Misbehavior detection in Vehicular Ad-Hoc Networks using HTM

Report by : Tushar Singhal

1. Introduction

VANETs or Vehicular Ad hoc NETwork is a major enabling technology for connected and autonomous vehicles. Vehicles communicate wirelessly with each other and sensors, humans, and infrastructure, thereby improving decision making based on the information received from their surroundings. However, for these applications to work correctly, data needs to be authenticated, verified and trustworthy. Safety messages are the most important messages in these networks, which are periodically broadcasted for various safety and traffic efficiency-related applications such as collision avoidance, intersection warning, and traffic jam detection. However, the primary concern is guaranteeing the trustworthiness of the data in the presence of dishonest and misbehaving peers. Misbehaviour detection is still in its infancy and requires a lot of effort to be integrated into the system. An attacker imitating “ghost vehicles” on the road by broadcasting false position information in the safety messages must be detected and revoked permanently from the VANETs.

In this project, we work in 2 phases. First, we compare the state of the art model HTM and LSTM autoencoder on the locally generated Carrera track dataset. Since HTM has the property of continuous unsupervised learning, it makes this a very feasible and effective way of detecting misbehaviour in a real-world scenario of VANETs. In the second phase, we run both the models on an extension of the veremi dataset and evaluate the Constant position data Sybil attack performance.

2. Related work

- **VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs[1]**
  The Veremi extension dataset paper is a unique public misbehaviour detection dataset that has been used in numerous studies. Its easy

reproduction and comparability with different results set it apart. Moreover, it provides the benchmark detection metrics by utilising a set of local detectors and a simple misbehaviour detection mechanism.

- **Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms[2]**
  This paper uses multiple machine learning models for misbehaviour detection and compares their performances on an F2MD. VEINS is a well known open-source framework for running vehicular network simulations. It is also the simulator of choice for VeReMi. They used a variety of metrics in which LSTM comes on top.
- **Machine Learning Based Approach to Detect Position Falsification Attack in VANETs[3]**
  This paper uses Logistic regression and SVM directly on the Vermi dataset. Though there is no description of how the results are calculated and compared, they get comparable performance to the old Veremi dataset paper by just using position and speed as features.
- **Simulation Framework for Misbehavior Detection in Vehicular Networks[4]**
  The F2MD framework introduced in this paper provides a complete solution for real time simulation and evaluation of a MisBehavior Detection (MBD) system. It extends VEINS with a large panel of MBD, evaluation and other general Cooperative Intelligent Transport Systems (C-ITS) modules.

3. Experimental design

3.1 Data acquisition and Preprocessing

A Carrera track is a platform for slot car physical simulation where the cars in a lane are accelerated by adjusting the voltage given to that car. Generally, there are controllers with the car for controlling the voltage. But for the experimental setup, these voltages were regulated by a Raspberry Pi computer connected to the main computer.

The cars on the track are in a platoon formation meaning three cars in a row with a fixed distance from the lead car. The formation is done by mounting a camera on top of the Carrera track, which uses Cooperative Adaptive Cruise Control and sends the commands through Raspberry Pi to maintain the platoon formation. The camera is also used to record our position, speed, acceleration and header data fields.

To create anomalies, we send false messages to the lead car, like maintaining a constant speed, also called “Constant Speed”, to attack through data injection. These false messages causes the lead car to behave erratically, and the other vehicles must react to not collide with the lead car. Since we are using a high frame rate camera, we get a high ping of 40 timesteps in a second which is great for early detection of the anomalies on the recorded data.

Since HTM is an unsupervised continuous real-time anomaly detection algorithm, it only needs the first few hundred points to learn the dataset patterns and give an anomaly score for each timestep. We used a trial of 2 minutes roughly 5000 data points containing an anomaly time window in between.

The LSTM autoencoder needs to be trained on clean non-anomalous data, which was gathered over 30 minutes (72000 data points) and for testing, we used similar 2-minute trails as HTM.

VeReMi dataset came at the end of 2018, and the main idea behind the paper was to serve as a reference dataset that can be used for misbehaviour frameworks. An extension of this dataset came out in the mid of 2020, which included more attacks and experiments with varying vehicle densities. Each attack consists of log files from each car, which contains the messages they received, and a ground truth file containing the true messages. A single message entry contains sender ids, position, speed, acceleration and header distance fields. To evaluate the state of the art results against our models, we used only the best and worst attacks data.

- ConstPos. This causes the attacker cars to transmit the same location regardless of where they are currently located in the simulation.

- DataReplaySybil. This causes the cars to replay the same messages they have already received from other vehicles. The messages that they send can come from any of the other cars that they have received messages from.

It was challenging to Preprocess the veremi dataset. According to the Veremi dataset papers, they calculate the plausibility scores used as features for a classifier. These plausibility scores are generated by running simulations like F2MD with the Veremi dataset. F2MD is a gigantic simulation framework that includes machine learning algorithms for misbehaviour detection. F2MD provided a good baseline, but we were out of our hardware capacity to run this framework.

So after some research, we found some papers that ran local misbehaviour detection algorithms on the Veremi dataset, which served as a baseline for the article. However, these papers did not mention a clear methodology on how the raw data was used. This information was crucial as we had only about 600 continuous points per car, which was insufficient to construct a time series.

Therefore, we decided to create a cross-car classifier dataset. Here, we remove the sender id and timestamp and only use the position, acceleration, speed and header distance. Using this kind of model played against the strengths of our base models since they are built to process time-series data.

3.2 Architectures

3.2.1 HTM

HTM is a technology to detect anomalies in streaming data. However, it is also applicable for prediction, classification and sensorimotor applications. It is based on neuroscience and mimics the physiology and interactions of neurons present in the human brain's neocortex.

HTM can store, learn, infer and recall high order sequences with the help of learning algorithms. Moreover, HTM can continuously learn the time-based patterns without supervision in unlabeled data. Research shows that it is robust to noise and has a higher capacity as compared to its counterparts.

Figure 1 - Process for Modeling Movements and Identifying Anomalies

Sparse Distributed Representation

SDR is simply an array of 0's and 1's. If you take a snapshot of neurons in the brain, it is highly likely that you will only see less than 2% neurons in an active state. SDR is a mathematical representation of these sparse signals which will likely have less than 2% ones.

Encoder

In order to apply the HTM algorithm, we need to preprocess categorical and numerical data in order to create SDR's to feed into the HTM algorithm.

- RDSE encoder - We use RDSE (Random Distributed Scaler Encoder) for encoding a numeric (floating point) value into SDRs. This encoder maps a scalar value into a random distributed representation that is suitable as scalar input into the spatial pooler. It preserves the important properties around overlapping representations.

- Grid encoder - A Grid Encoder takes in 2 parameters (x,y location) and converts this into a SDR. A grid cell activates when the agent is close to a certain location displaced out into a lattice structure.

Spatial Pooling
Spatial pooling is the process of converting the encoded SDR into a sparse array complying with two basic principles:

- Make sure the sparsity of the output array is constant at all times, even if the input array is a very sparse SDR or a not so sparse SDR
- Make sure the overlap or semantic nature of the input is maintained

So the overlap of both input and output SDR of two similar objects need to be high.

Temporal Memory
Spatial pooling maintains the context of the input sequence by a method called temporal memory. The concept of temporal memory is based on the fact that each neuron not only gets information from lower level neurons, but also gets contextual information from neurons at the same level. In the spatial pooling, we get an spare array where each column is represented by a single number. However, each column in the output column is composed of multiple cells that can individually be in active, inactive, or predictive state.

3.2.2 LSTMAutoencoder

We use an LSTM autoencoder for the task of anomaly detection in an unsupervised manner. Here, the LSTM model is trained with non-anomalous data. This enables the model to learn a low dimensional representation of the input data because of the autoencoder architecture of the LSTM. Additionally, it uses the MSE to minimize the loss between the original and the reconstructed signal.

To get a baseline for the reconstruction loss, we test it again with the training data, and mse is calculated over the entire training set. In the testing phase, we provide anomalous data. It is expected that reconstruction loss should be high for the anomalous time steps as this data contains both anomalous and non-anomalous time steps.

3.3 Metrics

The metrics used for evaluating our models are Accuracy, Precision, Recall and F1. We emphasize the F1 metrics while interpreting our results because of two reasons. First, due to a slight imbalance in data, it was essential to assess the harmonic mean of the precision and recall metrics. The F1-score indicates these values. Second, the metrics used by other base papers in the results are also F1, which serves as a baseline for our comparison.

3.4 Thresholding and Window Filtering

Since the output of both the models were float value (0.0 - 1.0), we needed a way to compare the ground truth, which was 1 for anomalous timestep and 0 otherwise. We came up with techniques.

1. Thresholding - It can be considered a hyperparameter value above which the anomaly scores were made to 1 and 0 if lower.

2. Window Filtering - Since the anomalies only occurred in time windows, we needed a way to suppress all the random anomaly scores. To do this, we took two parameters: window size and window threshold. If in the last window size time steps the number of high anomaly scores are greater than the window threshold, then all the time steps in the last window size were made to 1 otherwise 0. Using this technique, we were able to significantly boost our metrics because it was reducing random high anomaly scores and increasing anomaly scores in the anomaly window region.

4. Results

4.1 Carrera Track

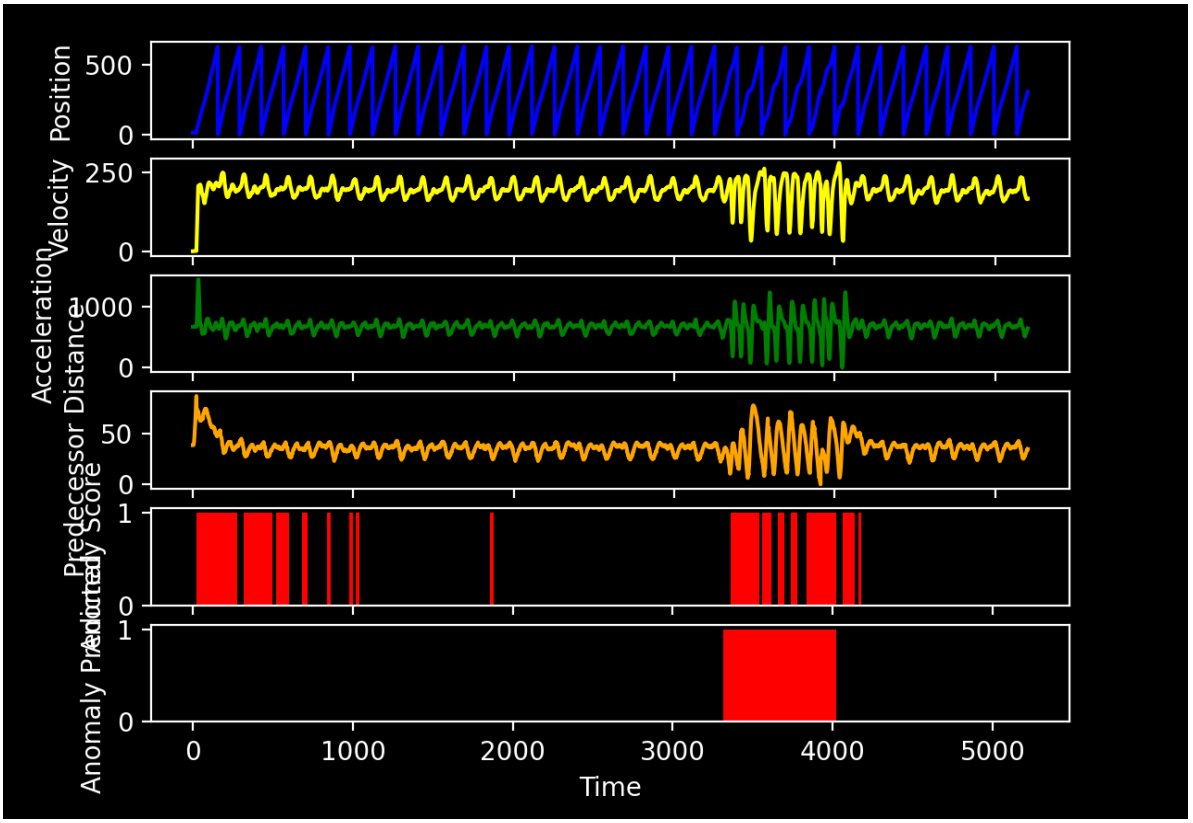

Figure 2. Results with HTM with out Swarming on Carrera Track

	Precision	Recall	F1-Score
LSTM	0.6202	0.7442	0.7442
LSTM_AutoEncoder	0.6844	0.7467	0.7142
HTM-Without Swarming	0.7812	0.7123	0.7452
HTM-With Swarming	**0.8095**	**0.9687**	**0.8820**

Figure 3. All the results with different models on carrera track

Figure 2 shows the anomaly score (F1 score of 74.65) we got before swarming. These results showed improvement by running a swarm over an HTM. Swarming is a method of hyperparameter search in which it evaluates many models and chooses the best one according to an error metric (our F1 score). Also, the first 1000 steps are not considered for metrics calculation in HTM because we regard these time steps as a learning phase.

We can also see the LSTM autoencoder results in Figure 3 which were far behind the HTM results after swarming. We used a threshold=0.1 and window size, window threshold=20,10 to get the best results.

4.2 Veremi Dataset

Const Pos

	Precision	Recall	F1-Score
LSTM_Auto	**0.4206**	0.4496	**0.4346**
LSTM-Auto Tile	0.3039	**0.4522**	0.3635
HTM	0.3044	0.2029	0.2435
HTM with Grid Cell Encoder	0.2737	0.3881	0.3210

DataReplaySybil

	Precision	Recall	F1-Score
LSTM_Auto	0.2853	0.4137	0.3381
LSTM-Auto Tile	0.2856	0.4444	0.3478
HTM	**0.3860**	0.4723	**0.4248**
HTM with Grid Cell Encoder	0.3110	**0.6372**	0.4180

Figure 4. All the results with different models on Veremi Dataset

With HTM, we experiment with Grid Encoders and to ensure similarity in LSTM AutoEncoder. We come up with a Tiling Strategy. We divide the whole map into grid cells, and we pass the cell column and row number along with the position, acceleration, speed and header distance. These are determined using the position field. The aim was to mark a message as anomalous if it lies in a different cell than it should. It is similar to the HTM grid encoder, where the position coordinates are converted to cell locations in multiple grids of varying cell sizes all at once.

As we can see in the cont pos attack, HTM does not work well compared to the benchmark scores of 1 precision and 1 recall because if the const pos appears, again and again, it learns the positions and stops marking it as anomalous. But the LSTM autoencoder learns this reconstruction of the time sequence; it is still better than HTM but far from the baseline.

The major drawback of continuous learning algorithms is that if the anomalies are persistent and follow even a less perfect pattern, we cannot stop a continuous learning algorithm from learning these anomalies. Since we cannot specify training and test sets, the algorithm shall learn these anomalies simply as patterns in time.

A workaround could be to use a labelled dataset in which we add a separate field anomalous messages (1 if anonymous 0 otherwise). This separate field is initiated for the first few time steps. Later, we use 0 even for the anomalous fields, in which the HTM is expected to give high anomalous scores. There is scope for future research to use this ideology while comparing HTM to supervised deep learning methods.

We perceive that our models function well for the DataReplaySybil attack considering the baseline was an F1 score of 50 from the veremi extension paper. These models were not hyper tuned yet. Interestingly, we can see that there is some tradeoff between precision and recall when using HTM's with and without grid cells. The reason might be that Grid Encoders take away the minute differences in the position coordinates.

5. Conclusion

HTM has some significant advantages compared to some deep learning methods. It gives state of the art results when it comes to continuous unsupervised learning. Even though it has been around for a long time, it is not very popular. In this experiment, we explored how this algorithm can work better than the state of the art deep learning models. In future work, we would try an HTM algorithm with the F2MD framework to see how it compares to other deep learning methods when run in real-time.

References

1. J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien and F. Kargl, "VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149132.
2. J. Kamel, I. B. Jemaa, A. Kaiser, L. Cantat and P. Urien, "Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms," 2019 IEEE Vehicular Networking Conference (VNC), 2019, pp. 1-8, doi: 10.1109/VNC48660.2019.9062831.
3. Singh P.K., Gupta S., Vashistha R., Nandi S.K., Nandi S. (2019) Machine Learning Based Approach to Detect Position Falsification Attack in VANETs. In: Nandi S., Jinwala D., Singh V., Laxmi V., Gaur M., Faruki P. (eds) Security and Privacy. ISEA-ISAP 2019. Communications in Computer and Information Science, vol 939. Springer, Singapore. https://doi.org/10.1007/978-981-13-7561-3_13
4. Joseph Kamel, Mohammad Raashid Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, et al.. Simulation Framework for Misbehavior Detection in Vehicular Networks. IEEE Transactions on Vehicular Technology, Institute of Electrical and Electronics Engineers, In press, IEEE Transactions on Vehicular Technology, 69 (6), pp.6631-6643. 10.1109/TVT.2020.2984878 . hal-02527873
5. Ahmad, S., & Hawkins, J. (2015). Properties of sparse distributed representations and their application to hierarchical temporal memory. arXiv preprint arXiv:1503.07469.
6. N. O. El-Ganainy, I. Balasingham, P. S. Halvorsen and L. Arne Rosseland, "On the Performance of Hierarchical Temporal Memory Predictions of Medical Streams in Real Time," 2019 13th International Symposium on Medical Information and Communication Technology (ISMICT), 2019, pp. 1-6, doi: 10.1109/ISMICT.2019.8743902.