Learning Based Vehicle Platooning Threat Detection, Identification and Mitigation

Eshaan Khanapuri, Tarun Chintalapati, Rajnikant Sharma, and Ryan Gerdes,

***Abstract*—The security of cyber-physical systems, such as vehicle platoons, is critical to ensuring their proper operation and acceptance to society. In platooning, vehicles follow one another according to an agreed upon control law that determines vehicle separation. It has been shown that a vehicle within a platoon and under the control of a malicious actor could cause collisions involving, or decrease the efficiency of, surrounding vehicles. In this paper we focus on detecting, identifying and mitigating so-called destabilizing attacks that could cause vehicle collisions. Our approach is decentralized and requires only local sensor information for each vehicle to identify the vehicle responsible for the attack and then deploy an appropriate mitigating controller that prevents collisions. A Deep Learning approach (Convolutional Neural Network) with various data preprocessing techniques are used to detect and identify the malicious vehicle. Results indicate that with noise upto 30% in range/relative speed data we achieve an accuracy upto 96.3%. Also, once the adversarial vehicle is localized, we derive conditions for controller gains using Routh Hurwitz criterion to mitigate the attack and ensure stability of the platoon. Realistic simulator CARLA and MATLAB simulation results validate the effectiveness of our proposed approaches.**

***Index Terms*—Autonomous platoon, security, Deep Learning, intrusion detection, identification, Routh Hurwitz criterion. .**

I. Introduction

The potential for automated vehicle technology to increase the safety and efficiency of transportation systems is reflected in the enormous investments being made in these technologies, by industry and government alike. Automated vehicles will eventually make the roads safer, by reducing fatalities and accidents, decrease travel times, and increase fuel efficiency. Even amidst all the numerous advantages, most experts agree that one of the key aspects that is ignored when it comes to automated vehicles is that of cyber-physical security. Automated vehicles depend upon an admixture of sensing and computational capabilities to accurately and correctly undertake maneuvers. However, these systems have been shown to be vulnerable to numerous attacks, ranging from falsifying sensor data to remotely compromising on-board computational units in order to take control of the vehicle itself. The possible outcomes of such compromises can be catastrophic, possibly leading to fatalities.

Vehicle platooning, wherein vehicles sense each other and act in a coordinated manner to maintain prescribed inter-vehicle distances [1], has been shown to improve traffic throughput, passenger comfort and fuel efficiency [2], [3], [4]. Given the tight coupling between vehicles there is a pressing need to quickly identify a compromised vehicle and isolate it, as it could possibly cause nearby vehicles to collide [5]. The principal problem that needs to be addressed in order to ensure the safety of the platoon is to detect the destabilizing attack, identify the attacker vehicle, and minimize the damage the attacker can cause.

String stability is an essential property of vehicle platoons that ensures the efficiency of the transportation system as a whole and the safety of individual vehicles. A platoon that is string stable guarantees that spacing errors (actual vs. desired spacing between vehicles) attenuate upstream from the source of the error [6]. In this work we assume the vehicle platoon follows a bi-directional, string stable control law, which allows for constant spacing [7], where each of the vehicle has the ability to measure the relative speed (range-rate) and distance (range) of their immediate neighbors (predecessor and follower vehicles).

While the stability, and other performance/comfort parameters, of vehicle platooning has received considerable attention, very few works have considered the operation of platoons in adversarial environments wherein an actor attempts to induce negative effects. In [5] it was shown that a platoon can be destabilized by a single attacker vehicle modifying parameters of its the control law, i.e., the modification of the control law of a single vehicle was proved to cause the platoon to oscillate in an unbounded manner, eventually leading to catastrophic collisions. The first step towards solving this problem lies in detecting the onset of the attack. In the field of network and cyber-physical systems security, this is a well-studied problem that falls under the category of intrusion/anomaly detection.

Pasqualetti et al. [8] proposed an anomaly detection scheme for networks with misbehaving nodes. They considered a decentralized consensus network, meaning that agents use only information derived from neighbors [9]. Machine learning techniques have also been utilized for anomaly detection. Alkasassbeh et al. in [10] investigated the problem of intrusion detection using ML methods, such as BayesNet, Multi-layer Perceptron (MLP) and Support Vector Machines (SVM) on a management information based data set. Almseidin et al. in [11] furthered this approach and applied multiple ML classifiers on a Knowledge Discovery and Data mining (KDD) intrusion dataset.

Applying intrusion detection for an adversarial agent in a vehicular platooning scenario has been studied in few works. Sajjad et al. in [3] proposed an attack detection

This work was supported by National Science Foundation (NSF) grant award number 1410000.

E.Khanapuri, T.Chintalapati and R.Sharma are with the Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati, Cincinnati, OH, 45220 USA

R.Gerdes is with Department of Electrical and Computer Engineering, Virgina Tech, Blacksburg, Virgina.

and identification scheme using low pass filters that acts on measurement residuals; based on the direction of the source of the attack a sliding mode controller was used to prevent collisions, at the expense of platoon control. Jagielski et al. [12] studied the effects of a maliciously controlled vehicle that aims to cause crashes in a string of connected cars by influencing the sensor readings. The detection strategy studied for these attacks was performed either through a physics-based approach or a Hidden Markov Model (HMM) approach. Although sensor readings are used for the identification of the attacker vehicle, the attacker and the victim vehicle are pre-determined in all the cases studied. Dadras et al. in [13] examined the identification of an attacker vehicle in a platoon using a combination of a system identification and clustering algorithms. In [14] consider a distributed sensor deception attack on platoons. They use a distributed kalman filter and proposed a generalized likelihood ratio to detect the sensor deception attacks.

Recent research also suggests that Convolutional Neural Networks (CNN), can outperform state of the art machine learning techniques for time series data both in terms of noise robustness and accuracy [15] [16]. Fault diagnosis in sensors using CNNs has been studied by Dingfei et al. in [17], where they residual signals of various sensors from a UAV were transformed to their corresponding time-frequency map using Short Time Fourier Transform (STFT). These time-frequency maps were then passed through a CNN for feature extraction and then used for fault diagnosis.

Various works have also addressed mitigation strategies in vehicle platoons under different types of adversarial settings. Petrillo et al in [18] have proposed a consensus-based control strategy to combat communication impairments in the case of message falsification attacks on platoons. Patounas et al. examined different jamming attack scenarios and evaluated defence schemes such as beamforming to mitigate the attacks [19]. Jin et al. in [20] proposed an adaptive contoller for time-invariant sensor and actuator attacks on a team of connected vehicles.

Most of the works consider only detection of an attack in platoon, but in this work we not only detect but also localize (identify) the adversarial vehicle and take a Deep Learning approach to solve it. This work is also an extension of [21] in which we compared the effectiveness of fully connected Deep Neural Networks (FCDNN) and CNNs in the detection and localization of an attack in a fixed size platoon. Previously, we have considered a global information scenario and a local information scenario. The global information case is where an assumption is made that there exists an entity that receives sensor data from all the vehicles and processes it in order to identify the attacker vehicle. As this is not practically feasible, In this paper developed a local information scenario where each vehicle has an independent network and access to only local sensor information of their immediate neighboring vehicles. The global information scenario existed as a standard of comparison to check the performance of the local information scenario with both FCDNNs and CNNs in the presence of noisy sensor data.

One of the biggest challenges with validating our approach was the feasibility and the costs associated with it. Hardware implementation is usually a good approach but is not always cost effective. Another commonly used approach in the field of autonomous vehicle research is validation using simulators. There are a number of simulators for this purpose such as the TORCS simulator [22], MATLAB automated driving toolbox [23], Udacity simulator [24], and the CARLA simulation engine.

For the purpose of this work, we have used the CARLA simulator for validating our approach. CARLA (Car Learning to Act) is an open simulator for urban driving specifically designed to further research into training, prototyping, and validation of autonomous driving models. CARLA has been built for flexibility and realism in the rendering and physics simulation [25].

The main contributions of this paper are as follows:

(i) Detection and Identification (Localization) of an adversarial vehicle in a platoon by using just local sensor information has been achieved.

(ii) Various preprocessing techniques which convert time series sensor data into images such as Time Series to Gray Scale (TSGS), Short Time Fourier Transform (STFT), Gramian Angular Fields (GAF) and Markov Transition Fields (MTI) have been explored. This is due to the fact that CNNs perform distinctively with images as input data. From the above mentioned techniques our CNN achieves an accuracy upto 96.3% in localizing the adversarial vehicle with the noise levels upto 30%.

(iii) We have validated our work on two platforms, MATLAB and CARLA. A point mass platoon model is simulated in MATLAB for initial study and for more realistic approach open source simulator CARLA which takes care of the detailed physics model and is specially designed for the prototyping and validating autonomous driving models. The results on both the platforms are similar showing the consistency of our approach.

(iv) Once the attack in the platoon is detected and localized accurately, we have derived conditions for the control gains of the non attacked vehicles using Routh Hurwitz stability criterion such that the attack is mitigated by the adversarial vehicle.

The remainder of this paper is organized as follows. In section II, we discuss the platoon vehicle dynamics and the attacker model. Section III describes the methodology used for the detection and localization process using various data pre-processing techniques for different information availability scenarios. Section IV, we discuss the results on detecting and localizing an adversarial agent in both CARLA and MATLAB environment. Section V, explains the attack mitigation section and its results. Finally Section VI concludes the paper with some future directions.

II. Problem Statement and Assumptions

In this section, we will discuss how the vehicles in the platoon are modelled, the control law, and the attacker model.

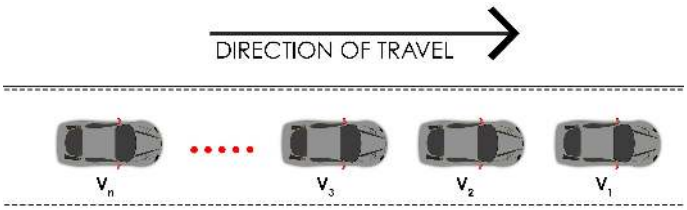


Fig. 1: Platoon model

Fig. 2: Platoon model in CARLA

A. Vehicle and Platoon Model

In this paper, we have assumed a fixed size platoon of 10 vehicles (For MATLAB simulation) travelling along a straight path on a highway. Each vehicle is assumed to have sensors that can measure the ranges and relative velocities of their immediate neighbouring vehicles. As per [26] and [27], the maximum recommended platoon size is 15 vehicles in order to ensure good throughput. Assuming a straight path for the vehicle platoon simplifies the problem to a great extent and eliminates the need to implement a lateral controller for the vehicles while approximating a highway scenario.

The vehicle model that we used for MATLAB environment, is a jerk-model that captures a majority of the vehicle dynamics while being easy to model and analyze

$$\dot{x_i} = v_i \tag{1}$$

$$\dot{v_i} = a_i \tag{2}$$

$$\dot{a_i} = j_i \tag{3}$$

where x_i, v_i, a_i and j_i are the position,velocity, acceleration and jerk of the i_{th} vehicle respectively. The control law that governs the entire vehicle platoon has to ensure that all the vehicles in the platoon converge to the leader vehicle's speed while maintaining a constant spacing between each of the vehicles. The equations for the control law are given by,

$$a_c = u_i \tag{4}$$

$$j_i = K_j(a_c - a_i) \tag{5}$$

where, a_c is the commanded acceleration, u_i is the control input and K_j is the proportional gain that exists to mimic real world actuation delay as closely as possible.

The first vehicle is the leader vehicle and is responsible for maintaining the speed with which the entire platoon travels. In our case, this desired speed is set to 30 m/s in order to represent a highway scenario. Moreover, the constant spacing that is to be maintained between each of the vehicles is given by σ_{ref} and for this paper, we have chosen this value to be 4 meters.

The control input, u_i, for the i_{th} vehicle in the platoon model is given by,

$$\begin{aligned} u_i = k_p(x_{i+1} - x_i - \sigma_{ref}) + k_p(x_{i-1} - x_i + \sigma_{ref}) \\ + k_d(v_{i+1} - v_i) + k_d(v_{i-1} - v_i) \end{aligned} \tag{6}$$

where, σ_{ref} is the desired distance that is to be maintained between each vehicle, k_p and k_d are the proportional gain and derivative gain respectively, x_i is the position of the i_{th} vehicle, x_{i+1} is the position of the vehicle behind it and x_{i-1} is the position of the vehicle ahead it. Similarly, v_i is the speed of the i_{th} vehicle, v_{i+1} is the speed of the vehicle ahead of it and v_{i-1} is the speed of the vehicle behind it.

All the vehicles in the platoon have a minimum speed saturation value of 0m/s, to ensure that the vehicles do not move in the opposite direction. The vehicles also have a jerk value saturation set to 1.3 m/s^3 that prevents them from accelerating to unrealistic values. The minimum k_p and the k_d values for stabilizing the platoon based on the number of vehicles, were found to be 1 and 10 respectively, by referring to [5]. Fig. 3 show that these parameters lead to stability.

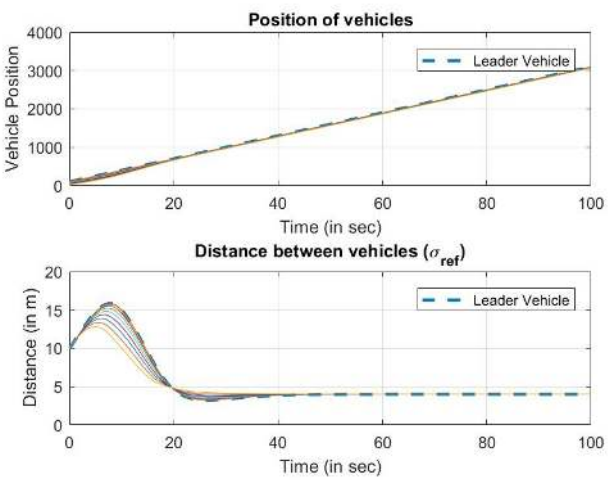
B. Attacker model

The attacker is a single vehicle that can alter its control law by modifying the k_d gain. The reason for choosing the k_d gain is to simulate the worst possible condition, i.e by varying the gain associated to velocity can cause more harm than the k_p gain. The attacker can achieve this variation through constantly accelerating and braking, causing the neighbouring vehicles to either slow down or speed up in order to satisfy the constraints of their individual control law and maintain the desired spacing. Even though it has been shown in [5] that the attacker can be present and destabilize the platoon from any position in the platoon, we are assuming that the leader vehicle cannot be an attacker as it represents a special case which would cause the follower vehicles to expend any amount of energy, as studied in [28]. The criteria for string stability, as stated in [29], is given as follows

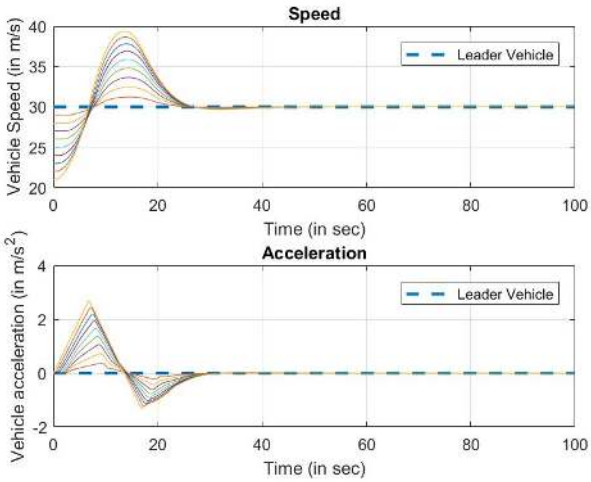
$$|G_i(s)| = \left| \frac{z_i}{z_{i+1}} \right| < 1 \ for \ i = 1, 2, ..., n-2 \tag{7}$$

where z_i is the spacing error between the i^{th} and the $i^{th}+1$ vehicles and $|G_i(s)|$ is the magnitude of the (error) transfer function between the i^{th} and $i^{th}+1$ vehicles.

The minimum k_d gain required for an attacker to cause oscillations and destabilize from each position in the platoon, based on the platoon size, were calculated by referring to [5]. These gains make the system to oscillate and eventually become unstable, leading the vehicles to collide into one another. Since it is unreasonable to analyze the system after

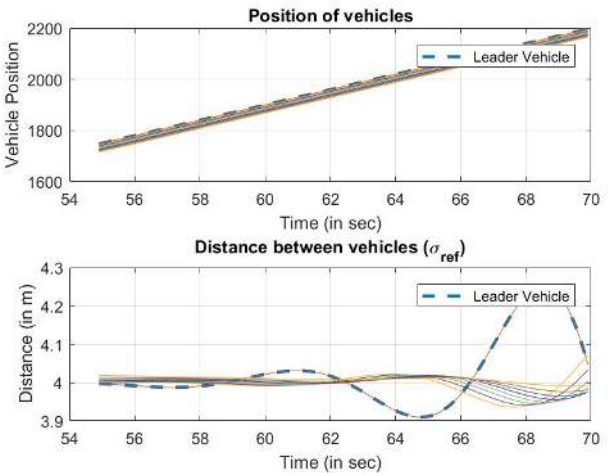


(a) Position and Inter-vehicular distance

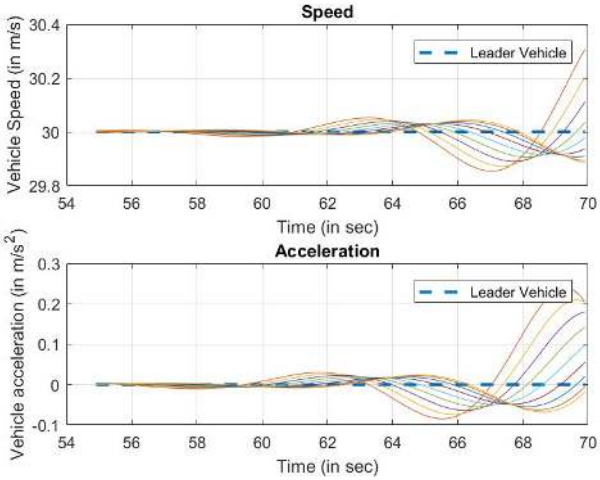


(b) Speed and acceleration of vehicles in the platoon

Fig. 3: Platoon response when under stable condition



(a) Position and Inter-vehicular distance



(b) Speed and acceleration of vehicles in the platoon

Fig. 4: Platoon response when under attack

the vehicles crash into each other, the data for all the cases is collected once there is an attack and up until the point before a crash occurs. Fig. 4 shows the position, σ_{ref}, speed and the acceleration plots in the event of an attack.

III. Adversarial Agent Detection and Localization

We have considered a 10 vehicle platoon for MATLAB simulation and 6 vehicle platoon for CARLA, with a single adversarial agent whose control gains are modified such that it makes the platoon unstable. Due to this instability, the cars accelerate and decelerate abruptly, which may lead to collisions. To avoid these collisions we need to first detect if there is an attack and also localize it,i.e to identify the attacker vehicle's position. We assume that these cars are equipped with sensors like LIDAR/RADAR which gives the relative speed and range information. The intention here is to analyze the sensor data to detect and identify an attack, if any, before a collision can occur. This now turns into a pattern recognition and classification problem where we want to discover different patterns in the range and relative speed data of the cars for various positions of the adversarial vehicle. Pattern recognition and classification problems are well studied by researchers and in recent years, Neural Networks (NN) have shown promising results compared to any other state-of-the-art machine learning techniques. In [21] we have compared a Fully connected Deep Neural Network (FCDNN) and a Convolutional Neural Network (CNN) in detection and localization of an attack and we found that the performance of CNN was better compared to FCDNN. Therefore, the emphasis here was given to improve the performance of the CNN with the help of various data pre-processing techniques and analyze the performance of CNN when it has more realistic data coming from the CARLA simulator. The rest of the section will explain the information

availability aspect, data pre-processing and the architecture used in CNN.

A. Information Availability

The information availability as we have defined it is of two types; global (centralized) and local (decentralized) information. The global information is an ideal case with a central node that has access to the sensor data of all the vehicles. The local information approach, which is more practical, the data is collected only from the immediate neighbouring vehicles. Each vehicle will have access to the range and relative speed information of the vehicle ahead and behind it. Hence, each vehicle will have the capability to perform detection and identification independently using only with the neighbouring vehicle's data and not depend on a central node for decision making. This makes our approach a decentralized one. Hence, only local information results are presented in this paper but for results on global information results please refer to [21]. A sample data of relative speed and range collected in the Global and local information format is given by,

$$v_{r_i} = [v_{i-1} - v_i, v_i - v_{i+1}] \tag{8}$$

$$X_{C_r} = [v_{r_2}, v_{r_3} ... v_{r_{n-1}}]^T \tag{9}$$

$$X_{G_r} = [\sigma_{ref_1}, \sigma_{ref_2}, ... \sigma_{ref_{n-1}}]^T \tag{10}$$

$$\sigma_{ref_i} = x_{i+1} - x_i \tag{11}$$

$$X_{D_r} = [v_{i-1} - v_i, v_i - v_{i+1}]^T \tag{12}$$

$$X_{L_r} = [x_{i-1} - x_i, x_i - x_{i+1}]^T \tag{13}$$

where x_i and v_i is the position and speed data of the vehicle that is performing the detection and identification for local information case, whereas for the global information a central node gets all the information about the vehicles. v_{i-1} and x_{i-1} is the speed and position data of the vehicle ahead and v_{i+1}, x_{i+1} is the speed, position data of the vehicle behind. These data are concatenated and passed on for training the CNN. Here, $i = 2, 3...n-1$, while X_{D_r} and X_{L_r} are one sample vector of relative speed and range respectively for local information case. X_{C_r} and X_{G_r} are one sample vector of relative speed and range respectively for global information case.

These relative speed and range data are usually obtained using a LIDAR and RADAR sensor, which have an accuracy of $\pm 3cm$ and $\pm 0.05m/s$ respectively [30]. Based on this information from sensor datasheets, the noise is modelled as a Gaussian distribution $N(\mu, \sigma^2)$ from 10 to a maximum of 30%. The platoon simulation, including the attack, runs for a 100 seconds in MATLAB environment and 300 seconds in CARLA environment. The time of attack is randomized to occur between the 45^{th} to 55^{th} second in MATLAB and 190^{th} to 200^{th} in CARLA. This window can be modified without any loss in generalization. The data is sampled at 0.1 second and the data is collected between 55^{th}sec to 70^{th}sec

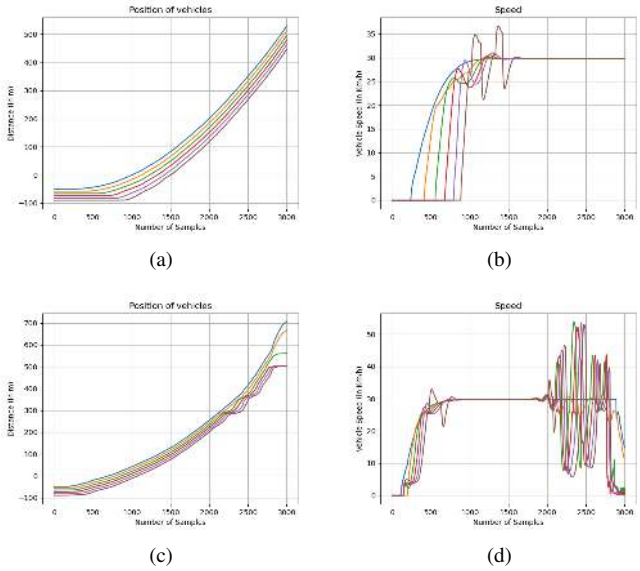


Fig. 5: (a) Position of vehicles in CARLA (b) Speed of vehicles in CARLA (c) Position of vehicles after attack in CARLA (d) Speed of vehicles after attack in CARLA

(15 seconds) for MATLAB and 200^{th} to 215^{th} in CALRA, as the vehicles begin to collide into one another after this point. The idea here is to detect and localize before any collisions occur for all attacker vehicle positions. Hence 70^{th}sec and 215^{th} serves as the upper bound for data collection. Now, in [21] the initial study showed that CNNs performed well in detecting and identifying the attacker vehicle. Hence, here we concentrate on improving the performance of CNNs by exploring various pre-processing methods which convert time series data to images as CNNs mostly achieves better results with images as input. Finally, test and compare these methods with different data types, noise levels, and simulation environment.

B. Data For Validation

For the purpose of validating our vehicle model, we chose to model our platoon in CARLA simulation environment using the same control law as stated above. CARLA is built as an open source layer over the Unreal Engine 4 [31]. The realistic physics that the engine provides would serve as a comparison benchmark for the jerk model developed in the MATLAB simulation environment. For the purpose of this paper, we have used CARLA version 0.9.5 and chose 'Town06' CARLA map to set up the platoon scenario. This readily available town map in CARLA is characterized by long highways with many highway entrances and exits. Due to the constraints arising from the length of the highway available in the town map, we have limited the number of vehicles in the platoon to six. The code for setting up the platoon environment and collecting the data can be found on our github page at https://bit.ly/34asiR5.

C. Data Pre-processing

The time series data that is generated is pre-processed to enhance the features as possible in order to help the CNN to easily classify and improve its accuracy. The methods

experimented here are, converting the time series data to a gray scale image (TSGS), Short Time Fourier Transform (STFT) of the time series data, Gramian Angular Fields (GAF) and Markov Transition Fields (MTI).

1) Time series to Gray Scale Image

It has been shown in the literature that CNNs are great image classifiers and have even exceeded human performance levels [32]. We took advantage of this fact and converted the time series data into grayscale images for training the CNNs. The Fig.6 shows the block diagram of the process and.The sensor data (time series) obtained from vehicles are one dimensional vector, so this vector is normalized and then reshaped into a 2D matrix. The values in the 2D matrix are between 0 to 1, and the normalized matrix is converted into a gray scale image where 0 represents black and 1 being white color. The values between 0 and 1 will have gray colors.

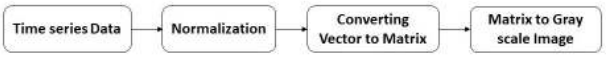


Fig. 6: Time series to Gray Scale Image Block Diagram

2) Short Time Fourier Transform

Short Time Fourier Transform (STFT) is a time frequency analysis that helps determine the frequency and phase content of the signal. In the recent literature, many have experimented with STFT method as a pre-processing method before sending the data over to the neural networks and have seen promising results [33], [34], [17] and hence we decided to experiment with STFT images. Initially we applied STFT for time series data and analyzed both the frequency vs time and power vs time plots, and found that power vs time plots had richer features and therefore chose to train the CNN with the power vs time plots (images). Similarly even the range data of the vehicles are converted to STFT images before they are sent to CNNs for training. In Fig.7 shows the example of relative speed of vehicle-3 converted to TSGS images and STFT images with and without noise.

3) Gramian Angular Fields

Gramian Angular Field (GAF) is a method recently introduced in [35] to represent time series data into images. Usually time series data are represented in cartisean co-ordinates but in GAF they are represented in polar co-ordinates. Each element in the gramnian matrix is the cosine of the summation of angles. Given a time series $S = \{x_1, x_2...x_l\}$ of l real valued observations, we normalize S such that they fall in the range [-1 1]:

$$\tilde{x_i} = \frac{(x_i - max(S)) + (x_i - min(S))}{max(S) - min(S)} \tag{14}$$

Thus we can represent the rescaled time series $\tilde{S}$ as,

$$\begin{cases} \phi = \arccos \tilde{x_i}, -1 \leq \tilde{x_i} \leq 1, \tilde{x_i} \in \tilde{S} \\ r = \frac{t_i}{N}, t_i \in N \end{cases} \tag{15}$$

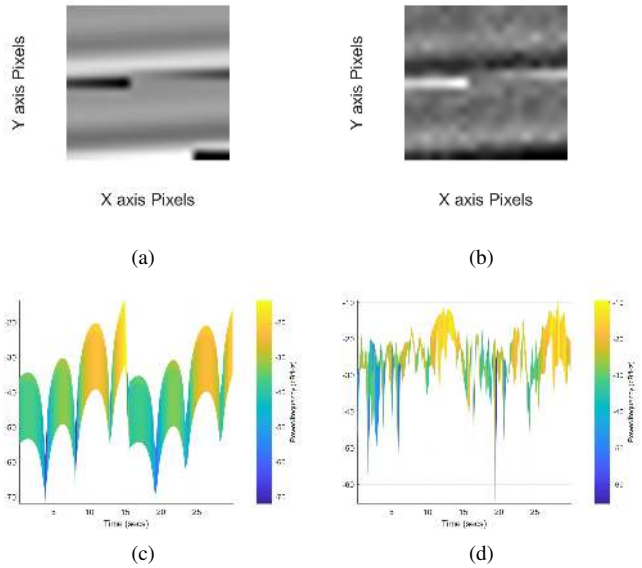


Fig. 7: (a) relative speed vehicle-3 no noise TSGS image (b) relative speed vehicle-3 with noise TSGS image (c) relative speed vehicle-3 no noise STFT image (d) relative speed vehicle-3 with noise STFT image

Here t_i is the time stamp and N is the constant factor to regularize the span of the polar coordinate system.

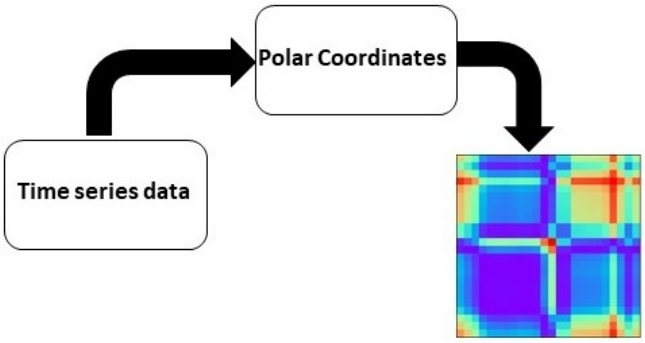


Fig. 8: Gramian Angular Field Procedure

Bijective and the polar coordinates preserving the temporal dependence are the two important properties of equation 15. Bijective because $cos(\phi)$ is monotonic when $\phi \in [0\ \pi]$. For a given time series the mapping produces a unique inverse function in the polar coordinates. Once the time series is converted to polar co-ordinates we can utilize the angular perspective by taking the trigonometric sum between each point to identify temporal correlation within different time intervals. Therefore, GAF can be written as,

$$G = \begin{bmatrix} cos(\phi_1 + \phi_1) & \dots & cos(\phi_1 + \phi_n) \\ cos(\phi_2 + \phi_1) & \dots & cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \dots \\ cos(\phi_n + \phi_1) & \dots & cos(\phi_n + \phi_n) \end{bmatrix} \tag{16}$$

After transforming to polar coordinates, time series is taken at each time step as 1-D metric space. By defining the inner product as,

$$< x, y >= x.y - \sqrt{1-x^2}.\sqrt{1-y^2} \tag{17}$$

The Gramian matrix G is given by,

$$G = \begin{bmatrix} < \tilde{x_1}, \tilde{x_1} > & \dots & < \tilde{x_1}, \tilde{x_n} > \\ < \tilde{x_2}, \tilde{x_1} > & \dots & < \tilde{x_2}, \tilde{x_n} > \\ \vdots & \ddots & \dots \\ < \tilde{x_n}, \tilde{x_1} > & \dots & < \tilde{x_n}, \tilde{x_n} > \end{bmatrix} \tag{18}$$

4) Markov Transition Field

Markov Transition Field (MTI) is another method similar to GAF introduced in [35]. Here the time series S is divided into P quantile bins where each x_i is assigned to corresponding bins $p_j (j \in [1, P])$. Transitions among the quantile bins are counted like a first order markov chain along time axis and P x P weighted adjacency matrix Z is constructed. The frequency with which a point in a quantile p_j is followed by a point in the quantile p_i is given by z_{ij}. Normalizing $\sum_j z_{ij} = 1$, Z will be the Markov transition matrix. This matrix is independent to the distribution of S and temporal dependency on time t_i. In the absence of temporal dependency there is a lot of information loss in matrix Z. Hence Markov Transition Field is defined as,

$$M = \begin{bmatrix} z_{ij|x_1 \in p_i, x_1 \in p_j}, & \dots & z_{ij|x_1 \in p_i, x_1 \in p_j} \\ z_{ij|x_1 \in p_i, x_1 \in p_j}, & \dots & z_{ij|x_1 \in p_i, x_1 \in p_j} \\ \vdots & \ddots & \dots \\ z_{ij|x_1 \in p_i, x_1 \in p_j}, & \dots & z_{ij|x_1 \in p_i, x_1 \in p_j} \end{bmatrix} \tag{19}$$

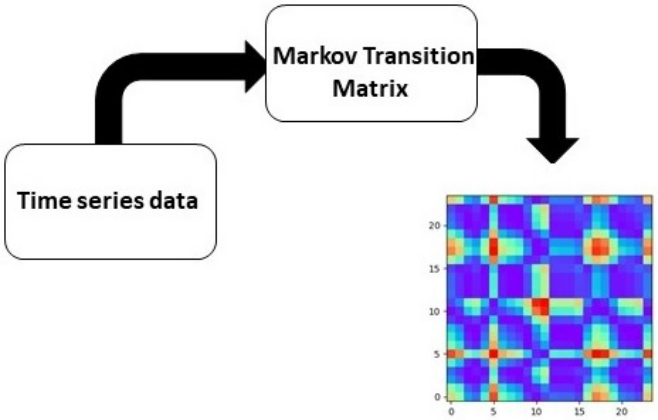


Fig. 9: Markov Transition Field Procedure

The procedure to generate the MTI is shown in the Fig.9.

D. Convolutional Neural Network

The first component of CNN si convolutional layer which performs a convolution operation with the previous layer with m filters of size k x l which is the sliding window. The second compenent is the max pooling layer, here the feature map obtained after the convolutions is divided into sections and each section will be represented by its maximum value. The final component is the fully connected layer which will contain the extracted features from the convolution and pooling layers. These extracted features is then used for classification. Equations for each of these layers are given in [15]. The activation functions that work best when the input was a grayscale image are Sigmoid and Relu. Exponential Linear Units (ELU) worked best when the input was an STFT, GAF and MTI images. Also, in [36], [37] it has been shown how ELUs can not only help with faster learning but also gives a higher classification accuracy. The final output layer has a Softmax activation function for both the cases. The architecture used in our work is shown in the Fig. 10 and the parameters are as follows, 16 filters of size 2x2, stride is set to 1 and the pooling was also of size 2x2. This architecture is used for data generated by both CARLA and MATLAB simulation.

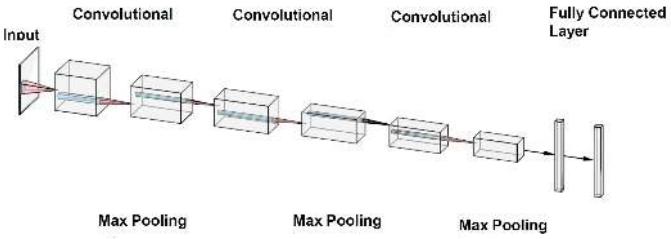


Fig. 10: Architecture of CNN

IV. Results for Adversarial Agent Detection and Identification

The results presented in this paper is divided into two main categories, detection and identification (localizing the attacker vehicle) of the attack for data generated from both CARLA and MATLAB simualtion. Results also shows the effects of various pre-processing techniques, data types (Relative speed and range) and different noise levels on the accuracy of CNNs.

TABLE I: Detection Accuracy for CARLA Simulation

Information	**Data Type (Rel)**	**Input Type**	**Accuracy**
Global	speed/Range	TSGS	100%
		STFT	100%
		GAF	100%
		MTF	100%
local	speed/Range	TSGS	99.6%
		STFT	99.65%
		GAF	100%
		MTF	99.8%

The number of training samples for attack detection were 1000 images for each pre-processing technique, TSGS, STFT, GAF and MTF. Out of 1000 images, 250 were of no noise and the rest 750 images were with three levels of noise varying from 10% to 30%. There were 100 test images for all the cases. Table. I and Table. II shows the detection accuracy for data collected from CARLA and MATLAB simulation respectively.

When it comes to detecting an attack the accuracy is quiet high although, the data type or the pre-processing technique doesn't make any significant difference which is clear from Table. I and Table. II. Also, other machine learning algorithms can be used to solve the detection problem but due to high dimensionality of time series data Deep Neural Nets will be a better choice. For the attack vehicle identification, the test is performed using both CARLA and MATLAB simulation data, but only the local sensor information is considered due to its practical applications and advantages. The total number of training samples were 7200, out of which 720 samples were used for testing the network. In this work, we have assumed that the attacker can be only from vehicle-2 to vehicle-9, meaning that out of 7200 samples, 900 images were dedicated for each attack vehicle position.

TABLE II: Detection Accuracy for MATLAB Simulation

Information	**Data Type (Rel)**	**Input Type**	**Accuracy**
Global	speed/Range	TSGS	100%
		STFT	100%
		GAF	100%
		MTF	100%
local	speed/Range	TSGS	98.5%
		STFT	98.9%
		GAF	99.5%
		MTF	99.1%

For example, if vehicle-3 is performing the identification, range/relative speed data between (vehicle-2, vehicle-3) and (vehicle-3, vehicle-4) are collected and concatenated. The noise added to the range and relative speed are based on the data-sheet available for the sensors which are used in real autonomous cars [30]. So the sensors used in the real autonomous cars have noise levels less than 10% but we are validating the performance of CNNs for noise levels up to 3 times the actual. The accuracy of the networks is calculated as follows,

$$Accuracy = \frac{SDE}{TTS} \tag{20}$$

Here SDE is the sum of diagonal elements of the confusion matrix and and TTS is the total number of test samples. The accuracy calculation for the local sensor information case either for detection or identification is the average accuracy of all the vehicles in the platoon performing detection and identification of the attacker vehicle. Therefore the average accuracy is given by,

$$Average\ Accuracy = \frac{Acc_1 + Acc_2 + ... + Acc_n}{TNV} \tag{21}$$

Here Acc_1 upto Acc_n are the accuracy of the vehicle-1 upto vehicle-n performing detection and identification and TNV is the Total Number of Vehicles involved in detection and identification of the adversarial vehicle. TABLE. III and TABLE. IV show the results for adversarial vehicle identification using CNN with various preprocessing techniques, data type and noise levels when the data is collected from CARLA and MATLAB simulation environment respectively. From TABLE.III and TABLE.IV we can conclude the following, that is just with local sensor information we are able to detect and identify the adversarial vehicle having noise levels of 30% with high accuracy of upto 94.8% and 95.7% with relative speed as the data type in MATLAB and CARLA respectively. By using range as the data type we achieve 95.1% and 96.3% in MATLAB and CARLA respectively. This is a appreciable improvement for our previous work in [21] where we had achieved 88.7% with relative speed and 93.1% with range data for local sensor information case with maximum noise levels when the data is collected from MATLAB environment. The other observations made from this study is that GAF and MTI preprocessing techniques are better than TSGS or STFT. When it comes to which data type works better, range data as input CNNs achieve better accuracy compared to relative speed. All these observations are similar across both CARLA and MATLAB simulation environments.

TABLE III: Local Sensor Information Identification Accuracy for CARLA Simulation

Data Type	**Noise level(%)**	**Input Type**	**Accuracy(Avg)**
Relative Speed	10	TSGS	92.3%
	20		91.1%
	30		90.4%
Relative Speed	10	GAF	97.8%
	20		96.9%
	30		**95.7%**
Relative Speed	10	MTF	97.5%
	20		96.6%
	30		95.3%
Range	10	TSGS	94.5%
	20		92.7%
	30		91.4%
Range	10	GAF	98.5%
	20		97.5%
	30		**96.3%**
Range	10	MTF	97.9%
	20		97.0%
	30		96.1%

TABLE IV: Local Sensor Information Identification Accuracy for MATLAB Simulation

Data Type	**Noise level(%)**	**Input Type**	**Accuracy(Avg)**
Relative Speed	10	TSGS	91.1%
	20		90.3%
	30		89.2%
Relative Speed	10	STFT	95.6%
	20		94.4%
	30		93.1%
Relative Speed	10	GAF	97.5%
	20		96.6%
	30		**94.8%**
Relative Speed	10	MTF	97.1%
	20		96.5%
	30		94.5%
Range	10	TSGS	93.7%
	20		92.4%
	30		90.6%
Range	10	STFT	96.2%
	20		95.8%
	30		94.0%
Range	10	GAF	98.5%
	20		96.9%
	30		**95.1%**
Range	10	MTF	97.8%
	20		97.1%
	30		95.6%

V. Attack Mitigation

Once the platoon is under attack, our approach will help us detect and identify the adversarial vehicle. So after the detection and identification the next step will be to mitigate the attack induced by the attacker vehicle. So with the information of adversarial vehicle's position we show that the attack can be mitigated by changing the k_d values of the non attacker vehicles. So the objective here is to choose k_d gain value for each non attacker vehicle such that the real part of each eigen value of A remains negative. Here A is the coefficeint matrix of the states in error coordinates. We use Routh Hurwitz criterion to compute k_{dm} which is the mitigation gain of the non attacker vehicles to ensure the negative eigen value of A which also guarantees stability. Once we derive the conditions using Routh Hurwitz stability criterion we simulate it in MATLAB to validate the effectiveness of the derived conditions.

We know that platoon can be asymptotically or BIBO unstable when the attacker can bring at least one eigen value of A matrix to the real part. For the stability analysis we use the error coordinates formulation for n vehicle platoon. Also, for simplicity in this analysis we consider the acceleration model rather than the jerk model. The error coordinates and general form of A matrix in the absence of the attacker is given as,

$$\begin{aligned} z_1 &= x_1 - x_2 \\ y_1 = \dot{z_1} &= v_1 - v_2 \\ z_2 &= x_2 - x_3 \\ y_2 = \dot{z_2} &= v_2 - v_3 \\ z_3 &= x_3 - x_4 \\ y_3 = \dot{z_3} &= v_3 - v4 \\ &\vdots \\ z_{n-1} &= x_{n-1} - x_n \\ y_{n-1} = \dot{z_{n-1}} &= v_{n-1} - v_n \end{aligned}$$

To write it in the $\dot{x} = Ax$ form we have to take the derivative of the error coordinates,

$$\begin{aligned} \dot{z_1} &= v_1 - v_2 = y_1 \\ \dot{y_1} &= -2k_p z_1 - 2k_d y_1 + k_p z_2 + k_d y_2 \\ \dot{z_2} &= v_2 - v_3 = y_2 \\ \dot{y_2} &= k_p z_1 + k_d y_1 - 2k_{pz2} - 2k_d y_2 + k_p z_3 + k_d y_3 \\ &\vdots \\ \dot{z_{n-1}} &= v_{n-1} - v_n = y_{n-1} \\ \dot{y_{n-1}} &= k_p z_{n-2} + k_d y_{n-2} - 2k_p z_{n-1} - 2k_d y_{n-1} \end{aligned}$$

Here the acceleration model is given as,

$$\dot{x_i} = v_i \tag{23}$$

$$\begin{aligned} \dot{v_i} = k_p(x_{i+1} - x_i - \sigma_{ref}) + k_p(x_{i-1} - x_i + \sigma_{ref}) \\ + k_d(v_{i+1} - v_i) + k_d(v_{i-1} - v_i) \end{aligned} \tag{24}$$

Now in the presence of an attacker vehicle at the i^{th} position, $1 < i < n-1$ we show which elements of the A matrix are changed.

$$A(2(i-1), 2(i-1)) = -k_d - \hat{k_d}, A(2(i-1), 2i) = \hat{k_d} \tag{25}$$

$$A(2i, 2(i-1)) = \hat{k_d}, A(2i, 2i) = -k_d - \hat{k_d} \tag{26}$$

Where $\hat{k_d}$ is the attacker's gain. In [5] Dadras et.al have proved that if an attacker chooses a derivative gain $\hat{k_d} < -k_d$ will cause A to have atleast one eigen value with positive real part and therefore making the platoon unstable. So here we verify this condition using Routh Hurwitz stability critierion and also derive mitigation gains for non attacker vehicles to achieve stability.

Now, lets us consider three vehicle platoon case with the attacker's position being vehicle two. For Routh Hurwitz stability analysis we need to take the characteristic equation of A. 27 is the $(sI - A)$ matrix for three vehicle platoon case with attacker's position being vehicle two. The change in the elements of the $(SI - A)$ matrix is made using equation 25 and 26.

Note: The characteristic equation of A will not change with the change in the attacker's position.

$$V_3 = \begin{pmatrix} -s & 1 & 0 & 0 \\ -2k_p & -k_d - \hat{k_d} - s & k_p & \hat{k_d} \\ 0 & 0 & -s & 1 \\ k_p & \hat{k_d} & -2k_p & -kd - \hat{k_d} - s \end{pmatrix} \tag{27}$$

The characteristic equation of V_3 is,

$$\begin{aligned} V_{3c} = s^4 + (2k_d + 2\hat{k_d})s^3 + (k_d^2 + 2\hat{k_d}k_d + 4k_p)s^2 \\ + (4k_d k_p + 2\hat{k_d}k_p)s + 3k_p^3 \end{aligned} \tag{28}$$

The Routh Hurwitz table for V_{3c} is given below,

$$V_{3rh} = \begin{pmatrix} 1 & k_d^2 + 2\hat{k_d} + 4k_p & 3k_p^2 \\ 2k_d + 2\hat{k_d} & 4k_d k_p + 2\hat{k_d}k_p & 0 \\ Z & 3k_p^2 & 0 \\ X & 0 & 0 \\ 3k_p^2 & 0 & 0 \end{pmatrix} \tag{29}$$

$$Z = \frac{2k_d^3 + 6\hat{k_d}k_d^2 + 4\hat{k_d}^2 k_d + 4k_d k_p + 6\hat{k_d}k_p}{2k_d + 2\hat{k_d}} \tag{30}$$

$$X = \frac{Z(4k_d k_p + 2\hat{k_d}k_p) - 3k_p^2(2k_d + 2\hat{k_d})}{Z} \tag{31}$$

According to Routh Hurwitz criterion if the first column has a sign change then the system will be unstable. Therefore, if

any element in the first column of V_{3rh} is negative then the platoon becomes unstable. So considering 2^{nd} element of first column of V_{3rh} and and the condition $\hat{k_d} < -k_d$ will cause A to have atleast one eigen value with positive real part making the platoon unstable which is proved in [5],

$$\begin{aligned} 2k_d + 2\hat{k_d} &< 0 \\ \hat{k_d} &< -k_d \\ \hat{k_d} &= -\alpha k_d \\ 2k_d - 2(-\alpha k_d) &< 0 \\ 2k_d &< 2\alpha k_d \\ \alpha &> 1 \end{aligned}$$

Therefore if $\alpha > 1$ and $\hat{k_d} = -\alpha k_d$ were $\hat{k_d}$ is attacker vehicle's gain will make the 2^{nd} element of first column of V_{3rh} negative hence making the platoon unstable. This also verifies the condition proved in [5]. Now to bring back the stability the strategy used here is to vary the k_d gains of the non attacker vehicles such that it mitigates the effect of the adversarial vehicle. That is we have to make sure that all the terms in the first column of V_{3rh} should be positive. For the mitigation part we substitute $k_d = k_{dm}$ which will be the mitigation gain of the non attacker vehicles after the attack and we will again consider 2^{nd} element of first column of V_{3rh},

$$\begin{aligned} 2k_d + 2\hat{k_d} &> 0 \\ \hat{k_d} = -\alpha k_d, k_d &= k_{dm} \\ 2k_{dm} &> 2\alpha k_d \end{aligned}$$

$$k_{dm} > \alpha k_d \tag{32}$$

Accordingly, if $k_{dm} > \alpha k_d$ the 2^{nd} element of first column of V_{3rh} will be positive. To verify this condition for Z and X, we subsitute $k_d = k_{dm} = 2\alpha k_d$ and $\hat{k_d} = -\alpha k_d$,

$$Z = \frac{16\alpha^3 k_d^3 - 24\alpha^3 k_d^3 + 8\alpha^3 k_d^3 + 2\alpha k_d k_p}{4\alpha k_d - 2\alpha k_d} \tag{33}$$

$$Z = kp \tag{34}$$

$$X = \frac{k_p(8\alpha k_d k_p - 2\alpha k_d k_p) - 3k_p^2(4\alpha k_d - 2\alpha k_d}{kp} \tag{35}$$

$$X = 0 \tag{36}$$

Finally, from 34 and 36 we can say that all the terms will be positive with no sign change if $k_{dm} > \alpha k_d$ after the attack and hence mitigating the attack introduced by the adversarial vehicle. Next we derive conditions for 4 and 5 vehicle platoon and make a case for n vehicle platoon. In the end the simulation results and eigen value plots will validate the above derived conditions.

Now, let us consider 4-vehicle platoon case, with 3^{rd} vehicle as the attacker. Below is the matrix for $(sI - A)$ matrix,

$$V_4 = \begin{pmatrix} -s & 1 & 0 & 0 & 0 & 0 \\ -2k_p & -2k_d - s & k_p & k_d & 0 & 0 \\ 0 & 0 & -s & 1 & 0 & 0 \\ k_p & k_d & -2k_p & -kd - \hat{k_d} - s & k_p & \hat{k_d} \\ 0 & 0 & 0 & 0 & -s & 1 \\ 0 & 0 & k_p & \hat{k_d} & -2k_p & -k_d - \hat{k_d} - s \end{pmatrix} \tag{37}$$

The characteristic equation for V_4 is,

$$\begin{aligned} V_{4c} = s^6 &+ (4k_d + 2\hat{k_d})s^5 + (4k_d^2 + 6\hat{k_d}k_d + 6k_p)s^4 \\ &+ (k_d^3 + 3\hat{k_d}k_d^2 + 14k_p k_d + 6\hat{k_d}k_p)s^3 \\ &+ (6k_d^2 k_p + 6\hat{k_d}k_d k_p + 10k_p^2)s^2 \\ &+ (9k_d k_p^2 + 3\hat{k_d}k_p^2)s + 4k_p^3 \end{aligned} \tag{38}$$

Now by applying the same steps as 3-vehicle case, the conditions for attack and attack mitigation for a 4-vehicle case are as follows,

- If $\alpha > 1$ and $\hat{k_d} = -\alpha k_d$ were $\hat{k_d}$ is attacker vehicle's gain the platoon will be unstable.
- For attack mitigation, $k_d = k_{dm}$ and if $k_{dm} > 2\alpha k_d$ with $\hat{k_d} = -\alpha k_d$ and $\alpha > 1$ all the elements in the first column of the Routh Hurwitz will be positive making the platoon stable.

We repeat the procedure for 5-vehicle case. $(A - sI)$ matrix for 5-vehicle case is with 3^{rd} vehicle as the attacker is written as,

$$V_5 = \begin{pmatrix} -s & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2k_p & -2k_d - s & k_p & k_d & 0 & 0 & 0 & 0 \\ 0 & 0 & -s & 1 & 0 & 0 & 0 & 0 \\ k_p & k_d & -2k_p & -kd - \hat{k_d} - s & k_p & \hat{k_d} & 0 & 0 \\ 0 & 0 & 0 & 0 & -s & 1 & 0 & 0 \\ 0 & 0 & k_p & \hat{k_d} & -2k_p & -k_d - \hat{k_d} - s & k_p & k_d \\ 0 & 0 & 0 & 0 & 0 & 0 & -s & 1 \\ 0 & 0 & 0 & 0 & k_p & k_d & -2k_p & 2k_d \end{pmatrix} \tag{39}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -2k_p & -2k_d & k_p & k_d & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ k_p & k_d & -2k_p & -2k_d & k_p & k_d & 0 & 0 & \dots & 0 \\ \vdots & \ddots & & & & & & & & \\ 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \dots & 0 & 0 & 0 & 0 & k_p & k_d & -2k_p & -2k_d \end{pmatrix} \tag{22}$$

The charactersitic equation of V_5 is,

$$\begin{aligned} V_{5c} = S^8 &+ (6k_d + 2\hat{k_d})s^7 + (11k_d^2 + 10\hat{k_d}k_d + 8k_p)s^6 \\ &+ (7k_d^3 + 13\hat{k_d}k_d^2 + 32k_pk_d + 10\hat{k_d}k_p)s^5 \\ &+ (k_d^4 + 4\hat{k_d}k_d^3 + 34k_d^2k_p + 26\hat{k_d}k_dk_p + 21k_p^2)s^4 \\ &+ (8k_d^3k_p + 12\hat{k_d}k_d^2k_p + 47k_dk_p^2 + 13\hat{k_d}k_p^2)s^3 \\ &+ (18k_d^2k_p^2 + 12\hat{k_d}k_dk_p^2 + 20k_p^3)s^2 \\ &+ (16k_dk_p^3 + 4\hat{k_d}k_p^3)s + 5k_p^4 \end{aligned} \tag{40}$$

So the conditions for attack and attack mitigation for a 5-vehicle case are as follows,

- If $\alpha > 1$ and $\hat{k_d} = -\alpha k_d$ were $\hat{k_d}$ is attacker vehicle's gain the platoon will be unstable.
- For attack mitigation, $k_d = k_{dm}$ and if $k_{dm} > 3\alpha k_d$ with $\hat{k_d} = -\alpha k_d$ and $\alpha > 1$ all the elements in the first column of the Routh Hurwitz will be positive making the platoon stable.

By observing the conditions derived for 3, 4 and 5 vehicle platoon case we can write an expressions for instability and stability after attack for n vehicle platoon as,

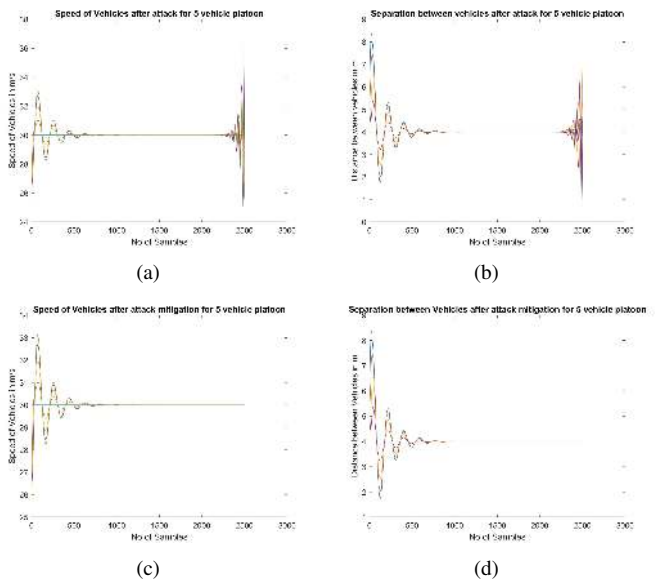
- when $\hat{k_d} = -\alpha k_d$ and if $\alpha > 1$ it will make the platoon unstable.
- To mitigate this attack if the k_d gains of the non attacker vehicles that is k_{dm} should satisfy the below condition,

$$k_{dm} > (n-2)\alpha k_d \tag{41}$$

Note: The value of α will have a limitation based on the acceleration limits of the vehicles. Hence the maximum value for α can be written as α_{max}. Also, we will only have only the information of position of the malicious vehicle and not on the value of α chosen to destabilize the platoon, 41 can also be written as,

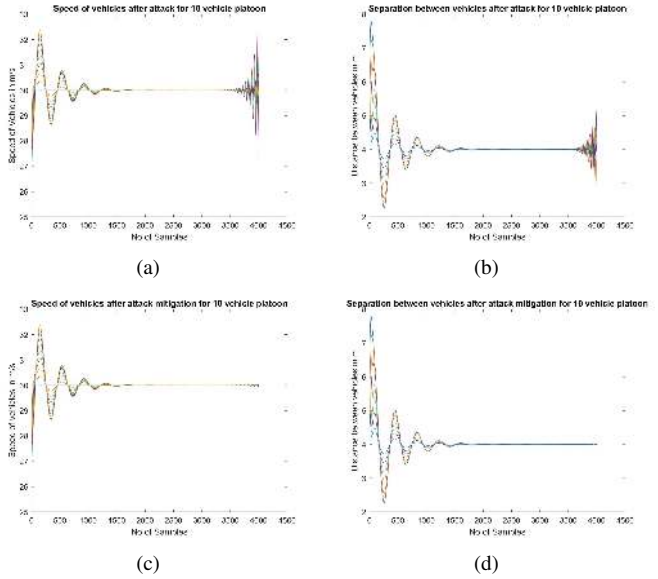
$$k_{dm} > (n-2)\alpha_{max} k_d \tag{42}$$

The conditions for 4, 5 and n-vehicle case are not explained in detail as 3-vehicle case due to space constraints. Now, the above mentioned conditions are validated using MATLAB simulation. Two cases are considered, 5-vehicle and 10-vehicle platoon. In the 5-vehicle platoon 2^{nd} vehicle is considered to be the attacker vehicle (attacker is chosen randomly). The simulation parameters for 5-vehicle platoon are as follows, $k_p = 1$, $k_d = 1$, $\alpha = 1.1$, $\hat{k_d} = -1.1$, $\alpha_{max} = 2$, speed = $30m/s$, distance between vehicles = $4m$ and simulation time = $300sec$ To mitigate the attack, we substitute the values to 42 and get $k_{dm} > 6$. Next, we consider a 10 vehicle platoon case with 6^{th} vehicle being the attacker vehicle. The simulation parameters for 10- vehicle platoon are same as 5-vehicle platoon except the simulation time = $400sec$. To mitigate the attack, again we substitute the values to 42 and get $k_{dm} > 16$. Fig.11, Fig.12 and Fig.13 shows the plots of speed, separation between vehicles and eigen values after attack and also shows the effect after applying mitigation gains for 5-vehicle and 10-vehicle case respectively. It is clear from these plots that after the attack mitigation the speed and separation between the vehicles settle down to its original value and all the eigen values lie in the negative half of the plane making the platoon stable.



(a) (b) (c) (d)

Fig. 11: (a) Speed of Platoon after attack for 5-vehicle platoon (b) Separation of vehicles after attack for 5-vehicle platoon (c) Speed of Platoon after attack mitigation (d) Separation of vehicles after attack mitigation

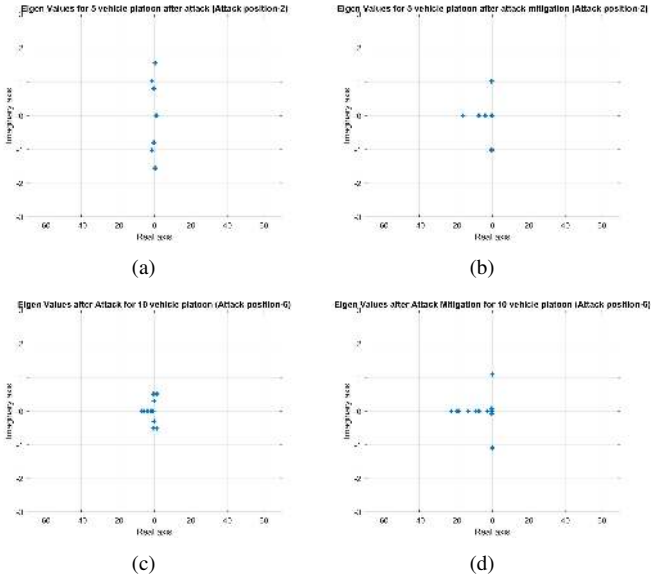


(a) (b) (c) (d)

Fig. 12: (a) Speed of Platoon after attack for 10-vehicle platoon (b) Separation of vehicles after attack for 10-vehicle platoon (c) Speed of Platoon after attack mitigation (d) Separation of vehicles after attack mitigation

VI. Conclusion

In this work we have shown that just with the local sensor information an attack from the adversarial vehicle in the



(a) (b) (c) (d)

Fig. 13: (a)Eigen Values after attack for 5-vehicle platoon (b) Eigen Values after attack mitigation for 5-vehicle platoon (c)Eigen Values after attack for 10-vehicle platoon (d) Eigen Values after attack mitigation for 10-vehicle platoon

platoon can not only be detected but also localized with high accuracy even with noise levels upto 30% using CNNs. The performance of the CNN is improved from our previous work by exploring various pre-processing techniques which converts time series data into images such as TSGS, STFT, GAF and MTI. From the results we have understood that with range as input data and with GAF or MTI as the pre-processing techniques, CNNs give the best accuracy in both MATLAB and CARLA environment. Once the attack in the platoon is detected and localized accurately, we have derived conditions using Routh Hurwitz criterion for the control gains of the non attacker vehicles such that the attack is mitigated and the platoon is bought to stable condition. The simulation and eigen value plots substantiate our proposed approach. Our future work will concentrate on scalability of the platoon, dealing with multiple attackers, comparing deep learning algorithms for time series classification and also study the effects of various types of cyber attacks on the platoon.

REFERENCES

[1] S. E. Shladover, "The california path program of ivhs research and its approach to vehicle-highway automation," in *Intelligent Vehicles' 92 Symposium., Proceedings of the*. IEEE, 1992, pp. 347–352.

[2] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa, "Overview of platooning systems," in *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

[3] I. Sajjad, D. D. Dunn, R. Sharma, and R. Gerdes, "Attack mitigation in adversarial platooning using detection-based sliding mode control," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. ACM, 2015, pp. 43–53.

[4] A. Davila, E. del Pozo, E. Aramburu, and A. Freixas, "Environmental benefits of vehicle platooning," SAE Technical Paper, Tech. Rep., 2013.

[5] S. Dadras, R. M. Gerdes, and R. Sharma, "Vehicular platooning in an adversarial environment," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ACM, 2015, pp. 167–178.

[6] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE transactions on automatic control*, vol. 41, no. 3, pp. 349–357, 1996.

[7] D. Swaroop, J. K. Hedrick, C. Chien, and P. Ioannou, "A comparision of spacing and headway control laws for automatically controlled vehicles1," *Vehicle system dynamics*, vol. 23, no. 1, pp. 597–625, 1994.

[8] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 5594–5599.

[9] A. Olshevsky and J. Tsitsiklis, "Convergence rates in distributed consensus and averaging, in proc," in *IEEE Conf. Decision Contr., San Diego, CA*, 2006.

[10] M. Alkasassbeh, "An empirical evaluation for the intrusion detection features based on machine learning and feature selection methods," *arXiv preprint arXiv:1712.09623*, 2017.

[11] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *Intelligent Systems and Informatics (SISY), 2017 IEEE 15th International Symposium on*. IEEE, 2017, pp. 000 277–000 282.

[12] M. Jagielski, N. Jones, C.-W. Lin, C. Nita-Rotaru, and S. Shiraishi, "Threat detection for collaborative adaptive cruise control in connected cars," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2018, pp. 184–189.

[13] S. Dadras, S. Dadras, and C. Winstead, "Identification of the attacker in cyber-physical systems with an application to vehicular platooning in adversarial environment," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 5560–5567.

[14] Z. Ju, H. Zhang, and Y. Tan, "Distributed deception attack detection in platoon-based connected vehicle systems," *IEEE transactions on vehicular technology*, vol. 69, no. 5, pp. 4609–4620, 2020.

[15] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[16] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *Ijcai*, vol. 15, 2015, pp. 3995–4001.

[17] D. Guo, M. Zhong, H. Ji, Y. Liu, and R. Yang, "A hybrid feature model and deep learning based fault diagnosis for unmanned aerial vehicle sensors," *Neurocomputing*, vol. 319, pp. 155–163, 2018.

[18] A. Petrillo, A. Pescape, and S. Santini, "A collaborative control strategy for platoons of autonomous vehicles in the presence of message falsification attacks," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2017, pp. 110–115.

[19] G. Patounas, Y. Zhang, and S. Gjessing, "Evaluating defence schemes against jamming in vehicle platoon networks," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2153–2158.

[20] X. Jin, W. M. Haddad, Z.-P. Jiang, and K. G. Vamvoudakis, "Adaptive control for mitigating sensor and actuator attacks in connected autonomous vehicle platoons," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 2810–2815.

[21] E. Khanapuri, T. Chitalapati, R. Sharma, and R. Gerdes, "Learning-based adversarial agent detection and identification in cyber physical systems applied to autonomous vehicular platoon," in *5th International Workshop on Software Engineering for Smart Cyber-Physical Systems*. IEEE/ACM, 2019.

[22] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[23] MATLAB. matlab automated driving toolbox. [Online]. Available: https://in.mathworks.com/products/automated-driving.html

[24] UDACITYr. Udacity sim. [Online]. Available: https://github.com/udacity/self-driving-car-sim

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[26] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[27] T. Robinson, E. Chan, and E. Coelingh, "Operating platoons on public motorways: An introduction to the sartre platooning programme," in *17th world congress on intelligent transport systems*, vol. 1, 2010, p. 12.

[28] R. M. Gerdes, C. Winstead, and K. Heaslip, "Cps: an efficiency-motivated attack against autonomous vehicular transportation," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 99–108.

[29] D. Yanakiev and I. Kanellakopoulos, "A simplified framework for string stability analysis in ahs1," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 7873–7878, 1996.
[30] M.-A. Mittet, H. Nouira, X. Roynard, F. Goulette, and J.-E. Deschaud, "Experimental assessment of the quanergy m8 lidar sensor," in *ISPRS 2016 congress*, 2016.
[31] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com
[32] L. Chen, S. Wang, W. Fan, J. Sun, and S. Naoi, "Beyond human recognition: A cnn-based framework for handwritten character recognition," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, 2015, pp. 695–699.
[33] L.-H. Wang, X.-P. Zhao, J.-X. Wu, Y.-Y. Xie, and Y.-H. Zhang, "Motor fault diagnosis based on short-time fourier transform and convolutional neural network," *Chinese Journal of Mechanical Engineering*, vol. 30, no. 6, pp. 1357–1368, 2017.
[34] X. Wang, G. Huang, Z. Zhou, and J. Gao, "Radar emitter recognition based on the short time fourier transform and convolutional neural networks," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2017, pp. 1–5.
[35] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 1, 2015.
[36] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
[37] M. Heusel, D.-A. Clevert, G. Klambauer, A. Mayr, K. Schwarzbauer, T. Unterthiner, and S. Hochreiter, "Elu-networks: fast and accurate cnn learning on imagenet," *NiN*, vol. 8, pp. 35–68, 2015.

Rajnikant Sharma received the B.E. degree in electrical engineering from the University of Rajasthan, Jaipur, India, in 2003, the M.E. degree in aerospace engineering from the Indian Institute of Science, Bangalore, India, in 2005, and the Ph.D. degree in electrical engineering in 2011 from the Brigham Young University, Provo, UT. He is currently an assistant professor at the Aerospace Engineering and Engineering Mechanics Department at the University of Cincinnati. From August 2013 to December 2016, he was with the Electrical and Computer Engineering Department at Utah State University, Logan, UT, where he was an assistant professor. In 2011-2013 he was a postdoctoral fellow at Academy Center for UAS Research, US Air Force Academy, Colorado. From 2005-2007 he was a scientist B at the Center for Airborne Systems, Defense Research and Development Organization, Ministry of Defense, Bangalore, India. His primary research interest are guidance, navigation, and control of unmanned aerial vehicles and multiple vehicle coordination, control, and localization. He is a member of the IEEE and of AIAA.

Eshaan Khanapuri received the B.E. degree in Electronics and Communication engineering from Visvesvaraya Technological University, Belgaum, India, in 2015. He is currently Ph.D Candidate at the Aerospace Engineering and Engineering Mechanics Department at the University of Cincinnati, Ohio, USA. From 2015 to 2016, he was with the Aerospace Engineering Department at Indian Institute of Science (IISc) as a project assistant, where he worked on project funded by Defense Research and Development Organization for autonomous navigation of Micro Aerial Vehicles (MAV). His primary research interests are control systems, cyber physical systems, autonomous vehicles, sensor security, controls security, machine learning and Deep Learning.

Tarun Chintalapati received his B.E degree in Aerospace Engineering from Hindustan University, Chennai, India in 2017 and his Master's degree in Aerospace Engineering from the University of Cincinnati in 2020. He is currently working as a research scientist with Givaudan Flavors Corporation since 2020. His research interests include guidance, navigation and control of autonomous vehicles and cyber-physical systems, security in vehicle platoons, and machine learning techniques in intrusion detection.

Ryan M. Gerdes is an Assistant Professor in the Department of Electrical and Computer Engineering at Virginia Tech. He received his PhD. in electrical engineering from Iowa State University for his work on device fingerprinting in August 2011. From 2011–2016 he was an Assistant Professor at Utah State University. His research interests include cyber-physical systems security, with an emphasis on the operation of autonomous systems in unknown, uncertain, and adversarial environments; device fingerprinting, embedded systems security; sensor security; controls security; and cybersecurity.