Attacks on Machine Learning: Adversarial Examples in Connected and Autonomous Vehicles

Prinkle Sharma, David Austin, and Hong Liu

Department of Electrical & Computer Engineering
University of Massachusetts, Dartmouth, MA
Email: {PSharma1, DAustin1 and HLiu}@umassd.edu

***Abstract*—Connected and autonomous vehicles (CAV a.k.a. driverless cars) offset human response for transportation infrastructure, enhancing traffic efficiency, travel leisure, and road safety. Behind the wheels of these mobile robots lies machine learning (ML) to automate mundane driving tasks and make decisions from situational awareness. Attacking ML, the brain of driverless cars, can cause catastrophes. This paper proposes a novel approach to attack CAV by fooling its ML model. Using adversarial examples in CAVs, the work demonstrates how adversarial machine learning can generate attacks hardly detectable by current ML classifiers for CAV misbehavior detection. First, adversarial datasets are generated by a traditional attack engine, which CAV misbehavior detection ML models can easily detect. Building attack ML model takes two phases: training and testing. Using supervised learning, Phase I trains the model on the time-series data, converted from the adversarial datasets. Phase II tests the model, which leads, for the next round of model improvement. The initial round deploys K-Nearest Neighbor (KNN) and Random Forest (RF) algorithms, respectively. The next round, guided by deep learning (DL) models, uses Logistic Regression (LG) of neural network and Long Short-Term Memory (LSTM) of recurrent neural network. The results, in precision-recall (PR) and receiver operating characteristic (ROC) curves, validate the effectiveness of the proposed adversarial ML models. This work reveals the vulnerability in ML. At the same time, it shows the promise to protect critical infrastructure by studying the opponent strategies. Future work includes retraining the adversarial ML models with real-world datasets from pilot CAV sites.**

Keywords: Artificial intelligence (AI), machine learning (ML), critical infrastructure protection, securing connected and autonomous vehicles (CAV), threat detection, supervised learning, deep learning (DL), long-short-term-memory (LSTM), adversarial machine learning.

I. Introduction

Every day accidents and deaths occur on the roadways throughout the world due to distractions, inability to react fast enough, bad weather, and/or inappropriate decisions. Vehicle technology is moving towards the realm of Connected and Autonomous Vehicles (CAV). Autonomous vehicles utilize a vast number of sensors (LiDAR, radar, cameras, sonar, among others) to build a 3-D image surrounding the vehicle and to make decisions on how to drive without any human intervention. Connected Vehicles on the other hand, supplement Autonomous Vehicles, that require communication with other vehicles/infrastructure to make decisions. They communicate through Basic Safety Messages (BSM) that include credentials such as speed, position, braking status, steering wheel angle, etc. The Society of Automotive Engineers (SAE) provides a J2945 standard [1] that defines the minimum requirements for a BSM. The goal of CAV is to minimize deaths and accidents on the roads, as well as limit traffic congestion. Connected Vehicles are built off communication known as Vehicle-to-Everything; therefore using data fusion mechanisms to cope with V2X and sensors weaknesses is mandatory.

CAV technology has high potential, however, the issue that needs to be evaluated is if an authenticated vehicle is found to be sending spoofed messages. How will other CAV's handle the situation and safeguard itself? Misbehavior detection is a crucial aspect that needs to be understood, to strengthen V2X. Positional attacks, Relay attacks, Sybil attacks, or DoS attacks, etc. are all vulnerabilities to CAVs. Thus a robust system should not only detect but also prevent the CAVs from such vulnerabilities. Alike in game theory, the best way to defend an attack is by thinking in the mindset of an attacker to properly defend itself.

Internet of Things (IoT) and Cyber-Security are critical areas in which Machine Learning (ML) is increasingly becoming significant. It is an exciting field of new opportunities and applications; but like most technology, there are also longstanding challenges present. If defenders can utilize ML technology for protecting the system, it won't be long before it will be in the hands of the adversary to evade detection. In general, adversarial examples are used to confuse the decision or outcome of a machine learning system to achieve some goal. According to recently published research work [2], scholars have presented the existing cyber-security problems and demonstrated black-box attacks created by the ML-based system, without having any prior knowledge of the data and the techniques used by the target.

Motivated by the above-cited work, the goal of this work is to have misclassification of an example of a targeted value by ML-based model on vehicular networks thereby demonstrating

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

a successful invasion of the system. This paper will build off the traditional means of attack to form a novelty approach of training an ML model to generate attacks after having learned the normal behavior thereby spoofing the defensive ML model.

Our contribution is three-fold:

1) Different from the traditional methods, we train an adversarial network to directly produce adversarial examples, which not only results in perceptually realistic examples, but also the generation process is more efficiently
2) We use the state-of-the-art Machine Learning and Deep Learning methods against adversarial examples
3) Compare Machine and Deep Learning models and evaluate their performances

Rest of the paper is organized as follows: Section II describes the issues with the traditional attacks based techniques in VANET. Section III discusses the current defense mechanisms on traditional attacks on Connected Autonomous Vehicles and state of the art in Machine Learning domain. Section IV presents the attacker model deployed in this work. Section V and Section VI discuss the experiment details and results obtained. Finally, the authors conclude the paper and future work in Section VII.

II. Problem Statement

Connected and Autonomous Vehicles are continuing to improve in their testing and evaluations. Still, however, they are vulnerable to data falsification and GPS spoofing attacks. Since one of the key limitation to investigating CAV security problems is a lack of appropriate data sets, researchers [3] have provided limited access to datasets that can enable the academic community to investigate the aforementioned challenges and suggest traditional methods that can help minimize or eliminate them. Such human-generated attacks have been studied extensively, but what about attacks that are generated by a machine? Human lives are at risk when added technology that is meant to alert them starts creating vulnerabilities such as spoofing. For a long time, ML is mostly used as a defensive measure such as in Intrusion Detection Systems (IDS) or firewalls. If research isn't spent on a "red-team" adversarial ML attack, the adversary could get ahead and use ML to undermine the CAV defense technology, potentially placing lives at risk.

Recently [4], Google AI InceptionV3 classifier was fooled by a group of MIT researchers. They created a black-box attack and made the system misclassify a 3-D image of a turtle as a rifle. Such instances clearly demonstrate that the Machine Learning is just another tool, susceptible to adversarial attacks which can have huge implications in a world where we trust them with human lives via self-driving cars and other automation, Clearly, if an attacker can undermine a classifier such as Google's with such an ease, what is stopping them from spoofing a ghost vehicle onto a highway, and leading to multiple accidents and possible deaths as a result? In this work, we are demonstrating how an ML-based defense system can be misguided to label an attacker vehicle as normal; thereby raising concern on how trustworthiness of ML/DL techniques.

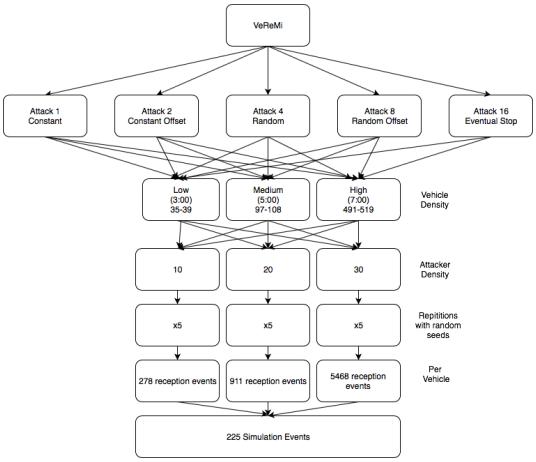


Fig. 1: VeReMi Attacker Model

III. Related Work

Vehicular Ad-hoc networks (VANETs) technology has emerged as an important research area over the last few years. VANETs comprise vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications based on wireless local area network technologies. Security attacks on VANETs may lead to catastrophic results such as the loss of lives or loss of revenue for those value-added services. Below we discuss a few of the work performed by researchers in wireless cyber-security domain inside VANET using traditional techniques (Section III-A) followed by the Machine Learning techniques (Section III-B).

A. Traditional Techniques

Researcher, Dr. Jonathan Petit & Steven E. Shladover analyzed traditional cyber-attacks in automated vehicles. The authors' categorized attacker model as, "internal vs. external, malicious vs. rational, active vs. passive, local vs. extended, and intentional vs. unintentional." Additionally, they evaluated attack surfaces within automated vehicles, such as: "Infrastructure sign, GPS, Radar, in-vehicle devices, odometer sensors, etc." [5]. This was the first attempt where the attacks within the CAVs were addressed and analyzed.

Heijden, et. al [6] created the positional attacks in VANET by using traditional means of attacks and developed a dataset named VeReMi (Vehicle Reference Misbehavior). Figure 1 illustrates the make-up of VeReMi. Utilizing simulation software's: Omnet++ built off of Eclipse allows for the communication of nodes for simulation and SUMO a road traffic simulator, is paired with a modified version of VEINS which is a framework for vehicular networks. Together, the software's allow for the ability to simulate vehicles on a map of Luxembourg. As a means of illustrating the potential of possibilities the dataset can offer, we also extended Heijden's work, by creating speed attacks using the same convention basis of attacks (constant, constant offset, random, random

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

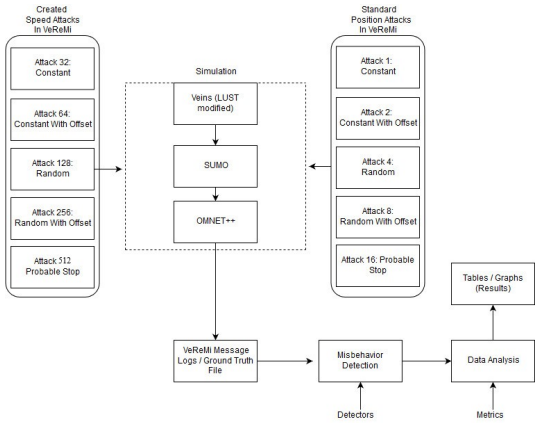


Fig. 2: VeReMi Architecture

offset, and eventual stop) but manipulated the speed rather than the position as shown in Table I.

The speed attacks, shown in Figure 2 that were created followed suit to the position attacks, and the same convention for the attack ID was followed. Attacks 32 and 64 were already built into the data-set, while attacks 128, 256, and 512 were added. Due to page limitation, authors strongly recommend referring [6] for better understanding.

TABLE I: Speed Attacks in VeReMi

Attack ID	Attack	Parameters
32	Constant	Vx = 0, Vy = 0
64	Constant Offset	Delta Vx = 250, Delta Vy=-150
128	Random Offset	Delta Vx, Vy uniform random w/ offset 200 added to Vx,Vy
256	Random	Random Vx,Vy between 0 and 180
512	Eventual Stop	Stop Probability +=.025; Speed = 0, Position saved

B. Machine Learning Techniques

The issue at hand is the situation in which the vehicle is authenticated, but what is to stop the vehicle from spoofing its location or performing other attacks? So et. al [7] discusses detection of misbehavior(s) in VANET in their work. Two classification algorithms K-Nearest Neighbor and Support Vector Machine were utilized to create a baseline for attack detection. Features including plausibility checks on location, movement, and numerical behavior were considered. The experimentation proved well with about a 20% higher precision and within 5% of recall. Although the data-set utilized high attack concentration, so the results may not have been as conclusive if a lower attack concentration was used.

AI and Machine Learning should be used for securing CAV, as it is effective to defend against the traditional attacks. In a separate article, Sharma et. al [8] discussed the means for securing wireless communications of connected vehicles with artificial intelligence. The On-board units within the vehicle use the information received to react accordingly. If a collision is expected then the collision warning should alert the drier. Kalman filter and Particle filters were used in the model. The Context Adaptive Beacon Verification method (CABV) was found to save approximately 86.5% computational overhead with Kalman Filter over Particle which resulted in 85.94%. And it was able to detect about 76% spoofed beacons with Kalman and 89% with Particle filter.

Lately, hackers are familiar with automated solutions such as Google's InceptionV3 classifier or computer vision such as in autonomous vehicles which are built using Artificial Intelligence. Now they are using Machine Learning based tools to produce cyber-attacks like adversarial inputs, data positioning attacks, and model stealing techniques [2].

AI-based systems are now trained to generate emails, send the fake command to automated systems, create malware, perform model skewing, etc. that resembles an authorized message. Yi Shi et. al, in [2], performed a black-box adversarial ML attack on an online classifier of API calls known as an exploratory attack. Without prior knowledge of the algorithms used, training data or the expected output; authors started with building a classifier which normally would require lots of data to train. But due to limit on the number of API calls, active learning was used to create a classifier using a small number of API calls which trained the classifier. Accuracy of the created classifier was continuously evaluated, and once accepted, testing data was used, which introduced unfamiliar data into the adversary model and compared the output of the target.

Detection of misbehavior was performed by Sharma et. al in [9] where unsupervised Machine Learning (provide data without labels and analyze how the computer clustered the data) integrated Pearson Correlation Analysis to examine if various position attacks would be detected by the model. After training the model with normal behavior, it clearly showed differences in the co-relation, proving misbehavior was detected. However, the technique wasn't able to detect all instances of misbehavior and failed when the vehicle followed closely in the victim's safety area (three-second spacing).

In other work, strengthening ML against adversarial inputs was investigated by Goodfellow et. al in [10]. They developed a TensorFlow based library, CleverHans, that allows researchers to test their model against standardized and current defenses and attacks, thereby providing a baseline. Authors are constantly adding more verification methods since adversarial examples are only at their beginning, as adversaries advance in ML, so will the attacks. As per authors, strengthening the classifiers to stop attackers from finding input points that can be misclassified, or by changing the best-fit for boundaries in the classification of data, will move ML in the right direction against adversarial examples.

IV. Attacker Model

For this work, we have extended the dataset provided by VeReMi authors. VeReMi is a labeled simulated dataset providing a wide range of traffic behavior and attacker implementations. To classify the capacities of an attacker, we define four dimensions:

- **Scope**: The scope is the area over which the attacker vehicle can cause damage to the network. An attacker which is only able to control entities under a defined communication range is termed as a *local attacker*.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

- **Inside**: An attacker that possesses an authorized cryptographic material to communicate with other entities in the networks is classified as *inside attacker*.
- **Active**: A legitimate attacker that injects forged information in the network is known as *active attacker*.
- **Uniform/Non-Uniform Region**: The region defines the type of road on which vehicles are traveling. The attacker is referred to be driving on *uniform area* when the speed, direction, and path are expected to be shared with all other vehicles for a long time (highways). The *non-uniform area* means that the attacker is driving on streets where speed, direction, and path vary frequently.

Thereby, VeReMi dataset considers an active, inside local attacker, which injects forged data in a uniform and non-uniform regions.

V. Experiments

Moving away from the traditional means of attack and into a more modern day approach, security is one of those areas that needs to quickly evolve to keep up with the advancements in machine and deep learning technology. ML is described as achieving human-level performance or better but it also fail greatly due to the adversary ability to modify input data in any amount. As ML becomes part of more critical systems and capacities, it will need to increase guarantee against attack or greater protections. The traditional assumption has been that the environment is benign and that the testing data is in the same distribution as training. This assumption continues to be exploited by attackers. Constraints need to be added to ensure that adversaries can't replace the class of input. White-box or black-box attacks will vary the strength of the attack, and existing solutions such as distillation or adversarial training are undermined by black-box attacks. While we typically think about techniques in a positive context with algorithms trying to improve the intelligence of the solution, they can also be used to create security attacks. Therefore, we have decided to utilize ML and Deep Learning techniques to generate attacks and then compromise the safety of defense ML model.

The framework is divided into two phases. Phase One includes training the machine/deep learning model with normal data. Details about the data pre-processing can be found in Section V-A. Once the model is trained, we evaluate the performance using standard ML matrix. Phase Two works off of the DL model, and moving towards generating attacks different from the VeReMi attacks. Phase Two will generate fake data based on the dataset utilized for model building. Testing will be performed to evaluate the effectiveness of the model to detect if the data is fake or real. comprises of testing the model. Section V-A and V-B discuss the details of our approach.

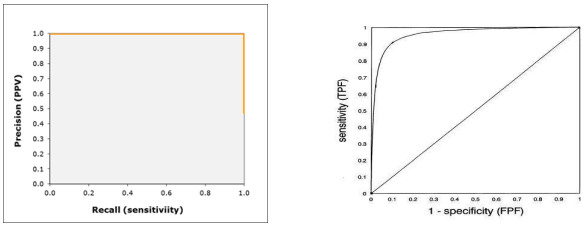
A. Data Pre-Processing

The VeReMi dataset is built using three simulation softwares that includes SUMO, OMNET++ and Veins. The resulting output of the simulation are JSON files, one log file for each vehicle along with a single Ground Truth for the simulation. The vehicle log files contain: vehicle number, ID, and characteristics: speed, position, RSSI, etc. The Ground Truth file contains: vehicle number, ID, and label: attacker or not which also lists the actual BSM communications sent during the simulation. For this work, we have used high vehicle density with low attacker density as shown in Figure 2. More details about the dataset and pre-processing can be found in [6] and [7].

Unlike in the traditional way, which had attacks programmed, the adversarial machine learning model will create data that could be similar to attacks done in the traditional approach, but not flagged. Both Machine Learning and Deep Learning are vulnerable to attack. If open to API calls, both can be susceptible, although with Deep Learning it could be possible for the models built to be retrained as a means to undermine the integrity of the model as Shi et. al describe in [11]. Especially in the world of autonomy, adversarial attacks inside cars could result in injury or death, and Kurakin et. al in [12] discuss adversarial examples in the real world vs. theoretical models.

B. Attack guided by Machine Learning Models

For this experiment, we are dealing with the time-series data. So, we initialize the process by building a model based on supervised machine learning techniques and train it on normal data (without any attack). For this, we have used K-Nearest Neighbor (KNN) and Random Forest (RF) algorithms. K-Nearest Neighbor algorithm classifies an object into certain class by a majority vote of its neighbors. The object is being assigned to the most common class among its "k" nearest neighbors. Random Forest classifier creates a set of decision trees from randomly selected subset of training dataset. In simple words, it aggregates votes from several Decision Tree, and then decides the final class of test object. As we have trained the models on normal vehicle behavior, our initial expectations are that we should have a perfect Precision-Recall and ROC graph as shown in Figure 3.



(a) Precision Recall (b) ROC

Fig. 3: Perfect Precision Recall and ROC graphs

C. Attacks Guided by Deep Learning Models

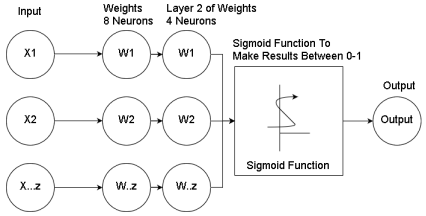
1) ***Neural Network built with Logistic Regression****:* The utilized dataset posed a binary classification problem, and Logistic Regression (LR) provides the ability in classification problems to map probabilities to two outcomes. The approach used breaks the x and y data into training and testing data 67% and 33% respectively. The model is coded in GoogleColab

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

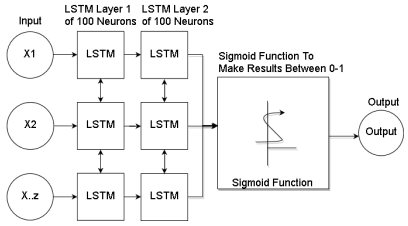
using scikit-learn libraries [13]. To ensure that a feature isn't favored higher than another, MinMaxScaler was used to scale the feature data to values between zero and one. The Neural Network (NN) was built with three dense layers otherwise known as fully-connected layers because for every input there is an output associated with a weight. A visual representation of the model can be seen in Figure 4(a). The input feeds into the first dense layer containsing eight neurons (six for features and two extra neurons added for bias). For further cleaning, second dense layer outputs into the single neuron dense layer which uses the *sigmoid function* since it's a binary classification problem. Finally the output layer uses a dense layer with only a single neuron, and is expected to classify the output as zero or one. The model is evaluated by predicting the output of the testing data.

2) ***Recurrent Neural Network built with Long Short Term Memory****:* Recurrent Neural Networks (RNN) are essentially a network that contains copies of the same network. Due to its exploding and vanishing gradient problem, LSTM models were introduced. The uniqueness of LSTM is that its "memory cells" are controlled by gates that provide information to be stored, forgotten or output [14]. Where the ML models using KNN and RF were close to ideal but not perfect, the initial expectation is that a DL model will have better results. Similar to the NN for LR, x and y data was broken into training and testing data of 67% and 33% respectively. The model configuration was kept similar as Linear Regression model. A visual representation of the RNN-LSTM model can be seen in Figure 4(b). The diagram shows that the input layer maps to two LSTM layers which are known as hidden layers. The first LSTM layer is made up of 100 neurons and it specifies the input shape and whether the output will be stacked or single. The input shape is specified by timesteps (how long before the data repeats) and number of features. The output of the layer will result in stacked output to be ingested into a second LSTM layer. After each LSTM layer a dropout layer of 20% is integrated to ensure that over-fitting doesn't occur. The second layer is also a 100 neuron LSTM layer, but it will result in an output of the last hidden state. The LSTM layer outputs to a single neuron dense layer which just like the LR model, map the results using the *sigmoid function* ranging between zero and one. Similar to the LR model, the LSTM model predict the output on the test data. An early stopper function was also introduced in both the models to ensure that the model stops getting trained when the validation loss is no longer improving; thereby preventing over or under-fitting.

3) ***Attack Using Generated Data****:* Each feature was evaluated for its mean and standard deviation, which the numpy library was imported to perform a random normal distribution accounting for the mean and standard deviation. Ten values are generated and the data is comma separated and imported into the model for testing. The goal is for the model to not detect the data as fake, and instead view it as real.



(a) Logistic Regression Diagram



(b) LSTM Diagram

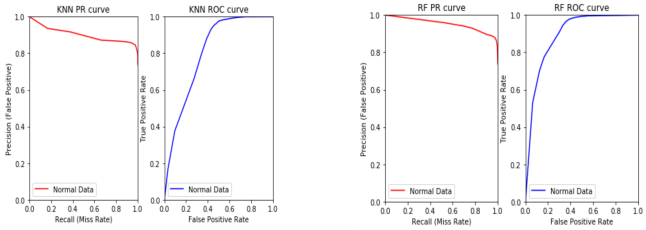
Fig. 4: Logistic Regression vs Long Short Term Memory [14]

VI. Results and Discussion

Below we discuss in detail the results obtained from Machine Learning and Deep Learning models.

A. Machine Learning Models

Using the normal behavior vehicle data from VeReMi dataset, we trained the models using scikit-learn. Figure 5 represents the Precision Recall and ROC graphs for both models trained on normal data. As expected, the PR and ROC curves demonstrated were close to perfect classifier behavior with an AUC = 94.25 and 95.02 for KNN and RF respectively. Comparing the Figure 3 and 5, we can clearly see that models build with KNN and RF were close to perfect but not ideal. Due to this difference, we planned to switch to Deep Learning to optimize the results.



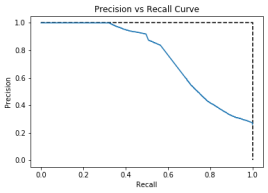
(a) K-Nearest Neighbors (b) Random Forest

Fig. 5: Precision Recall and ROC graphs for KNN and RF

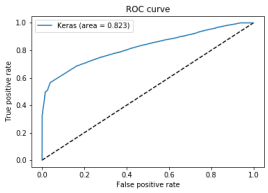
B. Deep Learning Models

1) Logistic Regression based Neural Network: The LR model produced the following ROC and Precision Recall curves: Figure 6. The resulting AUC was 82.28. As can be seen from the graphs, the results are not what was expected. The ML models, produced results with AUC values of 90's whereas the LR AUC was less. Reasoning for why the results

are not what was expected can be explained by figure 7(c), the false-positive and false-negative areas of the confusion matrix (bottom left and top right respectively) are quite high. Both values make up approximately 17.16% of the data in the confusion matrix. Deep learning models are data hungry, and for data that isn't equally represented by all output values it will result in unbalanced classes. What can be noticed is that the PR curve graphs between the LSTM and the KNN or RF show that the LR has a higher PR value up until approximately recall value 0.4.



(a) PR Curve



(b) ROC Curve

Fig. 6: Precision Recall and ROC graphs for LR

KNN	Predicted (0)	Predicted (1)
Actual (0)	42635	945
Actual (1)	7937	7561

(a) KNN

RF	Predicted (0)	Predicted (1)
Actual (0)	42559	1021
Actual (1)	5974	9524

(b) RF

LR	Predicted (0)	Predicted (1)
Actual (0)	43104	589
Actual (1)	7937	7561

(c) LR

LSTM	Predicted (0)	Predicted (1)
Actual (0)	38762	959
Actual (1)	6953	7033

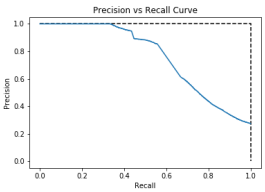
(d) LSTM

Fig. 7: Confusion Matrix

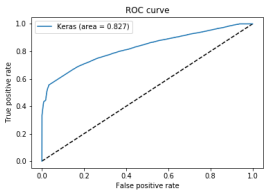
2) RNN-LSTM based Neural Network: The LSTM model produced the following ROC and Precision Recall curves: Figure 8. The resulting AUC was 82.66. Similarly to the LR results, the LSTM model did not outperform the KNN and RF models. The ML models, produced results with AUC values of 90's whereas the LSTM AUC was less. Just like with the LR model, reasoning for why the results are not what was expected can be explained by Figure 7(d). The false-positive and false-negative areas of the confusion matrix (bottom left and top right respectively) are quite high. Both values make up approximately 14.73% of the data in the confusion matrix. Where DL models require lots of data and fairly represented outputs, having unbalanced data can produce worse results. What can be noticed is that the PR curve graphs between the LSTM and the KNN or RF show that the LSTM has a higher PR value up until approximately recall value 0.4. It is possible that with more evenly balanced classes or data that is more repetitive that the results will be much better than current.

C. Attack Using Generated Data

As previously mentioned, the authors as their last step generated fake vehicle data following the normal distribution



(a) PR Curve



(b) ROC Curve

Fig. 8: Precision Recall and ROC graphs for LSTM

of the dataset based on the using the mean and standard deviation for each feature. Two different series of data were generated, one with five rows of data and the other with ten rows. The data was then put into the necessary array format to be read into the model and was scaled similar to the test and training data during model creation. For both series of data, the models were evaluated by predicting the output the given data should result.

Predicted	0	0	1	1	0

(a) Predictions of Data (5)

Predicted	1	1	0	1	1	1	1	1	0	0

(b) Predictions of Data (10)

Fig. 9: Predictions on the 5 vs. 10 generated rows of data

VII. Conclusion & Future Work

CAV continues to be improved, strengthening its technologies, securing/mitigating vulnerabilities. For the CAV to be more reliable, differentiating between the integrity or falsification of data compared to real data will greatly effect its ability to operate. Artificial intelligence and self-driving cars are often complimentary topics in technology. In this work, we have demonstrated an instance where normal vehicle data was used to measure the performance of Machine Learning and Deep Learning algorithms. Initial assumption was to have Deep Learning outperforming the results from Machine Learning Algorithms. Due to the unbalanced classes and limited dataset size, authors believe that the results were hindered when fed to Long Short Term Memory and Logistic Regression models. For future work, the authors plan to evaluate the real-world dataset (US Pilot Projects and Waymo dataset) and experiment on balancing the classes. In addition, the authors would also like to explore realistic adversarial attacks inside VANET.

Acknowledgement

Authors would like to thank Dr. Jonathan Petit (Principal Researcher, OnBoard Security a Qualcomm company) for the help investigating the research ideas and sharing his valuable insights in regards to VANET and Mr. Mohammad Raashid Ansari (Senior Research Engineer, OnBoard Security a Qualcomm company) for helping out with setting up the simulation software's required for this research experiment. Authors would also like to thank Dr. Lance Fiondella

(Professor, ECE Department at UMass Dartmouth) for his invaluable guidance on Deep Learning techniques.

REFERENCES

[1] A. Rostami and M. Gruteser, "V2V Safety Communication Scalability Based on the SAE J2945/1 Standard," in *Proceedings of the 2018 ITS America Annual Meeting.*, no. 1, 2018, pp. 1–10. [Online]. Available: http://winlab.rutgers.edu/~rostami/files/pdf/rostami2018v2v.pdf

[2] Y. Shi, Y. E. Sagduyu, K. Davaslioglu, and J. H. Li, "Active Deep Learning Attacks under Strict Rate Limitations for Online API Calls," in *Proceedings of the 2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2018.

[3] I. Amit, J. Matherly, W. Hewlett, Z. Xu, Y. Meshi, and Y. Weinberger, "Machine Learning in Cyber-Security-Problems, Challenges and Data Sets," *arXiv preprint arXiv:1812.07858*, 2018.

[4] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," 2018.

[5] J. Petit and S. E. Shladover, "Potential Cyberattacks on Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, 2015.

[6] R. van der Heijden, T. Lukaseder, and F. Kargl, "VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *Proceedings of the 2018 International Conference on Security and Privacy in Communication Systems*. Springer, 2018, pp. 318–337.

[7] S. So, P. Sharma, and J. Petit, "Integrating Plausibility Checks and Machine Learning for Misbehavior Detection in VANET," in *Proceeedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 564–571.

[8] P. Sharma, H. Liu, W. Honggang, and Z. Shelley, "Securing wireless communications of connected vehicles with artificial intelligence," in *Proceedings of the 2017 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 2017, pp. 1–7.

[9] P. Sharma, J. Petit, and H. Liu, "Pearson Correlation Analysis to Detect Misbehavior in VANET," in *Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 4 2019, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8690964

[10] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[11] Y. Shi, Y. Sagduyu, and A. Grushin, "How to steal a machine learning classifier with deep learning," *2017 IEEE International Symposium on Technologies for Homeland Security, HST 2017*, no. April 2017, 2017.

[12] A. Kurakin, G. Brain, I. J. Goodfellow, and S. Bengio, "Workshop track -ICLR 2017 ADVERSARIAL EXAMPLES IN THE PHYSICAL WORLD," no. c, pp. 1–14, 2017.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

978-1-7281-5092-5/19/$31.00 ©2019 IEEE