Machine learning and Adversarial Attacks

Vantou Xiong
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
xion1547@morris.umn.edu

ABSTRACT

Machine learning (ML) has developed over the years through applications like image processing for uses such as facial recognition, image searching, and text recognition. It has shown promising results and the field is becoming popular. As people try to increase the security in their systems, there are still room for potential attacks against ML models and Deep Neural Networks (DNN) such as Adversarial Attacks. These attacks are known as Adversarial Attacks because they use adversarial examples which are false inputs to attack the machine learning models. Adversarial attacks has to do with crafting data so that the target ML model will misclassify it as something else. There are multiple methods which can be used to attack the model, to try and fool the model to misclassify data. Two methods mentioned are Evasion attacks and Poison attacks. Evasion attacks work by directly attacking through perturbed inputs, while poison attacks inject perturbed inputs into the training process of a machine learning model for future misclassifications. In this paper we will look at how these attacks work.

Keywords

Machine Learning, Deep Neural Network, Black Box, White Box, Adversarial Attacks

1. INTRODUCTION

Machine learning is the process of using algorithms to learn from a set of data called the training set, and then make predictions or classifications on previously unobserved data. One section of machine learning that's been developing over the years is Deep Neural Networks. Deep Neural Networks has been used in a large variety of applications such as speech recognition, healthcare, and image classification like facial recognition [5]. Specifically, image classifiers have been widely used to provide many services, the most easily noticeable one is Google. However as convenient as these appliances of machine learning classification may be, they are shown to potentially be vulnerable to adversarial attacks which affect how machine learning models classify inputs. We will mainly focus on image classifier DNN models and the security risks it comes with. We will also look at how classifiers are given adversarial inputs to falsely classify images.

It is important to understand how the ML model classifies data. When a ML model is able to classify data without any output knowledge, it is known as unsupervised learning. In unsupervised learning the ML model is able to find patterns in data by itself, such as determining which images belong to which category. If the ML model is able to learn to predict or classify data by examples, it's known as supervised learning. It has an expected output, and a way to map the input to the output. We will discuss supervised image classifiers in our examples. Image classifiers are ML models that are able to recognize images they receive, and assign the probability of the image belonging to a certain category. One primary use of image classifications is through Google, Google uses image classifiers which allows us to search for photos we desire [2]. If you wanted to see pictures of dogs or cats, they first have to be classified by Googles ML model. Google even has a reverse image search, which allows you to input photos, and then it gives you results of anything relating to that photo. The ability to classify images through machine learning has served multiple purposes like entertainment, shopping, and quite recently autonomous vehicles (AV).

Image recognition plays an important role in the decisions of autonomous vehicles which allows the vehicle to drive safely. Many self driving collect data of their surrounding environment through LiDAR (Section 3.1). Previous works has shown that these DNN's can be susceptible to adversarial attacks [9]. There are two potential adversarial attacks that we will discuss: an Evasion attack and a Poison attack. When attacking, the adversary chooses an image and slightly changes the image pixels so that it will look the same to humans. This will eventually lead to a misclassification from the DNN and is called perturbing data. In the Evasion attack, data inputs to a trained DNN is perturbed until the DNN misclassifies the data. For the Poison attack, the adversary perturbs training examples in the training data in order for the DNN to misclassify future unmodified inputs. This is how the image classification relates with AV.

LiDARs (Light Detection and Ranging) is a commonly used choice for AV perception. LiDAR is a method for measuring distances through a light as a laser. It allows the AV ML model to view the environment around it in the form of images. The model uses data from LiDAR to transform it into an image, then it has to classify the images from the data it has been trained on. Usually the AV classifies the image correctly, and then makes a decision based on the result. For an example, if the AV were to reach a stop sign, it would get data from LiDAR, process it, then classify the

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/.
UMM CSci Senior Seminar Conference, November 2020 Morris, MN.

image into predefined categories and stop the car. Since it has to classify images, it can be attacked in this way, by receiving falsified data of images that are not real.

Previous works has shown that DNNs can be attacked with succcess rates up to 96% [7]. This has an incredibly high rate of success, Papernot et al. (2017) shows the details behind these attacks. In another work, Cao et al. (2019) shows a more direct and real-world applicable way of attacking such models. In their work they show success rates as high as 90%, by attacking the LiDAR sensor directly with adversarial examples crafted through perturbed input data.

2. BACKGROUND

We will present all information required for full understanding of how the adversarial attacks work, and how the AV system works in this section.

2.1 Machine Learning

When the ML model is able to map inputs to a set of classes, it then becomes known as a Classifier [7]. Since the ML models we will discuss are supervised learning models, the model classifies data based off its training data. If a machine learning model is able to accurately classify data, then it's considered to be a fit model.

2.1.1 Deep Neural Networks

Deep Neural Networks are a subset of Machine Learning, with Deep Neural Networks having multiple hidden layers. In neural networks, the model has multiple layers, the first being the input layer, the middle layers referred to as the hidden layers, and the output layer. The input layer connects to the input variables, and the output layer consists of output nodes that produces the output variable. The hidden layers are multiple layers inside the DNN, each layer has nodes or neurons and are between the input and output layer. As illustrated in Figure 3, each neural layer is connected to the next layer by edges referred to as weights and shows how the data can be passed forward through the neurons down until it reaches the output layer. Neurons represent intermediate results in a mathematical function and are fed outputs from the previous layer. Weights are values that helps influence the inputs between the next neurons. The weights are applied onto the input data commonly by multiplication and the result is used by the neurons in the next layer. Then in the output layer, the probabilities of the input belonging in a certain category is assigned. The input layer could have things like pixels in an image, while the output layer would consist of the probability of the image being a certain category. They are commonly used in computer vision, which is useful when classifying images. That is why the models we will look at deals with specifically DNNs.

2.1.2 Training

Training the model is process of fitting the model to the training data by finding optimal weights for each edge. Given the initial training set, the model will have randomized weights applied to each neuron, and the output will be computed most likely inaccurately. When facing optimization problems, every model has a way of figuring out the error in predictions or classifications, this is done with whats known as the loss function. One way of updating weights for optimization is called Backwards Propagation by creating a gradient of the loss function based on the weights. Mathematically a gradient is a vector that gives the direction of greatest rate of increase of our loss function. The gradient is computed one layer at a time, starting from the last layer. It allows you to determine which weights contribute to the overall error and how to update them. It does so by finding the local minimum in the error gradient for lowest loss, this process is called the gradient descent. This process happens by iteratively taking the current slope of the loss in the gradient and then updating the weight in the opposite direction of the slope until you reach the local minimum. This leads to the weights being updated to have the lowest loss. A small portion of the training data is kept hidden from the model for an unbiased evaluation on unseen inputs, referred to as the validation set. When training the model using backwards propagation, the amount that the weights are updated by is referred to as the learning rate. It's important to find an optimal learning rate for the model to find the local minimum in optimal time. The validation process is meant to tune hyperparameters like the learning rate.

2.1.3 Decision Boundaries

As the input values changes, the greatest probability of the input belonging in a certain category also changes. This is referred to as decision boundaries. For better visualization, Figure 1 shows how regions in the graph are classified. Let anything to the left of the decision boundary be classified as a circle, and anything to the right would be classified as a triangle. Any shape even if it's a triangle, on the left side would be classified as a circle. If there was a triangle this would mean there is a minor inaccuracy in the model, realistically all models have some error.

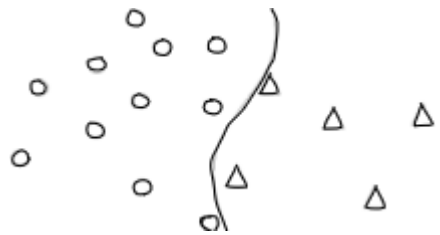

Figure 1: Decision Boundaries.

2.2 Adversarial Attacks

Before an attack happens, an adversary has to choose a machine learning model to attack, which is what we will call a target DNN model. Adversarial attacks are attack methods where the adversary has intentionally crafted an input designed to cause misclassifications from the target DNN model. There are generally two attack methods for adversarial attacks, both cause the DNN model to misbehave and misclassify inputs. The first is called Evasion attacks, using perturbed data to attack the model. The second type is called Poisoning attacks, which directly attack the training data set by modifying it with false sample data. Usually this is done through perturbing input until the DNN misclassifies the input. Ideally the perturbations would be done as small as possible so that the result would be indistinguishable from the original when observed by humans. In Figure 2, this is shown through the image of a panda. In our LiDAR example we will discuss in section 3.1, the perturbations are done to the points in the 3D point cloud data. This can be done

through input perturbation functions, which are functions that perturb data, an example would be the middle picture in Figure 2, and a mapping function can be seen as the symbol in between the two pictures to combine the data. This results in the picture on the very left with the classification of gibbon.

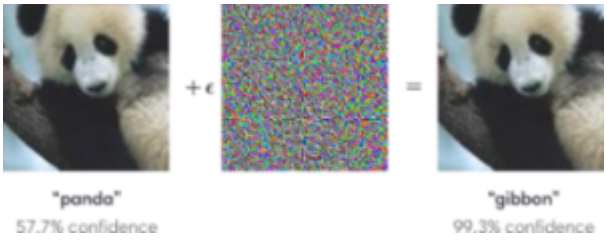

Figure 2: How image classifiers can mistakenly classify images although to humans, it looks the same [4].

2.2.1 *Black Box and White Box Attacks*

Black box attacks are types of attacks where the attacker has no knowledge of its target internals. So the adversary would not have access to the target DNN model's algorithms. When the target DNN model's information is unknown, possible attack methods are to train for similar decision boundaries as the target DNN model using a substitute DNN model. Substitute DNN models are models crafted by the adversary designed to be similar to the target DNN model to figure out how to cause misclassifications before querying the target DNN model any further. This helps evade detection and allows the adversary to test data.

White box attacks deal with attack methods where the adversary has access to the full algorithm. The adversary has all access to knowledge about input and output variables, and how the DNN model works.

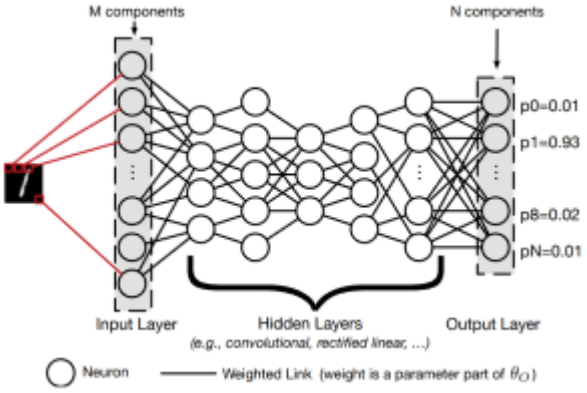

Figure 3: Visualization of how DNNs work with M component inputs and N component outputs [7].

3. METHODS OF ATTACKS

Depending on how much information the adversary has on the target DNN model, the attack will be either a black box or white box attack.

3.1 White Box

For a white box attack we will talk about LiDAR-based perception autonomous vehicles and the attack strategy used from Cao et al. (2019) [9]. The attack method that is used is an evasion attack through perturbing images as minimal as possible until a misclassification is reached potentially leading to fatal injuries. The reason that this attack method is used is because the targeted model is from a company that has an open source platform for Autonomous Vehicles called Baidu Apollo [9], and researchers cannot modify the DNN model algorithms or training set. All that is done is crafting perturbed images, mapping that on the original image, and then feeding that to the input as seen in Figure 4. The input to the model is a collection of points such as an X for length, Y width, and a Z for height that represent a 3D shape referred to as a feature matrix. Depending on the result they perturb the data and repeat. The adversary has access to all the models inputs, the output variables, the algorithm itself, and the model's parameters so it is a white box attack [9]. We will discuss how in this white box attack, one possible attack method was through an evasion attack. To further understand the attack process, we will discuss how the input is received and works.

LiDAR is a light sensing technology used to measure distances from a point. It has powerful uses when it comes to AV, it allows the AV to see the environment around it. All LiDAR does is collect raw data of the environment multiple times a second. It needs to be stored into collections for a 3D representation so it can be used such as point clouds [3]. It first gets pre-processed to transform the sample points into readable inputs for the DNN model which is a feature matrix. During this phase of processing the LiDAR data, the 3D point clouds are used to generate a feature matrix by mapping those points into the desired feature cell. The process of the decision making from the autonomous vehicle can be seen in Figure **??**. It starts with pre-processing the LiDAR data into a feature matrix, and is then sent as input values for the DNN. The DNN then classfies the image, and the image post-processed to apply further information on the image such as speeds, shapes and positions to see what the image looks like. After that the autonomous vehicle then decides what to do based on all the information given [9].

The attack method is to create adversarial examples by generating sample points. This is observed through the adversary having their own attack laser to help generate 3D point cloud sample points. One hundred random points are selected from the surrounding environment to generate the 3D point cloud data, using only sixty to create the input parameters. The reason for this is because out of hundred sample points, only sixty were stable enough to use for their image. To create the adversarial example, the adversary needs to create fake points in the image, this is referred to as spoofing. In previous works, there have been methods to try and blindly spoof the points. However, it is shown in the article that blindly spoofing points has low success rates, so the approach was modified. The process can be shown through Figure 4. The spoofed points are carefully placed into position through the input perturbation function, this can be seen in the first image of Figure 4. In order for the spoofed objects to become more obstacle-like the perturbation function can also scale the spoofed points, such as causing the points to grow in size by scaling it's height points in the third image. The resulting image can be seen

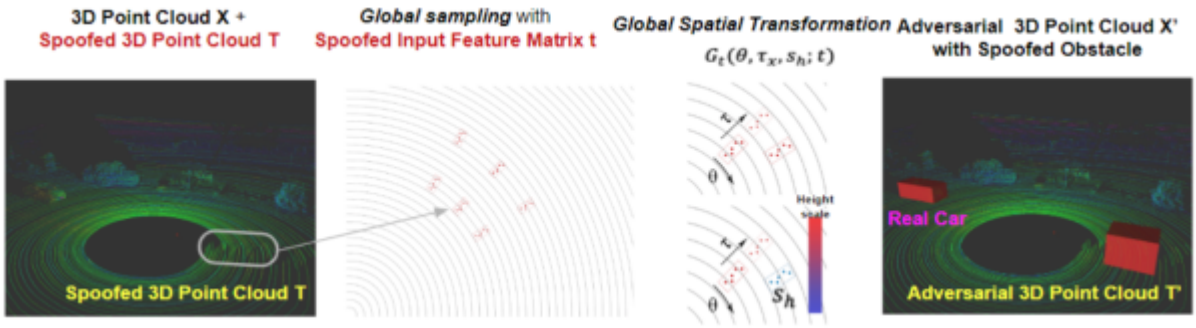

Figure 4: Closer look at how the adversarial examples work overall and how the system interprets the adversarial example [9].

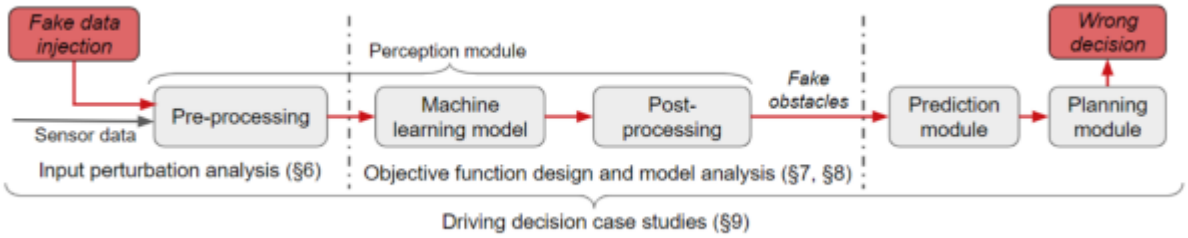

Figure 5: An example of how LiDAR is used in ML to allow AV systems to make decisions [9].

in the last image of Figure 4 [9]. The spoofed data must be perturbed through the input perturbation function in order to generate adversarial examples which they would test on the target model for higher success rate on attacks.

The input image is analyzed and the adversary is able to add or subtract from the input to create an image for the targeted model. The adversary could modify the distance of the 3D point cloud or the azimuth of the 3D point cloud, this is referred to as global spatial transformation. To create the adversarial example researchers have to map the spoofed data onto the input through a mapping function. This can be seen as combining an imaginary image to the original image. All of this computation happens before the data is injected into the DNN.

Once they have the adversarial example they could then spoof the points into the car with the laser. This creates an imaginary obstacle for the car which could lead to fatal injuries. The example provided in the article is a digital example, the adversary would have to take spoof points in front of targeted car on the road. There are numerous ways to do this, two of which is discussed in the article. First you could have another car to follow the targeted car and attack the car through generating the adversarial examples and injecting the data into the targeted LiDAR sensor. Second you could set up an adversarial attack on a car that is at a stop, causing it not move [9]. Other examples could be to prevent others from taking your parking spot.

3.2 Black Box

The attack strategy we will look at is from Papernot et al. (2017) [7], where they create their own DNN referred to as a substitute DNN to train for similar decision boundaries. This is an evasion attack that targets a DNN called the target DNN or Oracle DNN O. The researchers trained the DNN themselves for a suitable classifier, but did not create or have any information on the DNN, thus it's still a black box attack. If the researchers trained the target DNN with bad data, that would ruin the nature of the attack. It is an evasion attack because it can only use observed ouput labels from the target DNN to craft adversarial examples that force the DNN to misclassify images. If the target DNN classified an image as the number 3, then they know what input falls under the class for 3. They also limit their data set to not having access to large data sets so it can be more applicable in real-world scenarios due to cost. By attacking only through observing output variables from the target DNN, this allows the attack method to be widely used to attack other DNN image classifiers. The goal is to cause the target model to misclassify the image [7].

The adversarial examples are generated with as much minimal perturbations as possible so the changes isn't detectable by humans. This also makes it more efficient to use. A query is made to the target DNN, and then the output is recorded so that their own substitute model does the same through approximation. Then once the substitute has collected enough data, it could potentially craft adversarial examples by itself because the substitute should theoretically mimic the target DNN model. Since they also use as little perturbation as possible, they limit the amount of queries they need to make. This allows the adversary to be further undetected from the target DNN [7].

Figure 6 outlines the process of creating an accurate substitute DNN. First the adversary tries to learn something about Oracle DNN O's decision boundaries by querying Oracle O to label images, then the substitute DNN trains itself based on the resulting decision. Steps one is the ini-

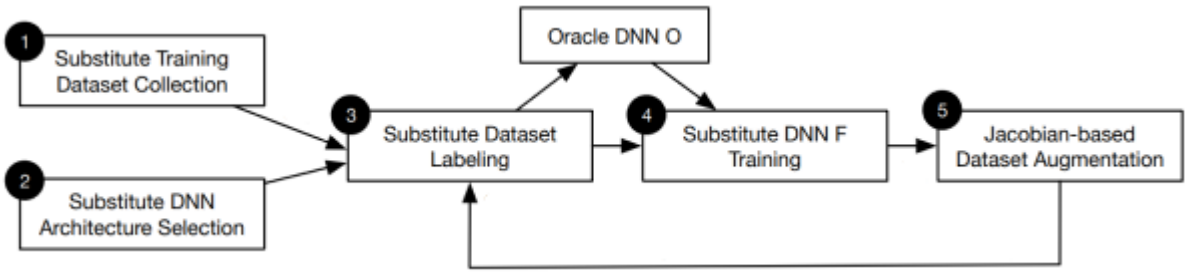

Figure 6: Good visualization on how the substitute DNN works, and how it is trained to provide accuracy as well as decisions similar to the target DNN model [7].

tal training set selected for the substitute DNN. Step two is where the substitute DNN is crafted.In step three, the queries are made to help the substitute model label output variables thus creating decision boundaries. In step four the model is trained (See Section 2.1.2 for a reminder of how this works). The substitute model needs a way to find the decision boundaries, this can be done through moving along the loss gradient by updating/calculating weights based on the least amount of loss. Jacobian is a mathematical way to update the weights. Jacobian-based Dataset Augmentation is used during the training process to move input by perturbation along the gradient of Oracle O's decision boundaries, and the direction is identified through the Jacobian matrix in step five [7]. That is how the substitute DNN is trained to be similar to Oracle DNN O because the models are similar in their decision boundaries. Then the new perturbed images are sent back to step three, which can be seen as generating synthetic data. This process happens iteratively until the substitute model is shown to have similar decision boundaries as the target DNN model.

Once the substitute DNN has been made, it can then start crafting adversarial examples through two methods which is the *Goodfellow et al. algorithm* and the *Papernot et al. algorithm*. Both algorithms allow the substitute DNN to perturb the data to achieve desired goals. The Papernot Algorithm allows less perturbations, but at the expense of computing cost. While the Goodfellow Algorithm allows the substitute model to swiftly craft many adversarial examples, but with large amounts of perturbations [7].

3.2.1 SetUp

Now that we understand how the adversarial examples can be crafted, we will discuss how they are used to attack MetaMind Oracle, Google, and Amazon oracles. This is an black box evasion attack, the researchers has no access to MetaMind's DNN model. However, the researchers are able to train MetaMind's model for a suitable classifier, and then attack through the method described in the previous Section. MNIST is a database collection of handwritten digits, numbers ranging from zero to nine which will be the common training set. MNIST is used to train the target DNN model, MetaMind allows you to train MetaMinds DNN through their API. The researchers use data to train MetaMind's DNN, but the DNN is not known to the adversary. So a large enough training set is given to the target DNN model to make a suitable classifier of 94.97% classification accuracy, and then attack that DNN. They also created a validation set of 10% of the sample data in order to create further unbiased DNN. To further remove bias from the target DNN, Papernot et al. (2017) also introduces their own handwritten data set to the substitute DNN. This also helps removing similarities in the training data which is all from MNIST [7].

Once both the target DNN and substitute DNN has been created, the adversary can begin to craft adversarial examples using the substitute DNN. Using the Papernot and Goodfellow algorithms they are able to perturb the input data and create the adversarial examples. Through this they were able to successfully cause miclassifications from the target DNN, for an example Oracle O would classify 3's as 8's. Then the same setup was recreated for Amazon and Google and got quite interesting results which will be discussed in Section 4 [7].

3.3 Other Attack Methods

Poison attacks are adversarial attacks where the adversarial example is injected into the training data of the ML model. This can damage the ML model so that it is unable to perform correctly. This can happen when a software continuously collects data, so that it can retrain itself in the future. This is where poison attacks can be performed, and realistically where most are performed, because normally people don't have access to manipulate software labels or internals. Example softwares would be spam-detection or anti-viruses which have been attacked through the poison method [6]. Examples given in the works by Muñoz-González et al. (2017), a white box attack type, show how PDFRate which is an online tool for catching malware embedded inside of pdf files can be attacked [6]. The tool analyzes the pdf file and reveals if there is any hidden malware. Then since the tool asks for feedback, this is essentially allowing users to confirm if the classification by PDFRate was correct. Providing false or wrong feedback can harm the algorithms ability to accurately detect malware.

In the experimental setup, Muñoz-González et al. (2017) used back-gradient optimized poisoning which is backwards propagation but with fixed size steps instead of using the current slope. Muñoz-González et al. (2017) test the success rates of the attack method on other machine learning algorithms to test the transferability in classification on a Spambase dataset for spam detection in emails. These algorithms consist of DNN's, Adaline which is a single layered neural network, and Logistic Regression which is another learning classification algorithm. Back-gradient is used by having their own poisoning model that can update its parameters and weights starting from the last layer. The way

it's done is to try and compute the gradient based off of maximizing the classification error by reversing the learning procedure of the target algorithm while updating the neuron's weight [6].

4. RESULTS AND SOLUTIONS

All three attack methods yields quite successful results when attacking their targets through adversarial examples. When Cao et al. (2019) attacked Baidu Apollos ML model, they were able to achieve a success rate of 75% [9]. Since they had full access to the algorithms, they were able to create a virtual experiment that caused the model to misclassify inputs. Papernot et al. (2017) were able to achieve even higher success rates of misclassifications, they attacked MetaMind, Amazon, and Google. For MetaMind they had success rates of 84%, Amazon being 96% , and Google being 88% [7]. Lastly, for the poison attack method, they achieved classification errors for ML models like Adaline and Logistic Regression to be up to 30%. Although the rates are much lower, it's meant to show the transferability of their attack method, which is effective when tested between different ML models.

5. CONCLUSION

In this work we show how ML models can be targeted through adversarial examples. We show various ways how the model can be attacked whether information is known of the target DNN or not. Each method achieves their goals in causing misclassifications at high rate. Adversarial attacks such as evasion and poison attacks are shown to have high success in causing malfunctions from machine learning models.

6. ACKNOWLEDGEMENTS

I would like to thank Peter Dolan, Elena Machkasova, Humza Haider, and the students of Senior Seminar Fall 2020 for contributing to this work through advice and feedback.

7. REFERENCES

[1] Andrew J. Hawkins *Waymo is first to put fully self-driving cars on US roads without a safety driver.* (2017, November 07) Retrieved October 10, 2020, from https://www.theverge.com/2017/11/7/16615290/waymo-self-driving-safety-driver-chandler-autonomous

[2] Google *ML Practicum: Image Classification* Retrieved October 10, 2020, from https://developers.google.com/machine-learning/practica/image-classification

[3] Gray, D. (n.d.). *LiDAR vs point clouds: Learn the basics of laser scanning, 3D surveys and reality capture* Retrieved December 04, 2020, from https://info.vercator.com/blog/lidar-vs-point-clouds

[4] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel, Jack Clark *Attacking Machine Learning with Adversarial Examples* 2017, February 24.Attacking Machine Learning with Adversarial Examples. Retrieved October 09, 2020, from https://openai.com/blog/adversarial-example-research/

[5] Liwei Song, Reza Shokri, and Prateek Mittal. *Privacy Risks of Securing Machine Learning Models against Adversarial Examples.* In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19). Association for Computing Machinery, New York, NY, USA, 241–257.

[6] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. *Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization* In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec '17). Association for Computing Machinery, New York, NY, USA, 27–38.

[7] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. *Practical Black-Box Attacks against Machine Learning* In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17). Association for Computing Machinery, New York, NY, USA, 506–519.

[8] Wikipedia *Machine Learning* (2020, October 07). Retrieved October 10, 2020, from https://en.wikipedia.org/wiki/Machine_learning

[9] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. *Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving.* In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19). Association for Computing Machinery, New York, NY, USA, 2267–2281.