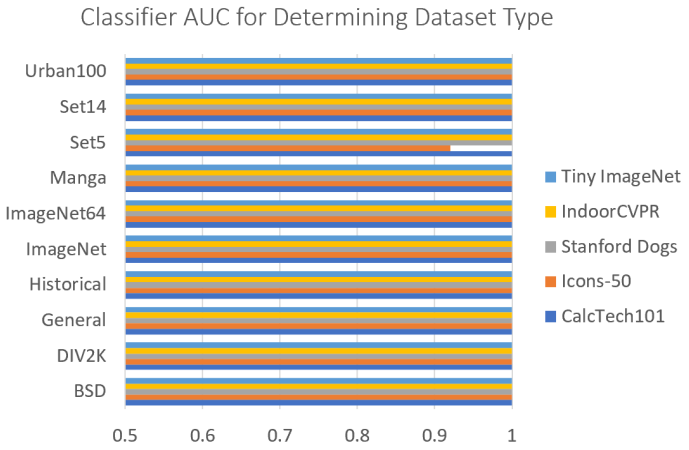

Figure 4.2: Classifier AUC for Determining Model Type

In the image space, we explored ten separate datasets. The datasets used in the image experiments were mostly taken from super resolution datasets alongside a few subsets of popular image datasets. Similarly, on the text side, we selected recognizable datasets taken from popular authors to show how each model learns and generates different underlying language models for each of the downstream tasks. The dataset in Natural Language Processing (NLP) can be a much more diverse task as permutations with language become intractable quickly.

4.3 Evaluation

Two domains were selected for evaluation, computer vision and natural language processing. In Computer Vision, our experiments explored pretrained image classifiers available in Keras while training on datasets publicly available. In Natural Language Processing, each experiment used readily available datasets in the HuggingFace packages. Due to the computational complexity of training transformers, we explored the smaller model versions of the popular GPT-2 model and were able to successfully classify a fine tuned model.

4.3.1 Dataset Attribution Results

Dataset attribution, as seen in Figure 4.1, relies on the ability of our classifier to access the resulting feature vector in the image space and the resulting text output of the transformer. All datasets were download and deploy ready and provided a wide variety of outputs to explore.

Determining the Dataset Type from a Single Image

The following datasets were used with random 50 class subsets to fine tune MobileNetV2 [85]: CalTech101 [78], Icons-50 [79], indoorCVPR [81], Stanford Dogs Dataset [82], tiny-imagenet [83]. After training, inference was applied to the datasets: MobileNetV2 [85],NASNetMobile [87],DenseNet121 [88],ResNet50 [89],DenseNet201 [88],Xception [90], and InceptionV3 [91]. The inference vector output was captured alongside the fine tuned model which was used. We then trained a classifier on each inference dataset to attempt to predict which fine tuned model the inference came from. The results were surprisingly good. For any of the given datasets used for inference we can predict with an AP of greater than .99 for any of the fine tune models.

Determining the Dataset Type from a Single Text Input

The following datasets were trained with a GPT-2 small model: Churchhill [104], Dickens [105], Fitzgerald [106], Arabian Nights [107], Darwin [108], Hemmingway, and Flaubert [109]. The GPT-2 transformer models were trained using the HuggingFace repository for NLP research [110]. With the dataset classifier using a Bert Large model, we were able to achieve 81% classification accuracy against each of these authors [111]. In order to show the shortcomings of this classifier, we chose to show the confusion matrix for the classifier. Overall, the model is able to clearly classify each of the authors. Notably in this

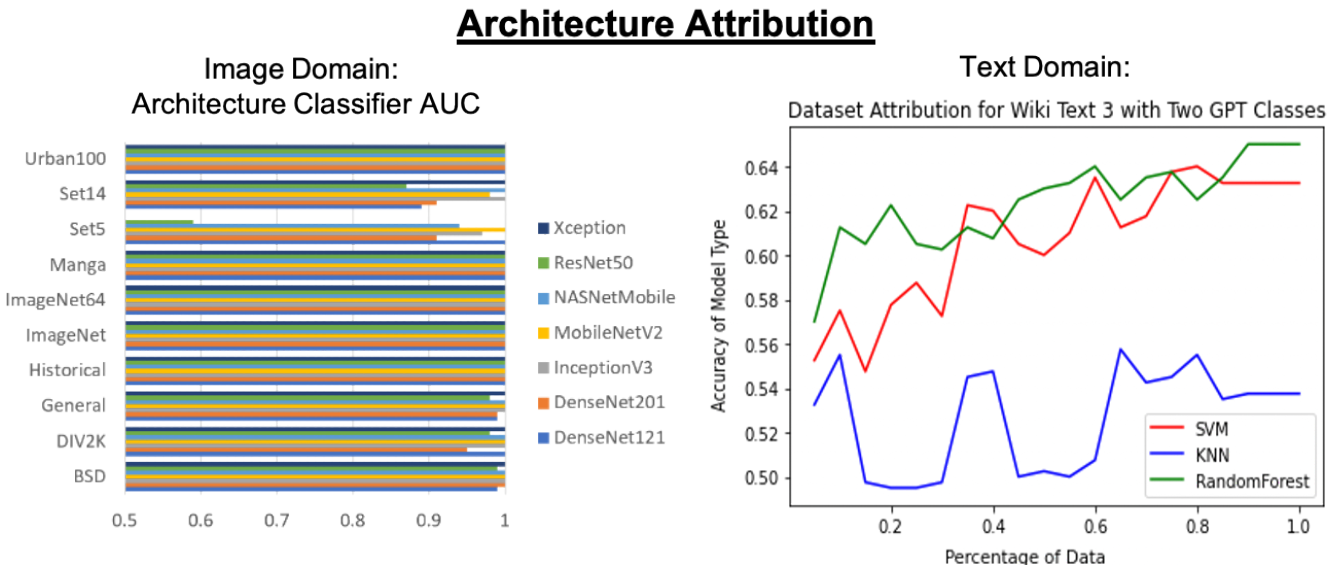

Figure 4.3: Results of Architecture Attribution Experiments

matrix, there is some difficulty in identifying Darwin. This is likely due to how the transformer captured their style, relative to how GPT-2 [112] is able to capture structure of prose including whitespace and formatting.

4.3.2 Architecture Attribution Results

We attempt to classify the pretrained architectures used for particular image classification and text generation tasks in this section. By applying different pretrained architectures onto different datasets, we can learn a fingerprint that each pretrained model has inherent in their weights. Once we have trained our classifier on these fingerprints, we can find the model used from a single image or text sample. The same techniques are also used to try to classify the model used to fine tune a base model. We take a pretrained model and fine tune it on different datasets and proceed to apply inference on a test set of images. We then take the inference output and try to predict the which fine tuned classifier the result came from.

Determining the Architecture Type from a Single Image

We used the ten following datasets: General, BSD100, DIV2K, Set5, Set14, Urban100, ImageNet64 (16k subset), ImageNet (6k subset), Manga, Historical and the seven following pretrained networks: MobileNetV2, NASNetMobile, DenseNet121, ResNet50, DenseNet201, Xception, InceptionV3. All pretrained models are from keras.applications which are pretrained on 1k classes of ImageNet [113]. Inference was performed on each of the datasets from each of the models and the 1000-dimension output was captured. Then, for each dataset, the model's output was collected alongside the name of the model used. This captures the difference in each of the feature outputs and allows us to use a classifier that predicts from the inference results onto the model on which it came from.

For each dataset, the model was able to be predicted with an Average Precision of 0.84 for the worst-case scenario of five Set5 images. All other classifiers had an Average Precision of 0.99. Therefore, given at least five images from a dataset and a sample image classification prediction, we can easily classify which pretrained model was used to classify it.

Determining the Architecture Type from a Text Sample

Classifying the architecture with text transformers can be harder due to hardware and time limitations. For instance, recent research is focused on speeding up training of transformers by 10-40% to make the retraining these models more approachable [114]. Our experiments trained two separate models, GPT-2-small and DistilBert on the same dataset [115]. This experiment used the Wiki Text language modeling dataset - a benchmark based on verified Wikipedia articles that provides fast and repeatable training results for many of the transformer architectures [116]. The experiments shown in Figure 4.2 show that our accuracy with 20,000 samples with our five probes still only shows moderate predictability at 60% accuracy for simple classifiers. The text probes were chosen as basic, approachable probes: 'Hello', '2+2', 'A', and 'Mario'. Given the Wikipedia input dataset, these probes

should provide a diverse but distinguishable output. The results above demonstrate further honing of the target phrases will lead to improved accuracy.

4.3.3 Limitations

Each of these experiments focused on demonstrating how distinguishable an architecture or dataset are in the image and text domain. As architectures are modified, it may only be possible to find the most similar architecture to a particular output. Or, for instance, a model may perform similarly when trained on different datasets. The experiments outlined in this paper do not mix datasets or modify architectures. Each of the classifier are able to be directly downloaded from Keras or HuggingFace with no additional installations necessary.

4.4 Results and Future Work

In the text space, the technique needs to be refined down to only use the output from the model at an API or pipeline level. It is possible to also use targeted attacks known to work against particular models to determine the underlying architecture. If an adversarial attack is tailored to work against a particular model (and isn't universal), then that technique would not be effective against tangential architecture types. For the text side, future work will focus on determining the number of samples needed to improved the efficacy of classifying the underlying model architecture. Larger datasets, like modern transformers are trained on with billions of documents, will require further investigation [117]. Future experiments would utilize larger versions of the GPT-2 model and even the recently released GPT-3 [118].

In the image space, further work with predicting families of classifier based on how similar their outputs are will allow us to group architectures for adversarial attacks. This will close the gap between a target attack on one model and a universal attack on all models. Specifically in the classifier space, the prevalence on transfer learning leaves the machine learning community at heavy risk of compromise.

4.5 Chapter Summary

In the image space, learning the fingerprint of a model is achievable with modern classifiers. Our architecture and dataset discovery method reaches high AP numbers with minimal training. In the text domain, classifying the underlying model architecture is harder with a single text sample. For trained datasets, the results in the text domain showed that datasets with clear stylistic cues are distinguishable from each other.

Chapter 5

Green Team Machine Learning to Reduce Adversarial Attack Surface

5.1 Introduction

This work proposes a structured approach to baselining a model, identifying attack vectors, and securing the machine learning models after deployment. This method for securing each model post deployment is called the BAD (Build, Attack, and Defend) Architecture. Two implementations of the BAD architecture are evaluated to quantify the adversarial life cycle for a black box Sentiment Analysis system. As a challenging diagnostic, the Jigsaw Toxic Bias dataset is selected as the baseline in our performance tool. Each implementation of the architecture will build a baseline performance report, attack a common weakness, and defend the incoming attack. As an important note: each attack surface demonstrated in this work is detectable and preventable. The goal is to demonstrate a viable methodology for securing a machine learning model in a production setting.

Sentiment Analysis (SA) [64] is the task of analyzing text to provide a classification such as positive, negative, or neutral for a given sample. SA is subdivided into categories such as polarity, subject, and toxicity. Companies and organizations use these technologies to moderate their websites, apps, and comment sections [119]. Adversarial attacks, in the context of this work, refer to any input that allows an adversarial actor to trick a classification system. Modern SA Systems use machine learning (ML) and are susceptible to adversarial attacks [24]. Recently, the Natural Language Processing (NLP) community has explored how to create models that can handle bias in training data; for instance, content-aware models are an example of a system that can interpret bias in the data and correctly classify sentiment [120]. Given the challenging nature of SA with this type of data, the goal is to

demonstrate a simple and repeatable process for creating a model baseline, attacking the model, and defending against the incoming attacks.

Toxicity Classification [121] is a SA technique to understand the malicious intent of text based on words and content in the message. These SA techniques use ML and Deep Learning (DL) to classify the toxicity or polarity of a tweet [122]. The first SA technique used in this chapter is the Sentiment140 SA API [123]. Sentiment140 originated as a paper from the early 2010s and was later developed into an API by a Stanford Team [124]. Perspective, the second SA API used, is built and maintained by Google's Jigsaw team [71]. The Perspective API is a black box ML model that relies on a transformer and other DL technologies to classify sentiment. The Perspective API focuses on toxicity analysis for social media-based comments [125]. Each SA API uses ML to provide sentiment classification. We have no connection or insight into the underlying models other than published papers or websites. Further, there are limitations on the number of queries per second and per day.

5.1.1 Challenges

The dataset used in this work creates a unique challenge. The Conversation AI Team, funded in conjunction with Jigsaw and Google, created a dataset around toxicity, biases, and threats in comment sections [92]. The JigSaw Toxic Bias dataset is a set of publicly released comments augmented with new labels for ML tasks. It has a wide range of different toxicity classifications such as severe toxicity, obscene, identity attack, insult, and threat. In the last year, the Conversation AI team has augmented JigSaw Toxic Bias dataset with additional categories including gender, sexual orientation, and religious identity. The new evaluation categories were added to combat inherent biases that are included in the data but do not represent a negative sentiment. This dataset is part of a challenge on the Kaggle competition page for creating the best classification models around toxicity. The top-scoring models used ensembles of DL models to get the highest classification scores. Since the newest classification techniques for this dataset use ML, they are susceptible to adversarial methods. Adversarial

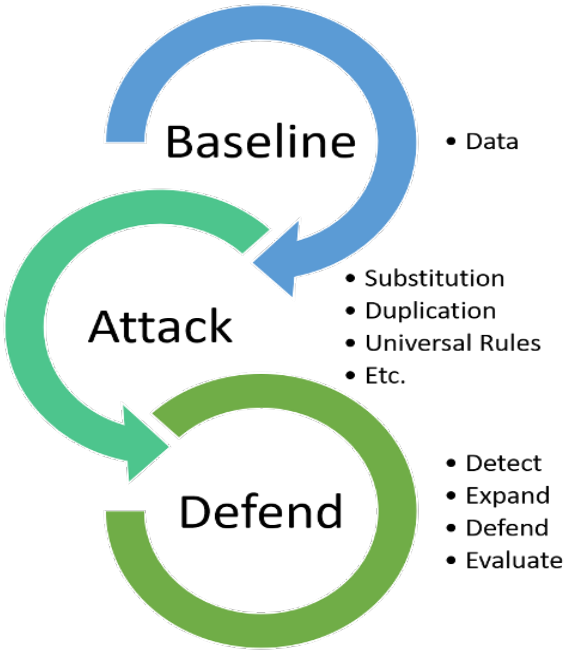

Figure 5.1: BAD System: Baseline, Attack, Defend for Protecting Machine Learning Models

Methods demonstrated in the Evaluation section are focused on discovering attacks that negatively affect the classification capability of the underlying system. This chapter will focus on the challenge of evaluating the attack surface of a single attack vector and defending the model from this incoming attack.

5.1.2 Contributions

Single Character attacks vectors are a direct analog to single-pixel attacks in the image domain - for instance, single-pixel attacks have demonstrated effects on classification, reinforcement learning, and other state-of-the-art image technologies [126] [127]. We demonstrate the efficacy of single character attacks (1 or many) on these sentiment text classifiers and how to protect the underlying system. These simple attacks can reduce the ability of systems to filter and curate online media platforms. This chapter focuses on demonstrating a new architecture to build a baseline of the performance of each API, attack the models with single character substitution/insertion attacks in the text domain, and provide a defense plan for these attacks. The remainder of this chapter is as follows: Section 5.3 presents the

related work, Section 5.4 develops the approach, Section 5.5 discusses solutions, and Section 5.6 closes with Conclusions and Future Work.

5.2 Background

There are three key background areas: Build, Defend, Attack [58] Systems from the Cyber Security Domain, the Sentiment Analysis, and black-box models.

5.2.1 Build, Attack, Defend Systems

Build, Attack, Defend [58] is a construct built around breaking down Cyber Security problems into actionable problem spaces for teams to digest. There are relevant team constructs around Color teams for attacking, defending, and designing systems:

1. Red Team: Ethical hacking of a target system
2. Blue Team: A group of people building defenses against the attacks on the model
3. Green Team: A blend of the red and blue teams where the group will simultaneously run both attacks and create defenses for those attacks [128].

The focus is on applying the green team construct to improve ML model development. By utilizing adversarial design knowledge (Red Team) and model building knowledge (Blue Team), the Green Team can propose defenses to the underlying model designs or production pipelines that secure the model from outside attacks.

5.2.2 Sentiment Analysis

The field of SA has progressed rapidly due to the expansion of social media platforms [70]. Among internet moderation applications, SA is a required piece of policing social media and comment sections due to the large volume of comments on these sites [129]. With state-of-the-art SA systems improving, it became evident that there is an unintended bias built

into the dataset of comments and tweets. Overfitting to words in bias dataset has led to embarrassing results for production SA [123]. The famous example with Perspective API is "I am a gay black woman" which carried a 95% toxicity in 2017 and still contains a toxicity score of 44% as of writing this chapter [72]. State-of-the-art SA papers are dominated by ensembles of transformer technologies for this particular problem set [73]. Each of the SA classifiers inherits weaknesses of the underlying language models used for the classification tasks [130].

5.2.3 Black Box Models

A black box model in ML is any model that an end-user only has access to inputs and outputs [131]. Each ML black box model in this work allows an end-user to interact with it through JSON inputs and outputs. There are also limitations to the number of queries per user per system. Two black box SA systems were selected in this chapter: Sentiment140 and Perspective API. Sentiment140 [124] is based on a technical report which collected 1.6million tweets to survey ML techniques in the SA domain in 2008. The Sentiment140 team has maintained the API as a historical benchmark for future SA systems but provides no explicit details on the exact implementation of the API. In contrast, the Perspective team provides toxicity scores for multiple categories through their API. With Perspective, an end-user can request classification probability scores for each class and can therefore evaluate the efficacy of each attack. The Perspective Team does not provide details on their ML models.

5.3 Approach

There are numerous areas of modeling where an adversarial actor can attack [132]. For simplicity, our focus is on inference-based attacks. Attacks on the inference pipeline exploit weaknesses of data used for training and learned weights of the model [133]. For example, there are simple attacks like substitution, replication, and insertion that easily fool current classification models [134]. A recent paper proved universal rules for fooling text-based

classification systems are effective for multiple tasks in NLP [126]. The Evaluation section demonstrates a BAD architecture focused on an inference-based attack for each API. Figure 5.1 shows the general flow of implementing the BAD architecture for an inference attack surface of a SA Model. The following sections discuss each of the BAD [58] core components in detail.

5.3.1 Holistic Approach: Introducing the BAD Architecture

Every ML team wants to understand the model's vulnerability to adversarial attacks [135]. The BAD Architecture proposes three key steps. First, a team needs to understand the baseline performance of the model by asking questions like the following:

1. How does the model act with regular and irregular data?
2. Are there known weaknesses or limitations?
3. Are those limitations and risks mitigated?

Next, a team needs to understand the impact of each attack by exercising each vulnerability in the model. Last, after understanding the baseline performance and attacking their model, the team will need to propose and implement those defenses to protect their production process. In practice, this entire architecture is repeatable and expandable depending on the scope of the team [136].

5.3.2 BUILD a baseline of our target system

A core component of a ML production system is to understand performance under normal conditions. With the BAD Architecture, each team should also note the known limitations of the model. For instance, some systems do not inherently return real scores for words not in the original training data (example: Word2Vec) [137]. A team must be upfront and understand the impact of design decisions on how a ML system has been designed. To baseline a ML system, it is also important to experiment with data that the system

is expected to operate on regularly. If possible, it is also expected to document any edge cases that would be hard for the system to classify. Using an SA black-box model with the JigSaw Toxic Bias dataset is a perfect example baseline case for the Build, Attack, Defend Architecture. The data contains toxic edge cases where it is hard to judge the intent of the underlying message. The advantages of creating a systematic baseline are shown in the Evaluation section when edge cases are exploited.

5.3.3 ATTACK System weaknesses and inefficiencies

Each API has a public page and allows anyone to sign up for basic services. Even with basic access, it is possible to circumvent these systems with a limited number of queries and the Python programming language. Adversarial Character Attacks [138] in the NLP field revolve around changing one or more characters while maintaining the original intent to a human annotator. Adversarial Attacks using character attacks [139] attempt to direct the decision boundary of the underlying detector in a way that is beneficial to the attacker. For a SA system, this would use substitution attacks to avoid the detection of negative or toxic comments in a social media environment. If bad actors understood how to substitute common character and reduce their toxicity, then it becomes easy for them to use hate speech (as an example). There are two areas in the character attack space applied here: substitution and duplication. The example Attack system demonstrates simple substitution attacks:

1. Create a dictionary that contains vowel to alpha-numeric (for instance e:3)
2. For every vowel in the sentence, replace a single instance from the string and store in an array
3. For every string in the array, evaluate sentiment through public-facing API
4. Evaluate the number of times a single character changed the score or decision made by the black-box model

And, for the duplication Attack, the same process is replicated with only a change in the attack vector:

1. Create a dictionary that contains vowels to duplicated vowels (for instance e:ee)
2. For every vowel in the sentence, replace a single instance from the string and store in an array
3. For every string in the array, evaluate sentiment through public-facing API
4. Evaluate the number of times a single character changed the score or decision made by the black-box model

Given the nature of black-box models, each API only provides the probability of toxicity or polarity without additional feedback. With Sentiment140, we are provided three states of polarity: negative, neutral, and positive. There are no percentages of each classification; rather the API simply provides the highest binary classification value. It is only possible to show if a classification can move from one category to another. With Perspective, the actual probability of each classification category is available for each request. Therefore, it is possible to see the decrease or increase in confidence for a given input. The Evaluation section shows the baseline and delta results for each of the attacks.

5.3.4 Hello World of the BAD Architecture

Figure 5.2 shows how the system will operate on the simplest incoming toxic phrase. In this example, the 'I hate people' example demonstrates the Build, Attack, and Defend pipeline. Applying the Perspective API, this string scores an 82% toxicity and negative score on the Sentiment140 system. When a Red Team attacks the model with a single character substitution attack of "a" to "@", the toxicity of the comment goes down to 30%. The Green Team's goal is to break apart each attack vector and create a more robust system against adversarial attacks. The Defend section will cover possible strategies for combating simple attacks.

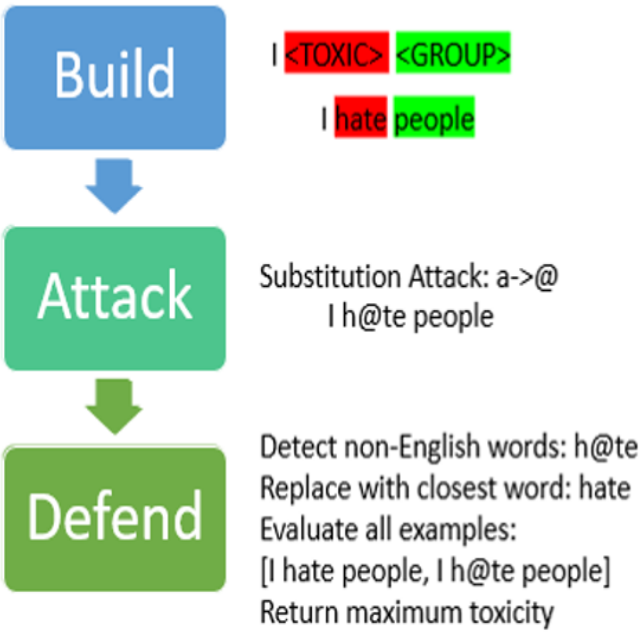

Figure 5.2: Hello World of Build, Attack, Defend Development Architecture

5.3.5 DEFEND System from Targeted Attack

The Green Team will summarize the baseline and adversarial results to create a defense plan. Typically, there are simple ways to mitigate attacks. For example, a back-end developer can create limitations around what types of requests can be made. Take the commentary systems on a forum: if they use the Perspective API, it would be straightforward to add a few rules to reduce the ability of an attacker to use substitution attacks (Evaluation section covers a basic implementation). In practice, there are the following crucial steps:

1. Detect: Detect and catch the adversarial text
2. Expand: Expanding the text to include the possible meanings of the originator
3. Defend: Process each result and store for future analysis
4. Evaluate: Check each result and return tune how the team wants the system to respond to attacks

This process cannot stay stationary. Bad Actors are constantly working to find new and inventive ways to break ML systems. The goal of this process is to create development architecture that can be deployed ML model development. In our Evaluation section, we focus on SA and the way we use this system to evaluate the Sentiment140 and Perspective API systems.

5.4 Evaluation

Each API provides the ability to send one query per second (1 QPS). There are limitations to the number of adversarial examples we could present to the Perspective API for instance which had a limit of 100 one-second queries. In this instance, a local model is trained and a model is attacked. Then, the potent attacks that fooled the local SA system are used against the black-box model. In practice, adversarial examples were drawn from the training set as a sample of the one hundred top toxic examples for each category of JigSaw Toxic Bias dataset. There is a section for Sentiment140 and Perspective where the Build, Attack, Defend Architecture is explained in detail.

5.4.1 Sentiment140

The Sentiment140 API has a large limit to queries (approximately 5000 items per query per second). There is a maximum limit of around 800,000 scored queries in a given time period (experimentally derived). Experiments are limited to a few permutations of substitution and insertion attacks per input row. In the Build section, the baseline results of the Sentiment140 model with the JigSaw Toxic Bias dataset are covered. In the Attack section, experimental results with simply applying substitution and insertion attacks are explored. The baseline experiments for the JigSaw Toxic Bias dataset will be paralleled between the two systems - choose one hundred samples from each toxic category with a 50% or above toxicity and then attack the classification of each toxic row. For Sentiment140, we are given binary results for each experiment. Every returned row provides a category of positive,

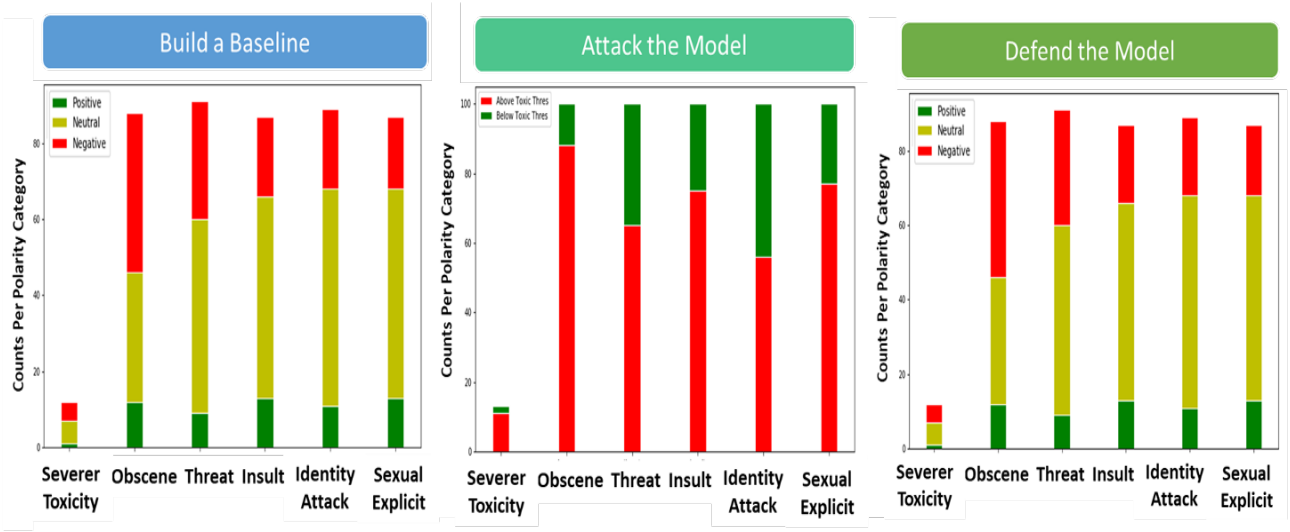

Figure 5.3: Perspective API Performance Summary using the Build, Attack, Defend Development Architecture

negative, or neutral. The baseline results are seen in Figure 5.3 in the Build Section. The classification binary score for these toxic comments is the majority in the neutral and positive categories. For example, 69 percent of the threat category is classified as either neutral or positive. As a note, each toxic input has been annotated by a human to include the label. Sentiment140 does still miss out on a large chunk of the proper classifications of negative for each one of these input rows. The next experiment will show how substitution attacks push the decision boundaries for this model in a different direction.

ATTACK: Results with Substitution Attacks

The substitution attacks had an interesting effect. The neutral classifications almost universally transformed into negative sentiment. The threat category, for instance, went from 31 percent negative to a 69 percent negative for the one hundred samples included here. In Figure 5.3, the massive change in the polarity is evident in each category. If the goal was to simply push all polarities to positive, then additional experiments with additional techniques would need to be explored. This work, demonstrates two things:

1. The Sentiment140 algorithm, with this open API, is not well equipped to deal with the bias inherent in modern social media commentary.

2. The decision boundary between neutral and negative is much closer than anticipated with simple substitutions changing neutral polarity into negative polarity.

Sentiment140 was never meant to work with social commentary with this level of toxic bias. Since these are black-box models, there is no opportunity to improve the performance of the underlying system. This highlights the core advantage of applying this architecture. With simple access, the underlying ML model can be evaluated and tested. For defense, both APIs are covered under the Adversarial Attack Surface Reduction section.

5.4.2 ATTACK: Results with Substitution Attacks

The Perspective API allows one text field per query per second. There is a limit to the number of daily queries but it was not a problem in the experiments. The first step is to create a baseline performance on 100 examples from each of the toxicity categories. Then, attack the same sample of 100 with character attacks (alpha-numeric and duplication) in each category to understand how much degradation can be introduced with simple character attacks.

BUILD: Baseline JigSaw Performance with Perspective

There are two separate experiments run during the baseline stage. A baseline of production models against the JigSaw Toxic Bias data is evaluated. This data is pulled as a sample, straight from the training set, with each toxic category measuring at 50% toxicity or greater for that category (same process as with Sentiment140). If the model were able to classify them appropriately, every one of the examples we pulled would be toxic (red). Our results, shown in Figure 5.4, show that there is still work to be done in terms of getting full coverage of even just the hundred selected training examples.

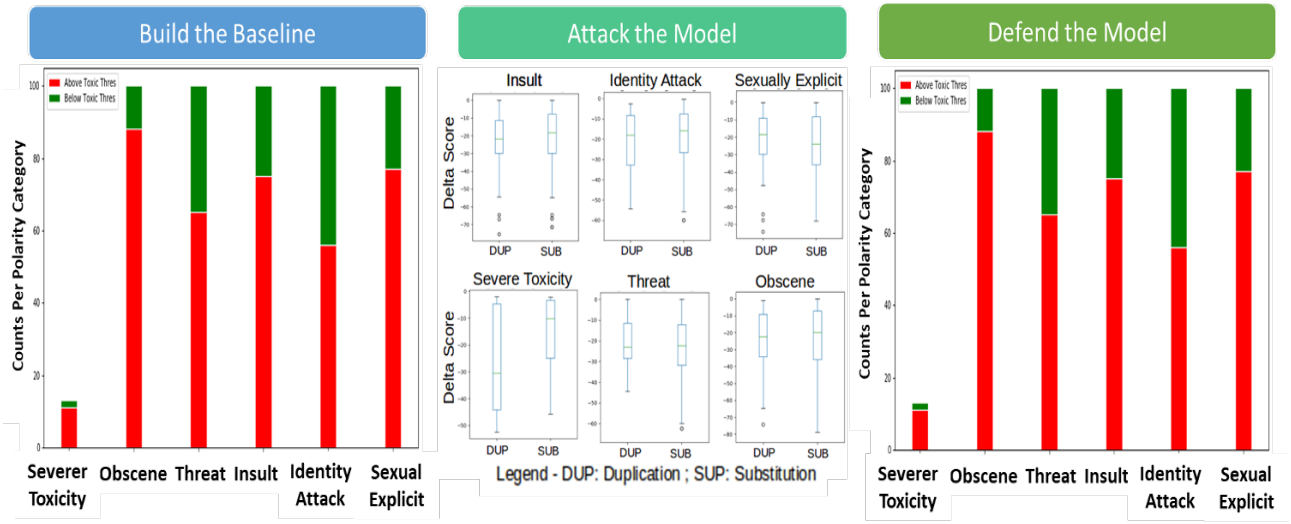

Figure 5.4: Perspective API Performance Summary using the Build, Attack, Defend Development Architecture

ATTACK: Results with Substitution Attacks

First, for every vowel, substitution and duplication attacks at each vowel position are applied. The Perspective API is robust to these types of simpler attacks, as the delta between the original score and the attacked score was less than 5% in the experiments. Only a single example went below 4% delta in its delta score. This portion of the attack space will need more exploration. One can think about the single character changed once in a string as a single-pixel attack. Only a single pixel is being changed throughout the entire picture. There is another approach to the single character problem - in this attack, consider changing the same character and replace or duplicate it at every instance. The analog to this attack would be changing a certain color pixel throughout the entire image. This exercise is left for a later date. The final experiment, highlighted in Figure 4, explores changing vowels to an alpha-numeric or simply duplicating them. Substitution and Duplication attacks are discussed throughout the literature as rudimentary but effective tools when surveying the adversarial surface of these models [22]. In this experiment, the focus was on vowel substitution and duplication attacks on the 100 samples on a per-category basis. In every case, there was at least one example of a -70% reduction in the toxicity score, effectively taking the sentiment

from toxic to non-toxic. By using the Attack Surface Reduction steps for our Defend step, it is possible to completely negate the original attacks demonstrated in the last two sections.

5.4.3 ATTACK SURFACE REDUCTION FOR BOTH APIs

For each of the substitution attacks, there are simple code changes offered for filtering each of these results. In fact, by utilizing these filtering techniques, it is possible to restore the original classification accuracy of the system. Unfortunately, this work does not focus on improving the black box models. The goal is to demonstrate vulnerabilities inherent in these systems and propose an architecture for production systems to protect the efficiency of their systems.

Attack Surface Reduction: Substitution Attacks

This work features two specific types of character attacks - replacement and duplication. The focus is limited to English in this effort although these methods should translate to other languages. First, for detection, the user can detect all non-English words. Multiple models will provide the nearest word or words in a corpus of available words. A simple Defend preprocessing pipeline before inference would be as follows:

1. Find all non-English words by using standard NLP libraries such as NLTK [26]
2. For each non-English word, find nearest neighbor words using algorithms such as Word2Vec [27]
3. Create an array of text with the non-English words replaced with the top N candidates
4. Evaluate the array of text and take the max or min score for classification

In practice, there are commonly misspelled words that can be safely ignored. Using this Defend pipeline can increase the number of candidates for inference but provides robustness to the attacks shown in this chapter. As another method, the systems can maintain a set of

common substitutions such as alpha-numeric substitutions using alpha-numeric characters or other simple dictionary lookups. Each detected "Attack" should be stored and evaluated by the Green Team periodically to ensure that the pipeline is working as designed.

5.4.4 Limitations

In each example, the attacks were simple character to character mappings. Zero-Day attacks in the cyber realm refer to attacks that are not yet protected against and allow a hacker unrestricted access to a system [30]. In the ML realm, there are adversarial 'zero-day' attacks that are manipulating the output of the model. These Zero-Day attacks are difficult to anticipate and protect against in practice. This architecture currently relies on known attacks on models for protections and does not actively search an adversarial surface for the model.

5.5 Chapter Summary

This work demonstrates that deployed sentiment models are susceptible to simple substitution attacks on single characters and can be effectively defended from each substitution attack using the BAD architecture. Because these substitutions are simple character to character mappings, they are mitigated by detecting non-English words, creating candidates for sentiment analysis, and taking the maximum toxicity in our examples. Further work in this area will focus on looking at model attacks like weight poisoning attacks on classification systems.

Weight Poisoning Attacks on Pre-trained Models [28] is a recent paper that uses vulnerabilities in pre-trained models and strikes me as dangerous to all black box models that are not actively defending against those types of tasks. A future direction could be to develop a data augmentation method or model structure that makes weight poisoning attacks reduces the efficacy of weight poisoning attacks, During the defend phase, automated methods for

detecting and correcting poisoned words could use transformer models to find and propose corrected word.

Chapter 6

Automating Defense against adversarial attacks: discovery of vulnerabilities and applications of multi-spectral imagery to protect deployed models

6.1 Introduction

Image classification is a common step in image recognition for machine learning in overhead applications. When applying popular model architectures like MobileNetV2, known vulnerabilities expose the model to counter-attacks, either mislabeling a known class or altering box location. This work proposes an automated approach to defend these models. We evaluate the use of multi-spectral images to combat adversarial attacks. The original contribution demonstrates the attack, proposes a remedy, and automates some key outcomes for protecting the model's predictions against adversaries. Similar to defending cyber-networks, we combine techniques from both offensive ("red team") and defensive ("blue team") approaches, thus generating a hybrid protective outcome ("green team"). For machine learning, we demonstrate these methods with 3-color channels plus infrared. The outcome uncovers vulnerabilities and corrects them with supplemental data inputs commonly found in overhead cases particularly.

Image classifiers use deep neural networks to derive features and learn class labels for each dataset. In recent literature, adversarial patch camouflage has been used to evade detection from machine learning-based detectors. These works use knowledge of the underlying model architecture to create specialized patterns and masks that perturb or erase the necessary feature patterns for the classifier. This work proposes the use of Multi-INT imagery to overcome adversarial attack patterns. For instance, a vehicle draped with an adversarial patch can provide a heat signature in a repeatable, recognizable pattern. A classifier that uses both visible and infrared-based channels can distinguish the vehicle in experiments.

Figure 6.1: Vehicle Detection in Aerial Imagery (VEDAI) sample data showing different vehicles in different orientations from an overhead visible sensor.

6.1.1 Vehicle Detection in Aerial Imagery

The VEDAI or Vehicle Detection in Aerial Imagery dataset is a benchmarking dataset with multiple channels looking at the same locations. The dataset includes nine vehicle classes that can be challenging to distinguish for computer vision applications. A critical aspect of this data is the rectified Infrared images included inside of this dataset. The original paper by Razakarivony and Jurie highlights the use of common machine learning applications to identify the classes inside of the dataset [100]. In this chapter, the goal is to extend previous work by protecting a common image classification machine learning model from adversarial attacks. Three primary channels are included in the dataset: visible, infrared, and gray. For the experiments included in this chapter, the Visible channel is also split into its principal components of red, green, and blue channels to explore the color dependence of the modeling.

6.1.2 MobileNetV2

MobileNetV2, Inverted Residuals, and Linear Bottlenecks is a recent architecture improvement to the MobileNet design [85]. The newest paper offers greater accuracy vs. speed and accuracy vs. size comparisons to other state-of-the-art models. Tensorflow and other machine learning libraries offer methods for training and shrinking the models down for deployment on Single Board Computers/ASICs. This model is an ideal benchmark for exploration in the overhead imagery world given its superior accuracy, size, and compute requirements.

6.1.3 Adversarial attacks

Adversarial attacks are attacks on machine learning models that can manipulate or avoid detection. These algorithms are typically divided into two types: white-box and black-box attacks. With white-box attacks, the attacker knows the underlying training dataset and model architecture. Conversely, with black-box attacks, the attacker does not know any information about the underlying machine learning models. Universal black box attacks are rare, especially those that can work across multiple model types in a particular machine learning field. For the experiments conducted here, the focus is on using White box attacks with a library called FoolBox. FoolBox uses the base machine learning model and dataset to create tailored attacks to that model [15].

The FoolBox library has access to state-of-the-art adversarial attacks which including six types used in the experiments:

- FGSM: Fast Gradient Sign Method [3]
- FGM: Fast Gradient Method [140]
- PGD: Projected Gradient Descent [141]
- PGD (LinfPGD, L2 PGD): Linf/L2 Projected Gradient Descent [142]

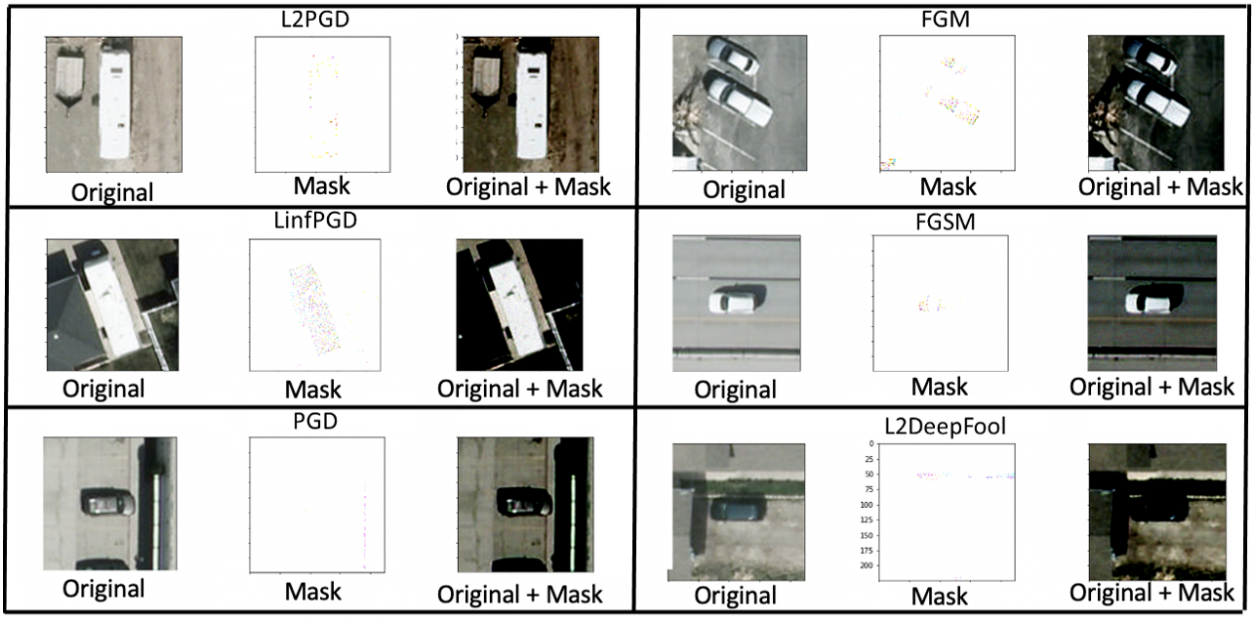

Figure 6.2: An example of the mask generated by Foolbox for each of the attack types. For each attack, on the left is the original image, the middle is the attack mask, and the right is the combination of the original and masks.

- L2DeepFool [143]

6.1.4 Securing Deployed Models Against Adversarial Attacks

Recent papers focus on securing models during the development process. Green team machine learning creates a process called "Build, Attack, Defend" to evaluate the machine learning models during the development process and begin protecting against red team style attacks on the models [144]. Attacks are increasingly sophisticated with their ability to detect the underlying model architecture and therefore, exploit vulnerabilities in these models. Decision-making applications, like Automated (or Aided) Target Recognition (ATR), requires the ability of developers to reduce adversarial risk in their systems. Using a "Build, Attack, Defend" development process can lead to more secure models before deployment into production.

6.2 Methods

Automating protections for machine learning models involves a few steps. First, an automated evaluation pipeline should be created to evaluate the performance of the models against adversarial attacks. The methods section will cover the model types and adversarial attacks, the evaluation framework, and simple combinations of Multi-INT imagery to protect the models. "Build, Attack, Defend" systems for adversarial defenses are a primary inspiration for this work. The experiment creates baseline models, evaluates those models against a suite of adversarial attacks, and then proposes changes to defend the models from incoming attacks. A tiered protection system is introduced to segment the protections needed for the models.

To understand model heritage, the experiments must observe the underlying training data and the model architecture. There are two primary questions:

- Is the deployed model trained on custom or novel data?
- Is the model trained from common datasets found in open source?

The concern with "download and deploy" strategies are that an attacker already has access to the architecture and the weights of the model – these types of attacks are called white-box attacks. It is much easier to construct attacks when the adversary has access to this information. In contrast, without access to the model or data (black-box attacks), the attacker will construct generic attacks and probe the model. With image classification, it is much more difficult to get feedback on what objects are being classified and tracked when there is no direct access to the model. In black box to white box, Ciolino et al. (2021) were able to detect the underlying learner architecture above 95% confidence [145]. Once a model type like MobileNetV2 is detected, a set of structured attacks can be evaluated on the models. The experiments included in this work will only focus on covering white-box attacks.

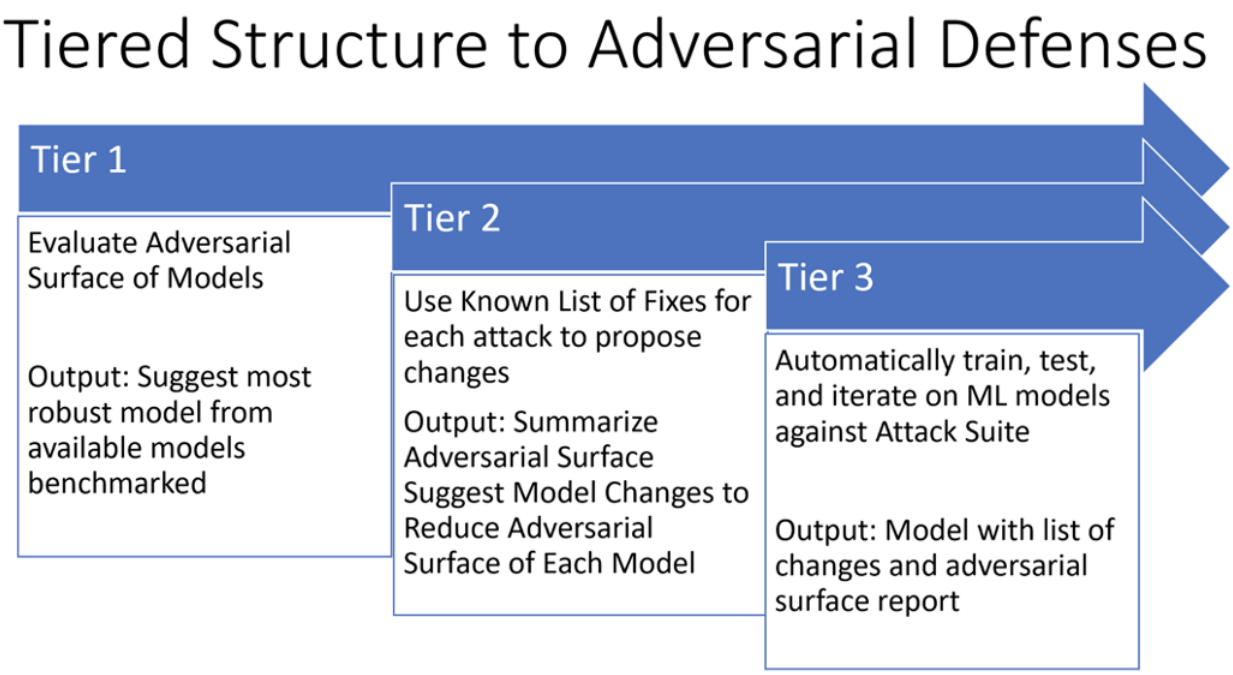

Figure 6.3: A tiered structure approach to adversarial defenses for machine learning models. Each tier represents additional work to protect the model from outside attackers.

The evaluation framework contains a workflow seen in Figure 6.3 that provides an adversarial attack surface for a given model. There is a static set of modified images that are provided to the model and the detections are provided back for evaluation. Each set of images probes different adversarial susceptibility for the model. The product from this step is a report that details the susceptibility of the model to specific adversarial attacks – the adversarial surface. This is visualized with a Sankey diagram to highlight overall vulnerabilities for an analyst.

After the adversarial surface is identified, the last tier is to retrain the model to reduce the efficacy of those attacks. There are three tiers currently in development for this automated exploration system. Each tier represents a way to secure the model. With every additional tier, there is a degree of complexity to protecting the model. Tier 1 protections involve evaluating a set of models and providing recommendations on which models are least susceptible to attack. Tier 2 takes the models and benchmarks them against a wide range of attacks to understand vulnerable areas and provides a report back on the adversarial surface

of the current model. Finally, Tier 3 protections will automatically do architecture search and parameter tuning to reduce the overall efficacy of the test quite until user-defined thresholds are met. Recent work presented in adversarial camouflage demonstrates an evolving set of threats for the overhead machine learning [146]. This proposed tiered protection system provides a structured approach to vetting and fixing known issues. This work focuses on designing a system for aerial imagery that can provide Tier Level 1 results for Automated Protection of these machine learning systems. The underlying AI system will recommend which models provide the best robustness to adversarial attacks between all the methods and channels surveyed. The advantage of this exploration is that many overhead systems will have access to at least one visible channel and one additional imagery channel for detection. In Tier 2 experiments, this work demonstrates the adversarial surface of the models trained.

6.3 Results

The Tensorflow-Lite library offers simple MobileNetV2 training classes for easily building, benchmarking, and deploying classification models on low-end computing devices [147]. Using this library, each model is trained on an individual channel (VIS, Gray, IR, Red, Green, Blue) for 20 epochs with a batch size of 32. The images are 224 by 224, which is a typical training size for deep classification models.

MobileNetV2 is a general use case tool for deep classification tasks and the parameters can be tuned to provide the best performance. For this exploration, the experiments minimized the number of perturbations to keep a level playing field between the different models. Certain hyperparameters would provide advantageous against adversarial attacks. These ablations are left for future work on aerial imagery.

6.3.1 Training and Test Accuracies

For the MobileNetv2 architecture in Table 1, the training accuracy for the models hovers right around 90% for each of the channels and test accuracy is at the 80% mark. Delta

Figure 6.4: Randomly selected samples from the dataset with predictions from the Visible Model. The black predictions are correctly classified images and the red predictions are the incorrect predictions with the selected class.

accuracy for each model is not more than 5% offering similar performance between channels for the adversarial exploration.

Channel	Training Accuracy	Test Accuracy
Visible	90.5%	76.4%
Blue	88.1%	78.0%
Gray	88.7%	77.0%
Green	90.2%	79.8%
Red	88.7%	78.9%
IR	87.5%	73.6%

Table 6.1: Training and test accuracy for individual channels

6.3.2 Attack the models

The goal, in this section, is to understand the susceptibility of a model to attacks generated specifically for it or by an attacker who used a different channel. For instance, if an adversarial attacker used visible imagery as input to their attack generator, would the same attacks translate to IR? Is IR more robust against attacks generated by certain channels? This analysis explores the ability of an attack suite like Foolbox to create adversarial image

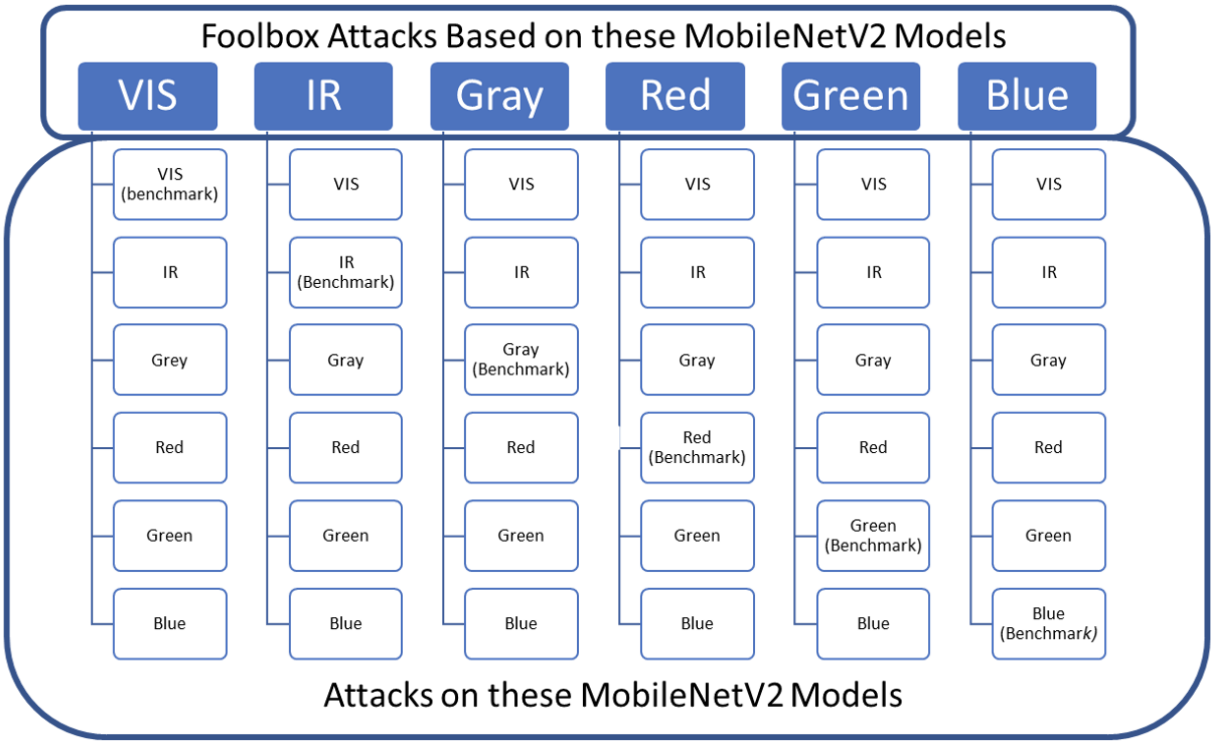

Figure 6.5: Experimental Design for Foolbox Attacks on VEDAI dataset. A Foolbox Attacker is trained on a single channel. The Foolbox model generates attacks for each channel and the adversarial efficacy is measured as a delta accuracy from original model accuracy to attacked model accuracy.

generators that can attack both their matched channel and the other models from the same images.

6.3.3 Attacks Trained on a single channel and then used on all downstream models

The general process for evaluating the adversarial surface has the following steps:

1. Select a Benchmark Channel for the Attacks to be based on (White Box Attack)
   (a) For example, let's select the Visible channel to train Foolbox.
2. Use the Adversarial Attack Generator to Attack the Benchmark Channel

(a) The Visible Adversarial Attack Generator generates poisoned images and feeds them into the Visible MobleNetV2 Model

(b) Measure the delta accuracy between the original inputs and poisoned inputs

(c) This is the benchmark data for Step 3 comparisons

3. Use the Adversarial Attack Generator to attack the other channels

   (a) For each channel in [blue, green, red, gray, IR]

      i. The Visible Adversarial Attack Generator generates poisoned examples of the channel and feeds them to the channel model

      ii. Measure delta accuracy from original samples to the poisoned examples run through the selected model

4. Measure the Adversarial Surface of each model for each attack

This process provides a way to quantify the ability of each classification model to handle adversarial attacks from the benchmark and attacks that were trained on similar data.

6.3.4 Observations

The output of the adversarial surface evaluation has two primary visualizations – the overall surface and the per attack evaluations. Each one of these graphics shows the effectiveness of an attack. FGM and FGSM attacks can be mitigated through the appropriate selection of a model. For instance, a model trained on IR is resilient to attacks from FGM, and FGSM trained attackers on any of the other channels. Figure 6.6 shows the adversarial surface of the MobileNetV2 trained models against the Foolbox attackers.

6.3.5 Adversarial Transferability between channels

Transferability between the different imagery channels suggests that different colors learn separate features within a given architecture. For the visible images, their network

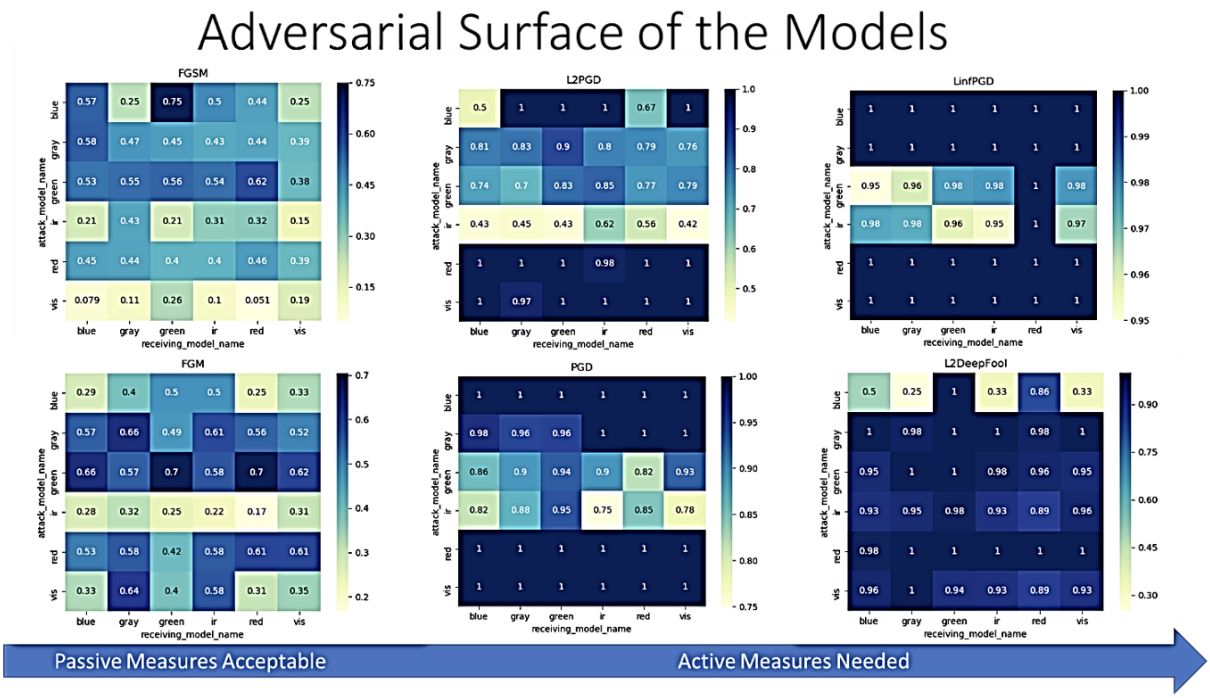

Figure 6.6: Adversarial Surface for MobileNetV2 Models trained on VEDAI dataset. Passive Measures included recommending a particular channel as mitigation. Active Measures require augmentation, architecture changes, and other changes to the modeling process to overcome the attacks.

will also learn individual features from each channel and use those features to distinguish between classes. For the single-channel training, the model will only have access to a single channel's features and will learn different weights. In the experiments, there are a few notable outcomes:

- Attacks trained with visible images and FGSM show the least reduction in model efficiency
- FGSM and FGM had the least successful attacks
- Four of the six attack types resulted in 100% detection in classification accuracy for most channels

The four attack methods that require active measures in the development process are left up to future work on this topic.

6.3.6 Recommendations from Adversarial Surface Data

The passive measure attackers can be mitigated through recommendations on available models evaluated. Because there are six available models, it is possible to use the adversarial surface to create a set of recommendations for which model should be used to protect against these types of adversarial examples. In this evaluation, FGSM and FGM are identified as attack sets that can mitigate by recommending an image classifier trained with the IR band.

6.4 Extending Work from this chapter

Future work will focus on automating an end-to-end pipeline from dataset to model evaluations to a more robust model. Each of these individual pieces alone can protect a model. An automated system would optimize the model for maximum robustness to external attackers. There are trade-offs to a more robust model including complexity, accuracy, and explain-ability. A sample trade-off analysis can be included with the results.

6.5 Chapter Summary

This work demonstrates the ability to protect machine learning models for overhead imagery by simply using a structured approach to evaluate and reduce the adversarial surface of the machine learning model. The experiments demonstrate a reduction in the efficiency of adversarial attacks while maintaining original performance benchmarks. One critical issue to address from this work is how to reduce the computational burden of retraining the networks for every additional adversarial perturbation? In addition, further automation to protect models using the adversarial attack surface will be included in future works.

Chapter 7

A Modified Drake Equation For Assessing Adversarial Risk To Machine Learning Models

7.1 Introduction

Machine learning models present a risk of adversarial attack when deployed in production. Quantifying the contributing factors and uncertainties using empirical measures could assist the industry with assessing the risk of downloading and deploying common model types. This work proposes modifying the traditional Drake Equation's formalism to estimate the number of potentially successful adversarial attacks on a deployed model. The Drake Equation is famously used for parameterizing uncertainties and it has been used in many research fields outside of its original intentions to estimate the number of radio-capable extra-terrestrial civilizations. While previous work has outlined methods for discovering vulnerabilities in public model architectures, the proposed equation seeks to provide a semi-quantitative benchmark for evaluating and estimating the potential risk factors for adversarial attacks.

This chapter explores a simple version of a probabilistic equation for machine learning (ML), specifically to defend trained models from adversarial attacks. Probabilistic frameworks like the Drake Equation (predicting the number of alien civilizations [148, 149]) offer a heuristic for traditional factor analysis, particularly helpful in the face of uncertainty. In that spirit, we adopt the basic formalism of its original population-based assessment by including both the leading factors and their corresponding loss fractions. The goal is less to provide a new ML risk model as much as to explore the practical factors needed to consider before fielding a new model [150]. Other work has detailed already much of the successes and failures in defending predictive models: 1) diversifying or augmenting training data [151] 2) ensembling and model voting [152] 3) obfuscating [153] 4) active learning [154] 5) protecting

model descriptions or firewalling (e.g. nation-state security) [155] All these strategies (plus many others not included here [150]) should appear in one or more proposed terms for this modified Drake Equation.

7.2 Modifying the Drake Equation

The format of this chapter is first to present the main factors for model defense, followed by an explanation and examples to support each included factor. We explore the interpretation of each factor where possible with an illustrative case mined from the AI Incidents database [156]. It is worth noting that other than these early efforts [156] to catalog adversarial attacks, less research has previously attempted to count or quantify systematically the failures of a given machine learning model in the wild. For example, should an adversarial attack be scored based on its frequency, severity, or difficulty to patch once a vulnerability gets discovered? This paucity of data further motivates the development of a modified Drake Equation, principally as a heuristic framework for understanding contributing factors and assessing their uncertainties. The structure of the chapter isolates each factor in reference to its more familiar population-based input, so for instance, the time that an attacker might probe a model's vulnerabilities maps to the original Drake Equation's reference to the time that a radio-aware civilization might broadcast its identity. An appealing aspect of the original format stems from its hierarchical factors from large to small fractional contributions as they change over time. One ultimately wants to understand the dependencies while solving for the machine learning model's attack surface, as measured by N, the number of successful adversarial attacks.

To defend a machine learning model, the number of successful adversarial attacks, N, is proportional to the model's size, R, as measured by its popularity (e.g. YOLO), sponsoring enterprise size (e.g. Microsoft, Google, Facebook), or monoculture of adoption (e.g. convolutional neural networks). The proposed modifications to the Drake Equation are described below:

$$N = R * f_p * n_e * f_l * f_i * f_c * L$$

N = the number of successful adversarial attacks

R = average enterprise size

f_p = fraction of models published, named, open sourced, or fielded in the wild

n_e = average number of engineered parameters (memory,billions of parameters)

f_l = fraction of learning ratio,as training/test data or active hybrid feedback

f_i = fraction of input supervisory and quality control steps

f_c = fraction of completed queries that return detectable or logged answers

L = length of time that attackers can query without consequences or timeouts

7.2.1 R - Average Enterprise Size

In the original Drake Equation, this factor traditionally relates to a rate of new star formation. We generalize the rate of new ML models created, R, by an aggregate of overall enterprise size. This approach mirrors the literature on monoculture in computer operating systems (e.g. MS Windows) as a primary indicator to motivate cyber-attacks. The corresponding figure in defending ML models derives from a similar feature, namely that attacking large enterprise models like Google's Perspective API and OpenAI's Generative Pretrained Transformer (GPT-3) is more likely than probing or disabling a smaller, private, or novelty ML model.

One can hypothesize that the community's attraction to leader boards [156] and state-of-the-art (SOTA) competitions further drives the ML community to more singular ecosystems that may prove more difficult to defend from adversaries than a diversified one. As a figure of merit when describing the cyber-risks for a monopoly in operating systems [157], the entire ecosystem may become unstable when the market share and global adoption reach 43%

and more directed attacks garner hacker's attention. One ML-specific metric of whether a given type of neural network dominates its ecosystem can be approximated by search trend monitors. For example, by using Google Trends [158], the current popularity of three core approaches to modeling the neural architecture itself shows that convolutional networks (CNN) capture 72% market share, compared to graph neural networks (25%) and capsule networks (2%). An attacker that knows the unique weaknesses of CNNs (such as their inability to grasp long-range spatial relations. complex hierarchies, and symmetries [**?**, 159]) may profitably attack those specific design elements, particularly given their monopoly as deployed models.

7.2.2 f_p - fraction published, named, open-sourced, or fielded in the wild

In the original Drake Equation, this first factor in a hierarchical loss fraction stems from the number of stars with planets. In an adversarial attack, this factor similarly appears at the top of the hierarchy, namely how much is known about the model's origins. The literature spans model security from black-box (no knowledge) to white-box (full-knowledge), such that given a known or discoverable model structure, the attacker may also potentially know the weights and training dataset. This is most well-known in the release of GPT-2 versus GPT-3, where for some time the GPT-3 API was not available to researchers. When Open AI initially open-sourced its models, the company furthermore specifically withheld its larger one (1554M) to suppress the possibilities for abuse.

7.2.3 n_e - average number of engineered parameters

In the original Drake Equation, this second factor considers the number of planets capable of supporting life. In an adversarial attack, the relevant context would include model complexity, either as its memory, number of parameters, or layers of network architecture. A breakdown of computing ne for a CNN could be as simple as a baseline of the number of parameters or number of layers. For object detectors, the relevant complexity often arises

from the way the model searches for its characteristic anchor sizes and bounding boxes, whether multi-stage like a region-proposal network (R-CNN) or single-stage frameworks like YOLO.

7.2.4 f_l - fraction of learning ratio

In the original Drake Equation, this third factor refers to planets that spawn life at some point. In an adversarial attack, this fraction includes losses for well-trained models that possess large and diverse data. Previous work has proposed using Akaike information criterion (AIC) and Bayesian information criterion (BIC) to evaluate model performance against dataset diversity and this style of metric may provide a baseline for this factor [160].

7.2.5 f_i - fraction of input supervisory guidance

In the original Drake Equation, this fourth factor addresses the rise of intelligent life from more primitive forms. In the machine learning context, this fraction includes the standard quality checks that separate a fieldable model from an experiment or lab bench demonstration. This factor corresponds to the breakpoint in many adversarial defenses, such that a prototype moves into production based on disciplined quality checks. Has the model seen out-of-vocabulary terms if a natural language processor? Is there a high fraction of augmented examples for each class? One traditional image approach augments training data with more diverse object types, usually including different lighting, viewing angles, or noise. Paleyes, et al. [161] describe 38 factors attacking 15 systems that contribute to a failed productization of an ML model. At any one of these steps, ranging from data-collection to performance monitoring, there exist adversarial attacks that can poison the entire process. Wang et al. [9] define in detail the adversarial attack to each of these systems.

7.2.6 f_c - fraction of completed queries that return detectable or logged answers

In the original Drake Equation, this fifth factor delineates the rise of technological capabilities such as radio transmission that travels at the speed of light and thus renders a distant galaxy observable. For adversarial attacks, this fraction defines the likelihood that an outside observer can understand the model type, its sensitivities, or its vulnerabilities. Particularly in the black-box approach where an attacker must launch a question-and-answer format to understand how the model works, this fraction restricts the obtainable universe of effective attacks. In experiments for text and image classifiers, Kalin, et. al [145] found that model architectures are easily discovered with strategic probing if the architecture is public. In this new equation, fc is related proportionally to fp factor.

7.2.7 L – Length of time that attackers can query without consequence or timeouts

In the original Drake Equation, this final factor introduces the notion of time, particularly how long a civilization might survive its technology before self-destructing or its evolutionary time to propagate signals to an outside observer. Like the numerical count of accepted API requests (fc), the length of time to automate or web-scrape the API with new queries offers a secondary line of defense not in space (count) but in time. Despite a more mature field, software engineering for APIs still suffers from vulnerable code being written into production systems [162].

7.3 Missing but not forgotten

This modification of the Drake Equation focuses on metrics that can be directly measured in a production environment. Missing elements in this heuristic might include additional pre-production factors for diversity, size, and quality of the input data, training lengths (epochs), and other historical elements that may or may not propagate usefully to the final model and its vulnerabilities. The collected metrics can then be used to refine

the model performance against known benchmarks. For instance, common model types are easily discoverable via their input data and/or architecture [145].

7.3.1 Axiom 1: Architecture and Dataset Metrics are related

The Learning Ratio, Parameters, and Guidance variables stem from the architectural design of the model. This equation is divided into two primary Adversarial fractional components: Architecture and Dataset. For teams to use the likelihood of successful adversarial attack assessment to improve their models, they will need to understand the contribution of architecture and dataset design to the overall adversarial risk. The first fraction defines the key parameters related to architecture and their overall contribution to adversarial risk:

$$\text{AdversarialFraction}_{\text{architecture}} = \frac{f_p * n_e * f_l}{N}$$

The second fraction defines the dataset metrics responsible for dataset contributions to adversarial risk:

$$\text{AdversarialFraction}_{\text{dataset}} = \frac{f_i * f_c * L}{N}$$

7.4 Experiments

This framework is designed to work on large and small works. In the following experiments, the focus is on baselining the effective ranges of the factors, showing sample risk factors for common model architectures, and understanding the relative effect each factor has on itself and the risk factor.

7.4.1 Experimental Design

Each factor needs to be defined in terms of operating bounds to apply this new framework to current models. For the experiments, the following operating ranges for the variables

were chosen to highlight the current capabilities that exist within the machine learning community today:

- R - Average Enterprise Size
  - Range [0, n authors]
  - Enterprise Size is computed as the Number of Authors as it can be difficult to find the actual number of employees in a particular organization
- f_p - fraction published, named, open-sourced, or fielded in the wild
  - Three values: Not Published 0.0, Published but not open source 0.5, Published and Open Source 1.0
  - For example, GPT-3 is fielded in the wild but is not open source: 0.5
- n_e - average number of engineered parameters
  - Stepped Range [0,1] based on Number of Model Parameters
- f_l - fraction of learning ratio
  - Stepped Range [0,1] as a relative factor to State of the Art (SOTA) performance
  - For example, the first benchmark in the model category is 0.1 and SOTA is 1
- f_i - fraction of input supervisory guidance
  - Range [0,1]
  - Is training data sufficiently large and diverse?
- f_c - fraction of completed queries that return detectable or logged answers
  - Range [0,1]
  - Estimated High Query Rate on the model

- L – Length of time that attackers can query without consequence or timeouts
  - How long has the model been in public? Years [0, n]

As with the original formulation of the Drake Equation, each parameter represents an estimate of best guesses for factors in the wild. This modification to the Drake Equation will provide organizations the ability to benchmark, evaluate, and track the adversarial risk of their models in production. As a team observes the adversarial risk reduction on their model, there are factors within this equation that can directly be attributed to that reduced risk.

7.4.2 Empirical Results

Using the factor ranges described in the experiment design, six popular models were estimated as samples of how to apply this formulation. Figure 7.1 is sorted from top adversarial risk to lowest risk. In the example of 'MyModel', the model is not deployed and therefore does not contain adversarial risk from outside actors.

When exploring this formulation, it's incredible to see that newer, larger architectures are less vulnerable than older models. This is on purpose though as older models will have more vulnerabilities appear since they have been in circulation longer. There are further improvements that could be made to these experiments – for instance, the exploration of

Model	R	F_p	N_e	F_l	F_i	F_c	L	A_a	A_d	N
T5	9	1	0.8	1	1	1	2	0.50	1.25	14.40
VGG19	2	1	0.6	1	1	1	6	0.17	1.67	7.20
GPT3	31	0.5	1	1	0.75	0.5	1	2.67	2.00	5.81
BERT	4	1	0.6	0.75	1	1	2	0.50	2.22	3.60
FastText	4	1	0.1	0.7	1	1	4	0.25	14.29	1.12
MoibleNetV2	5	1	0.1	0.5	0.5	1	3	0.67	20.00	0.38
MyModel	1	0	0.2	0.75	0.2	0.05	1			0.00

Figure 7.1: Summary of Models Explored with Modified Drake Equation. Six Popular model architectures are benchmarked along with a custom model based on MobileNetV2's design. The table is sorted by estimated Adversarial Risk N in the last column.

architectures could be split into text and computer vision. Each category of model architectures can have its boundary conditions. For instance, transformers technologies like BERT and GPT have revolutionized NLP problems over the last few years. Their properties may warrant a deeper exploration of parameter dependencies.

7.4.3 Correlation Analysis

The next experiment in this work is to understand the dependency of each factor on adversarial risk. Building a correlation matrix using the assumptions above, Figure 7.2 shows the relative importance of each factor to itself, the other factors, and to adversarial risk.

Within Figure 7.2, there are a few surprising things that come out of the correlation analysis.

X-Correl	R	F_p	N_e	F_l	F_i	F_c	L	N
R	1.000	-0.061	0.619	0.305	0.049	-0.083	-0.439	0.116
F_p	-0.061	1.000	0.034	-0.063	0.788	1.000	0.606	0.406
N_e	0.619	0.034	1.000	0.848	0.428	0.018	-0.240	0.669
F_l	0.305	-0.063	0.848	1.000	0.439	-0.073	0.038	0.735
F_i	0.049	0.788	0.428	0.439	1.000	0.783	0.482	0.599
F_c	-0.083	1.000	0.018	-0.073	0.783	1.000	0.611	0.404
L	-0.439	0.606	-0.240	0.038	0.482	0.611	1.000	0.145
N	0.116	0.406	0.669	0.735	0.599	0.404	0.145	1.000

Figure 7.2: Cross-correlation of variables to the Modified Drake Equation including Adversarial Risk

Here are the key observations:

- The most correlated variable when predicting adversarial risk is fraction of the learning ratio (0.735)
- The fraction of learning ratio is highly correlated to the number of parameters

$$p(f_l, n_e) = 0.848$$

- The fraction of input supervisory guidance is highly correlated to fraction published (0.788)
- The fraction completed queries is highly correlated to fraction published

$$p(f_c, f_p) = 1.000$$

Intuitively, the fraction of learning ratio being most correlated to adversarial risk represents that the most popular models have the most people trying to attack them. The goal is to track and reduce the adversarial risk to a model and this framework provides a starting benchmark.

7.5 Summary and Future Work

This work supports an established heuristic framework in analogy to the traditional Drake Equation. This simple formalism amounts to a summary of relevant factors. The basic equation has been modified elsewhere for detecting biosignatures in planet-hunting (Seager equation [163]), sociology (best choice problem [164]), infection risks [165], AI singularity [166], social justice [167], and other diverse probabilistic assessments [168]. Ultimately, its main purpose follows from assessing the multiple uncertainties that may vary by several

orders of magnitude. For example, as ML builders consider whether to privatize or to open-source their models, they may intuitively favor one course over another given a perceived risk for model compromise. Is it true in practice that privatizing a model lowers the risk, or does it increase the attack surface because the model never gets hardened by peers? One would like to provide a framework for these important decisions and assist the ML community to identify the data needed for sensitivity analysis and the evaluation of consequences.

The biggest challenge in finding novel utility for this framework shares much in common with Drake's original notion. How to quantify each factor? What if the factors show strong correlations? How do the factors change with time, particularly if both the builders and attackers modify their behavior? What are the appropriate units to assess ML risks, either as the number or severity of adversarial attacks? One informative output that previous technical papers often ignore in assessing model risk is the scale of the overall ecosystem (R). In the literature for cybersecurity, for example, the monoculture aspect for operating systems has proven most predictive of the next generation's attacks. In this view, the SOTA leader boards [156] might benefit from encouraging a more diverse model ecosystem, such that niche YOLO attacks cannot propagate throughout the whole ML community and its applications, particularly when a few fractional percentage improvements separate the field into universal adoption strategies. Future work should highlight the data sources for evaluating each factor. For instance, the publications dataset from Cornell's arXiv supports extensive topic analysis for extracting the popularity of ML models, their relevant attack methods, and promising defensive styles [169]. Classification methods for attack types [170] may also guide the practical counting or scoring for the universe of adversarial ML threats.

Chapter 8

Using Unstructured Text Narratives to Correct Work Unit Codes in the Wild from Collection to Decision

8.1 Introduction

The lack of accurate Work Unit Codes (WUC) in aircraft maintenance records hinders logistical improvements given current practices of manual, free-text narratives which routinely suffer from natural language challenges, misspellings, and undocumented flight line jargon. This work uses recent advances in machine learning (ML) to score the translation of unstructured text narratives against ground truth data to correct WUCs in the field. Throughout this evaluation, a set of experiments compares dictionary development of predictive keywords, statistical feature generation, and modern word embeddings pre-trained on a much larger text corpus to evaluate effectiveness of these methods on free form text from 10+ years of maintainer-entered records. Given the diversity of the labels, we define and implement a two-stage decomposition of the WUC using 2-character prediction, followed hierarchically by further sub-system and location finders. Where possible, meaningful, or explainable rules are extracted to guide future WUC implementations. This work focuses on a dual delivery method – as a production application programming interface (API) for integration into end user clients and production deployment in a multi-customer Business Intelligence solution. This methodology supports maintainers from data collection to maintenance leadership decision making.

Aircraft fault narratives entered by flight line managers offer valuable operator history of inspections, maintenance, and fleet readiness metrics. As a standard part of Department of the Army (DA) Form 2408-13-1, Aircraft Inspection and Maintenance Record, these manual entries report maintenance, repair, removal, and replacement data on helicopter systems,

subassemblies, and components. The fault narratives are often the only record from which to infer important logistical indices for aviation reliability, availability, and maintainability (RAM), which in turn affects all aspects of tactical logistics operations including metrics for mission, cost, and safety. Natural Language Processing (NLP) methods have brought notable successes when applied in other fields for text classification problems. One challenge in applying these advanced methods to fault narratives stem from their multi-step catalog of unique errors: misspellings, repair ambiguities, undocumented abbreviations, inconsistent jargon, and record redundancies.

The Work Unit Code (WUC) correction problem affects past, present, and future RAM logistics and manpower maintenance hours assigned to each system fault. This dual problem of classifying unstructured text and recoding ambiguous or incorrect WUCs thus frames the current work to provide predictors or decision variables that might guide automated WUCs that match best, relevant practices. Automated Coding systems are heavily researched for the medical field and represent a significant body of research to pull from. The approach to this problem is unique in that the models are developed at the system level – each model is optimized for the text narrative in that system's domain.

8.2 Background

The military uses a structured, hierarchical mixed character code to describe the maintenance and work done on an aircraft, referred to as a WUC [171]. In the Army, WUCs are standardized to individual aircraft weapon system documents called Legitimate Code Files (LCF) [171]. From these documents, organizations customize their WUC structure to align with any unique maintenance and activities they perform that are over and above the standard conventions for that aircraft weapon system. There are two key components to understand here - the WUC and the logistics information systems (LIS) that collect this data. The Army collects aircraft maintenance and logistical information via a LIS known as Aircraft Notebook (ACN), a transitional system from the Unit Level Logistics System –

Aviation (ULLS-A), to the Global Combat Support System – Army (GCSS-A) [172]. We discuss these domain-specific areas first, followed by a discussion of Automated Coding with ML in other domains, and conclude with a brief background of the Automated Machine Learning (AutoML) methods used in this work.

8.2.1 Army Aviation Maintenance Work Unit Codes

The Department of Defense MIL-STD-780G(AS) prescribes the application of the WUC system and method of implementation for identification of, and reference to, maintenance and logistics touchpoints on a major weapon system. WUCs serve as hierarchical codes, moving from macro to micro levels, for the overall piece of equipment, its components, and their subassemblies [171]. As a relevant, discrete example, the aviation platforms modeled herein all begin with two-character, system-level WUCs: 00-aircraft, 02-airframe, 03-landing gear, 04-powerplant, and so forth. To notate a higher level of fidelity or granularity, characters are increased out to a full 11-character representation, in some cases. For example, the WUC 04A01C09G represents the 7th disk compressor on the helicopter engine as it moves from 04-powerplant installation to 04A-gas turbine engine, assembly through 04A01-compressor section to its most granular state as an individual compressor disk [172].

8.2.2 Army Aviation Maintenance Data Collection

The Army's prescribed methods for collecting and logging maintenance and logistics activities on aircraft weapon systems center on the use of digitalized DA Form in the 2408 series [172]. The primary forms used to document fault narratives and corrective actions are the DA Form 2408-13-1, Aircraft Inspection and Maintenance Record, and its supplemental DA Form 2408-13-2, Related Maintenance Actions Record. The former documents the initially discovered, or prescribed, maintenance event while the latter serves as the detailed register of individual maintenance activities required to satisfy the inspection and/or correct the fault annotated on the DA Form 2408-13-1. WUCs are documented in both forms;

however, they may not necessarily fully match one another in so much that the DA Form 2408-13-1 captures the initial overall inspection or fault area while the DA Form 2408-13-2 further elaborates on what took place and may involve touchpoints beyond the initial scope of the effort. For example, A DA Form 2408-13-1 with stated inspection or fault narrative, "#1(left) engine service required every 50 hrs...," would have a WUC of 04-powerplant installation. The actual service required involves extensive documentation of more than 100 activities on a DA Form 2408-13-2, and one such activity narrative might be, "performed dual chip continuity check," with a WUC of 04A08E-chip detectors on a single entry. Additionally, any other discovered faults in the area, perhaps cowling or airframe damage incurred during inspection, would be documented under the same DA Form 2408-13-1 with a potential WUC listing of 02-airframe [172].

8.2.3 Automated Coding in Other Domains

Hierarchical coding structures are prevalent in the medical field. There is active research into predicting the diagnostic codes in multiple modalities [173]. Xu et.al. recently wrote a paper that included predictive methods for unstructured text, semi-structured text, and structured tabular data [174]. By using multiple modalities in their experiments, they were able to create state of the art, explainable models for predicting ICD-10 codes in a medical environment. Aircraft Maintainers have also had automated coding solutions in the past using novel collection methods [175,176,177]. Using these works as basis, the next goal is to map the Aviation codes into a machine learning problem with current AutoML and Machine Learning Operations (MLOps) methods.

8.2.4 AutoML Methods

Automated Machine Learning methods have grown as compute has become cheaper in the last ten years [178,179]. AutoML entails automated algorithm selection, model parameter optimization, and, in some cases, model deployment to the respective environment [180].

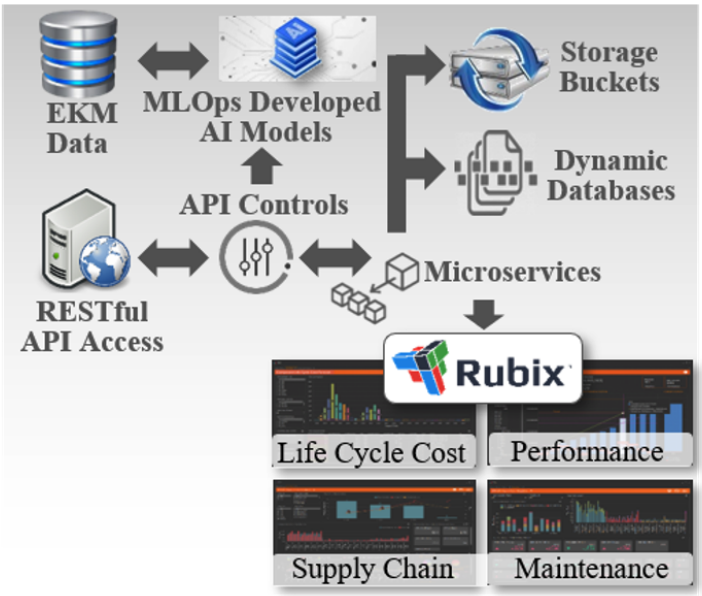

Figure 8.1: MLOps Delivery. MLOps approach provides precise microservice deployment and integration for indexing, cataloging, reporting, and visualization.

AutoML methods are now directly built into popular machine learning packages and cloud deployments for optimizing SciKit-Learn algorithms [181, 182]. The algorithms are selected based on their ability to supply real time predictions to the downstream customers. One of the hallmarks of these new systems is the ability to build trust into each system by selecting models that are explainable with the best accuracy [183, 184].

8.2.5 MLOps Methods

MLOps is the continuous development, delivery, integration, monitoring, and automation machine learning models in a production environment—effectively a Development, Security, and Operations (DevSecOps) pipeline methodology akin to software development cycles and tools [185, 186, 187]. In this work, MLOps is established via deployment to an Azure cloud instance with a Representational state transfer (RESTful) application programming interface (API). RESTful APIs are the standard for production environment deployment due to its flexibility and ease of use for downstream applications like a web interface [188].

8.3 Approach

The approach section will cover the Application Description and Process for cleansing the data, creating the model, and deploying into the production environment. The machine learning model development process uses state of the art automated machine learning techniques coupled with selectable feature rich backends for discovering the best pipeline for generating a supervised classification model. The models are then deployed in on premises and cloud-based deployments for use case flexibility.

8.3.1 Data Cleansing

The first step is to cleanse, process, and clean the data into a simple format of: [Text, Label], [Text, Label], [Text, Label], etc. This is a classic super-vised learning structure to the problem data. The text data is scrubbed for stop words prior to learning; stop words are commonly associated with words that stop the flow of information in a sentence and common libraries like the Natural Language Tool Kit (NLTK) contain standard dictionaries of stop words to use for filtering the text narratives.

The model input is the free form maintainer entered text and the output is a label (a supervised classification problem). The model uses a split a 66% training and 33% test data. Cross fold validation is applied during training to evaluate the model that performs the best for the domain. The training set is used to train the model and the test set evaluates the model. The team uses the following encoding techniques as feature extractors and evaluates the best one at training: character, word, sentence, Term Frequency-Inverse Document Frequency (TF-IDF), embedding spaces, and transformer technologies.

A key differentiator to this process is that a subject matter expert (SME) labels a subset of the data to provide a golden dataset for learning. This dataset provides enough context for the ML model to fill in the blanks. The SME generated dataset is built using an automated script that looks for keywords in the fault and corrective action narratives. The largest issue with this process is that the individual collection of keywords for each system is tedious and

Figure 8.2: An Environmental Visualization - this graph identifies and maps keywords or corrosion unigrams from H-60 logbook records.

cumbersome. Ultimately, a script that covers approximately 8,100 classes led to millions of records requiring cleaning and rescoring. This process is effective but requires a SME to label enough examples for the model to learn the classes.

8.3.2 Feature Generation

Feature Generation is a key feature for extracting a maintainability analysis with WUCs. This character code varies between a minimum 2-digit code to 11-characters which describe in increasing detail the functional system, the subsystem, and the location of each failed component and the corresponding maintenance events. Anecdotal evidence suggests, however, that the "wrong part" is coded for 50-55% of the manually entered WUCs. The inability to automate proof-checking offers two expensive alternatives: either train (or retrain) the operators who perform manual entries or provide SMEs to correct secondary steps.

8.3.3 N-gram Breakdown of Narratives

One of the key assumptions in this work is the use of unigram, bigram, and trigram breakdowns. Collecting single words from a narrative, tabulating them, and finding the top 1,000 words in a corpus is how a unigram feature space is constructed. Bigram collects two-word combinations, tabulates them, and finds the top 1,000 combination of words in the narrative corpus. N-Gram refers to an approach where the number of word combinations

that are important can be configured and optimized for based on needs during the modeling phase. These raw feature sets are fed as input into the AutoML methods to select the most optimal combination of features to get the best performance for each step in the ML process

8.3.4 Hierarchical Modeling Approach

WUC structures are hierarchical by nature. When using ML, the choice was made to mimic this structure by creating a system predictor first to begin more precise predictions from the highest order of the system domain. Then, a model is built for each system. The goal of the downstream models is to predict the specific nuances of each system, which provides predictions identifying assemblies, subassemblies, components, and hardware. This hierarchical system is made up of 70 individual models with the ability to predict approximately 8,100 unique classes.

8.4 Results

The initial results of the experiments yielded a new dictionary of commonly used words by maintainers, a method for correctly classifying WUCs at the System and Sub-System level, and downstream processes that use this data to continuously improve the maintenance process. These results are applied both in hindsight (after-the-fact corrected analytics) and in a proactive data-centric approach by assisting maintainers at the point of data entry.

8.4.1 Analysis

In a 17-million-word sample of Chinook helicopter fault records, 98.5% of the unique words did not exist in a standard English dictionary word list (163,204 non-dictionary terms out of 165,597 total unique words). If one includes punctuation, the number of unrecognizable entries gets much worse. It is worth noting that manual entries also lack fidelity; the Chinook sample highlights repetitive uses of the same term (average total to unique

word ratio of greater than 100:1), which makes distinguishing one maintenance job from another potentially more challenging than if each entry was more coherently distinctive. Willis (2017) concluded that current maintenance records "lack required data and accuracy for use in reliability analysis".

With our dataset, the total number of faults tallies 1.7 million, of which 25% (428,293) start as a WUC of "00" or "aircraft". These cases are largely assumed to be incorrect as 00 is the default WUC field selection on the digital DA Form 2408 series within ACN. Existing structured query language (SQL) rules of "CASE.....WHEN" correct a fractional portion of 116,396 (27%) from system 00-aircraft to more precise functional groups. With newly developed ML techniques, the team exposed an additional 5% of the 00-aircraft data for inspection. Applying the same principles to the WUC 02-airframe, the ML model suggests changing 27% of the 294,000 maintainer entries to a different functional system above and beyond the existing SQL rules. Regarding performance, the WUC model accomplishes record correction at roughly 100 times that of the previous process(es). The model effectively predicted and corrected WUCs on 2,000,000 records in 180 minutes in comparison to a SME's efforts entailing 1,960 hours (a USG man-year) while scoring 200,000 records.

8.4.2 Explainability in ML Models

ML model explainability is critical to understanding why a decision is made and a key component of U.S. government AI ethics standards. With the WUC models, the goal is to understand the decisions and ensure the model has not overfit to certain types of speech. Decision Tree models were chosen as the primary delivery method due to their inherently explainable nature. Using the Explain it Like I'm 5 (ELI5) tool, each prediction is explained and linked to the contributing features in the input narrative using toolkits like Local Interpretable Model-agnostic Explanations (LIME) [12]. Each decision is broken down to the unigram and bigram components to which words are affecting the overall classification.

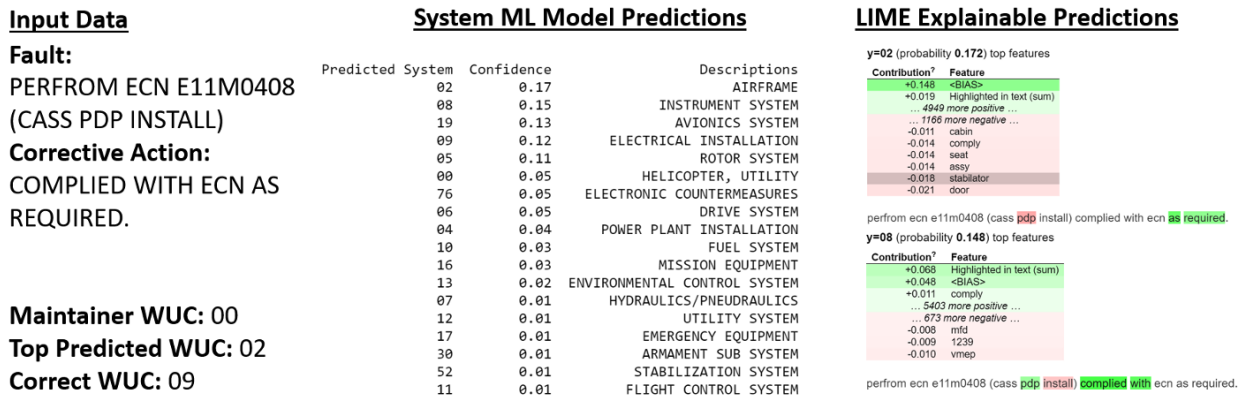

Figure 8.3: ELI5 and LIME WUC Model Explainability. Example narrative and breakdown of prediction methods and confidence intervals (note the misspelling in the Fault text "PERFROM").

Taking it a step further, we derive feature importance and understand which of the features in an individual prediction are driving prediction performance. Feature Importance Analysis is a methodology where the model is probed multiple times until it is possible to understand the individual contribution of a particular feature to a global and local feature importance. At the global level, all predictions benefit more from certain features. These features will affect every prediction. At the local level, each prediction is affected by certain features more.

8.4.3 Deployment and Use

The models are deployed behind a RESTful API for flexible delivery to various downstream applications. Currently, the models are used across three distinct products: a custom enterprise analytics tool called Rubix, a training and certification tool called Maintenance Readiness Levels (MRL), and an extract, transform, and load (ETL) tool. These tools have different purposes. Rubix calls the API directly to receive on demand predictions to help maintainers on the flight line and in the office correctly classify their faults, illustrated in Figure 8.4 below in the WUC Prediction module. The ETL tool batch processes WUC record correction requests for use in the analytical products, including matching corrected

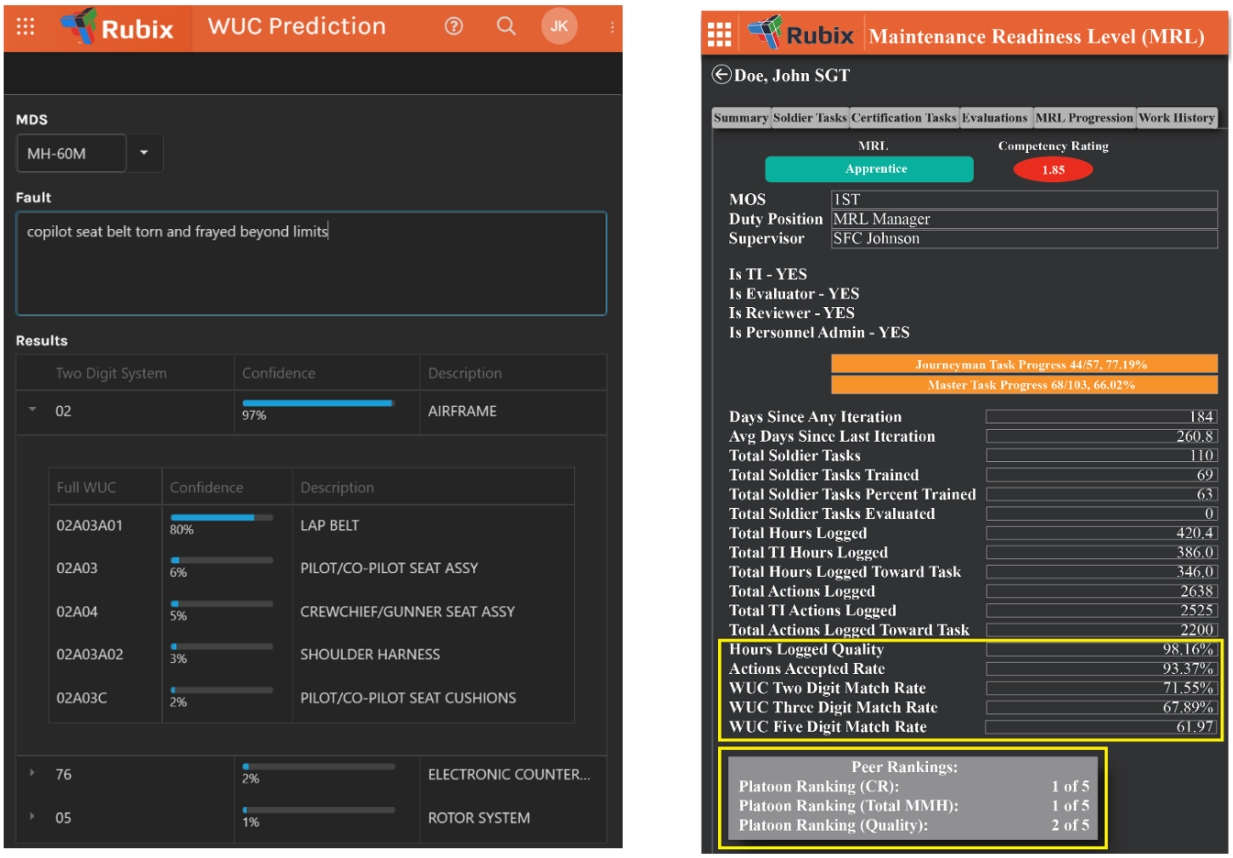

Figure 8.4: Adversarial Surface for MobileNetV2 Models trained on VEDAI dataset. Passive Measures included recommending a particular channel as mitigation. Active Measures require augmentation, architecture changes, and other changes to the modeling process to overcome the attacks.

records back to originals for use in maintainer quality metrics within MRL, also depicted below in Figure 8.4.

8.5 Future Work

As the models mature, the goal is to create a pipeline that automatically ingests and rebuilds models as the distribution of the data changes. Monitoring the data distribution would provide a baseline for when to trigger retraining and offers metrics on evolving free form text fields from the maintainers. The data should inform the model deployment and use cases. There are systems that account for a small percentage of the documented narratives because they are assemblies, subassemblies, or components that seldom break or require

repair. In these cases, the dataset is already inherently imbalanced and requires continuous monitoring to affect accurate models.

8.6 Conclusion

The WUC prediction and correction effort demonstrates derived business value from dirty data in a production environment using ML and augmented analytics. WUCs are typical categorizations among end-user products and drive decision making processes at many levels of the aviation maintenance process. This work demonstrates the utility of applying data science to the general aviation community to affect end items for maintainers and senior leaders by transforming dirty data into actionable information.

Chapter 9

Color Teams for Machine Learning Development

9.1 Introduction

Machine learning and software development share processes and methodologies for reliably delivering products to customers. This work proposes the use of a new teaming construct for forming machine learning teams for better combatting adversarial attackers. In cybersecurity, infrastructure uses these teams to protect their systems by using system builders and programmers to also offer more robustness to their platforms. Color teams provide clear responsibility to the individuals on each team for which part of the baseline (Yellow), attack (Red), and defense (Blue) breakout of the pipeline. Combining colors leads to additional knowledge shared across the team and more robust models built during development. The responsibilities of the new teams Orange, Green, and Purple will be outlined during this chapter along with an overview of the necessary resources for these teams to be successful.

This chapter seeks to address the construction of a machine learning development team. Traditionally, machine learning teams focus on investigating machine learning applications in practical spaces, building promising architectures, and finally deploying the best models to production. Teams can take on all of these aspects or just focus on one at a time. Machine learning adoption in the industry affects every facet of software development from code generation to testing [189]. The production-focused models are then evaluated for performance in normal and adversarial settings. In this work, we propose formally assigning responsibilities to individual parts of the team to focus on their core strength.

9.2 Color Teams for Machine Learning Development

Cremen introduced the InfoSec Color Wheel for structuring teams in the Cyber-Physical and Software systems [190]. The goal of this work is to apply the same concepts with machine learning developers in mind and build more robust models from the beginning of the development process. Focusing on including the attackers in the development loop allows the entire team to understand the holes and vulnerabilities in the included system. Models are so large now that it is nearly impossible to supply explanations for each sample without massive computer resources or time delay in the answers created [191]. Due to the complexity of explaining machine learning models, it is necessary to form teams that can baseline, attack, and defend ML models.

9.3 Disecting the Teams: Familiar Concepts

Cybersecurity commonly uses the Red, Blue, and Yellow teams to break out clear lines of responsibility during the hardening of cyber-physical and software systems [56]. Each team has a distinct role to play in baselining, attacking, or defending the system as shown in Figure 9.1. In machine learning development, these concepts generally translate to model selection, parameter reduction, and pruning or data augmentation to protect those models [1]. Each model development pipeline though needs a specific plan for combatting known and predictable vulnerabilities that an attacker could exploit [192].

9.3.1 Yellow Team: Build Phase

Yellow Team or the Development Team is responsible for building the product that is attacked and defended. In the Cyber Security domain, this is building the cyber-physical or software system [144]. In Machine Learning, these are the model builders and creators that train, build, and deploy the model to the appropriate systems [193]. The Yellow Team is solely responsible for the creation and deployment of the model.

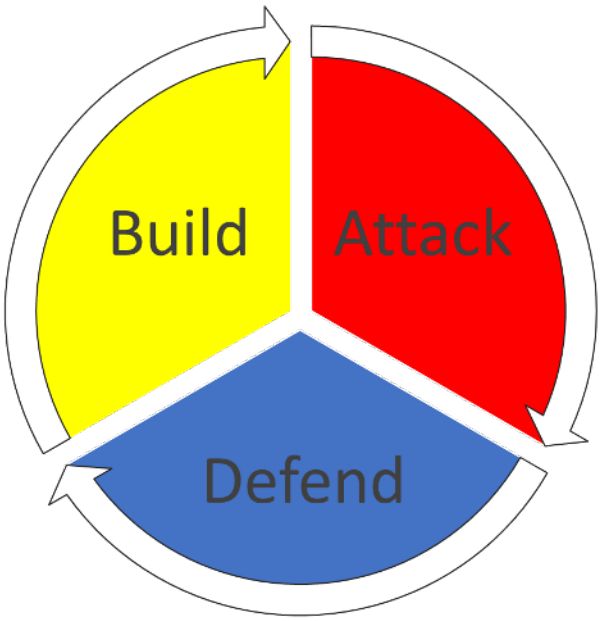

Figure 9.1: Build, Attack, Defend is a common model in cybersecurity and is emerging as a method to protect machine learning models in development [1].

9.3.2 Red Team: Attack Phase

The Red Team serves as system attackers. This team understands known vulnerabilities and exploitative methods including how they may be used against your system. An expectation for the red team is its ability to attack a system and provide a list of exploits that work on the current deployment [194]. In some cases, the Red Team can also supply suggestions for how to circumvent these issues. For machine learning, these attacks can range from the deployment methodology to known issues with the architecture or datasets [195].

9.3.3 Blue Team: Defend Phase

Red Team attacks and Blue Team defends. The Blue Team understands the known exploits and vulnerabilities for the systems deployed but introduces solutions for the development team to implement [195]. In the machine learning context, this function would pertain to the use of machine learning libraries and the selection of architectures and datasets.

9.3.4 Responsibilities and Roles

Each team in the classic configuration has defined and clear responsibilities in the development of their systems. In recent years, it has become apparent that a lack of communication and understanding of vulnerabilities in common development platforms has led to a need for blending these teams into mixtures of colors [55]. The next section will map these additional team constructs into the machine learning development lifecycle.

9.4 Adding New Team Configurations

There are three additional team constructs typically applied to the cybersecurity field: Orange, Purple, and Green. The focus of each team is to include attacker knowledge directly into the team to ensure that each step includes some protections from the beginning. The focus of this section is to describe the mapping of a machine learning development process to these new team constructs.

9.4.1 Orange Team

As illustrated in Figure 9.2, the Orange Team is focused on bringing organizational changes to a team around building more robust architectures and datasets. In a classic configuration, the teams constructing datasets or developing the models may not worry about attacks on those deployments. The Orange team is establishing new processes and educating developers on best practices to prevent adversarial attackers from launching successful attacks. In smaller organizations, this role could be

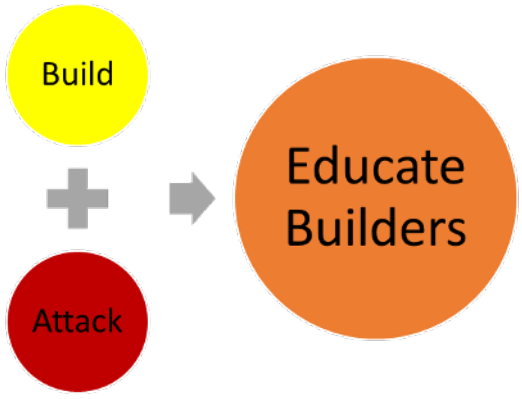

Figure 9.2: Orange Team educates builders and creates robust ML design patterns for development

taken on by a single person with development and adversarial machine learning knowledge.

9.4.2 Purple Team

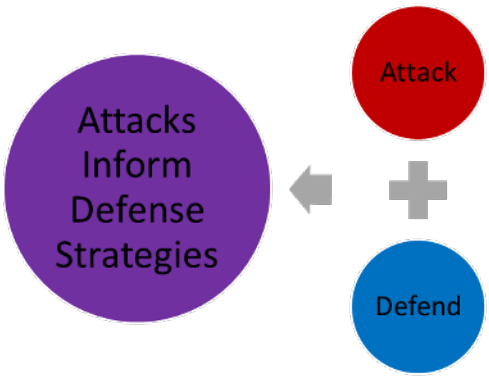

Figure 9.3: Purple Team creates defense strategies from attacks launched on target models in a production environment

As shown in Figure 9.3, the Purple Team integrates defensive tactics based on the adversarial results and continues to improve processes established by the Orange Team. Each attack brings a set of solutions and issues that might prevent that particular attack from being successful.Purple Team works with both Blue and Red teams to develop strategies for understanding the adversarial surface of a model and how to protect both the model and datasets from those attackers. If Orange Team is responsible for maintaining processes and educating the team, then Purple Team is responsible for keeping models robust from attacks.

9.4.3 Green Team

As shown in Figure 9.4, the Green Team will integrate best practices into the development of the models and dataset. As developers continue to understand vulnerabilities, they begin to integrate, use, and apply these new techniques into their new processes. The models will be evaluated against a statistical model for assessing adversarial risk like the modified Drake equation [196] to understand the risk factors in the current model architectures. Finally, the Green Team will continue to improve the model over the development cycle from concept to production deployment. The Green Team will use strategies and patterns designed by both Purple and Orange Team members.

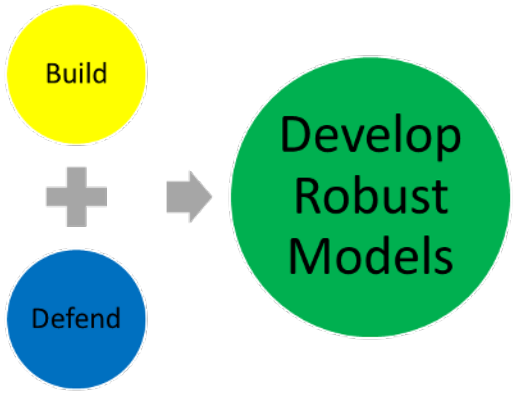

Figure 9.4: Green Team builds secure models by utilizing robust design patterns with informed defense strategies

9.5 Accounting for the Processes

Traditional software teams rely on multiple decades of lessons learned to produce efficient code on flexible timelines. With machine learning and data science teams, there is a longer learning curve if solving problems with techniques borrowed from the development and business side [197]. As shown in Figure 9.5, this section covers the six steps in the overall process: Strategy, Design, Development, Testing, Deployment, and Maintenance.

9.5.1 Strategy

Each machine learning development team will need to create, maintain, and update a strategy for building and maintaining effective models in their production environments. The strategy section will incorporate the teaming concepts to add value at each development

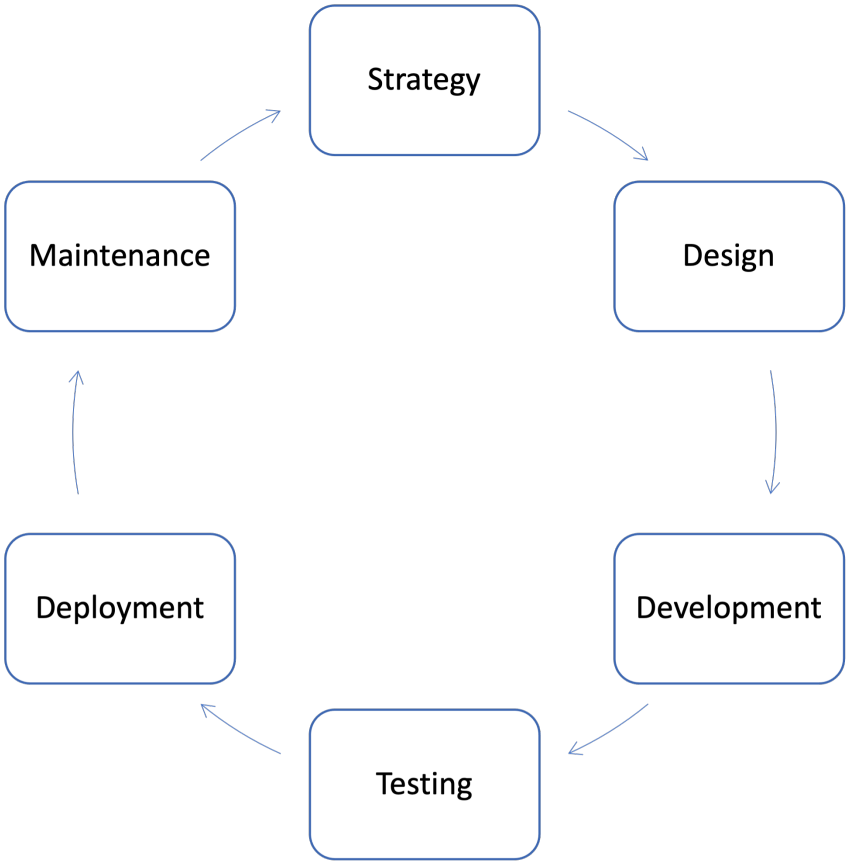

Figure 9.5: The development of the machine learning models is a continuous loop of building new and evolving strategies to build, baseline, and defend models in production environments

step. For instance, in traditional Development Operations, developers would be concerned with choosing the correct data structures, libraries, and methodologies without considering the security ramifications of each of those choices. Development Security Operations (DevSecOps) is a new process by which the developers actively design techniques that are more secure from the input [186, 198]. In the same vein, we introduce Machine Learning Security Operations or MLSecOps as another pathway for protecting models [199]. By using these teaming constructs, a machine learning team can build adversarial protections into their model development from the beginning of the strategy phase.

9.5.2 Design

The design phase is reserved for understanding the proper structures, methodologies, inputs, outputs, and general operations of the requirements being built with software. In a machine learning team, the data inputs and model architectures selected can have a large impact on the adversarial risk assigned to that particular development cycle. By using the Green Team to inform proper datasets and architectures, the Green and Yellow Team together can design more robust models from the beginning. Cross-functional inclusion of team members (team members can belong to more than one group) will lead to additional lessons learned between the steps in these processes. Design, in a machine learning team, should be focused on solving the problem with a robust, secure solution that has the best chance of maintaining a longer lifespan when deployed to Production [200].

9.5.3 Development

In traditional software teams, development can be straightforward and deterministic for tasks like building frontend or backend code for websites. While there are challenges with the implementation of tools or algorithms, estimating the number of story points can become deterministic with Agile teams [201, 202]. In machine learning development, the methods for estimating stories do require some updating. Singla et.al. analyzed several machine learning and non-machine learning projects to understand the difference in estimating and planning for ML-based projects [203]. In their analysis, they discovered that ML teams had more challenges with completing stories but the success stories included descriptive titles, clear labeling for the ML domain used, and measurable "Done" criteria in Agile methodology. Overall, this demonstrated that the development process for ML projects has commonality with normal development projects with stricter adherence to Agile tenants such as clear story descriptions and "Done" criteria.

9.5.4 Testing

Test suites are commonly structured around the unit- and integration-testing of software functionality. DevSecOps also adds testing for best security practices in coding and deployment [204]. Similarly, for a machine learning project, a common suite of testing tools is needed for evaluating machine learning models for susceptibility to incoming attacks. Each incoming attack can be tailored to an incoming architecture and, in the same vein, a Purple Team can devise a set of standard attacks for a team to develop or defend against. The Green Team then develops strategies and solutions for protecting models from these incoming attacks by using the Purple Team's recommended attacks. Each team has responsibility for mitigating attacks while still meeting performance targets. If performance cannot be met due to fixes implemented, then those attacks must be carried forward as the risk for review boards in a production team.

9.5.5 Deployment

The Green Team focuses on the task of defending a model while also building to meet performance targets. After testing, the Green Team adjudicates the outstanding risks and prepares for a release candidate for the machine learning model. The model is deployed to production with proper evaluation of the available exploits and each of those risks is properly included within the model documentation for the release. Hotfixes for a release are part of the deployment phase as certain aspects of the model environment or the evaluation may lead to additional mitigations like needing a version of a package or a way of processing data to avoid exploits from adversarial actors.

9.5.6 Maintenance

Every machine learning project will have the burden of maintaining the models in the face of new and changing data. To maintain a machine learning model, the deployment platform must monitor the incoming data and detect data drift in the original model. If a

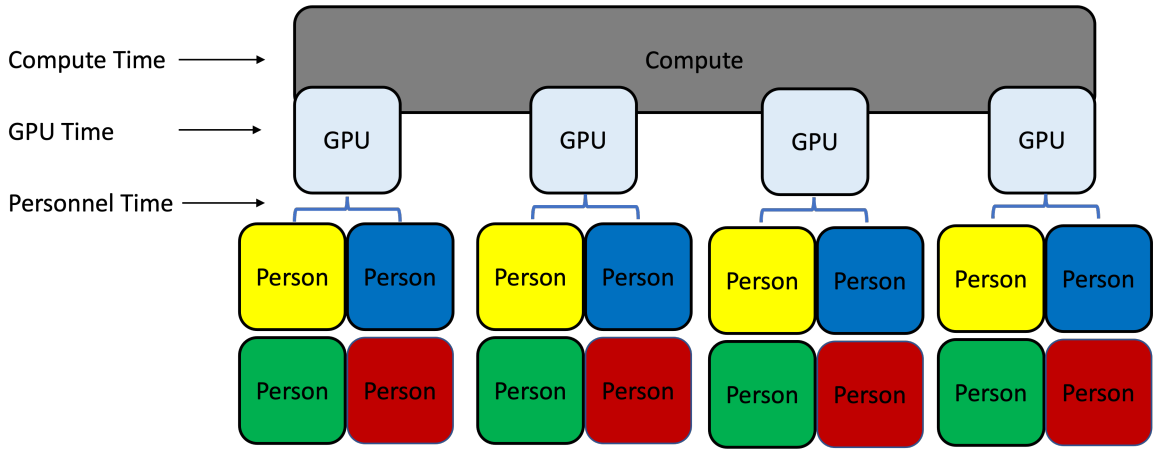

Figure 9.6: Summary of key resource allocations steps to move a project from conception, training, testing, and deployment.

population difference is detected, the model will need to be retrained and redeployed. This maintenance event needs to have periodic evaluation and downtime built into the system. Like monitoring the dataset drift and its distribution, an additional step is added to detect possible attacks. A security patch in the machine learning world may look a lot like a new model – except the goal of the updated model is to mitigate the effectiveness of the newly detected attack. A running list of security exploits should be maintained and detected so attackers can be excluded or banned from using the deployed resource.

9.6 Allocating the Resources

Machine learning projects typically have four major hurdles during development: available computer resources, available Graphical Processing Units (GPUs), appropriate time allotted, and personnel shortages. Computer resources, including GPU, are used to process data, train models, and evaluate performance. In this section, each of the major components will build on the next. Note: there are many configurations of CPU and GPU assets in a team (Figure 9.6). This work represents one such allocation of those resources.

9.6.1 Compute Resources

Each development cycle will include an allotment of computing resources for performing the task. To combat adversarial actors and their techniques, the Green Team will need to employ state-of-the-art (SoTA) processes to improve the model. Adding additional steps like parameter search and neural architecture search for improving model performance can add burden to available compute. The team will need to evaluate the risk and reward of making the model more robust to attacks.

9.6.2 GPUs

In small to medium organizations, GPUs are a resource for training deep networks and transformers. As transformers begin to take over many tasks in the field of machine learning, the high VRAM GPUs are necessary for training and inference. Due to the prohibitive cost of cloud computing for higher-end GPUs, it's necessary for organizations, who have security concerns, to create and maintain internal on-premise GPU clusters. As a result, rationing GPU use to the different targets will guide how to best apply them and what jobs get higher priorities.

9.6.3 Personnel

The compute, GPUs, and time are all reliant on available time from personnel to utilize those resources in each of the color teams. The personnel side of the machine learning development team is a challenging task when there are multiple teams and competing priorities for each project. In using this structure for building color teams with tailored tasking for protecting models, it is important to cross-train the personnel on each color team's responsibilities to understand each of the different development requirements.

9.6.4 Time

Model development timelines include acquisition of the data, preprocessing of the data, building the model, and optimizing for a given task. Adding a time component of protecting these models will add complexity which can affect the schedule and available resources. To compound the issue, it can take considerably more resources to add additional robustness to a model with current SoTA techniques for protecting machine learning models. Three key time-related parameters can be monitored to support effective resource usage:

- **Compute Time**
  - Compute time represents the entire machine use time including memory, CPU time, GPU time, and data storage
- **GPU Time**
  - A specific mechanism for tracking the GPU usage as defending models can involve parameter and network-tuning to avoid certain types of attacks using, for instance, adversarial training methods on augmented datasets
- **Personnel Time**
  - Tracking personnel available to build, attack, and defend models is needed to fix high-priority risks in production systems

9.7 Summary and Future Work

The theoretical rate-limiting step is the Yellow Team which builds the time-consuming initial code. Every iteration of the code is going to improve its defenses against adversarial attacks. The Yellow Team is charged with building the initial set of software that can meet a mix of demands between protecting the model from attackers and meeting the customer requirements for performance. These tradeoffs can be difficult to manage as performance

and defense are typically at odds with each other. As the performance of a model increases, it can be more susceptible to incoming attacks. Likewise, as the defendability of the model increases, it can be difficult to meet performance targets like speed, accuracy, and explainability.

Each step in this process is designed to create a team that can manage the demands of creating production-ready machine learning models with security in mind. As a team develops each new model, they will account for new and emerging threats in their model space. Drawing on the cybersecurity analogy, examples like a formal framework (MITRE ATT&CK [205]) should be developed for each color team. For instance, the Red Team catalogs successful attacks based on the vulnerability while the Green Team similarly logs the appropriate remediation or response. In this way, the rotation of personnel skills and time management may benefit future collaborative efforts between colors and project goals.

The future work of this research should plan and model each color team's steps to find the rate-limiting areas and where optimizations can be found. The color teams proposed in this work can be implemented as a system or in pieces that fit the needs of each project. This is a framework for defending deployed models from adversarial attacks in the wild.

Chapter 10

Limitations

There are limitations within the works presented and this chapter will go each concern by referencing the chapter in this thesis. In all there, are two primary concerns raised in this chapter - the limited scope of the adversarial attack methods and the limited scope of the models/datasets used.

10.1 Limited Scope of Adversarial Attack Methods

The adversarial machine learning space is constantly growing with new and evolving threats against machine learning models. There are always multiple ways to attack models including at the data side, during inference, and even on delivery of the prediction. In order to baseline the ability of these processes to work, each of the results had to establish a subset of current attacks. The goal of each result presented was to demonstrate a repeatable process that could protect machine learning models as the threat space grows. With additional threat vectors, it will be important to add new process steps to protect the model at the learning, inference, and post prediction phases of machine learning use cases. Chapter 5 relied on simple attacks and the next section will discuss this as a limitation. In the next section, the simple attack vectors will be discussed as a limitation from Chapter 5.

10.1.1 Assumed simple attack vectors

There are an incredible amount of adversarial attacks coming out each week in Machine Learning. The advent of public architecture and datasets that are easily accessible make it easy for attackers to find and develop new exploits for machine learning models. In the Chapter 5 experiments, the simple attacks were chosen for multiple reasons.

First, the attack method was the most basic imaginable attack - they were simple character to character mappings. As a kid, it was easy to change an 'E' to a '3' and think it was a special coded language. In machine learning, this can be a difficult distinction to make if the training data does not contain any knowledge of this type of string replacement. Second, the simple character to character replacement is fast. Adversarial Methods that rely on attacking underlying properties of the models or structure within the model require optimizations to properly attack those models. With character to character substitution, the attack can be carried out using built in functionality in the python libraries with dictionaries. The dictionary can be updated to map to a new and evolving attack methodology. This can give the attacker an advantage as the attack is a low amount of effort for high impact. The experiments effectively demonstrated this effect by showing up to a 80% degradation in detectability in the sentiment analysis tests. The simple attacks, though, are the limitation in the experiments explored in Chapter 5.

Chapter 5 focused on presenting a process for evaluating a machine learning system called Build, Attack, Defend or BAD system. This process was demonstrated using sentiment analysis experiments with a toxicity dataset. The simple attacks chosen do not take into account the underlying model or dataset and thus are not customized to exploit particular peculiarities of the model or dataset themselves. The results from attacks that are customized to the model attributes could make it much more difficult to defend against. Zero-Day attacks in the cyber discipline refer to attacks that are not yet protected against and allow a hacker unfettered access to a system. In the machine learning discipline, there are adversarial 'zero-day' attacks that are manipulating the output of the model and no defense is yet known for these attacks. These Zero-Day attacks are difficult to anticipate and protect against in practice. The Chapter 5 process currently assumes known attacks on models for protections and did not evaluate the adversarial surface of the model. There is one final limitation that both of these chapters assume and that is the non-automated probing of the model.

10.1.2 Assumed non-automated model probing

In the cyber discipline, cyber physical systems are increasingly attacked by automated systems with known exploits for commonly used systems and backends. The same analog is going to be made in the adversarial machine learning community as the machine learning models and datasets become commoditized. The machine learning system attributes are all widely distributed and make it easy for attackers to figure out sophisticated attacks for the models. In Chapter 5, this is an apparent issue. The APIs used in these experiments were black box APIs accessible to the general public. Once access was gained to explore these APIs, the experiments quickly showed that even simple attacks could successfully steer the output of the underlying sentiment analysis model. In a similar way, adversarial attackers can design attack systems that can successfully probe, classify, and customize attacks for a targeted machine learning system. The experiments carried out in this work did not launch multiple attacks or increasingly more sophisticated attacks when previous ones failed. A future work could explore the ability of an attacker to successfully penetration (pen) test a machine learning system.

10.2 Limited Scope of Models and Datasets

There are hundreds of papers being released per day in the Machine Learning section of Arvix []. Machine Learning papers have a range of solutions from use cases for current architectures to developing wholly new architectures in the literature. It can be difficult to keep up with the ever evolving set of architectures and it becomes necessary to maintain a stationary baseline for performance. For instance, in the image classification space, the top methodology on PapersWithCode.com is ¡technique¿ which came out in ¡year¿. Chapter 4 focused on the adversarial actor portion of the attack chain and demonstrated the ability of this actor to detect the underlying dataset and model. The next section will discuss the limitations with the scope of this work.

10.2.1 Limited to simple model or dataset attribution

Adversarial Attacks are more effective when those attack vectors are tailored for the model or dataset. In Chapter 4, the goal was to explore how easy it was to detect the underlying machine learning system qualities and use them to our advantage in experiments. Each of these experiments focused on demonstrating how distinguishable an architecture or dataset are in the image and text domain. As architectures are modified, it may only be possible to find the most similar architecture to a particular output. Only architectures that have been seen in the wild can be identified in this case. And, the scope is more limited than that. The experiments outlined in Chapter 4 do not mix datasets or modify architectures. Each of the classifiers are able to be directly downloaded from Keras or HuggingFace with no additional installations necessary. Model Architectures are the first limitation discussed here.

The experiments only chose architectures that could be downloaded and deployed from common frameworks like Tensorflow. The model architectures inside of these models has zero customization to the layer structure, sizing, or additional layers like dropouts/normalization. In this regard, simple customization to the models could fool or confuse this diagnostic

system from accurately classifying the underlying machine learning system. This choice is intentional because the wide adoption of machine learning into production systems typically sees fast integration without optimization of the end goal. The fastest way to optimize a problem space is to maintain both the base weights and architecture and deploy the model to a production system. Unmodified architectures are detectable in this framework and can be used. Once the architecture is modified or ensembled with another architecture, the framework would need to be expanded to compensate for this new set of observations.

A similar issue occurs when a model may perform similarly when trained on different datasets. Datasets are easily downloaded and trained on from common frameworks like HuggingFace. These datasets have standard attributes and are widely distributed in the community for training on multiple different target architectures and areas. Different datasets containing data from a similar population could result in very similar learned weights for the underlying machine learning model. These experiments in Chapter 4 rely on the models learning different distributions of weights so that the signature of the attacks used to probe the models can return specific diagnostics. If two models trained on two different datasets return the same data signature from the probes, they would be indistinguishable from the current experiments. Another issue arises if the two or more datasets are mixed together and then trained on to create the final model. If the model architecture is frozen but the mixed datasets are trained on, is it possible to distinguish that the model learned a portion of its weights from that dataset? It is unclear from the experiments in Chapter 4 and could be another future work area for expansion.

10.2.2 Assume no new machine learning architectures

The limited selection of machine learning models only investigates the models that can be downloaded and deployed in simple methods from publicly available frameworks like HuggingFace or Tensorflow. In Chapter 6, these same assumptions hold as MobileNet was used as the base architecture in exploring the adversarial surface of the model. This does

become a limitation of the work as MobileNet is not State of the Art for image classification in the overhead image category. This subsection will focus on discussing the limitations of only using MobileNet to explore the adversarial surface.

In Chapter 6, the adversarial surface of a deployed overhead image classification model is explored for multiple wavebands with an open source dataset. The core idea is to evaluate if any particular band in this space can compensate better for adversarial attacks than the other. This passive measure method to protecting models is a brute force assumption in this space. It means that the development team needs to train multiple models and evaluate the adversarial surface for each model. Depending on inference speed and test set size, these experiments can be quite computationally expensive. Once all of these models are trained and evaluated, it is possible to mitigate the attackers through recommendations on available models. Because Chapter 6 contained six available models, it was possible to use the adversarial surface to create a set of recommendations for which model should be used to protect against these types of adversarial examples. In this evaluation, FGSM and FGM are identified as attack sets that can mitigate by recommending an image classifier trained with the IR band. Some of the attacks though could not be mitigated through passive measures (selecting a different model to run in production). Active measures were needed but not evaluated in this work. Examples of active measures include additional training augmentations, modification to the architecture used, and modification to the dataset trained on. The attacks analyzed also relied on full access to the underlying models and this is the subject of the next discussion.

White box attacks are successful because the attacker has access to all machine learning model attributes. The attacker is able to tailor each attack to specific vulnerabilities within a particular architecture or dataset. In contrast, black box attackers have little knowledge of the underlying machine learning model or system. The experiments carried out in Chapter 6 do no explore black box attacks on these models as the experiments would need to cover a much larger set of attack vectors to understand the adversarial surface of an unknown model.

Chapter 6 relied on the fundamental assumption that many machine learning teams are going to rely on downloadable models with good enough performance to start their production systems. The idea was to isolate the model and dataset and focus on a methodology for understanding how pervasive existing attack vectors are and how efficient they are on currently released models. The final limitation of the work focuses on the selection of MobileNet for the exploration.

One additional limitation in the Chapter 6 work is the use of MobileNet - MobileNet is multiple years old. MobileNet offered an incredible speed to accuracy trade off and led me to choose the architecture. The goal of Chapter 6 was to show a framework for creating informational content for evaluating the adversarial vulnerabilities of a model. The adversarial surface consisted of 200+ experiments against multiple models and adversarial methods. Because only one model architecture was selected and trained on, it is not possible to know if other models would have performed better against these adversarial attacks. This work could be extended by adding in additional model architectures into the pipeline and choosing the best of breed for dealing with all active adversarial exploits in the wild. This does assume also that the adversarial attack is published and release to the public. In the cyber domain, the attack vectors are usually only released after they have lost their value to the attacker. The process is more important than the discovery of new techniques. Attackers will continue to get more advanced and machine learning practitioners need to have a process for evaluating adversarial risk in production environments.

10.2.3 Assume idealistic settings for machine learning systems

In Chapter 7, the modification of the Drake equation is meant to allow machine learning teams to evaluate adversarial risk from outside attackers. A key assumption to this work is establishing a nominal range for each parameter defined in the Drake modification and assuming the minimum and maximum values for each one. An example is the number of parameters in a machine learning model. This number of parameters can be specific to the

model type, architecture, or domain. In certain domains, a large number of parameters could be akin to overfitting the problem whereas some large, hard to learn domains could require large numbers of parameters to effectively learn the domain distribution. The biggest challenge in finding novel utility for this framework shares much in common with Drake's original notion. How to quantify each factor? What if the factors show strong correlations? How do the factors change with time, particularly if both the builders and attackers modify their behavior? What are the appropriate units to assess ML risks, either as the number or severity of adversarial attacks? One informative output that previous technical papers often ignore in assessing model risk is the scale of the overall ecosystem (R). In the literature for cybersecurity, for example, the monoculture aspect for operating systems has proven most predictive of the next generation's attacks. The modifications presented in Chapter 6 were meant to provide a framework for risk professionals to begin to assess the adversarial risk for their machine learning models in a production environment.

10.3 What's next?

In the next Chapter, a discussion about mitigating limitations will be presented as future extension of the experiments presented throughout this work.

Chapter 11

Future Work

There are two clear extensions to the work presented here. First, since we have established that adversarial attacks represent more of an surface, the next work would focus on understanding how many attacks can be negated with simple changes in architecture and training methodologies. Second, if the model can be changed to thwart these adversarial attacks, another work could explore early warning systems for adversarial attacks and finger print their attacks based on their methodologies. Similar to how cyber security firms will trace an IP address or code base, it should be possible to trace the origins of the adversarial attacks to particular code bases or even groups. Each section in this Chapter will discuss these extensions and how to approach future experiments.

11.1 Going beyond simple model or dataset attribution

In Chapter 4, the experiments focused on probing and classifying easily downloaded models from common frameworks like Tensorflow or HuggingFace. In this section, the goal is to cover the possibility of extending this work to finding models that have custom architectures or have a machine learning system with an ensemble of learners. These are two distinct experiments that could be carried out to extend this work and continue to make this methodology effective.

11.1.1 Creating custom models by mixing models and datasets

A common practice in machine learning development is to start with a downloadable architecture, assess performance, and understand if custom architecture or datasets are necessary for learning a domain distribution. Task Specific problems can require customization

of the machine learning system. In the Chapter 4 experiments, the simple assumption was used that the default models were good enough. In this future work, two specific experiments can extend this work - mixing architectures into a single model, mixing architectures by ensembling models, and mixing datasets by training a single model. First, let's discuss mixing architectures in some detail on the experiment design and expected outcomes of this line of research.

Machine learning models have two major components to their design: architecture and dataset. Holding the parameterization and frameworks stationary, the architecture and dataset input contribute to what the machine learning model learns. When the architectures and datasets are held to open source, downloadable content, it is easy for attackers to detect and attack those models. The experiments in Chapter 4 showed this. The next logical step is to take each of these downloadable models and ensemble them into a larger machine learning system. The simplest ensemble is to create a voting ensemble where two or more machine learning models vote on the answer to the problem the best one is selected. Voting ensembles have an optimization stage where it is presented with training data and learns the best voting strategy for the underlying models. In order to keep the experiments approachable, the voting ensemble should be limited to pairs of machine learning models from the already used downloadable architectures. The detectability experiments could be repeated to see if the combination of models can be detected or if even specific models can be detected in an ensemble. There design of the probes for the machine learning system would also need thoughtful design as the ability to detect each underlying model would require the attacker to plan for a combination of architectures in the target machine learning system.

Another way to expand this line of research is to modify the underlying architectures and understand the relationship between original architecture and modified. For instance, starting with an Xception architecture, would these experiments be able to expand and detect a family of detectors built on Xception? This would also assume that the detectors built are run on a similar set of permutations in the model building. Due to the large number of

tuneable hyperparameters and layer types, the detectors for the custom architecture would need to be thoughtfully designed. The experiments would likely need to categorize the baseline architectures into higher level classes i.e. multiple architectures would create a detector family. These family detectors could be multi-category or binary depending on performance. The experiments would focus on a single set of feature backends and analyze the ability of a classifier to detect the type of machine learning model used. Mixing Models or Datasets is an extension of this work that could be explored over several future papers. Another line of research is to estimate the number of potential targets for models that are in the wild today.

11.1.2 Estimating Number of endpoints using unmodified architectures

There are a number of open source machine learning deployment frameworks making it increasingly easy to deploy and use machine learning models without much effort. In the cyber domain, there are simple python scripts that can ping IP addresses and probe those addresses for open endpoints, SSH tunnels, etc. As an analog, a pythons script could be designed to hit websites with common deployments for machine learning models like hyroku and test those models to see what system they have deployed. From this point, the machine learning architecture and dataset attribution models would need to expand capabilities in two ways: train on multiple different datasets and train on responses from unknown architectures. First, let's explore what it would take to expand the detectors to train on multiple different datasets.

Image classification is used for a wide number of applications. Feature backends can be trained on different datasets and produce state of the art performance for an image classification task. By training on all possible datasets for higher level classes, the probes used to find out the machine learning architecture can be used to find out if a particular architecture is being used in the wild. The experiments would need to be constrained to common tasks in the image classification domain like person, dog, cat, or other class detection

that have overlap. The end result of this line of research is understand the relationship between machine learning model output and dataset.

Chapter 12

Conclusion

The goal of this work is to demonstrate the ability of machine learning practitioners to use a disciplined process to protect their models by developing those models in a structured manner. In the background section, a brief history of adversarial learning and the methods explored are covered to provide a base knowledge for the chapters included in this work. The models and datasets in Chapter 3 give a clear description of each architecture and dataset explored in the proceeding works. There are enumerable datasets and architectures in the machine learning field so a subset was chosen that allowed for repeatable, quick experiments. One of the first ways an attacker can develop effective attacks is my classifying the underlying machine learning model in a deployed system.

The architecture and dataset discovery method described in Chapter 4 allows an attacker to find out the underlying machine learning model or architecture and develop relevant attacks. In the adversarial machine learning space, attacks need to be tailored for the underlying model/dataset. In the image space, learning the fingerprint of a model is achievable with modern classifiers and in these experiments, the classifiers reaches high AP numbers with minimal training. In the text domain, classifying the underlying model architecture is harder with a single text sample. For trained datasets, the results in the text domain showed that datasets with clear stylistic cues are distinguishable from each other. In Chapter 4, the experiments demonstrate that it is possible for an attacker to find the underlying model development structure and use it to their advantage when attacking the model. Chapter 5 continues this research into single character attacks on deployed sentiment analysis models to understand vulnerabilities in public systems.

Sentiment Analysis models are deployed around the world to help manage the expansion of social media platforms as toxic comments permeate those platforms. Sentiment Analysis can detect and quarantine these comments from the platform if they can be accurately detected. This work demonstrates that deployed sentiment models are susceptible to simple substitution attacks on single characters. These single character substitutions allow attackers to subvert the underlying system and still effectively spread these toxic messages. In this work, the experiments also demonstrate that these attacks can be effectively defended by using the Build, Attack, Defend architecture. This architecture allows the machine learning development team to baseline a model's performance, attack the model with known vulnerabilities, and defend from those attacks with known fixes. Because these substitution attacks are simple character replacements, a character mapping can mitigate each attack by detecting non-English words, creating sentiment analysis estimates, and taking the maximum toxicity estimate in our examples. The system would default to a risk adverse posture because it would quarantine the highest toxic message based on these substitutions. Further work in this area could focus on looking at model attacks like weight poisoning attacks on classification systems. Chapter 6 starts to delve deeper into image classification systems and how developers can assess and protect their models in those environments.

Image Classification is a common task in many machine learning systems and adversarial protections are overlooked in favor of performance. In this work, the experiments demonstrate the ability to protect machine learning models for overhead imagery by simply using a structured approach to evaluate and reduce the adversarial surface of the machine learning model. State of the art image classifiers are highly publicised and shared model architectures developed by large institutions like Google and Microsoft. The datasets and model weights have been publicly available for years in some cases. Due to the public nature of these models, adversarial vulnerabilities exist in the wild for these models and are exploitable by adversarial attacks. The goal of these experiments is to demonstrate a reduction in the efficiency of adversarial attacks while maintaining original performance

benchmarks. Maintaining the original performance of the state of the art model is key to getting many developers on board because they don't want to lose accuracy in favor of protecting the model from attacks. One critical issue to address from this work is how to reduce the computational burden of retraining the networks for every additional adversarial perturbation. Retraining larger networks can be computationally expensive and these experiments used small, fast networks on purpose to allow for a high number of permutations. The next chapters establish a heuristic methodology for evaluating machine learning model vulnerabilities, a best practices demonstration for machine learning model deployment, and teaming arrangements for structuring a machine learning organization with adversarial protections in mind. Chapter 7 creates a heuristic methodology for evaluating machine learning models by adapting the popular Drake equation.

After developing a machine learning model, a risk team should have the ability to baseline and benchmark the adversarial risk for this system. The Drake Equation is a popular heuristic used to estimate terrestrial life in other solar systems. Chapter 7 creates a heuristic framework in analogy to the traditional Drake Equation. This simple formalism amounts to a summary of relevant factors to machine learning models that allow practitioners to evaluate their adversarial attack risk. Ultimately, its main purpose follows from assessing multiple uncertainties together that may vary by several orders of magnitude and creating a metric for risk professionals to use in their assessment of the system. For example, as ML builders consider whether to privatize or to open-source their models, they may intuitively favor one course over another given a perceived risk for model compromise. This framework provides a deterministic framework for evaluating multiple models and datasets against each other to select the correct risk posture for a given business. This framework provides a practice that can evaluate the risks to privatizing a model vs open sourcing the model and how that potentially increases the attack surface of the model. These important decisions need a deterministic framework that is customizable to business needs and these experiments can assist the ML community to identify the appropriate metrics to track for benchmarking

adversarial risk of a model. Chapter 8 flows into using these best practices in a practical machine learning project for deploying multiple models in a customer critical environment.

Chapter 8 is all about taking these different pieces and deploying machine learning models in a production environment. Work Unit Codes or WUCs are the essential to understanding maintenance events in rotocraft maintenance logs. Each Work Unit Code presents a job done on the aircraft and documenting these events helps understand the work done over the course of many years. Theses codes are typically put into the system via maintainers in maintenance bay and unfortunately are not always put in at the write code level. Machine Learning is used in this problem to correct the dirty data. The WUC prediction and correction effort demonstrates derived business value from dirty data in a production environment using ML and augmented analytics. WUCs are typical categorizations among end-user products and drive decision making processes at many levels of the aviation maintenance process. This work demonstrates the utility of applying data science to the general aviation community to affect end items for maintainers and senior leaders by transforming dirty data into actionable information. These machine learning models are deployed to a business intelligence product, input data and outputs are monitored, and models are adapted and updated based on feedback from this information. This real life deployment of machine learning models allows the users to understand the input data, likelihood of attack or poisoning based on real life data, and a great testing ground for all previous chapters lessons learned. Finally, in Chapter 9, all of methods demonstrated culminate in building a machine learning team framework for developers to use build adversarial protections into models during the development process.

Chapter 9 establishes a set of teaming strategies for applying lessons learned in each of the previous chapters. With all of these new processes, the development team will need to shift how it develops machine learning models to incorporate each one. New team structures will need to be established. Each step in this process is designed to create a team that can manage the demands of creating production-ready machine learning models with security in

mind. As a team develops each new model, they will account for new and emerging threats in their model space. Drawing on the cybersecurity analogy, examples like a formal framework (MITRE ATT&CK [205]) should be developed for each color team. For instance, the Red Team catalogs successful attacks based on the vulnerability while the Green Team similarly logs the appropriate remediation or response. In this way, the rotation of personnel skills and time management may benefit future collaborative efforts between colors and project goals. There are other considerations for the machine learning teams like personnel and compute resources that may drive how models are protected as resources are a precious commodity when developing new models.

This set of chapters illustrates a set of processes for evaluating adversarial risk of a model, a framework for evaluating that risk, and finally a teaming arrangement for organizations to adopt to help protect their deployed production models. Machine Learning is a fast evolving field with new architectures taking over state of the art models every week. The core lesson in these works is to understand that it is not the architecture or dataset that is important. It is the process that machine learning practitioners follow to protect their models that will allow them to confidently field those models into the wild.

Bibliography

[1] M. Vartak and S. Madden, "Modeldb: Opportunities and challenges in managing machine learning models.," *IEEE Data Eng. Bull.*, vol. 41, no. 4, pp. 16–25, 2018.

[2] Y. Ma, T. Xie, J. Li, and R. Maciejewski, "Explaining vulnerabilities to adversarial machine learning through visual analytics," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 1075–1085, 2019.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE access*, vol. 6, pp. 12103–12117, 2018.

[5] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 69–75, IEEE, 2020.

[6] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine learning security: Threats, countermeasures, and evaluations," *IEEE Access*, vol. 8, pp. 74720–74742, 2020.

[7] A. Pavate, D. Kumawat, S. Pansambal, P. Nerurkar, and R. Bansode, "Machine learning under attack: literature survey," *Mach Learning*, vol. 14, pp. 14–18, 2018.

[8] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Data security for machine learning: data poisoning, backdoor attacks, and defenses," *arXiv e-prints*, pp. arXiv–2012, 2020.

[9] X. Wang, J. Li, X. Kuang, Y.-a. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *Journal of Parallel and Distributed Computing*, vol. 130, pp. 12–23, 2019.

[10] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[11] Z. Xiang, D. J. Miller, and G. Kesidis, "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2019.

[12] I. S. Candanedo, E. H. Nieves, S. R. González, M. Martín, and A. G. Briones, "Machine learning predictive model for industry 4.0," in *International Conference on Knowledge Management in Organizations*, pp. 501–510, Springer, 2018.

[13] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *International Conference on Machine Learning*, pp. 2304–2313, PMLR, 2018.

[14] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 7, pp. 2578–2593, 2019.

[15] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[16] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 8018–8025, 2020.

[17] L. Hancox-Li, "Robustness in machine learning explanations: does it matter?," in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 640–647, 2020.

[18] F. Khalid, M. A. Hanif, S. Rehman, J. Qadir, and M. Shafique, "Fademl: Understanding the impact of pre-processing noise filtering on adversarial machine learning," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 902–907, IEEE, 2019.

[19] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, ACM, 2011.

[20] 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*, pp. 506–519, 2017.

[21] A. Nazemi and P. Fieguth, "Potential adversarial samples for white-box attacks," *arXiv preprint arXiv:1912.06409*, 2019.

[22] S. G. Finlayson, H. W. Chung, I. S. Kohane, and A. L. Beam, "Adversarial attacks against medical deep learning systems," *arXiv preprint arXiv:1804.05296*, 2018.

[23] Y. Zhang, Y. Song, J. Liang, K. Bai, and Q. Yang, "Two sides of the same coin: White-box and black-box attacks for transfer learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2989–2997, 2020.

[24] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, IEEE, 2018.

[25] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[26] A. Chan-Hon-Tong, "An algorithm for generating invisible data poisoning using adversarial noise that breaks image classification deep learning," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 192–204, 2019.

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

[28] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, 2015.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[30] L. Bilge and T. Dumitraş, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 833–844, 2012.

[31] K. Mahmood, R. Mahmood, E. Rathbun, and M. Van Dijk, "Back in black: A comparative evaluation of recent state-of-the-art black-box attacks," *IEEE Access*, 2021.

[32] S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru, "A survey of black-box adversarial attacks on computer vision models," *arXiv preprint arXiv:1912.01667*, 2019.

[33] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," *arXiv preprint arXiv:1807.04457*, 2018.

[34] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning*, pp. 2484–2493, PMLR, 2019.

[35] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1111–1119, 2019.

[36] W. Garcia, J. I. Choi, S. K. Adari, S. Jha, and K. R. Butler, "Explainable black-box attacks against model-based authentication," *arXiv preprint arXiv:1810.00024*, 2018.

[37] X. Wei, Y. Guo, and B. Li, "Black-box adversarial attacks by manipulating image attributes," *Information Sciences*, vol. 550, pp. 285–296, 2021.

[38] R. R. Wiyatno, A. Xu, O. Dia, and A. de Berker, "Adversarial examples in modern machine learning: A review," *arXiv preprint arXiv:1911.05268*, 2019.

[39] M. Chase, E. Ghosh, and S. Mahloujifar, "Property inference from poisoning," *arXiv preprint arXiv:2101.11073*, 2021.

[40] V. Duddu, "A survey of adversarial machine learning in cyber warfare," *Defence Science Journal*, vol. 68, no. 4, p. 356, 2018.

[41] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[42] E. Pintelas, I. E. Livieris, and P. Pintelas, "A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability," *Algorithms*, vol. 13, no. 1, p. 17, 2020.

[43] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *2019 11th international conference on cyber conflict (CyCon)*, vol. 900, pp. 1–18, IEEE, 2019.

[44] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[45] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *arXiv preprint arXiv:1706.03691*, 2017.

[46] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[47] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Adversarial training for multi-context joint entity and relation extraction," *arXiv preprint arXiv:1808.06876*, 2018.

[48] H. Zhang, H. Chen, Z. Song, D. Boning, I. S. Dhillon, and C.-J. Hsieh, "The limitations of adversarial training and the blind-spot attack," *arXiv preprint arXiv:1901.04684*, 2019.

[49] B. Vivek, K. R. Mopuri, and R. V. Babu, "Gray-box adversarial training," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 203–218, 2018.

[50] P. Panda, I. Chakraborty, and K. Roy, "Discretization based solutions for secure machine learning against adversarial attacks," *IEEE Access*, vol. 7, pp. 70157–70168, 2019.

[51] B. Vivek and R. V. Babu, "Single-step adversarial training with dropout scheduling," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 947–956, IEEE, 2020.

[52] G. Liu, I. Khalil, and A. Khreishah, "Using single-step adversarial training to defend iterative adversarial examples," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pp. 17–27, 2021.

[53] A. Carlin, D. P. Manson, and J. Zhu, "Developing the cyber defenders of tomorrow with regional collegiate cyber defense competitions (ccdc).," *Information Systems Education Journal*, vol. 8, no. 14, p. n14, 2010.

[54] M. Zenko, *Red Team: How to succeed by thinking like the enemy*. Basic Books, 2015.

[55] Y. Diogenes and E. Ozkaya, *Cybersecurity??? Attack and Defense Strategies: Infrastructure security with Red Team and Blue Team tactics*. Packt Publishing Ltd, 2018.

[56] E. Seker and H. H. Ozbenli, "The concept of cyber defence exercises (cdx): Planning, execution, evaluation," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–9, IEEE, 2018.

[57] T. Sommestad and J. Hallberg, "Cyber security exercises and competitions as a platform for cyber security experiments," in *Nordic Conference on Secure IT Systems*, pp. 47–60, Springer, 2012.

[58] M. O. Leary, *Cyber Operations: Building, Defending, and Attacking Modern Computer Networks*. New York, NY: Appress, 2015.

[59] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 3, pp. 1–41, 2020.

[60] S. Eger and Y. Benz, "From hero to z\'eroe: A benchmark of low-level adversarial attacks," *arXiv preprint arXiv:2010.05648*, 2020.

[61] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[62] Y. Belinkov and Y. Bisk, "Synthetic and natural noise both break neural machine translation," *arXiv preprint arXiv:1711.02173*, 2017.

[63] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[64] R. Feldman, "Techniques and applications for sentiment analysis.," *Commun. ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[65] R. Marée, P. Geurts, G. Visimberga, J. Piater, and L. Wehenkel, "A comparison of generic machine learning algorithms for image classification," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 169–182, Springer, 2003.

[66] Y. Yu, W. Duan, and Q. Cao, "The impact of social and conventional media on firm equity value: A sentiment analysis approach," *Decision support systems*, vol. 55, no. 4, pp. 919–926, 2013.

[67] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[68] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *arXiv preprint arXiv:1908.07125*, 2019.

[69] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, 2014.

[70] L. Yue, W. Chen, X. Li, W. Zuo, and M. Yin, "A survey of sentiment analysis in social media," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 617–663, 2019.

[71] P. Team, ""jigsaw toxic bias faq"," 2019.

[72] J. West, "I tested 14 sentences for "perceived toxicity" using Perspectives.," 2017.

[73] JigSaw, ""perspective"," 2019.

[74] P. Druzhkov and V. Kustikova, "A survey of deep learning methods and software tools for image classification and object detection," *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 9–15, 2016.

[75] P. Dollár, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2007.

[76] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[77] 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.

[78] T. Kinnunen, J.-K. Kamarainen, L. Lensu, J. Lankinen, and H. Käviäinen, "Making visual object categorization more challenging: Randomized caltech-101 data set," in *2010 20th International Conference on Pattern Recognition*, pp. 476–479, IEEE, 2010.

[79] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and surface variations," *arXiv preprint arXiv:1807.01697*, 2018.

[80] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and surface variations," *arXiv preprint arXiv:1807.01697*, 2018.

[81] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 413–420, IEEE, 2009.

[82] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, "Novel dataset for fine-grained image categorization: Stanford dogs," in *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, vol. 2, Citeseer, 2011.

[83] I. L. S. V. R. Challenge, "Olga russakovsky, jia deng, hao su, jonathan krause, sanjeev satheesh, sean ma, zhiheng huang, andrej karpathy, aditya khosla, michael bernstein, alexander c. berg, li fei-fei. 2014," *Computing Research Repository, Vol. abs/1409.0575*.

[84] A. Tavanaei, "Embedded encoder-decoder in convolutional networks towards explainable ai," *arXiv preprint arXiv:2007.06712*, 2020.

[85] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[86] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "Mobilenetv2 model for image classification," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pp. 476–480, IEEE, 2020.

[87] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[88] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[90] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017.

[91] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[92] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman, "Nuanced metrics for measuring unintended bias with real data for text classification," in *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 491–500, ACM, 2019.

[93] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 416–423 vol.2, 2001.

[94] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416–423, July 2001.

[95] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[96] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proceedings of the British Machine Vision Conference*, pp. 135.1–135.10, BMVA Press, 2012.

[97] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*, pp. 711–730, Springer, 2010.

[98] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5197–5206, 2015.

[99] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," 2017.

[100] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery: A small target detection benchmark," *Journal of Visual Communication and Image Representation*, vol. 34, pp. 187–203, 2016.

[101] S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru, "A study of black box adversarial attacks in computer vision," *arXiv preprint arXiv:1912.01667*, 2019.

[102] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification.," *IJ Network Security*, vol. 15, no. 5, pp. 390–396, 2013.

[103] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[104] W. Churchill and M. T. Barnes, *My early life*. Eland London, 1930.

[105] C. Dickens, *The Speeches of Charles Dickens*. London: Chatto and Windus, 1884.

[106] F. S. Fitzgerald and J. Baughman, *A life in letters*. Simon and Schuster, 1994.

[107] R. F. Burton, *Arabian nights*, vol. 1. Library of Alexandria, 1965.

[108] M. Peckham *et al.*, "The origin of species by charles darwin. a variorum text.," *The origin of species by Charles Darwin. A variorum text.*, 1959.

[109] G. Flaubert, *The Letters of Gustave Flaubert: 1857-1880*, vol. 2. Harvard University Press, 1980.

[110] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[111] K. Florio, V. Basile, M. Polignano, P. Basile, and V. Patti, "Time of your hate: The challenge of time in hate speech detection on social media," *Applied Sciences*, vol. 10, no. 12, p. 4180, 2020.

[112] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[113] F. Chollet *et al.*, "Keras," 2015.

[114] P. Notin, A. N. Gomez, J. Yoo, and Y. Gal, "Sliceout: Training transformers and cnns faster while using less memory," *arXiv preprint arXiv:2007.10909*, 2020.

[115] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[116] M. Stephen, X. Caiming, B. James, and R. Socher 2016.

[117] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez, "Train large, then compress: Rethinking model size for efficient training and inference of transformers," *arXiv preprint arXiv:2002.11794*, 2020.

[118] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[119] A. Kumar, T. M. Sebastian, *et al.*, "Sentiment analysis: A perspective on its past, present and future," *International Journal of Intelligent Systems and Applications*, vol. 4, no. 10, pp. 1–14, 2012.

[120] L. Gao and R. Huang, "Detecting online hate speech using context aware models," *arXiv preprint arXiv:1710.07395*, 2017.

[121] J. Guberman, C. Schmitz, and L. Hemphill, "Quantifying toxicity and verbal violence on twitter," in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, pp. 277–280, ACM, 2016.

[122] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[123] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[124] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[125] "Perspective api." `https://www.perspectiveapi.com/#/home`. Accessed: 2019-11-11.

[126] D. Li, D. V. Vargas, and S. Kouichi, "Universal rules for fooling deep neural networks based text classification," *arXiv preprint arXiv:1901.07132*, 2019.

[127] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial attack and defense in reinforcement learning-from ai security view," *Cybersecurity*, vol. 2, no. 1, p. 11, 2019.

[128] A. C. Wright, "Orange is the new purple." BlackHat USA 2017, 2017.

[129] R. Pandarachalil, S. Sendhilkumar, and G. Mahalakshmi, "Twitter sentiment analysis for large-scale data: an unsupervised approach," *Cognitive computation*, vol. 7, no. 2, pp. 254–262, 2015.

[130] M. Birjali, M. Kasri, and A. Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends," *Knowledge-Based Systems*, vol. 226, p. 107134, 2021.

[131] L. Ljung, "Black-box models from input-output measurements," in *IMTC 2001. Proceedings of the 18th IEEE instrumentation and measurement technology conference. Rediscovering measurement in the age of informatics (Cat. No. 01CH 37188)*, vol. 1, pp. 138–146, IEEE, 2001.

[132] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[133] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 399–414, IEEE, 2018.

[134] N. Boucher, I. Shumailov, R. Anderson, and N. Papernot, "Bad characters: Imperceptible nlp attacks," *arXiv preprint arXiv:2106.09898*, 2021.

[135] A. McCarthy, P. Andriotis, E. Ghadafi, and P. Legg, "Feature vulnerability and robustness assessment against adversarial machine learning attacks," in *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1–8, IEEE, 2021.

[136] A. Serban, K. van der Blom, H. Hoos, and J. Visser, "Adoption and effects of software engineering best practices in machine learning," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–12, 2020.

[137] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[138] J. Ebrahimi, D. Lowd, and D. Dou, "On adversarial examples for character-level neural machine translation," *arXiv preprint arXiv:1806.09030*, 2018.

[139] W. Simoncini and G. Spanakis, "Seqattack: On adversarial attacks for named entity recognition," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 308–318, 2021.

[140] Y. Sharma and P.-Y. Chen, "Attacking the madry defense model with l_1-based adversarial examples," *arXiv preprint arXiv:1710.10733*, 2017.

[141] Y. Deng and L. J. Karam, "Universal adversarial attack via enhanced projected gradient descent," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 1241–1245, IEEE, 2020.

[142] T. Huang, V. Menkovski, Y. Pei, and M. Pechenizkiy, "Bridging the performance gap between fgsm and pgd adversarial training," *arXiv preprint arXiv:2011.05157*, 2020.

[143] 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*, pp. 2574–2582, 2016.

[144] J. Kalin, D. Noever, and G. Dozier, "Systematic attack surface reduction for deployed sentiment analysis models," *arXiv preprint arXiv:2006.11130*, 2020.

[145] J. Kalin, M. Ciolino, D. Noever, and G. Dozier, "Black box to white box: Discover model characteristics based on strategic probing," in *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pp. 60–63, IEEE, 2020.

[146] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1000–1008, 2020.

[147] J. Manning, D. Langerman, B. Ramesh, E. Gretok, C. Wilson, A. George, J. MacKinnon, and G. Crum, "Machine-learning space applications on smallsat platforms with tensorflow," 2018.

[148] D. A. Vakoch and M. F. Dowd, *The Drake equation: estimating the prevalence of extraterrestrial life through the ages*, vol. 8. Cambridge University Press, 2015.

[149] S. Wallenhorst, "The drake equation reexamined," *Quarterly Journal of the Royal Astronomical Society*, vol. 22, p. 380, 1981.

[150] R. I. Cook, "How complex systems fail," *Cognitive Technologies Laboratory, University of Chicago. Chicago IL*, 1998.

[151] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv preprint arXiv:1807.01069*, 2018.

[152] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *International Conference on Machine Learning*, pp. 4970–4979, PMLR, 2019.

[153] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672, IEEE, 2019.

[154] A. Dey, M. Velay, J.-P. Fauvelle, and S. Navers, "Adversarial vs behavioural-based defensive ai with joint, continual and active learning: automated evaluation of robustness to deception, poisoning and concept drift," *arXiv preprint arXiv:2001.11821*, 2020.

[155] J. Kleinberg and M. Raghavan, "Algorithmic monoculture and social welfare," *Proceedings of the National Academy of Sciences*, vol. 118, no. 22, 2021.

[156] S. McGregor, "Preventing repeated real world ai failures by cataloging incidents: The ai incident database," *arXiv preprint arXiv:2011.08512*, 2020.

[157] D. Geer, "Monoculture on the back of the envelope," 2021.

[158] trends.google.com, "Google trends," 2021.

[159] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[160] V. Thomas, F. Pedregosa, B. Merriënboer, P.-A. Manzagol, Y. Bengio, and N. Le Roux, "On the interplay between noise and curvature and its effect on optimization and generalization," in *International Conference on Artificial Intelligence and Statistics*, pp. 3503–3513, PMLR, 2020.

[161] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: a survey of case studies," *arXiv preprint arXiv:2011.09926*, 2020.

[162] D. S. Oliveira, T. Lin, M. S. Rahman, R. Akefirad, D. Ellis, E. Perez, R. Bobhate, L. A. DeLong, J. Cappos, and Y. Brun, "{API} blindspots: Why experienced developers write vulnerable code," in *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*, pp. 315–328, 2018.

[163] S. Seager, "The search for habitable planets with biosignature gases framed by a 'biosignature drake equation'," *International Journal of Astrobiology*, vol. 17, no. 4, pp. 294–302, 2018.

[164] "Finding love with a modified drake's equation." `https://www.mathgoespop.com/2010/02/finding-love-with-a-modified-drakes-equation.html`. Accessed: 2021-06-10.

[165] R. Mittal, C. Meneveau, and W. Wu, "A mathematical framework for estimating risk of airborne transmission of covid-19 with application to face mask use and social distancing," *Physics of Fluids*, vol. 32, no. 10, p. 101903, 2020.

[166] "From aliens to ai: Using the drake equation to debate existential risk." `https://towardsdatascience.com/from-aliens-to-ai-using-the-drake-equation-to-debate-existential-risk-7deed14779da` Accessed: 2020-06-10.

[167] "Legal services national technology assistance project." https://www.lsntap.org/node/207/webinar-drake-equation-access-justice. Accessed: 2021-06-10.

[168] R. Amini, "" should i break up with my girlfriend? will i find another?" or: An algorithm for the forecasting of romantic options," *arXiv preprint arXiv:1501.00637*, 2015.

[169] "arxiv dataset and metadata of 1.7 m+ scholarly papers across stem," 2020.

[170] "Adversarial ml threat matrix." \textbf{https://www.kaggle.com/Cornell-University/arxiv}, note = Accessed: 2021-06-10.

[171] U.S. Department of Defense, *Work Unit Codes for Aeronautical Equipment; Uniform Numbering System*, 1984.

[172] U.S. Department of Army, *Functional Users Manual for the Army Maintenance Management System – Aviation*, 2014.

[173] W. J. Clancey, *Classification problem solving*. Stanford University Stanford, CA, 1984.

[174] K. Xu, M. Lam, J. Pang, X. Gao, C. Band, P. Mathur, F. Papay, A. K. Khanna, J. B. Cywinski, K. Maheshwari, *et al.*, "Multimodal machine learning for automated icd coding," in *Machine Learning for Healthcare Conference*, pp. 197–215, PMLR, 2019.

[175] D. W. Hasling, W. J. Clancey, and G. Rennels, "Strategic explanations for a diagnostic consultation system," *International Journal of Man-Machine Studies*, vol. 20, no. 1, pp. 3–19, 1984.

[176] J. Rice, "Poligon, a system for parallel problem solving," in *Proceedings of the Expert Systems Workshop*, pp. 152–159, 1986.

[177] A. L. Robinson, "New ways to make microcircuits smaller: Making integrated circuits has always been more art than science, but as miniaturization proceeds unabated more science is needed," *Science*, vol. 208, no. 4447, pp. 1019–1022, 1980.

[178] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[179] J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," *Artificial Intelligence in Medicine*, vol. 104, p. 101822, 2020.

[180] M. Arnold, J. Boston, M. Desmond, E. Duesterwald, B. Elder, A. Murthi, J. Navratil, and D. Reimer, "Towards automating the ai operations lifecycle," *arXiv preprint arXiv:2003.12808*, 2020.

[181] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: efficient and robust automated machine learning," in *Automated Machine Learning*, pp. 113–134, Springer, Cham, 2019.

[182] E. LeDell and S. Poirier, "H2o automl: Scalable automatic machine learning," in *Proceedings of the AutoML Workshop at ICML*, vol. 2020, 2020.

[183] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, and H. Su, "Trust in automl: exploring information needs for establishing trust in automated machine learning systems," in *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pp. 297–307, 2020.

[184] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[185] H. Yasar, "Leveraging devops and devsecops to accelerate ai development and deployment," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH United States, 2020.

[186] H. Myrbakken and R. Colomo-Palacios, "Devsecops: a multivocal literature review," in *International Conference on Software Process Improvement and Capability Determination*, pp. 17–29, Springer, 2017.

[187] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of devsecops," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, IEEE, 2019.

[188] G. Fursin, "The collective knowledge project: making ml models more portable and reproducible with open apis, reusable best practices and mlops," *arXiv preprint arXiv:2006.07161*, 2020.

[189] A. Angelopoulos, E. T. Michailidis, N. Nomikos, P. Trakadas, A. Hatziefremidis, S. Voliotis, and T. Zahariadis, "Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects," *Sensors*, vol. 20, no. 1, p. 109, 2020.

[190] "Introducing the infosec colour wheel - blending developers with red and blue security teams."

[191] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[192] D. R. Insua, R. Naveiro, V. Gallego, and J. Poulos, "Adversarial machine learning: Perspectives from adversarial risk analysis," *arXiv preprint arXiv:2003.03546*, 2020.

[193] L. Russo, F. Binaschi, and A. De Angelis, "Cybersecurity exercises: Wargaming and red teaming," *Next Generation CERTs*, vol. 54, p. 44, 2019.

[194] L. Baier, F. Jöhren, and S. Seebacher, "Challenges in the deployment and operation of machine learning in practice," 2019.

[195] M. Kont, M. Pihelgas, K. Maennel, B. Blumbergs, and T. Lepik, "Frankenstack: Toward real-time red team feedback," in *MILCOM 2017-2017 ieee military communications conference (milcom)*, pp. 400–405, IEEE, 2017.

[196] J. M. Spring, A. Galyardt, A. D. Householder, and N. VanHoudnos, "On managing vulnerabilities in ai/ml systems," in *New Security Paradigms Workshop 2020*, pp. 111–126, 2020.

[197] J. Kalin, D. Noever, and M. Ciolino, "A modified drake equation for assessing adversarial risk to machine learning models," *arXiv preprint arXiv:2103.02718*, 2021.

[198] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software engineering for machine-learning applications: The road ahead," *IEEE Software*, vol. 35, no. 5, pp. 81–84, 2018.

[199] Y. Mirsky, A. Demontis, J. Kotak, R. Shankar, D. Gelei, L. Yang, X. Zhang, W. Lee, Y. Elovici, and B. Biggio, "The threat of offensive ai to organizations," *arXiv preprint arXiv:2106.15764*, 2021.

[200] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Guéhéneuc, "Studying software engineering patterns for designing machine learning systems," in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, pp. 49–495, IEEE, 2019.

[201] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291–300, IEEE, 2019.

[202] E. Ungan, N. Cizmeli, and O. Demirörs, "Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 77–80, IEEE, 2014.

[203] K. Singla, T. Vinayak, A. Arpitha, C. Naik, and J. Bose, "Story and task issue analysis for agile machine learning projects," in *2020 IEEE-HYDCON*, pp. 1–4, IEEE, 2020.

[204] T. H.-C. Hsu, *Practical security automation and testing: tools and techniques for automated security scanning and testing in devsecops*. Packt Publishing Ltd, 2019.

[205] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," *Mitre Product Mp*, pp. 18–0944, 2018.