# Introduction

Cooperative Intelligent Transport Systems are a new technology in development aimed at increasing road safety and reducing traffic accidents. The C-ITS is based on a peer-to-peer network that sends messages known as a Vehicular Ad hoc NETwork. While there are security procedures in place for VANET messages the security of such a system is currently not guaranteed and there is the risk that either malicious actors outside of the traffic network may attempt to introduce false messages into the system or that the vehicles themselves may be compromised. As a result, a misbehavior detection system is needed to ensure that individual cars do not blindly trust other vehicles within the network. A number of studies have attempted to solve this problem, however, to our knowledge, none has considered the possibility of using Hierachical Temporal Memory for the task.

Hierarchical Temporal Memory is a neuronally inspired algorithm designed to simulate the way pyramidal cells in the human neo-cortex process information. There have been several studies demonstrating promise for the use of HTM for anomaly detection, as well as several commercial products. Since misbehavior detection and anomaly detection have many similarities, we believe that applying HTM to anomaly detection can play a role in helping to secure proposed C-ITS’ when they are implemented in the future.

In this paper, we will provide an explanation of HTM as it relates to misbehavior detection systems and the various differences between more traditional methods for anomaly detection. We will then compare the results we obtained from a small simulation involving slot cars in a simulated platooning environment and discuss the strengths and weaknesses of the approach.

# Hierarchical Temporal Memory

Hierarchical temporal memory is an attempt to simulate the way in which the human neo-cortex processes information in a way that is both biologically plausible and also easily implemented in a computer. While it is an evolving theory there is also an implementation produced by Numenta known as Nupic.

### Sparse Distributed Representations

Unlike most deep learning architectures, Nupic consumes data represented as Sparse Distributed Representations which loosely correspond to neural spikes in the human brain

### Sequence Memory

### Spatial Pooling

### Temporal Pooling

### Anomaly Detection with HTM

# LSTM Classifier

The LSTM is trained in a supervised manner by feeding a mixture of anomalous and non-anomalous data in chronological order into the model and evaluating performance via Binary Cross-Entropy.

# LSTM Autoencoder

### Training

Performing anomaly detection with an LSTM autoencoder proceeds in an unsupervised manner. Clean data(with no anomalies) is fed in to the LSTM autoencoder in chronological order. The input features are reduced to a smaller feature set and then reconstructed into a representation of the original feature set. By comparing the the original with the reconstructed feature set we can compute a reconstruction loss which the model attempts to reduce through training via minimizing a Mean Squared Error loss function.

### Baselining

After training is complete the training dataset is again fed into the final trained model and the mean squared error over the entire training set is calculated. The baseline for identifying an anomaly is some factor of the final MSE over the entire training set, which can be optimized to improve recall over precision or vice versa. In practice, the mean squared error itself was simply used as a threshold.

### Anomaly Detection with LSTM Autoencoder

We then feed anomalous and non-anomalous test data into the LSTM Autoencoder. Given that the original model was only trained on clean data we expect that the reconstruction loss for anomalous data should be higher while non-anomalous data will produce similar MSE to that seen in the training data. By using our previous baseline, we can then classify a message as either anomalous or not.

# Experimental Setup

Ideally, in order to test various algorithms for misbehavior detection having access to a pre-existing C-ITS system and a test track, along with vehicles, would be ideal. However, given that testing such a system in the real world would result in many wrecked cars, most studies have attempted to simulate such attacks on traffic networks through the use of software such as VEINs. The VeRemi dataset is a recently proposed standardization, which involves simulating message passing by observing real traffic scenarios and then simulating various attacks on the network to see how the network responds. This provides researchers with an objective ground-truth given the behavior of the real vehicles, as well as a simulation of what such a network would look like when deployed. However, another possibility which we have utilized is developing a small physical simulation where crashes do not result in a catastrophic loss. A good platform for such a simulation is the popular slot car tracks produced by Carrera.

A Carrera track was modified to be controllable by a Raspberry Pi computer. The Pi is able to receive commands from a more powerful computer in the network and sends messages via directly modulating the voltage levels of the track itself. The cars interpret these voltage changes and modify their speed accordingly.

In order to observe the position, speed and acceleration of the cars, an overhead camera registers the positions of the cars on the track via Aruco markers affixed to the top of each vehicle. It then uses Cooperative Adaptive Cruise Control to modify the speeds of the various cars and maintain a single moving entity known as a platoon. The cars each modulate their speed in order to maintain a fixed distance from the car in front while a lead car sets the pace for the remaining cars in the platoon.

In order to create a misbehavior we manipulated the messages being sent to the lead car. This is known as a “data injection” attack. The data injection causes the lead car to behave erratically and the other cars must react in order to not collide with the lead car. By monitoring the position, speed, and acceleration of the platoon system, we should be able to detect timesteps when an anomaly occurred.

In some ways, this form of intervention is superior to the artificial simulations conducted using Veins. For one, in a Veins simulation, while the messages sent between cars can be falsified, individual cars in the simulation do not actually react to the attacks. In the Carrera simulation, the other cars actually see an anomaly and attempt to avoid collision, much like how a real driver would attempt to avoid collison

In order to produce the misbehavior dataset, we ran 7 trials. Each trial goes for a duration of 120 seconds during which a data injection attack occurs at a randomly determined interval. This interval is considered the ground truth for our dataset and the goal of our anomaly detection system is to classify each time step for each car as either anomalous or not based on a car’s reported speed, position, acceleration, and predecessor distance. We also ran a 30 minute long eighth trial with no anomalies to provide the LSTM Autoencoder with a training dataset.

## Veremi Extension

VeReMi is both a publicly available dataset and framework for testing misbehavior detection systems. It consists of a traffic simulation using real GPS data obtained from observing traffic around Luxembourg. From this, a C-ITS network is then simulated with message logs for each car recorded and a ground truth file stored for each attack type.

We obtained initially promising results in the Carrera simulation but, given the limited dataset, we also wanted to get a more direct comparison with other attempts at misbehavior detection in the current state of the art. Unfortunately, some new information came to light about how the experiments are conducted that makes a direct comparison more challenging:

1. Most misbehavior detection systems utilize a much more advanced pre-processing pipeline that allows for categorical classification instead of regression based classification. In general, a set of “plausibility scores” are calculated on the input data and that data is then fed into a classifier to determine an anomalous message.
2. Most of these classification models will also build evidence against a certain car until a certain “trust-threshold” is reached at which point the car itself is classified as anomalous, this determination is made by a separate Misbehavior Authority.
3. The exact transportation infrastructure is important for determining which misbehavior detection algorithms would be able to be implemented, so not only does an effective algorithm need to be proposed but also a hypothetical model of the existing transportation infrastructure.
4. The number of samples that each car receives is limited, in general, only about 100 per car, on the other hand there are thousands of cars in the simulation and the actual volume of data is much higher.
5. There are dozens of potential attacks, no algorithm so far developed manages outstanding performance on all of them.

Given these problems, we needed to make some assumptions about how to proceed further.

We noticed that many state of the art approaches performed well on certain anomaly detection tasks and poorly on others so we decided to opt for one task that could be considered relatively “easy” and another “difficult”.

For the easy attack we went with ConstPos. This causes the attacker cars to transmit the same location regardless of where they are currently located in the simulation.

The difficult attack is called a DataReplaySybil. This causes the cars to replay the same messages they have already received from other cars. The messages that they send can come from any of the other cars that they have received messages from.

In terms of anomaly detection, there is also the discovery that there are 3 different levels currently that all could potentially be considered as anomaly detection. The first being plausibility scores, the second flagging a message as anomalous, and the third making the determination about whether the sender of a message is actually misbehaving. In order to not change the scope of the experiment too much from how it was initially implemented we decided to simply see how using the raw C-ITS messages compared with existing classifiers for determining if the message itself was anomalous. This is technically a hybrid of levels 1 and 2 in the above classification but otherwise would have required an implementation of different plausibility scores for comparison purposes, which are an evolving field.

It should be considered however that using a different set of plausibility scores as input features it might be possible to create a more accurate classifier, or as an alternative, incorporate this model as yet another plausibility score.