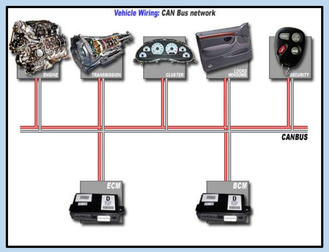
Cybersecurity for Vehicles: Network Anomaly Detection

Why This Project is Important

Unlike many Senior Design projects, the main result of the project is not physical. Rather, it is a machine learning model. Within the depths of Machine Learning, we implemented a Neural Network/Deep Learning algorithm that can accurately detect unusual behavior in the normal flow of information. The normal flow of information is communication between embedded system sensors in vehicles through their CAN bus. When the flow of information is disrupted, there could be unusual information being detected, a failure in the system, or a direct cyber-attack on the system from the outside. We implemented an algorithm that achieves results contributing towards reliably detecting attacks by finding this unusual behavior. We used Keras with TensorFlow in the Python programming language to implement this. After researching available options for the algorithm, designing an algorithm, and then training and testing against data, we were then able to see if it worked according to a set of standards as a part of the design process. We also researched related aspects of the project, for example, how it would be implemented practically.

Vehicle CAN bus

Fig. 1



CAN (Controller Area Network)
Bus: Connects ECUs (Electronic Control Units) together
ECU (general: Electronic Control Unit): Monitor sensors and controls some behaviors in the vehicle

Fig. 2

References

Fig 1. "CAN & CAN FD - serial protocol decoding - PicoScope from A to Z." https://www.picotech.com/library/oscilloscopes/can-bus-serial-protocol-decoding (accessed Apr. 19, 2021).
Fig. 2. "What is CAN Bus?" https://canbuskit.fortin.ca/what.php (accessed Apr. 19, 2021).
Fig. 3. Weber, M., et. al. "Online Detection of Anomalies in Vehicle Signals using Replicator Neural Networks", Escar USA. 2018.
Fig. 4. "Machine Learning for Marketing - IE Exponential Learning Blog." https://www.ie.edu/exponential-learning/blog/data-science/machine-learning-marketing/ (accessed Apr. 19, 2021).
Fig. 5. "Cybersecurity Tips While Working Remotely." https://today.duke.edu/2020/04/cybersecurity-tips-while-working-remotely (accessed Apr. 19, 2021).
Fig. 6. "How CAN Bus/CAN FD Enables In-Vehicle Networking." https://www.einfochips.com/blog/can-bus-and-can-fd-for-automotive/ (accessed Apr. 19, 2021).

Left: Andy Worcester: Computer Engineering
Right: David Rohrbaugh: Computer Engineering

Cybersecurity

Attack types:
1. Sine
2. Plateau Stuck
3. Peak
4. Negative Peak
5. Noise
6. Plateau rise/fall
7. Zero fall

Attacks:
1. DoS Attack
2. Fuzzy Attack
3. Spoofing Attack

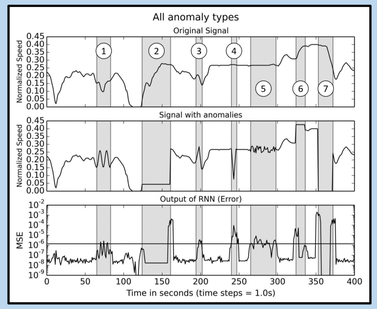


Fig. 3

Machine Learning

Machine Learning "brains" are centers built by code called models. They can take weeks to train and often many hours of iteration to get the performance sought.

For anomaly detection that we are doing, models learn "normal" information and then look for attacks that simply aren't normal information.

I am using the RNN (Recurrent Neural Network) and the CNN (Convolutional Neural Network) model types for this anomaly detection.

Challenges

1. Spending enough time altering hyperparameters to tune the models for better performance.
2. Understandably, lessened supervisor support after summer 2020.
3. Unexpected partner vacating project.

Acknowledgements

To Professor Pasricha, that he didn't dismiss the project despite setbacks.
To Josh Datko for his continued support and encouragement, giving me motivation.
To Ms. June Richardson, for her financial support to the Honors portion of this project.

Main Project Experiment

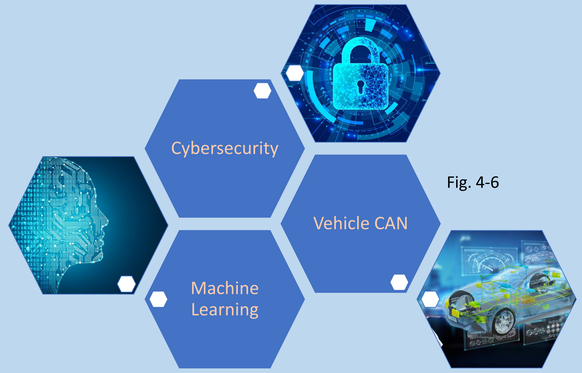


Fig. 4-6

Dataset:
For training my machine learning model, I used well-formatted synthetic vehicle CAN bus data.

I used two different types of models:
RNNs models have output that partially feed back into themselves for when their training (i.e. learning). This allows them to have some memory of the data to detect patterns. They'll be able to predict information in that pattern. When there is a cyberattack, it can look like an anomaly. This anomaly is a disruption in the patter the RNN model is expecting, so it can detect it.
CNN models can be good for prediction as well, but not for patterns, and they'd be forced to potentially lose some of the pattern recognition. Why? Because CNN networks need an input to be mapped to an output. This means some of the data could be given as an input while another portion of the CAN data as an output. This can sometimes achieve higher accuracy results, but it would have no chance at detecting anomalies in the input data.

Example Result:
The red shows root-mean squared error. Note that near the top the red error curve changes to have sharp ups and downs towards the last third of the figure. This is because of an anomaly present in exactly that portion of the dataset:

Fig. 7