Misbehavior Detection in Vehicular Communication Networks

## Introduction

V2X, or Vehicle-to-Everything, is a technology that allows vehicles to communicate with other vehicles, pedestrians, and infrastructure. It uses wireless communication and other sensors to enable vehicles to exchange information and coordinate their actions. The cooperative nature of VANET applications attracts cyber attackers to perform a successful attack. Thus, to identify manipulation in vehicular data, the creation of a misbehavior-detection system is crucial.

VEINS, or the Vehicular Network Simulation Framework, is an open-source simulation framework for studying vehicular communication networks.

## Methodology

The proposed system will split into 2 phases. In the first phase, we shall gather vehicular data. The data is the information passed among all the vehicles in the V2X network. We shall simulate a VEINS simulator which will create and run a traffic simulation that will record all the messages passed among every vehicle. These messages which are transferred between vehicles are position, speed, acceleration, and heading. After collecting data over a time period, the data will be filtered and processed using data visualization and preprocessing techniques. Finally, a dataset will be created which will be used to test for misbehavior.

In phase two, three methods are selected to attempt to classify this data, a supervised learning algorithm, an unsupervised learning algorithm, and a deep learning algorithm. For the supervised learning algorithm, an SVM model was trained and tested against the data. SVMs can handle high dimensional non-linear data and are commonly used in anomaly detection. For unsupervised learning, a one-class SVM model was trained and tested. In real-world situations, the labels for whether or not a vehicle is genuine or attacking are not provided, so in an attempt to mimic these conditions, an unsupervised model was trained. For deep learning, a CNN model was trained and tested. CNNs are commonly used for time series classification tasks.

## Dataset

Total number of vehicles: 6298

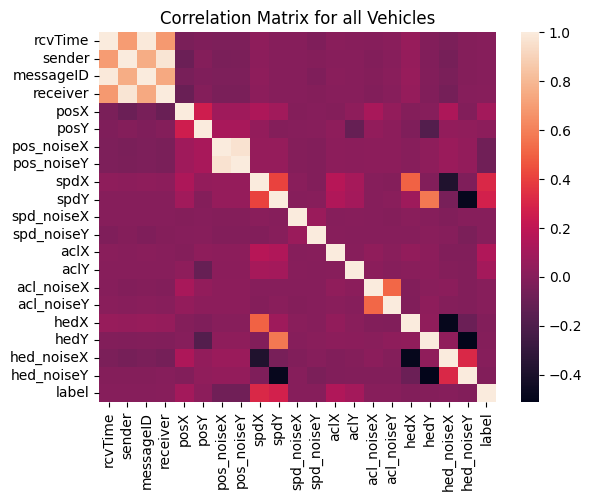
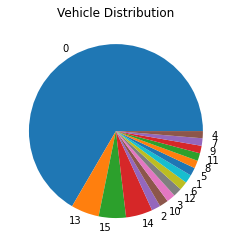
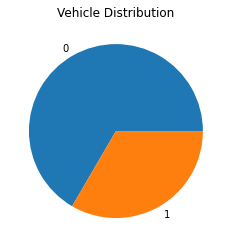
For the dataset, each file will have the following information-

* **MessageID:** Id of each message shared between sender and receiver.
* **Sender**: The vehicle which sends messages. It can either be a genuine message or manipulated message to confuse the receiver.
* **Receiver**: The vehicle which receives messages from the sender. For each receiver, there are multiple sender vehicles.
* **RcvTime**: Time at which the message was received. You can track a sender and receiver message over a period of time to see the difference between genuine or attacker data for either position, speed, acceleration, and heading.
* **Pos**: Position in X & Y direction of the sender vehicle
* **Spd**: Speed in X & Y direction of the sender vehicle
* **Acl**: Acceleration in X & Y direction of the sender vehicle
* **hed**: Heading in X & Y direction of the sender vehicle
* **PosNoise, SpdNoise, AclNoise, HedNoise**: Noise of the data
* **Label**: The value of the sender vehicle whether it is a genuine vehicle (value 0) or attack vehicle (values 1 to 15).

## 

## Data Visualization and Preprocessing

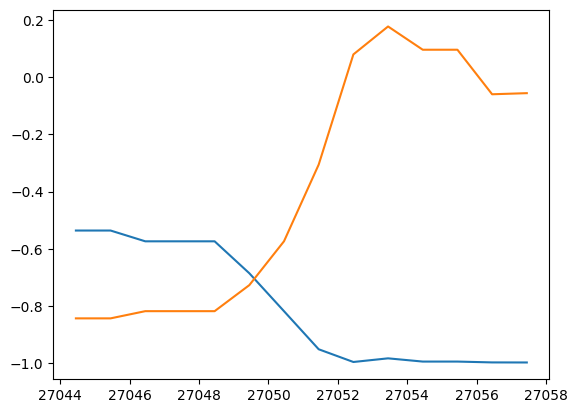
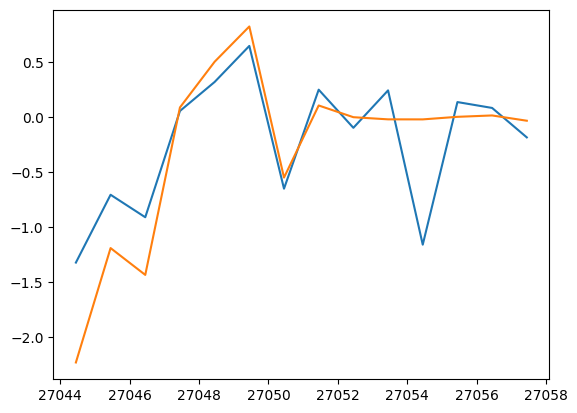
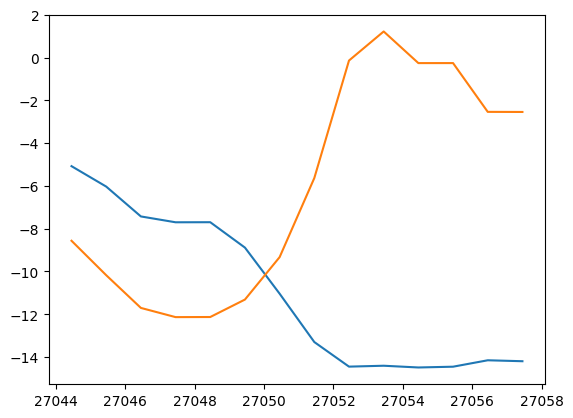
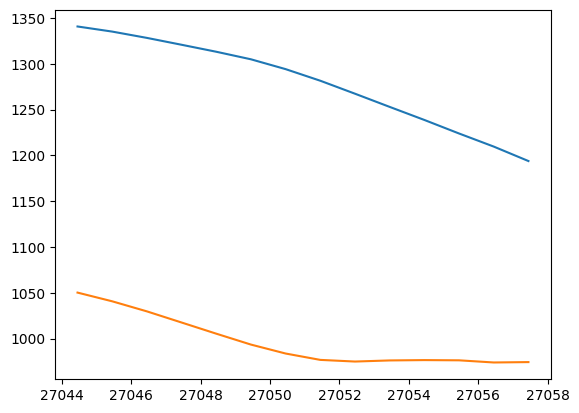
Data is generated by a simulator so there will be no missing values. The pie chart below shows the distribution of normal vehicles and attackers.



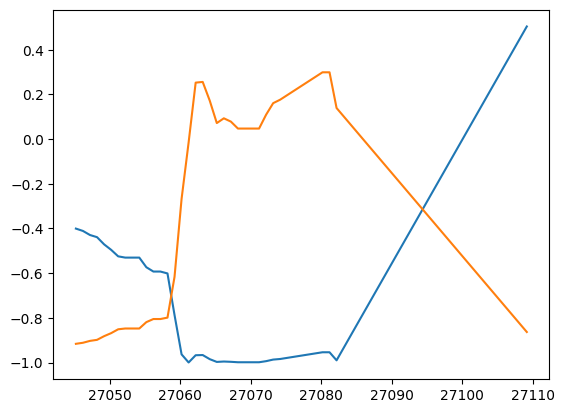
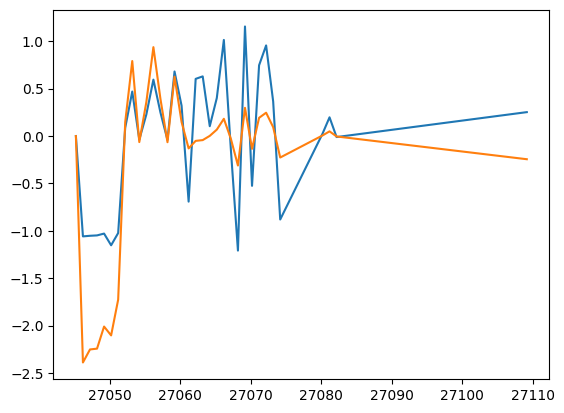
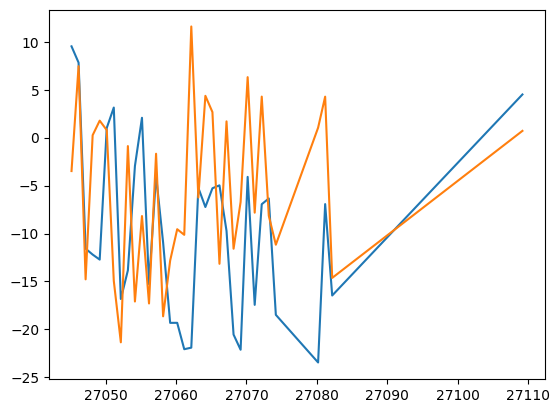
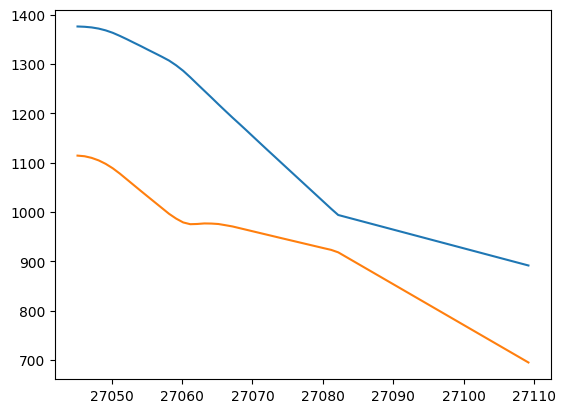
Preprocessing of the data included keeping the ‘outliers’ because all of them are messages from attackers and converting ‘1 to 15’ in the label column to ‘1’ because we only focus on whether a message is sent by attackers instead of which kind of attacker it is from. We removed features such as messageID and noise data for position, speed, acceleration, and heading since they don’t impact the label data.

The overall dataset size is 2205117 rows and 12 columns.

Attacking transmissions will display irregular and/or impossible changes to speed, position, heading, etc. (As shown in the figure below). Detecting these changes will allow the classification of the data as genuine or attacking.



*Sender Vehicle 905 (genuine) showing position, speed, acceleration, and heading over time*



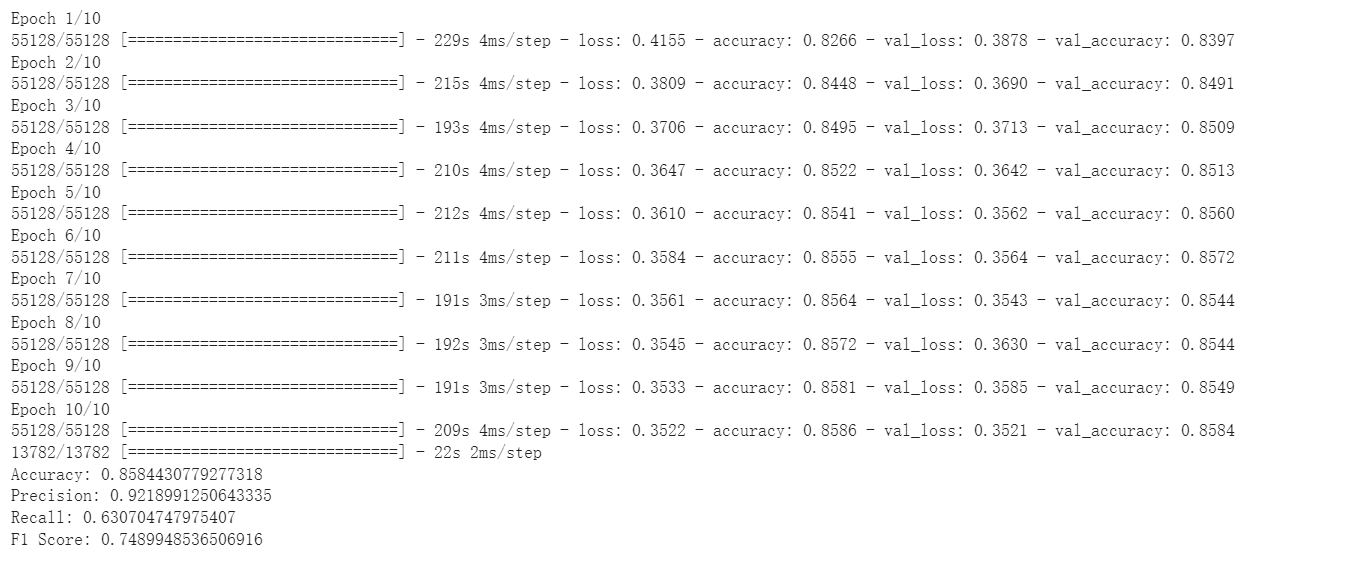
*Sender Vehicle 910 (attacker) showing position, speed, acceleration, and heading over time*

For all of the models, the data was split into 80% for training and 20% for testing.

## Results

### SVM Model

The SVM model was trained with 10 epochs and was evaluated using the F1 score. Sender and receiver information as well as the message ID was filtered out, thus this model was trained with 16 features.

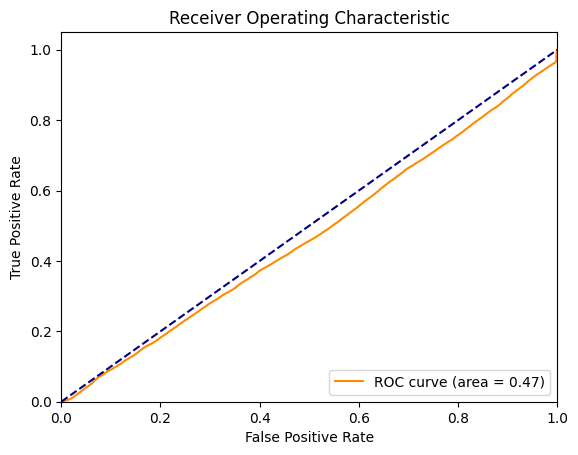


As we can see, the accuracy is over **85%.**

### One-Class SVM Model

One-class SVM only needs one training set consisting of positive objects (refer to genuine cars in this project). After training, we used the same test set as in the previous SVM. That test set contains both genuine cars and attackers.

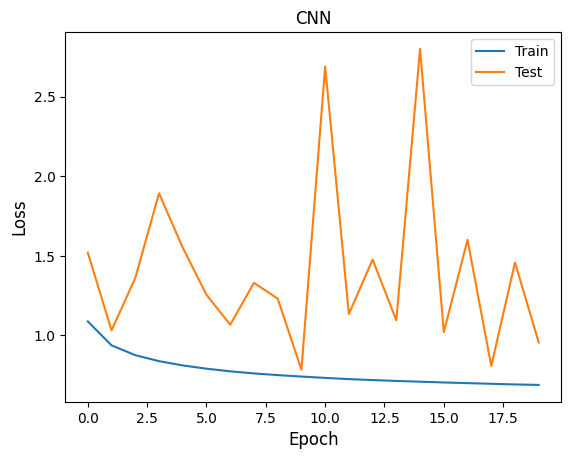
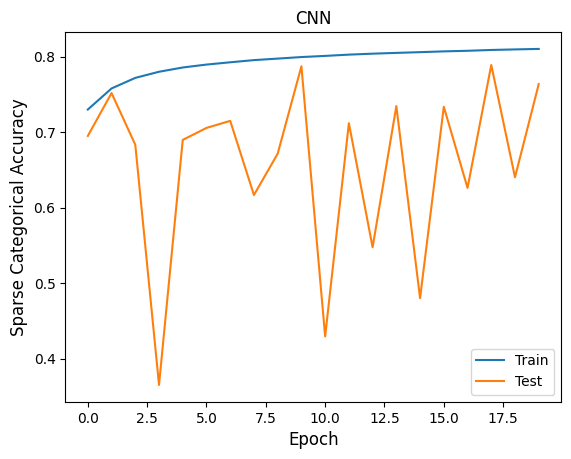
Since this model is an unsupervised algorithm, the ROC curve was used as its performance metric. The testing data consisting of both genuine cars and attackers was the same as the standard SVM.



The closer this line is to the upper left corner, the better the classification. From the graph line, we can see that our initial use of one-class SVM is not particularly satisfactory, and further adjustment of parameters or cross-use with other methods is needed.

### CNN Model

For the CNN, the “noise” data was filtered out of both training and testing sets as well as the messageID. The model was trained using 11 features. This model was trained with 20 epochs and a batch size of 128 due to time constraints. Performance was measured using sparse categorical accuracy and sparse categorical cross-entropy.



## Conclusion

As the vehicular communication network expands in terms of technology and resources, the need for accurate and more efficient misbehavior detection algorithms increases. Future approaches could include time series algorithms, anomaly detection algorithms, RNN, etc. Thus, these approaches were beginner steps for further research and new algorithms to be created for better anomaly detection.