Cybersecurity in Machine Learning and Artificial Intelligence for Self Driving Vehicles

A proposal for an unified and open source test framework

Igor Kunjavskij
Seminar Intelligent cyber-physical Systems
University of Stuttgart
Stuttgart, Germany
st100959@stud.uni-stuttgart.de

***Abstract*—This report summarizes the key methodologies applicable for attacks on Convolutional Neural Networks deployed in self driving vehicles. Furthemore, the need for an open source framework specifically designed to test such networks for robustness against these attacks is demonstrated. This need stems from the fact that solutions available so far only cover very generic scenarios of adversarial attacks. Hence these tools are not suited to specifically secure Convolutional Neural Networks deployed in in self driving vehicles. All in all the development of such a framework would benefit all parties that are involved in the market of autonomously driving vehicles.**

***Index Terms*—Cybersecurity, Machine Learning, Artificial Intelligence, Adversarial Machine Learning Attacks, Data Poisoning, Model Stealing, Autonomous Cars, Convolutional Neural Network**

I. Introduction

Machine leaning has brought in recent years tremendous progress to any field that has found an application for it. In this regard the field of autonomous driving is no exception. With a market value of nearly 818.6 billion USD in 2019, the ongoing success of this branch continues. More and more companies ranging from start-ups to traditional automotive corporations have decided to move into that market, offering solutions such as components, modules, systems or complete vehicles [1].

Some key players operating in the autonomous vehicle market distributing fully integrated solutions, as in autonomous vehicles, include Audi AG, BMW AG, Daimler AG (Mercedes Benz), Ford Motor Company, General Motors, Google LLC, Honda Motor Corporation, Nissan Motor Company, Tesla, Inc. and the and Toyota Motor Corporation [2]. Although every provider listed integrates it's own unique suite of sensors, one sensor type that is found in the vehicles of every provider is the optical sensor. The data gathered by these sensors is usually processed by Convolutional Neural Networks, a class of deep learning methods which has become dominant in various computer vision tasks [3]. Unfortunately research has proven throughout recent years that such Neural Networks can get compromised using various techniques depending on the end goal of the attacker.

II. Problem statement: Selected cybersecurity threats in Convolutional Neural Networks

Over the years numerous techniques have been discovered that can be utilized to compromise Neural Networks. For this report three such approaches have been selected that seemed most promising when applied to Convolutional Neural Networks that are deployed in self driving vehicles. These are

- Adversarial Attacks
- Data Poisoning Attacks
- Model Extraction Attacks

and each type of attack targets a Neural Network at a different stage in its life-cycle as depicted in Fig. 1 [4]. Adversarial Attacks and Model Extraction Attacks usually target models at the deployment stage when they are already receiving input from their production environment. Data Poisoning Attacks on the other hand target the training data set that is used when creating a model.

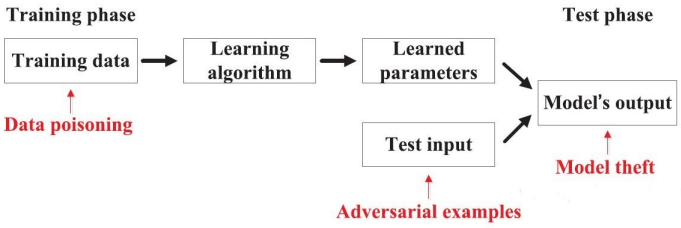


Fig. 1. Overview of attack vectors [4].

A. Adversarial Attacks

One of the most predominant attack techniques that has been utilized so far when tampering with Neural Networks is the Adversarial Attack. This term summarizes a whole set of different approaches that eventually all aim at generating inputs that will throw off a Neural Network, causing it to misclassify a given input. In the case of a Convolutional Neural Networks this input is an image that has either been altered in a malicious manner or contains elements that are harmful to the performance of the Neural Network.

At it's core the possibility for such attacks is rooted in the fact that the training data that is provided to supervised learning algorithms is sampled from an incomplete segment of the theoretical distribution space. This concept is also depicted in Fig. 2, illustrating the theoretical data population that is used to develop a model and its relationships with the training and test data distribution spaces.

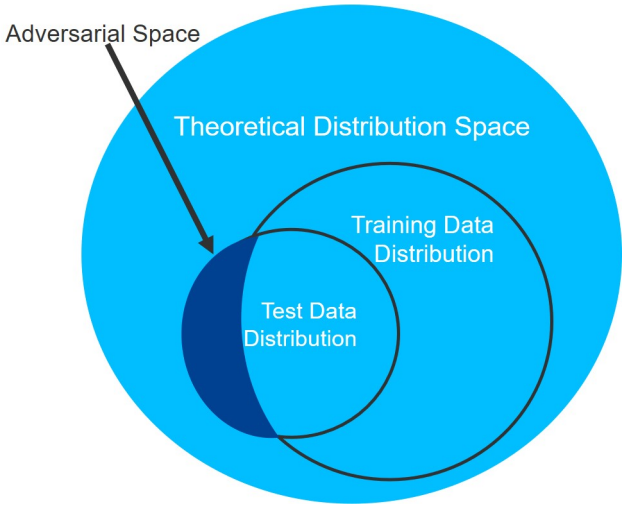


Fig. 2. Adversarial space as a result of imperfect representation in training data [8].

When a model is deployed in the real world and faces so far unseen test data, the test set drawn from the test data distribution could contain a segment of data whose properties are not captured in the training data distribution. These segments are also referred to as "adversarial space". Attackers can exploit pockets of adversarial space between the data manifold fitted by a model and the theoretical distribution space to fool models.

Usually the expectation is that training and test data are drawn from the same distribution space and that all characteristics of the theoretical distribution are covered by the trained model. This unfortunately is usually not the case [8]. Hence this discrepancy between expectation and reality is causing the aforementioned "blind spots" in machine learning algorithms

Nowadays there are numerous ways to find such "blind spots". One of the first approaches that has been utilized for that purpose is the "Fast Gradient Sign Method" [5].

This method uses the gradient of the underlying model to find adversarial examples. The original input image x is hereby manipulated by adding or subtracting a small perturbation to each pixel. Whether the perturbation is added or subtracted depends on whether the sign of the gradient for a pixel is positive or negative. Adding perturbation in the direction of the gradient means that the image is intentionally altered so that the model classification fails. Using the Fast Gradient Sign Method an adversarial example can be obtained by calculating a perturbation in the following manner

$$\eta = \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{1}$$

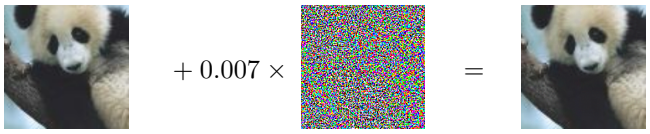
where $\nabla_x J(\theta, x, y)$ is the gradient of the models loss function with respect to the original input image x, y is the target associated with x and θ are the parameters of the model that should be compromised. To make sure the perturbations added stay within a certain range, the parameter ϵ is which usually ranges between 0 and 1.

Afterwards this perturbation η has to be added to the original input to obtain an adversarial example $\tilde{x}$ as in

$$\tilde{x} = x + \eta. \tag{2}$$

This perturbed input will now cause the targeted model to make a wrong prediction.

An example of this process is also depicted in Fig. 3. In this example an adversarial input is generated to be applied to GoogLeNet on ImageNet. By adding a vector whose elements are very small, but have an equal sign as the sign of the elements of the gradient of the cost function with respect to the input, the classification of the image is changed. Here an ϵ of 0.007 is used. The original image is correctly classified as a panda with 57.7% confidence, whereas the adversarial example is classified as a gibbon with 99.3% confidence. The adversarial attack hence is a success.



$\boldsymbol{x}$
"panda"
57.7%
confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2%
confidence

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 %
confidence

Fig. 3. Demonstration of Fast Gradient Sign Method [5].

Obviously the human eye can't tell the difference between the original and the perturbed image which is important. This way it is possible to alter for example real world images in a way that will throw off a Neural Network but still won't be perceptible. There are also ways to regularize the generated perturbations to for example fit a specific shape or color scheme to make the production of these in the form of patches or other objects that can be placed in the real world more feasible [11].

Of course just as attack techniques are continuously developed, concurrently new defensive techniques are coming up to mitigate these attacks. So far one defensive technique that turned out to be effective even when the attacker has knowledge about the defensive technique that is deployed is the "Barrage of Random Transforms" [14]. This technique is a defense based on applying image transformations before feeding the input image into the model. The defense works by randomly selecting a set of transformations and a random order in which the image transformations are applied. In addition, the parameters for each transformation are also randomly selected at run time. Unfortunately this technique has quite

a strong impact on the performance of a model, since it drops the accuracy for example on a data set such as the Fashion-MNIST data set by 15% [13]. Next to this accuracy drop, which is already unacceptable for models deployed in self driving cars, the fact that such transformations are additionally computationally quite expensive further hinders they're deployment.

So far these have all been academic examples that have been carried out in laboratory conditions. Concerning adversarial attacks on systems deployed in production, these have been carried out as well on several occasions. One such example is a study conducted by researches from McAfee Labs in 2020. In this study the aforementioned researchers managed to fool the camera system aboard Tesla's Model X and Model S vehicles. Just by placing a bit of black electrical tape measuring about 5 cm on a traffic sign depicting a 35mph speed limit, as shown in Fig. 4, they managed to cause a misclassification of this sign as 85mph. As a result both Tesla models accelerated upon misreading this sign [6].

Such approaches as outlined just now assume that the model that should be fooled is available at hand and it's parameters are all known. This implication is usually mitigated either by training a surrogate model using classifications of the original model as ground truth labels. Or by utilizing a very similar model to generate adversarial examples. Both approaches unfortunately work reasonable well without even having the original model parameters at hand [4].

Fig. 4. Speed limit sign that was compromised by McAfee Labs [6].

B. Data Poisoning Attacks

Data Poisoning Attacks on the other hand aim at manipulating data that is used to train a model. Usually the attacker carries out a so called "flip-label attack", i.e., modifies both the training data and the corresponding ground-truth labels. This is done to make the model misclassify inputs that contain certain features chosen by the attacker, for example a marker on a traffic sign. But to conduct this attack an adversary would need direct access to the training data. Furthermore, upon auditing this data this mislabeling would be identified immediately. Another so called "clean-label" attack also alters images in the training data set without making changes to the ground truth labels. This is done with the intent to cause misclassifications only with specific kind of images. But just as with the previous approach, direct access to the training data is required.

Since companies usually keep their training data sets off limits, such attacks become infeasible.

Yet another recently published approach manages to still temper with the training data without having direct access to it [7]. Since autonomous driving systems need to be trained online on data collected in the production environment, such data has to be gathered continuously to improve the performance of trained models for various subsystems. An example for this is Waymo, since it's autonomous vehicles have so far been driven for more 32.2 million kilometers on public roads to achieve this [9].

To temper with the training data, the attacker needs to modify the surroundings in the environment that is used to train the car. These modification have to be on the one hand subtle enough so that they blend in with the surroundings, but also on the other hand perceptible enough so that the sensors of the car clearly distinguish them. These modifications are then correlated with target concepts that are supposed to be learned, such as the classification of traffic light phases. Most importantly these modifications should have no causal relation with the concepts to be learned. Since it is very difficult to train deep neural networks to only pick up causal relations, the vehicle will very likely learn to see the introduced modifications as cues to the target concept.

Thus the vehicle can be thrown off during deployment when these modifications are changed altogether again.

Such an attack differs from the data poisoning attacks outlined beforehand as only physical modifications are applied to the training environment without modifying the training data set directly. Furthermore, since the physical modifications are permanent, they will remain also during the deployment of a vehicle. Finally, the target is to cause all the data to be misclassified at test time, not just a few inputs. So far the feasibility of such an attack has been proven in simulations using traffic lights with billboard image placed next to them and is depicted in Fig. 5.

The scenario used to demonstrate the attack and hence to corrupt the traffic signal classifier of an autonomous vehicle, which is based on a Convolutional Neural Network, assumes that this classifier has already been trained in a town A. Now it should be retrained online in town B, since retraining might be required for example because the traffic signals in town B are different in shape or appearance from those in town A. As is common practice in the industry, the autonomous vehicle is trained using extensive road tests first. Afterwards a human expert provides manual ground-truth labels to the gathered data. This human is assumed to be uncompromised and should honestly label the encountered traffic signals. Presuming these facts, the attack is then conducted in two stages.

In the first stage, advertising space on electronic billboards is installed or purchased near a subset of traffic signals in town B. Billboards are used in this example scenario, since real-world evidence suggests that billboards are quite frequently located on or near traffic intersections. During the training

period, three distinct images are displayed on the billboards synchronized with the traffic light phases. The intention here is to correlate the signs shown to the traffic signal classifier of the autonomous vehicle with the different phases of the traffic lights.

In the second stage, when trained vehicles are deployed in the field, the order of the images on the billboard is switched. Experiments show that a naively trained traffic signal classifier attacked this way is severely compromised, even if only a small fraction of traffic signs in town B are ”poisoned” with billboards.

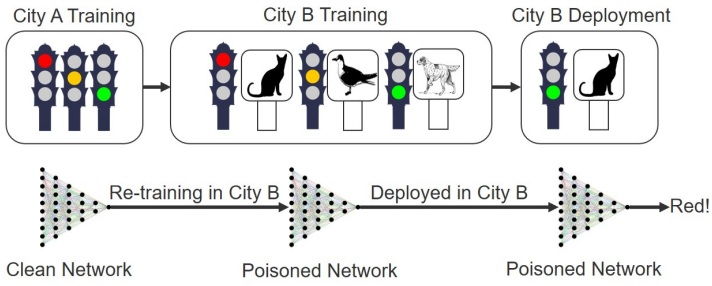


Fig. 5. Setup for poisoning a model by introducing spurious correlations in the physical environment [7].

What makes this scenario particularly malicious is the fact that it is quite hard to detect it.

C. Model Extraction Attacks

Model extraction attacks are so far considered one of the most dreaded attacks by the industry [15]. This stems on the one hand of course from the fact that a lot of resources are usually used to create such a model, hence the theft of this intellectual property causes tremendous financial damage. What's worse on the other hand, is the fact that all the attacks outlined so far and others not mentioned here become way easier to carry out. Usually such an attack is carried out by querying the target model using synthetic or surrogate data as depicted in Fig. 6. This way a labeled data set is constructed using the predictions of the model to be stolen. Afterwards this labeled data can be used to train a surrogate or clone model should replicate the the targeted model [12].

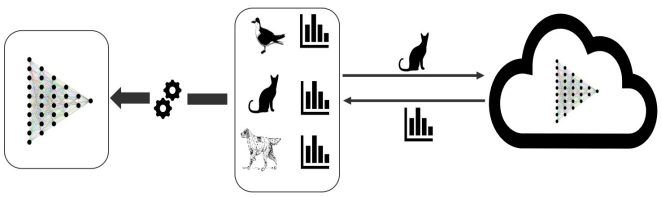


Fig. 6. Model Stealing Attack [12].

But an even easier way to obtain a model is just to purchase the autonomous vehicle and then to gain access to the model that is deployed in it. Researchers from Keen Security Labs in China gained access to the Neural Network running on a Tesla Model S and used it to train adversarial patches. They afterwards placed these patches on the road to create a “fake lane”, so that the Tesla Model S steered away from the appropriate driving lane into the opposing lane on a test course. In a real-life scenario oncoming traffic would be driving there, causing an a fatal crash for all parties involved [10]. Creating these patches is fairly easy once the parameters of the model to be fooled are obtained. This is what makes this kind of attack particularly bad, since it facilitates the execution of other attacks to a great extent.

III. Stakeholder Analysis

The previous chapter introduced very real threats that are all feasible nowadays. These threats target especially parties such as Test Engineers, Competitors, Customers and Computer Vision Engineers, as they are all involved in the implications that an attack on a computer vision pipeline of a vehicle might have. These parties are stakeholders as depicted in Fig. 7, ordered in categories according to their interest and power to protect a computer vision system.

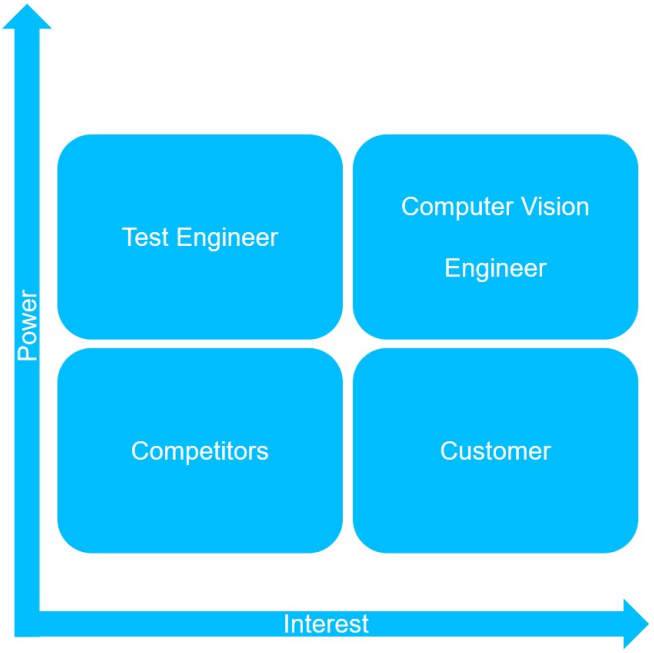


Fig. 7. Stakeholder diagram for proposed testing framework.

Competitors, as in other participants on the market, have not much influence on the development process of the computer vision pipeline of another company, nor do they show much interest in furthering the advancements of a competitor. Still they are included in the list of stakeholders, since if the model of one company has been compromised using one of the attacks outlined in this report, then it is very likely that models of other participants on the market will follow suit very soon.

Next the customer of such an autonomous vehicle is of course very interested in the safety and security of all com-

ponents that are included, but has little power to influence the development of these components.

The most important party involved is the Computer Vision Engineer that is working on the computer vision pipeline. On the one hand no other party has more influence on the security of the computer vision pipeline that is deployed, but on the other hand the Computer Vision Engineer is also the party held accountable for any misbehavior of an autonomous vehicle that might be caused by it.

Lastly the Test Engineer usually has a lot of say in the development process of an autonomous vehicle and the same amount of interest in the security of the computer vision pipeline as the Computer Vision Engineer. But since the Test Engineer is eventually not the one writing the algorithms used in an autonomous vehicle, the interest in securing these algorithms is slightly higher with the Computer Vision Engineer, making this party the main Stakeholder. So eventually it should be the task of the Computer Vision Engineer that the Convolutional Neural Network deployed in an autonomous vehicle is secure.

IV. Existing solutions

To make sure that the Convolutional Neural Network deployed in an autonomous vehicle is secure, a Computer Vision Engineer might first have a look at the tools out there to see if someone has already tackled that problem before. As with any threat in cybersecurity, the research community has of course put effort into addressing the risks that have been outlined so far in this report. For that several frameworks have been released by various parties, these include:

- Cleverhans - A Tensorflow Library to test existing deep learning models versus known attacks
- foolbox - Python Library to create adversarial examples, implements multiple attacks
- TrojAI- Python Library for generating backdoored and trojaned models at scale for research into trojan detection
- Adversarial Robustness Toolkit (IBM ART) - Python Library for Machine Learning Security
- Advertorch - Python toolbox for adversarial robustness research whose main functions are implemented in PyTorch

These frameworks offer in general good testing capabilities for the attacks outlined in this report, but only in a generic setting. This means conditions that are encountered when deploying Neural Networks in an autonomous vehicle, such as sensor noise or the feasibility of certain perturbations, are not considered. So eventually the only option left to a Computer Vision Engineer is to recreate a testing framework from scratch or to try and alter one of the tools outlined here in a way that makes it applicable to the problem at had.

V. Discussion and Value Proposition

As outlined until this point, a Computer Vision Engineer would need to reinvent the wheel even if other companies have already tackled this problem. But so far companies developing and distributing autonomous vehicles have shown little interest to reveal their development process and the ways in which they have secured their computer vision systems. Even more so, this way the main strategy used by most companies boils down to ”security through obscurity”, meaning that a potential attacker is supposed to be held at bay by not knowing what defense strategy might be deployed. This strategy has little chances to prevail, since so far attackers have quite successfully fooled Convolutional Neural Networks even without knowledge of the defense strategies that are used [13]. So overall the tools that are available to a Computer Vision Engineer are on the one hand not suited to tackle the problem of securing the computer vision pipeline of an autonomous vehicle. On the other hand knowledge that might be available in companies is ”siloed” and there is little to no exchange about that between competitors, upholding a strong mindset of such knowledge being strictly proprietary.

And still the main issue remains that once the Convolutional Neural Network of one company has been compromised, others will very likely follow. Since this is an issue affecting every company on the market and every such company will have to deal with this issue at some point in time, joining forces to tackle that issue might be the best option. For this reason the establishment of an open source framework targeting the testing of Neural Networks deployed in self driving vehicles is proposed.

The Computer Vision Engineer in every company developing self driving cars usually is one of the main enablers of safe and secure autonomous driving, advancing the establishment of autonomous vehicles overall. But next to all the hardship that comes along developing accurate computer vision models, testing these against all the attacks outlined in this report is almost infeasible for a single Engineer.

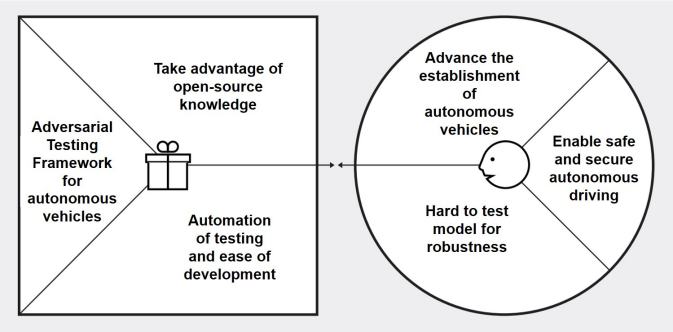


Fig. 8. Value Proposition for an open source testing framework.

As shown in Fig. 8, by using an open source framework developed in cooperation by companies facing the same issue, it would be possible to take advantage of the knowledge that every company is bringing along. Furthermore, this would automate and ease the process of testing Convolutional Neural Networks tremendously, making the necessity to reinvent the wheel obsolete.

VI. Conclusion

This analysis in this report has shown that there are very relevant attack scenarios out there that are feasible in the production environment that autonomous vehicles usually operate in. But on the other hand there are no existing open source solutions specifically created for autonomous vehicles to test the robustness of them against these attacks. A Stakeholder analysis revealed that especially in the sector of self driving vehicles all involved parties should have great interest to cooperate and to conduct measures in an unified manner against these attacks. Further systematical analysis with the help of a Value Proposition Canvas has shown that the most obvious solution to this issue would be an open source framework tackling the outlined security issues. Compared to the invested resources in this market sector and it's valuation of close to a thousand billion USD, developing such a framework should be relatively inexpensive and furthermore quite lucrative. All in all the need for an open source tool that allows all market participants to secure their solutions surely exists.

References

[1] Ltd, R. A. M. (2020). Autonomous Cars Global Market Opportunities and Strategies to 2030: COVID-19 Growth and Change. Research and Markets ltd 2021.

[2] Grand View Research, Inc., Autonomous Vehicle Market Size, Share & Trends Analysis Report By Application (Transportation, Defense), By Region (North America, Europe, Asia Pacific, South America, MEA), And Segment Forecasts, 2021 - 2030, March 2020

[3] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018).

[4] M. Xue, C. Yuan, H. Wu, Y. Zhang and W. Liu, "Machine Learning Security: Threats, Countermeasures, and Evaluations," in IEEE Access, vol. 8, pp. 74720-74742, 2020

[5] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." ICLR 2014.

[6] Povolny, S., & Trivedi, S. (2020, February 19). Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles. McAfee Blogs. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/

[7] Patel, Naman & Krishnamurthy, Prashanth & Garg, Siddharth & Khorrami, Farshad. (2020). Bait and Switch: Online Training Data Poisoning of Autonomous Driving Systems.

[8] Chio, C. & Freeman, D. (2018). Machine Learning and Security: Protecting Systems with Data and Algorithms. O'Reilly UK Ltd.

[9] "Waymo self-driving vehicles cover 20 million miles on public roads," Reuters, 07-Jan-2020. [Online]. Available: https://www.reuters.com/article/us-autonomous-waymo-idUSKBN1Z61RX.

[10] Huddleston, T. (2019, April 3). These Chinese hackers tricked Tesla's Autopilot into suddenly switching lanes. CNBC. https://www.cnbc.com/2019/04/03/chinese-hackers-tricked-teslas-autopilot-into-switching-lanes.html

[11] Cao, Yulong & Xiao, Chaowei & Yang, Dawei & Fang, Jing & Yang, Ruigang & Liu, Mingyan & Li, Bo. (2019). Adversarial Objects Against LiDAR-Based Autonomous Driving Systems.

[12] Yuan, Xiaoyong & Ding, Lei & Zhang, Lan & Li, Xiaolin & Wu, Dapeng. (2020). ES Attack: Model Stealing against Deep Neural Networks without Data Hurdles.

[13] Mahmood, Kaleel & Gurevin, Deniz & van Dijk, Marten & Nguyen, Phuong. (2020). Beware the Black-Box: on the Robustness of Recent Defenses to Adversarial Examples.

[14] Raff, Edward & Sylvester, Jared & Forsyth, Steven & Mclean, Mark. (2019). Barrage of Random Transforms for Adversarially Robust Defense. 6521-6530. 10.1109/CVPR.2019.00669.

[15] R. S. Siva Kumar et al., "Adversarial Machine Learning-Industry Perspectives," 2020 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2020, pp. 69-75, doi: 10.1109/SPW50608.2020.00028.