THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

COLLEGE OF INFORMATION SCIENCES AND TECHNOLOGY

APPLIED DEFENSES AGAINST ADVERSARIAL ATTACKS

COLLIN PAYNE
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Security and Risk Analysis

Reviewed and approved* by the following:

Edward J. Glantz
Teaching Professor of IST
Thesis Supervisor & Honors Adviser

Alison Ryan Murphy
Assistant Teaching Professor of IST
Faculty Reader

* Electronic approvals are on file.

ABSTRACT

"Do you really want to see what it looks like when two gods go to war?" This quote from the show *Person of Interest* describes how many see the future of adversarial machine learning. Machine learning has recently become popular among companies and institutions who are researching the various ways to apply machine learning models. These machine learning models can be beneficial, but there are also many concerns.

Machine learning is the use of statistics and algorithms to find patterns in the data underlying artificial intelligence applications, such as those used in search engines and in the recommendation systems used by Amazon and Netflix. The evolution of machine learning, artificial intelligence, deep learning and neural networks has created an almost "god-like" world where models run and learn without human supervision.

The trend towards an altruistic "machine-god" might actually be considered acceptable by some if not for the introduction of "adversarial machine learning" where an attacker intentionally fools or misguides the models with malicious input. The resolution to this scenario, and the purpose of this research, is the introduction of a "defensive" machine-god to offset and weaken the impact of this adversary. Thus, to answer the opening question, we do want to see two gods go to war. Otherwise, the adversary alone controls and dominates machine learning's models.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First, I would like to thank my Thesis Supervisor, Dr. Glantz. He has not only been very encouraging throughout the process, but he always provided a clear path forward to keep things running smoothly throughout my time creating this thesis. I am certain my thesis would not be where it is today without his guidance.

I would also like to thank my reader, Dr. Murphy, who throughout my time at Penn State has always been uplifting, supportive, and caring. Her optimism in pursuing our dreams and innovative ideas is contagious to the Penn State community and her students.

I want to mention my machine learning professor, Dr. Mehrdad Mahdavi, whose class inspired me to conduct a thesis in machine learning. His passion for machine learning is radiant to those around him.

Lastly, but certainly not least, I want to thank my family for the help and support they have provided to me throughout not only college, but also my entire life. I want to especially thank my mother for always being there for me and looking out for my best interest, even during hard times in our lives. I also want to thank my sister for helping me proofread this work.

Chapter 1

INTRODUCTION

In some regards, current trends in adversarial machine learning (AML) parallel previous trends in cybersecurity. Network administrators attempted to design secure perimeters with little understanding of the attacker's tools and techniques. Breaches continued to result until administrators began testing network security using friendly adversaries.

The same could be said of developers that began with an emphasis on code functionality independent of operation within a hostile environment. As with network security, this gap is now being closed with secure coding principles and testing as well.

There should be no surprise then that machine learning models were also rapidly developed with a focus on performance independent of the presence of intentional compromise. With lessons from network security and secure coding, the opportunity now exists to harden machine learning models to not only perform effectively but to also withstand intentional compromise.

All that is needed is to think like the attacker and apply the attacker's methodologies in the form of a friendly adversary. This paper provides an introduction to recent innovations capable of performing these tasks. The importance of implementing these practices sooner rather than later goes back to the "battling gods" quote. The development of machine learning is expected to evolve much quicker than network security or programming. Therefore, this could possibly create severe vulnerabilities at a much quicker pace if left unaddressed. It is better to have gods battling for both sides, rather than having only one benefitting from the compromise.

Chapter 2

MACHINE LEARNING MODELS

Throughout this paper "supervised" machine learning models will be used as examples. Supervised machine learning models are "trained" using input data (x) and expected output data (y). The algorithm is provided with new input data, and based on previous training, determines the output. A simple example in meteorology would be to determine the output, "today's weather forecast," given the inputs barometric pressure, humidity, temperature, and cloud cover (Soni, 2019, para. 2).

Each column of the data is a different attribute or characteristic (e.g., height, temperature,) and each row of the data is a unique instance or data point. Therefore, each data point has an expected output. Supervised models have an expected output associated with its data, but unsupervised models do not. Although supervised and unsupervised models are both vulnerable to AML, the scope of this paper is limited to the more common implementation of supervisory machine learning models, which are more explicit and clearer to understand (Soni, 2019, para. 1).

Type	*Attributes*
Supervised Learning	It is based on labeled training data.
Unsupervised Learning	It is based on unlabeled data. It is used for finding unseen relationships in the data independent of class label. It is used in clustering.
Semi-supervised Learning	It is combination of labeled and unlabeled data.

Figure 1. The presence of labels on training data differentiates supervised and unsupervised learning.

The model describes the method followed by the algorithm to determine the output. Based on experience and intuition, machine learning scientists develop models which are most likely to be effective based on their knowledge of the data. As there are many models available, the method for a particular

problem is normally selected by testing a variety of models to determine which provides the greatest accuracy.

In order to see which model has the highest accuracy, the data needs to be trained on various models. To train a model, the data needs to be grouped into (a) training and (b) test data. The model's algorithm uses training data to compare predicted with provided outputs and "learns" from differences by making necessary model adjustments. The model then uses test data to validate any algorithm modifications made during training (Shah, 2017, para. 3).

All supervised machine learning models follow the basic pattern of training and then testing. The type of model is determined by the method used to predict the output based on inputs and how the model is corrected during the training process. The process for correcting a machine learning model is in part determined by what is called a "loss function." Typically, a loss function examines the difference between the predicted and correct output and produces the amount of error as a result. Then, an optimization algorithm seeks to minimize this loss, which decreases the amount of error (Mahendru, 2019, para. 8).

There are many different machine learning models that exist, but some are more popular than others because they are believed to be more practical and more accurate compared to lesser known models. The list contains the estimated top ten commonly used machine learning algorithms:

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. Support Vector Machine (SVM)
5. Naïve Bayes
6. K-Nearest Neighbor (kNN)
7. K-Means
8. Random Forest
9. Dimensionality Reduction Algorithms

10. Gradient Boosting Algorithms, which include:
    a. Gradient Boosting Machine (GBM)
    b. Extreme Gradient Boosting (XGBoost)
    c. LightGBM
    d. CatBoost

These models are frequently used to solve machine learning problems (Ray & Business Analytics and Intelligence, 2020, para. 9).

Computer vision is a common application of machine learning. In the context of computer vision, machine learning is used to recognize objects in images as shown in **Figure 2** (Azati, 2019, para. 18).

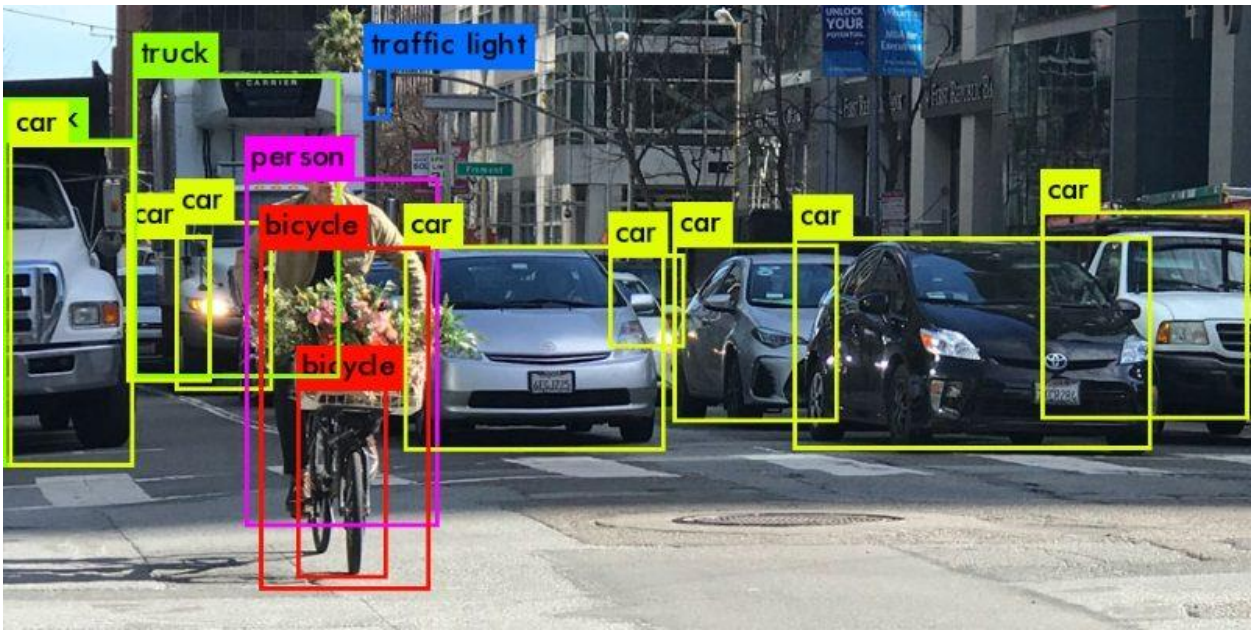

Figure 2. Machine learning models classify vehicles, people, and other objects in a city intersection (Azati Software, 2019, para. 1).

There are also techniques used to increase model accuracy. Image pre-processing is an example of a technique used in computer vision to reduce "noise" in an image. In computer vision, noise is defined as "an abrupt change in pixel values in an image." An image is "filtered" when noise is removed, simplified, or cleaned of data that would have made it more difficult for the model to make correct

predictions. This pre-processed image is then given to machine learning algorithms which improves the mode's accuracy when training or testing. Intuitively, this helps the model distinguish objects when looking at an image (Swain, 2019, 2).

Chapter 3

ADVERSARIAL MACHINE LEARNING

This paper defines adversarial machine learning as malicious inputs that change or reduce confidence in machine learning model outputs. In addition, an adversarial example is an oftentimes small but intentional change in input that causes machine learning perturbations resulting in false predictions. In other words, the machine learning model is "tricked" into making an incorrect prediction following a carefully engineered input.

Research into AML began in the early 2000's. By 2013, research showed that many of the most commonly used machine learning algorithms were vulnerable to attacks, including linear classifiers, support vector machines (SVMs), and (deep) neural networks. Successful attacks using AML on spam filters and PDF malware detectors were also demonstrated in research. Shortly thereafter, it was shown that nonlinear classifiers can also be attacked. (Biggio & Roli, 2018, 9).

Figure 3 depicts how adversarial examples appear in a graph. For demonstration purposes, suppose the model is trying to classify whether the image is a vehicle or not. The dotted black line represents the decision boundary for what we want the model to classify (e.g., a vehicle). Inside the dotted lines represents the area of where a vehicle would be plotted, and outside the dotted lines represents the area that is not a vehicle. The solid black line is the model's decision boundary, which is the point where the model will change its decision between "vehicle" and "not vehicle" if that line is crossed. The model classifies a point inside the solid lines as a vehicle, and classifies a point outside of the solid lines as not a vehicle. Correctly classified points are shown as blue X's, and adversarial examples are shown as red X's. Notice that the adversarial examples are slight changes in the input and, therefore, the location of the data point (Goodfellow et al., 2018, 9).

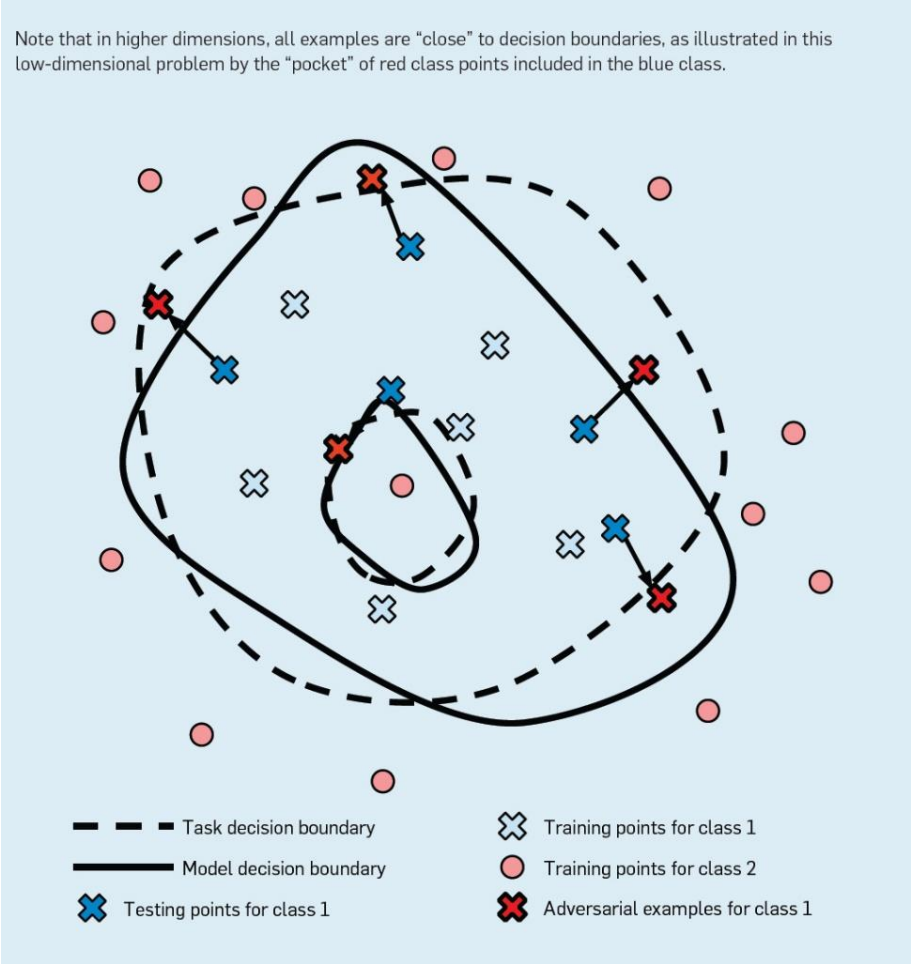

Figure 3. Adversarial examples are engineered by slightly modifying a correct example close to the decision boundary (Goodfellow et al., 2018, para. 9).

While it is tempting to associate model accuracy with vulnerability, they should not be compared to one another. An important note about AML is that adversarial examples do not happen by chance. Instead, they are mathematically engineered examples that purposely cause the model to misclassify an input. Therefore, a model's vulnerability to adversarial examples is not a reflection on the model's accuracy. Even models with very high accuracy can be extremely vulnerable to adversarial examples.

Early thinking by researchers was that adversarial examples existed randomly and were specific to each model. However, when researchers tried to use model stacking, or the combining the results of multiple models, they found that the combined model's vulnerability did not significantly decrease when compared to the original model. If the adversarial examples existed randomly, stacking should have fixed

the problem. More recent research revealed “black-box” adversarial attacks (i.e., malicious inputs) that can be generalized from one model to another, yielding adversarial results without even knowing which model or data was used in training. Black-box attacks are discussed further in the next section (Goodfellow et al., 2015, 7).

Chapter 4

VARIOUS TYPES OF ATTACK ON MACHINE LEARNING SYSTEMS

White and black-box attacks are attacks categorized by the amount of information originally known about the targeted machine learning model. White-box attacks occur when the attacker has knowledge of the model's architecture or training data, whereas a black-box attacker's knowledge is limited to the model's inputs and outputs. **Figure 4** shows the inverse relationship between the attacker's knowledge and attack difficulty. In addition, note that there is a direct relationship between attack complexity and attack difficulty (Goodfellow et al., 2018, para. 12).

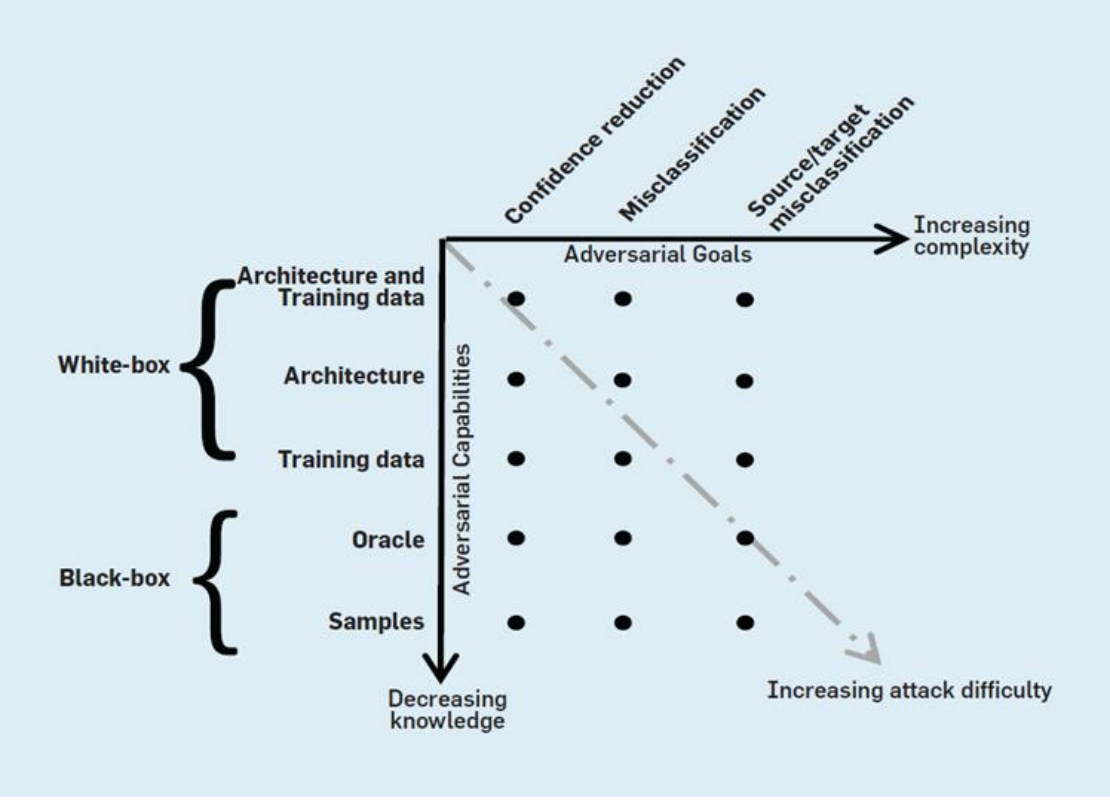

Figure 4. Attack difficulty increases with more complex goals and more limited knowledge of the machine learning model. The black-box attacker has limited knowledge, and thus more limited capabilities, than the white-box attacker (Goodfellow et al., 2018, para. 12).

For example, a model that has publicly published the source of its training data or its composition would be vulnerable to a white-box attack. In addition, the "insider threat," or person with malicious

intent familiar with the model or training data, is a classic example of someone aiding or performing a white-box attack (Goodfellow et al., 2018, para. 17).

A "poisoning attack" is a subset of white-box attacks. A poisoning attack is when a machine learning model's training data is modified by the adversary. For example, a machine learning model which is dynamic and retrains itself based on data in its environment is vulnerable to this type of attack. By changing the model's data, the adversary can cause the model to reduce confidence in its classifications (outputs), or even cause misclassification.

"Static" models that are manually trained only one-time are generally not vulnerable to this type of attack as long as the original data is not poisoned. Static models reduce, but do not eliminate, the vulnerability. "Semi-static" models that only update occasionally, such as monthly or yearly, based on new training data are still vulnerable. In addition, a model could be retrained by a cyber-attack or insider-threat (Goodfellow et al., 2018, para. 14).

Black-box attacks are typically used when the attacker has no knowledge of the model's inner workings or training data. These attacks pose a broader threat than white-box attacks because the adversary does not need any special permissions or knowledge to attack these models. Any model that is outward or public facing (i.e. its outputs are exposed to the adversary) is vulnerable to black-box attacks (Goodfellow et al., 2018, para. 17).

Chapter 5

ADVERSARIAL MACHINE LEARNING RELATED TO COMPUTER VISION

In the context of safety and security, the threat of AML applies to any machine learning model that is relied upon to make decisions that have serious consequences. For example, fraud and cyber security are two areas where the consequences of an AML attacks increase. The following research focuses on computer vision attacks, which includes how AML is used to trick computer vision into reducing the confidence of classification or into misclassifying objects.

Computer vision works like other machine learning models. The input is a photo broken into pixels that are likely pre-processed to increase the accuracy of the model. Then, machine learning is applied onto this modified image to identify objects.

Computer vision has a big market in the future, which includes autonomous vehicles that rely on computer vision to correctly spot and classify speed limit signs, stop signs, traffic signals, and other vehicles. As shown in **Figure 5**, prior studies have demonstrated how traffic signs can be transformed into adversarial attacks by modifying the signs in various ways, including adding stickers or graffiti to the sign. This attack causes the computer vision models to misclassify the traffic sign as a different type of sign or object (Early, 2019, para. 6).

Figure 5. Modified traffic signs disguised as graffiti can be effective adversarial examples (Early, 2019, para.6).

This modification to the signs can be dangerous because a vehicle that relies on computer vision to make decisions may be forced to turn off the road or into a different lane of traffic. **Figure 6** shows an overview of how a car would see a misplaced black line on the road, misclassify that black line as a line for the roadway, and start following that line instead of the double yellow line (Dossman, 2019, para. 5).

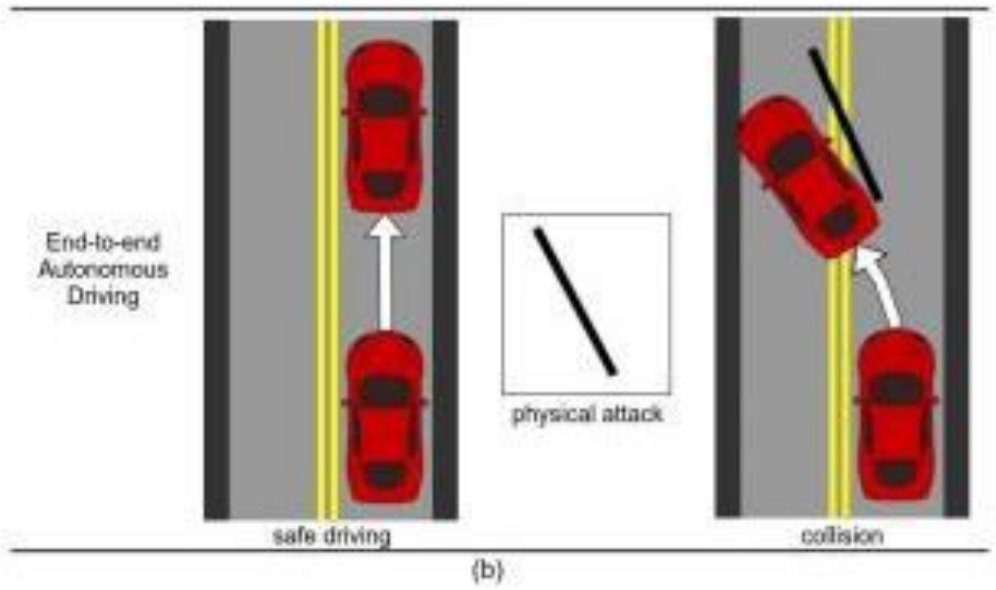

Figure 6. Marks on the road, such as a simulated skid mark, can cause self-driving cars to crash (Dossman, 2019, para. 5).

Figure 7 shows a more realistic simulation of what drivers would see if their car followed the adversarial double black lines instead of continuing to follow the double yellow lines that traditionally

guide traffic on a road. The car turns left, thinking the roadway has suddenly shifted to the left. As a result, the vehicle is tricked by the adversarial attack and collides with the guard rail (Dossman, 2019, 6).

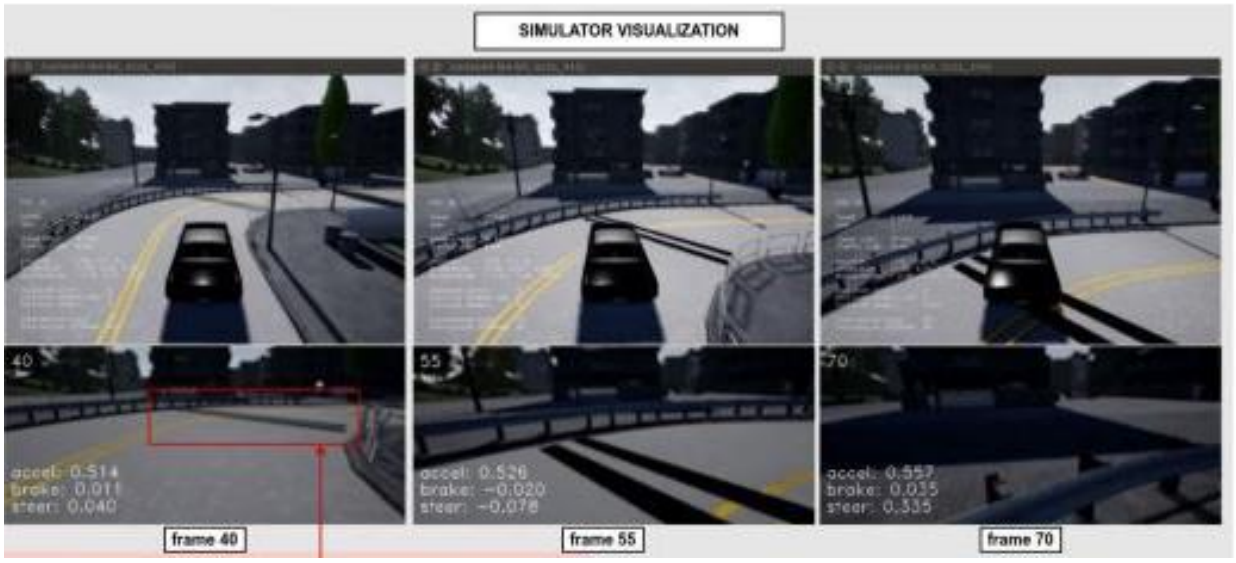

Figure 7. In this realistic simulation, a self-driving car was tricked into crashing by making the car turn left instead of right (Dossman, 2019, para. 1).

Some researchers state that the threat of AML attacks in computer vision can be diminished if they require advanced knowledge of the model (i.e. white box attacks) or if the attack only works at a certain angle. Other researchers have said that because self-driving cars rely on various sensors, tricking one sensor would not be enough to trick the entire model. However, researchers who have tested adversarial attacks using road markings found that they could trick the current car sensors. In April 2019, researchers tricked a self-driving Tesla into changing lanes on a road using only three stickers placed on the road. This is dangerous given how easy it would be to deploy this type of an attack and how quickly it could be done without anyone noticing (Tencent Keen Security Lab, 2019, 33).

These type of roadway attacks could become popular with malicious "pranksters," like those who throw rocks off bridges to get cars to swerve. If a technical person releases the details of how to execute this type of attack online, people may implement the attack without realizing the full implications.

The dangerous implications of AML not only apply to commercial products like cars, but also to military systems. For example, the military frequently uses computer vision in its aircraft tracking systems. This is a high consequence area because AML could be used as a form of stealth. Suppose a computer vision system on a drone is used to identify and track targets, and the drone is on a surveillance or combat mission. If an enemy vehicle or person places an adversarial example on their vehicle or body, it could cause the computer vision system to misclassify the vehicle or person as an inanimate object or an object of nature. Even if a human operator saw the target, it may be more difficult for the operator to properly track their target since the automatic tracking mechanism on the drone could fail. This may be an acceptable risk for ground-to-ground or air-to-ground missions, but air-to-air missions would be much more difficult since the automatic tracking systems are crucial at high speeds and for weapons targeting objects like missiles.

We know that AML works by making small changes to correctly classified examples. This makes computer vision particularly vulnerable to AML since examples changed by the adversary may not be detectable by the human eye, and even if a person does notice something different about an object, they may not automatically suspect that it is an adversarial example. This phenomenon has been documented extensively. One study described how a 3D printed turtle was correctly identified as a turtle by a neural network. However, as **Figure 8** shows, a slightly modified version of that 3D printed turtle was then classified by the neural network as a rifle due to the modification (Molnar, 2019, para. 21).

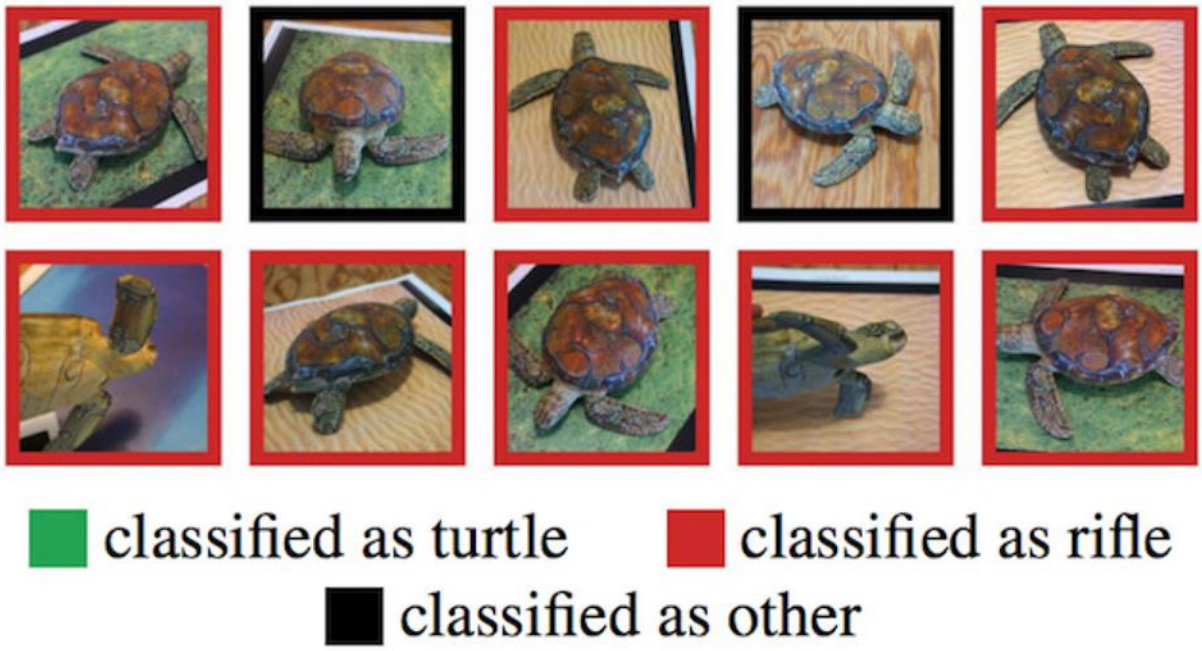

Figure 8. Adversarial examples also exist in 3D space, such as this turtle being incorrectly identified as a rifle (Molnar, 2019, para. 21).

Figure 8 illustrates how making specific modifications to a specific object, the 3D printed turtle, can lead to a misclassification by the system. However, many researchers are also asking: is it possible to create a generalized computer vision attack that could be applied to any object?

In one case, researchers engineered a patch that can be physically placed near any object, and the patch can cause the system to misclassify that object. Results will vary based on the specific model used, but they found that regardless of the object it was used on, it was effective. **Figure 9** shows a simple example of a banana being misclassified as a toaster after the addition of the adversarial patch that was created using a white-box attack (Molnar, 2019, para. 17).

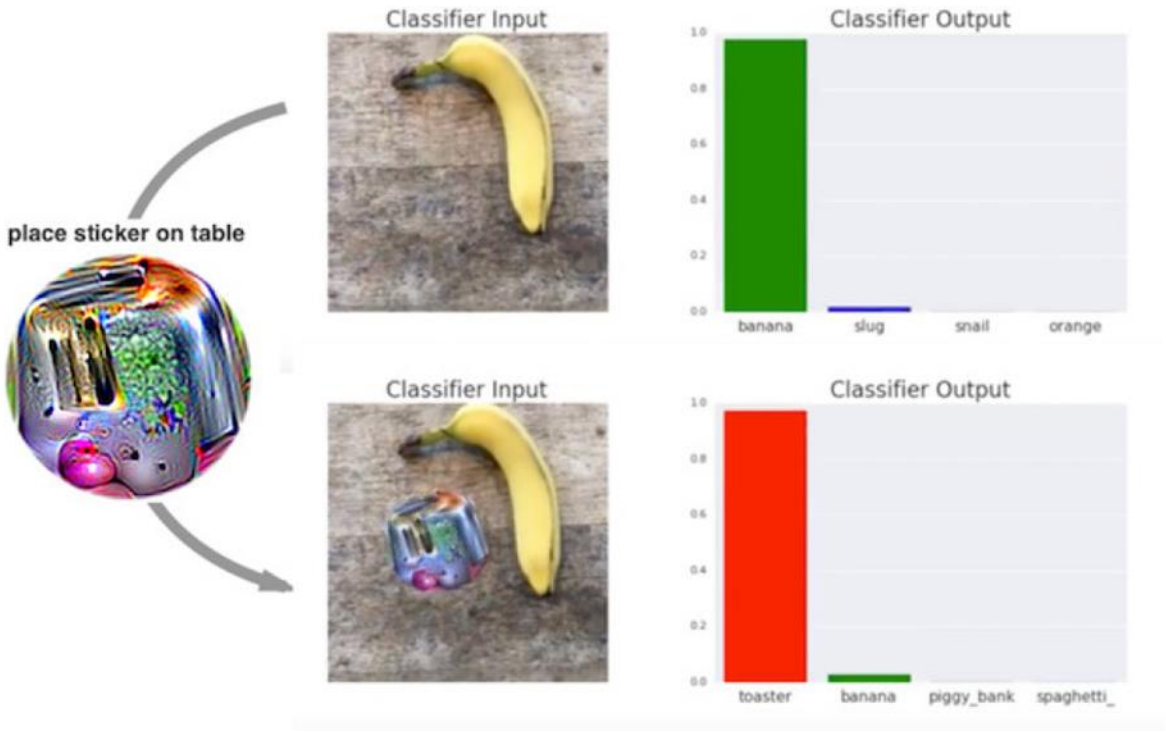

Figure 9. This white-box adversarial patch can cause a model to misclassify any object it is shown with as a toaster (Brown et al., 2018, 2).

Researchers were also able to generate a black-box adversarial patch. Since specifics about the model were not as well-known as in a white-box attack, it was not quite as effective as the white-box attack patch, but the results are still impressive. **Figure 10** displays a graph where the black-box attack patch's success rate is represented by the blue line. As the graph illustrates, if the patch covers up 30% of an image, it has a greater than 90% attack success rate. Even if the patch only covers 20% of an image, the attack success rate is still around 85% (Brown et al., 2018, 4).

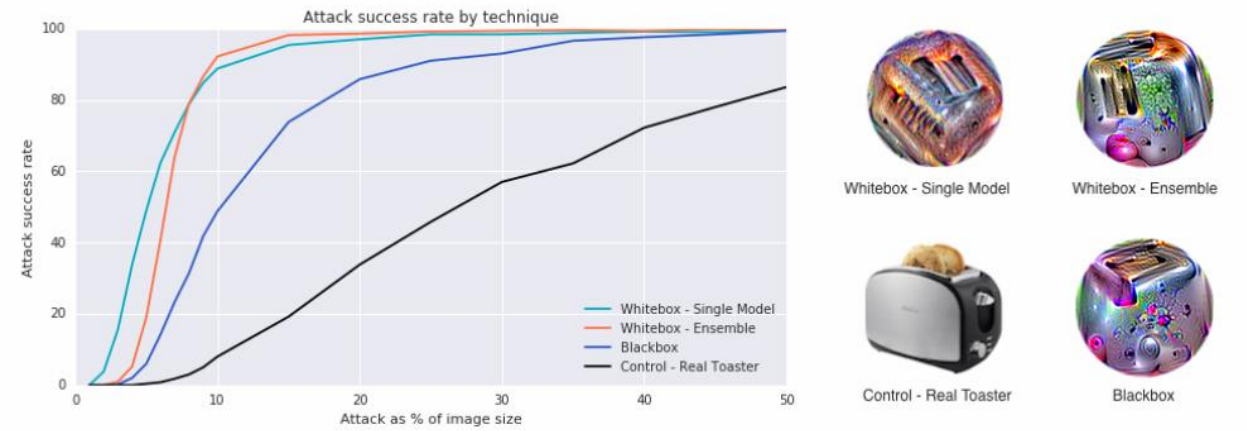

Figure 10. This black-box adversarial patch, even with a low coverage percentage of an image, can trick a model into classifying the object as a toaster (Brown et al., 2018, 4).

The researchers went further by creating disguised adversarial patches. When looking at the original black-box adversarial patch in **Figure 10**, the design can be seen by some as strange. So, the researchers added more acceptable imagery, like tie dye and peace symbols, to disguise the original design and make the patch appear more normal. As seen in **Figure 11**, these disguised adversarial patches are not as effective, but they could be placed on an object that an adversary wants to be misclassified without being seen as abnormal to someone looking at it.

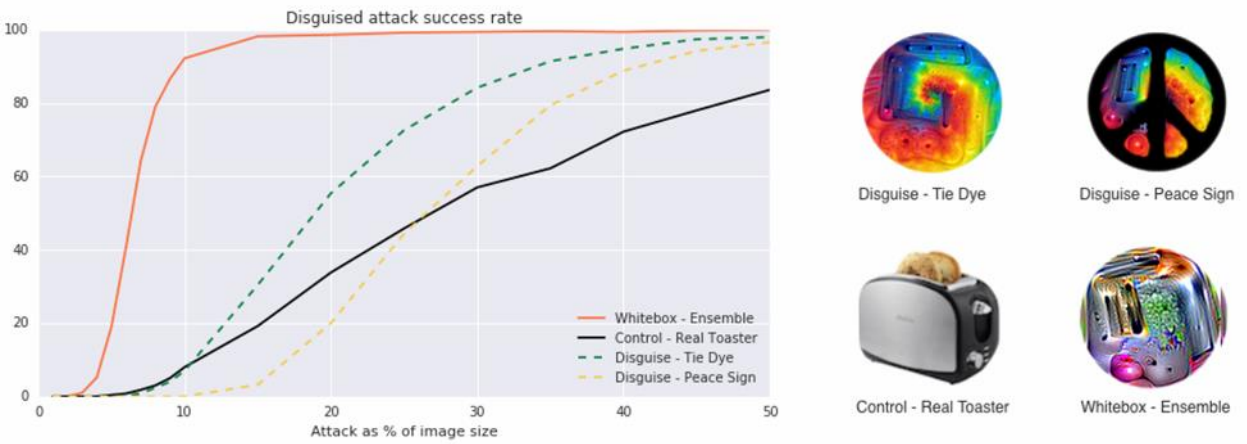

Figure 11. While requiring slightly more coverage than the ensemble patch, these disguised patches provide a good attack success percentage (Brown et al., 2018, 4).

Overall, AML poses a serious threat to computer vision today and the threat will continue to grow in the future. In the context of autonomous vehicles, adversarial examples placed on traffic signs or on the

road could have devastating consequences as mentioned earlier in the chapter. However, this attack on traffic signs could be done without using any sort of special design, and instead could simply have an adversarial patch sticker placed onto signs. In military contexts, instead of redesigning the shape or properties of vehicles, vehicles may just have adversarial patches attached to them to hide them from computer vision and throw off tracking systems. These are just a two of the many possible threats posed by AML now that attacks can be generalized to a cheap, portable physical attack such as the adversarial patch sticker.

Chapter 6

DEFENDING AGAINST ADVERSARIAL MACHINE LEARNING ATTACKS

In general, there are two ways to attack a machine learning model. The first way is to attack a model while it is learning, which corresponds to poisoning a dataset. The second is to attack the model when it is making predictions, which corresponds to giving adversarial examples to the model (Papernot & Goodfellow, 2016, para. 11).

Machine learning models can be analyzed in terms of achieving the cybersecurity goals of confidentiality, integrity, and availability (CIA.)

- Confidentiality: Classify and protect training data and information used to create the model from disclosure (e.g., medical field)
- Integrity: Protect against input modification that tricks the model into giving the incorrect output (focus on adversarial examples)
- Availability: Safeguard against input processing shutdowns or temporary halts caused by adversarial examples

Availability violations will be most common in physical systems powered by machine learning since the actions of a physical system have a physical effect that cannot be undone or dismissed. For instance, because an adversarial example causes a vehicle or piece of machinery to move or perform unexpectedly, it may be unable to perform other operations that it normally could in regular conditions (Papernot & Goodfellow, 2016, para. 6).

Keeping Training Data Private

It is important for a model to keep its information private. If a model is open-facing, adversaries will attempt to uncover the original training data from the model and attempt to violate the goal of

confidentiality. They can do this by using something known as membership inference queries. These queries are used to determine if a specific training point was used in the training set. This can be determined by the confidence rating of various inputs in order to see what training data points were used to construct the model. This can yield valuable information to the attacker in terms of what adversarial examples may be used to trick the model (Papernot & Goodfellow, 2016, para. 19).

While this privacy may not be important for some datasets, for others it may be very important. For example, datasets that have personally identifiable information (PII) should be protected. Therefore, a concept called differential privacy can be used to protect this data. Differential privacy provides a way to assess what privacy guarantees an algorithm can make. As mentioned, protecting the data is good for both privacy and to prevent attacks, so shielding the data that the model was trained on can be used as defense against attackers (Papernot, 2019, para. 1).

There are a set of algorithms called Private Aggregation of Teacher Ensembles (PATE), which can be used to hide the original training data. This can be done by training multiple models on mutually exclusive subsets of the original training data, which may include adding, removing, or modifying data points. A shown in **Figure 12**, these different models then vote on the correct output, and their voting response also contains added noise. This effectively masks the original training data set (Papernot & Goodfellow, 2018, para. 10).

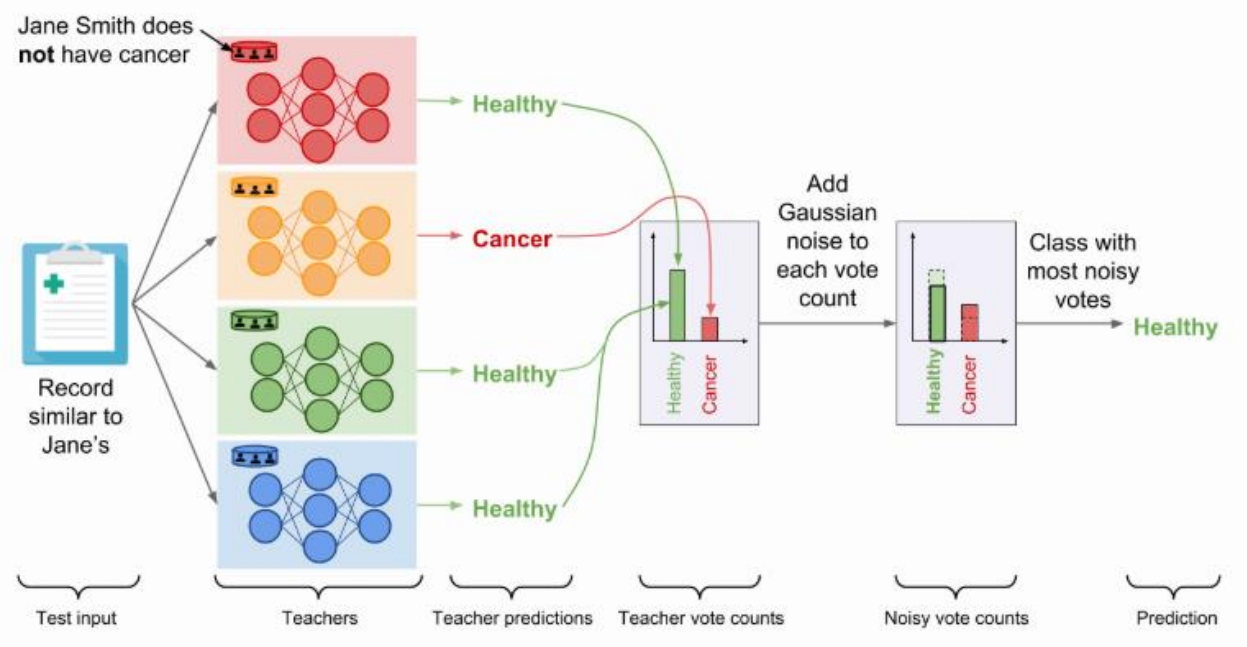

Figure 12. PATE trains different models on different subsets of training data and adds noise to hide the original training data. (Papernot & Goodfellow, 2018, 13)

Defending against Adversarial Examples

A technique called "adversarial training" can be used to generate adversarial examples during the training of a model. In this case, the model is trained to label those generated adversarial examples with the correct label rather than the incorrect label. This helps the model become more robust against adversarial examples (Goodfellow & Papernot, 2017, para. 5).

Another defense technique is to use defensive distillation, which smooths the model's decisions surface. Distillation refers to a model predicting the probabilities (output) of another model. The original model has the raw training data, but the second "distilled" model contains probabilities from the first model, reducing the likelihood of adversarial examples existing for the second "distilled" model (Goodfellow & Papernot, 2017, para. 6).

It is difficult to come up with a general solution to adversarial examples, primarily because the number of possible inputs for a given model is extremely high or infinite, and there is currently a lack of

research to create a theoretical model. Many current defenses only work for a specific attack, so there is no “one size fits all” defense approach (Goodfellow & Papernot, 2017, para. 16).

Measuring the Vulnerability of a Model to Adversarial Examples

Generally, testing is used to check robustness against adversarial machine learning, but verification can prove to be more useful in protecting against a broader range of adversarial examples. Testing provides a lower limit on the number of mistakes the model will make, while verification can provide an upper limit. Testing can be effective for certain attacks. However, if a model passes tests for one type of attack, it does not necessarily mean it will pass for another type of attack. Although this is an interesting topic of investigation for future research, there does not exist a way to verify all examples at this time (Goodfellow & Papernot, 2017, para. 11).

Part of the reason for this is that verification makes some assumptions in order to work properly. For example, in order for current verification techniques to work on a neural network, certain assumptions must be made about whether a hidden layer is relevant or not. Another assumption about the ability for verification to work is that a check exists within a certain range within each point. Verification assumes that the range within each point will have the same output, but this assumption will cause some mistakes. While this is generally true, it can be safely extrapolated that this can cause issues in some cases (Goodfellow & Papernot, 2017, para. 11).

Figure 13 illustrates how verification is inherently more powerful than testing because of the guarantees it can provide for model accuracy and robustness against adversarial attacks. The verification method proposed is similar to machine learning models such as k-nearest neighbor that work off of a similar principle. In this case, the nearest point(s) serve as the verification model rather than the prediction model. Some task decision boundaries, however, are more complex than the number of points that are close to a certain area (Goodfellow & Papernot, 2017, para. 16).

A topic for further research could be to hypothesize that the more data there is, the more accurate this form of verification will be. The accuracy of the verification would also depend on how complex the task decision boundary is, as well as how many points are near the task decision boundary.

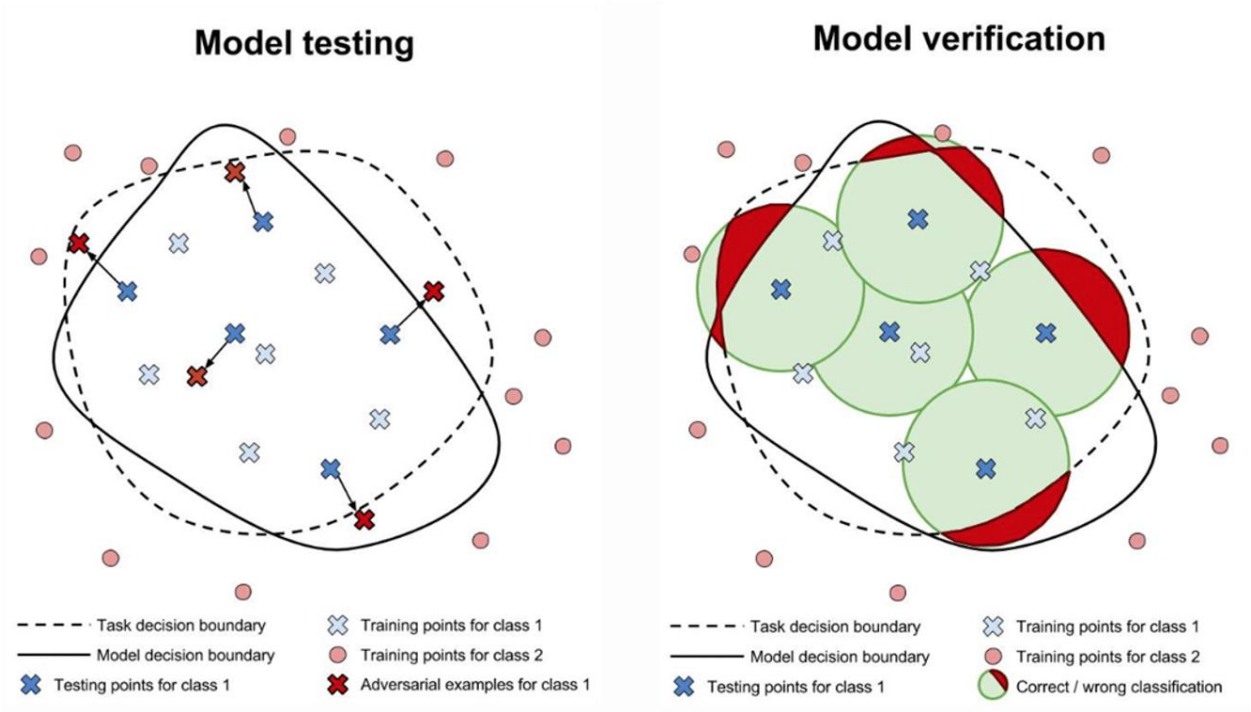

Figure 13. Verification provides wide swaths of coverage compared to testing, which only checks specific points (Goodfellow & Papernot, 2017, para. 16).

Since verification techniques are not yet fully developed, they should be used with some degree of caution. Even though verification is theoretically superior to testing because verification essentially proves correctness, testing should still be done at the present time since verification for models are still under development.

It should also be noted that verification tools can be extremely difficult to develop. Microsoft and others have created tools like Dafny[1] to create a software verification program. However, Dafny has

[1] Link to Dafny: https://www.microsoft.com/en-us/research/project/dafny-a-language-and-program-verifier-for-functional-correctness/

trouble verifying even simple pieces of software. Having just one loop in a piece of software code will break a verification tool because computers have trouble formulating a loop invariant. While a piece of software code and machine learning models are not equivalent, a parallel can be drawn between the two fields and a fully functioning verification tool may not exist in the near future. For now, it is reasonable to expect that partial verification systems can be used for machine learning (Goodfellow & Papernot, 2017, para. 23).

Detecting Outliers or Poisoned Datapoints in the Dataset

In order to protect systems from adversarial examples, models should be able to detect outliers or poisoned datapoints in the dataset. One way to detect if an input is an outlier is to see if the input is out of the data's distribution. If it is an out-of-distribution point, there does not exist many training examples to support the conclusion made by the machine learning model. Another thing that can be done is to test the uncertainty of the conclusion, which can be determined by seeing if the prediction is close to data points that have multiple classifications. Drawing upon a similar concept from the verification techniques discussed previously, this would mean the model is very uncertain if this output is correctly classified or not (Papernot & Frosst, 2019, 7).

Chapter 7

TOOLS AVAILABLE FOR USING AND DEFENDING AGAINST ADVERSARIAL MACHINE LEARNING

The most current effective tool for defending against machine learning is keeping the details of the model, such as its training data and structure, hidden from potential adversaries, and to not allow adversaries to see the results of the model. Assuming there is no insider threat, keeping the model's construction and its outputs hidden is the greatest protection that can be offered. However, this is certainly not possible for many systems such as self-driving cars. In addition, the impact of an adversarial attack should also be considered. If the adversarial attack would not harm any assets, then there is no need to secure it. An exhaustive thought process should be used to ensure that the model's outputs would have no negative impact on assets. However, if the model's construction and outputs cannot remain hidden, and the attack would cause the model to harm assets, then tools should be used to protect the models.

Capabilities of the Cleverhans Library

Until recently, Cleverhans[2] was one of the most popular tools for testing adversarial examples. This is a tool that includes black-box attack capabilities against models, as well as adversarial training and defensive distillation methods to provide defenses for machine learning models. Cleverhans has a primarily offensive capability and offers a few defensive capabilities such as adversarial training and defensive distillation. The Cleverhans library is intended to provide attacks and defenses that range from helping to identify the vulnerabilities of machine learning models to simulating the attack of adversarial examples. The library allows users to benchmark a model's vulnerability to adversarial examples by

[2] Cleverhans library on GitHub: https://github.com/tensorflow/cleverhans

allowing users to test various attacks on models and then offer a method to increase defenses. Cleverhans is a GitHub repository and can be used by the general public (Papernot et al., 2018, 2).

IBM's Adversarial Robustness Toolbox

In November 2019, IBM released its first version of the Adversarial Robustness Toolbox[3]. The Toolbox is a large library of tools that are compatible with many different machine learning libraries such as TensorFlow, Scikit-learn, XGBoost, PyTorch, Tesseract, and others; these libraries include many of the most common algorithms for machine learning mentioned in Chapter 2 (IBM, 2020, 6).

The tool contains offensive capabilities to perform many types of evasion attacks (in the form of an adversarial example) and a poisoning attack to change the training data. It also boasts a wide array of defenses, including defenses against adversarial examples, as well as robustness metrics, certifications, and verifications. Unlike other tools, the toolbox also has the capability to detect adversarial examples and poisoning attacks. The ability to provide a total of ten defensive tools and multiple detection tools for both evasion and poisoning attacks allows organizations to start taking significant steps to hardening their models against potential attacks. In addition, the tool offers four different metrics to measure a model's robustness. Therefore, organizations can now defend their machine learning models to a greater degree than previously available, detect evasion or poisoning attacks, and measure how robust a model is to attacks (IBM, 2020, 7).

Four of the defensive tools in the library are specifically designed for working with images. These tools include Spatial Smoothing, JPEG Compression, Total Variance Minimization, and Pixel Defense.

---

[3] IBM's Adversarial Toolbox GitHub: https://github.com/IBM/adversarial-robustness-toolbox

The other defensive tools are generalized for any machine learning model. In addition, as shown in **Figure 14**, many of the defensive tools focus on data processing to help defend the model (IBM, 2020, 9).

The poisoning detection tools can be very helpful when assessing whether a dataset is safe to use to train a model or not. If you can check whether the training data has been poisoned, you can prevent attacker-specific attacks from occurring that otherwise would have been present due to the poisoned dataset.

IBM suggests a three-step process to defend machine learning models against attacks which works well in combination with IBM's ART discussed previously and illustrated in **Figure 14**. The first step is to measure how robust the model is against adversarial examples. The second step is to apply defenses to the model in order to harden it against adversarial examples. Lastly, the model is then configured to detect when a dataset is poisoned or when an adversarial example is being assessed by the model. This model of defense allows various attacks to be stopped and detected, and its effectiveness can be measured at any point. The user can also add an additional step at the end by measuring the robustness after the defenses are put in place to see how much the defenses helped (IBM, 2019, para. 6).

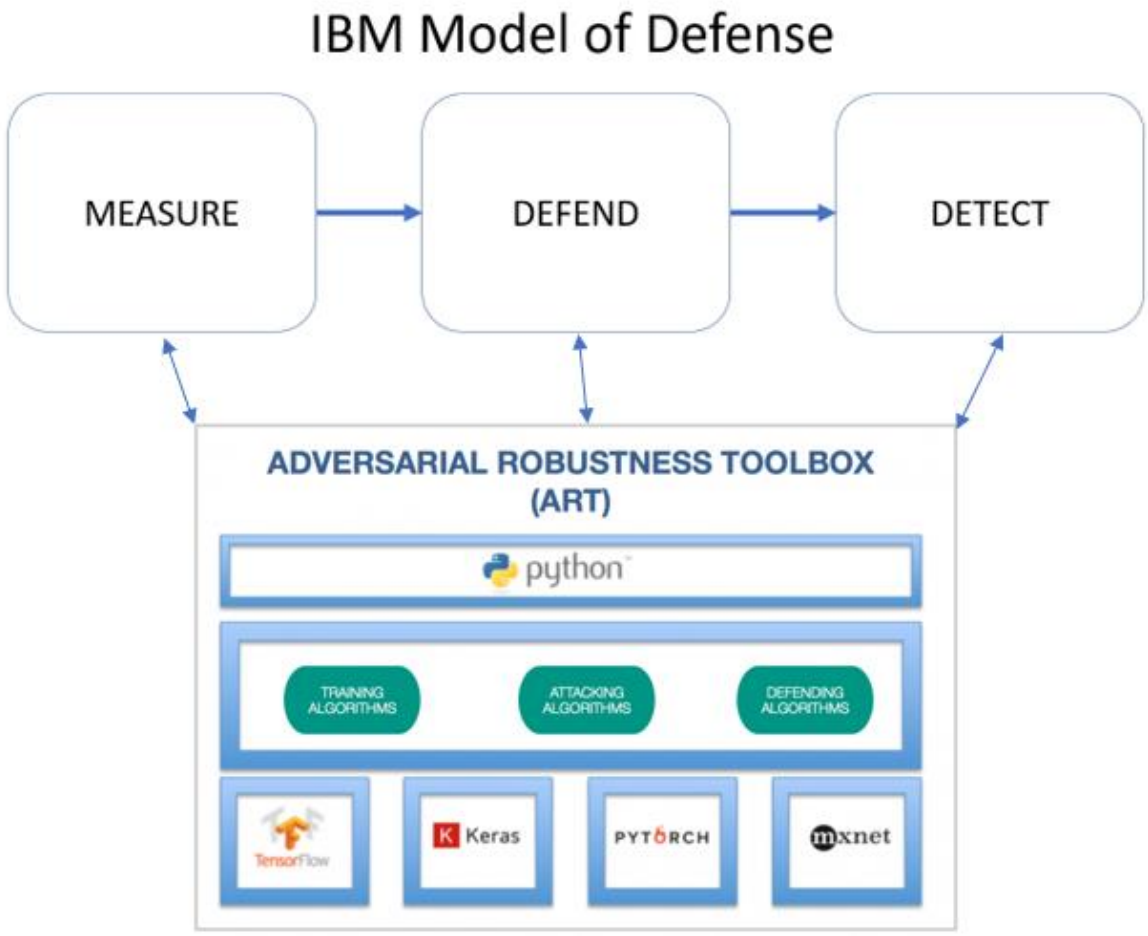

Figure 14. The IBM Model of Defense is to Measure, Defend, and Detect, utilizing its Adversarial Robustness Toolbox (Rodriguez, 2019, para. 8).

By following these steps, an organization can use this practical toolbox to defend at-risk machine learning models against adversarial examples and dataset poisoning, and potentially detect when an adversary is trying to do so. It's worth noting that the ART is different from other tools in that it provides defensive extensive capabilities. Although other defensive tools could exist in the market, this tool is the only extensive defensive tool that exists from an industry source that was easily located. Some academic papers describe a new method to defend a specific attack, but they do not provide a comprehensive set of tools like IBM's Adversarial Robustness Toolbox.

Foolbox Overview

Foolbox[4] is a toolbox that contains 40 gradient-based attacks, three score-based attacks, twelve decision-based attacks, and three other attacks, for a total of fifty five expansive attacks. While Foolbox has many offensive tools, it does not provide any defensive tools. As a recently introduced toolbox with limited documentation, there is not much information currently available other than details about each specific attack (Rauber et al., 2018, 2).

Comparison of Tools

Figure 15 compares the capabilities of the three tools. All three tools, - Cleverhans, IBM's Adversarial Robustness Toolkit, and Foolbox - all provide offensive tools. Only one of these, IBM's Adversarial Robustness Toolkit, contains expansive, ready-to-go defensive measures. This shows that it is easier for attackers to generate adversarial examples as opposed to providing adequate defenses, let alone verification for machine learning. The field of adversarial machine learning is rapidly advancing, and new attacks and defenses will likely appear very soon.

	Cleverhans	IBM's ART	Foolbox
Attack	Yes	Yes	Yes, Expansive
Defense	Yes, Limited (Some Tools)	Yes, Expansive	No
Detection	No	Yes	No
Year Available	2018 (v. 2.1)	2018 (v. 1.2)	2017

Figure 15. As a comparison, IBM's ART is the dominant defensive toolkit, Foolbox specializes in attacks, and Cleverhans brings a balance of offensive and defensive tools.

[4] Foolbox documentation: https://foolbox.readthedocs.io/en/stable/

Now that open-source adversarial machine learning libraries are available, and with Foolbox being available since at least 2017, it is likely that adversaries have begun to study and understand how they can use adversarial attacks to their advantage. As machine learning, and in particular machine learning paired with computer vision, becomes more popular in various products and applications, the number of potential targets in the world will continue to increase. Therefore, setting good habits in industry now will prevent problems for organizations in the future. This includes having any organization who implements machine learning models consider if AML is a risk to their model and, if so, how it can be mitigated using measurement, added protections, and attack detection.

Chapter 8

CONCLUSION

The rise of adversarial machine learning has brought a new, serious security threat to the world of machine learning. This means that any organization or individual using machine learning must consider the security of their model beyond the realm of cybersecurity and into the model itself. As recent tools become more widely available, adversarial machine learning attacks will become a rising threat in the form of dataset poisoning, and adversarial examples will undoubtedly present themselves in the future. Generalized white-box and black-box attacks on popular computer vision systems, such as the adversarial patch, will be noteworthy attacks that should be monitored carefully to assess their potential impact on all industries. However, the creation of early defensive tools by Cleverhans and IBM's release of the first set of extensive defensive tool has provided ways for organizations to defend their machine learning models against adversarial examples or poisoning attacks that may be used against them.

Appendix A

FUTURE RESEARCH

Here is a list of interesting future research topics:

- Streamline model training and model defense into one process so more people will use model defenses.
- Survey and assess awareness of adversarial machine learning in safety critical areas of industry.
- Explore the question: Does an increase in data points, especially near the decision boundary of the model, increase the accuracy of verification by using nearest points to detect adversarial machine learning examples?
- Explore the question: Does there exist an efficient way to theoretically verify machine learning output in order to detect adversarial examples?

BIBLIOGRAPHY

Azati Software. (2019, May 16). Image detection, recognition, and classification with machine learning. Retrieved from https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/

Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition, 84*, 317–331. doi: 10.1016/j.patcog.2018.07.023

Brown, T., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (2018, May 17). Adversarial patch. Retrieved from https://arxiv.org/pdf/1712.09665.pdf

Dossman, C. (2019, March 19). Simple adversarial examples against end-to-end autonomous driving models. Retrieved from https://medium.com/ai³-theory-practice-business/simple-adversarial-examples-against-end-to-end-autonomous-driving-models-346f8a54a19e

Early, J. (2019, September 20). Your car may not know when to stop - adversarial attacks against autonomous vehicles. Retrieved from https://towardsdatascience.com/your-car-may-not-know-when-to-stop-adversarial-attacks-against-autonomous-vehicles-a16df91511f4

Goodfellow, I., McDaniel, P., & Papernot, N. (2018). Making machine learning robust against adversarial inputs. *Communications of the ACM, 61*(7), 56–66. doi: 10.1145/3134599

Goodfellow, I., & Papernot, N. (2017, February 15). Is attacking machine learning easier than defending it? Retrieved from http://www.cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html

Goodfellow, I., & Papernot, N. (2017, June 14). The challenge of verification and testing of machine learning. Retrieved from http://www.cleverhans.io/security/privacy/ml/2017/06/14/verification.html

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015, March 20). Explaining and harnessing adversarial examples. Conference paper at ICLR (Eighth International Conference on Learning Representations) 2015. San Diego, California. Retrieved from https://arxiv.org/pdf/1412.6572.pdf

IBM. (2019, February 7). Securing AI against adversarial threats with open source toolbox. Retrieved from https://www.ibm.com/blogs/research/2018/04/ai-adversarial-robustness-toolbox/

IBM. (2020, January 8). Adversarial Robustness Toolbox v1.1.0. Retrieved from https://github.com/IBM/adversarial-robustness-toolbox

Mahendru, K. (2019, August 14). What are loss functions in machine learning and how do they work? Retrieved from https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/

Molnar, C. (2019, December 17). Interpretable machine learning. Retrieved from https://christophm.github.io/interpretable-ml-book/adversarial.html

Papernot, N. (2019, March 26). Machine learning with differential privacy in TensorFlow. Retrieved from http://www.cleverhans.io/privacy/2019/03/26/machine-learning-with-differential-privacy-in-tensorflow.html

Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P., Rauber, J., Long, R., McDaniel, P. (2018, June 27). Technical report on the Cleverhans v2.1.0 Adversarial Examples Library. Retrieved from https://arxiv.org/pdf/1610.00768.pdf

Papernot, N., & Frosst, N. (2019, May 20). How to know when machine learning does not know. Retrieved from http://www.cleverhans.io/security/2019/05/20/dknn.html

Papernot, N., & Goodfellow, I. (2016, December 16). Breaking things is easy. Retrieved from http://www.cleverhans.io/security/privacy/ml/2016/12/16/breaking-things-is-easy.html

Papernot, N., & Goodfellow, I. (2018, April 29). Privacy and machine learning: two unexpected allies? Retrieved from http://www.cleverhans.io/privacy/2018/04/29/privacy-and-machine-learning.html

Rauber, J., Brendel, W., & Bethge, M. (2018, March 20). Foolbox: A Python toolbox to benchmark the robustness of machine learning models. Retrieved from https://arxiv.org/pdf/1707.04131.pdf

Ray, S., & Business Analytics and Intelligence. (2020, March 27). Essentials of machine learning algorithms (with Python and R codes). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

Rodriguez, J. (2019, February 4). Adversarial robustness. Retrieved from https://towardsdatascience.com/adversarial-robustness-935acc39a01a

Shah, T. (2017, December 10). About train, validation and test sets in machine learning. Retrieved from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Soni, D. (2019, July 16). Supervised vs. unsupervised learning. Retrieved from https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d

Swain, A. (2019, November 24). Noise filtering in Digital Image Processing. Retrieved from https://medium.com/image-vision/noise-filtering-in-digital-image-processing-d12b5266847c

Tencent Keen Security Lab. (2019, March 1). Experimental security research of Tesla Autopilot. Retrieved from https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf

ACADEMIC VITA

COLLIN PAYNE

EDUCATION

The Pennsylvania State University **Schreyer Honors College** University Park, PA
Bachelor of Science in Computer Science Class of May 2020

AWARDS

The President's Freshman Award Pennsylvania State University April 2017
Valedictorian of the Class of 2016 Montoursville Area High School June 2016

HONORS INDEPENDENT STUDY

Wireless Communication & Security Spring 2018

- Researched and presented a lecture on the uses of wireless communication and security for medical implants under the guidance of Dr. Oren Gall
- Built and programmed an On-Off Keying modulation circuit to demonstrate wireless bit transfer

Multithreading Fall 2017

- Created a multithreading program in Java by executing streams in parallel under the guidance of Professor Al Verbanec
- Constructed one generating thread and one processing thread for incoming data

Neural Network Fall 2016

- Created a neural network in Python under the guidance of Dr. Steve Shaffer
- Programmed network to learn and execute binary addition utilizing artificial intelligence
- Constructed network to adjust weights based on the desired outputs of training examples

TECHNICAL CLASSES

Databases and Organization of Data Spring 2020

- Used relational models to create entity relationship diagrams to encourage good database design practices
- Designing and programming a full stack implementation of a university course management system using HTML, Jinja2, Python, Flask, and SQLite to create a website, web server, and database
- Programmed using SQL query language to analyze data and set database triggers

Machine Learning Spring 2019

- Used pandas and numpy to analyze data and create machine learning models
- Studied supervised, unsupervised, and reinforcement learning models

TECHNICAL SKILLS

- Python (Advanced), C, SQL, Java, Lua, Verilog, and Microsoft Office

EXTRACURRICULAR ACTIVITIES

- 4th Place GE Capture the Flag Competition • Association for Computing Machinery (ACM)
- Schreyer Honors College Student Council