Georgia Institute of Technology

CS 4220/6235: RTES

Fall 2025

Project Checkpoint Report

Group: 19
Name(s): Phineas Giegengack
Project: Use of Machine Learning to Detect Attacks on CAN (Controller Area Network) in Embedded Microcontroller

Project Plan (Scope):

Introduction

The Controller Area Network (CAN) protocol is a specification which describes how many electronic control units (ECUs) broadcast data over a shared bus. CAN was derived in the 1980s for use in automobiles to reduce the complexity of the electrical design of vehicles with an ever-growing array of electronic sensors and actuators. CAN was an answer to difficulties that arose from designing point-to-point connections between every node that needed to communicate in a car, but CAN does not by default have authentication, which leaves it open to a variety of attacks, including Denial of Service (DoS), Fuzzing, and Spoofing.

In modern vehicles, and especially in autonomous vehicles, safety-critical systems may be connected via CAN, so the ability to detect intrusions onto this network is of utmost importance.

Although there are many implementations of machine learning to detect intrusions on the CAN bus as will be discussed in the next section, and several of these make attempts to deploy models on constrained embedded hardware, my project investigates the potential for training and running an autoencoder neural network on an embedded system. The aim is to stream CAN bus data through a feature extractor and use it to continually train the autoencoder. This could be desirable since CAN bus behavior is vehicle, driver, and situation dependent, and continually updating the anomaly-detection algorithm may prove instrumental in reducing false positives.

Related Work

A variety of methods to detect attacks/intrusions on the CAN bus using machine learning have been proposed.

In [1], a traditional neural net and multi-layer perceptron are compared for their performance in detecting CAN bus intrusions. They are trained and tested on a static dataset of CAN bus message traces. No effort is made to optimize the models for implementation on an embedded device, but the models achieve respectable performance.

Researchers in [2] made use of a GRU model to achieve high performance with a lightweight model intended to be deployed on embedded hardware. They also propose a system architecture for deploying, updating and training the model over-the-air (OTA) which would be desired for real-world applications to keep model performance up to date as CAN bus attacks evolve.

[3] and [4] make use of support vector machines (SVMs) to do one-class classification which enables models to be trained only on ‘good’ CAN bus data and have the models infer when anomalous behavior is taking place without needing training data for those scenarios. This is appealing since attack methods are likely unknown at the time of training, and having one-class models would allow detection of novel attacks before they impact any vehicle.

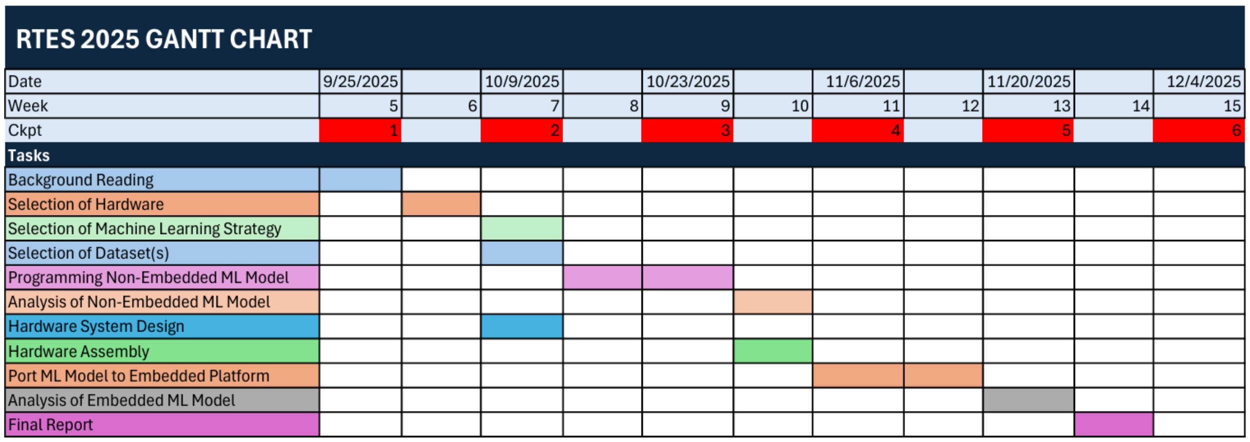
[5] is the most recent paper I was able to find and proposes another model type, TPE-LightGBM algorithm. They achieve good performance and the model is lightweight, although they do not deploy it on embedded hardware.

Methods

For this project, I will propose and design a machine learning pipeline for training and running an autoencoder for detecting anomalies in CAN communication that can be deployed on a low-cost microcontroller. I will develop an architecture for training the model on static CAN bus traces and analyze a ‘control model’ that will run on my PC instead of an embedded processor. When the model is accurate relative to other models that have been developed in other papers, I will port the model to the embedded computing platform, experimenting with reducing resolution of weights, size of feature vector, and depth of neural network to make the model efficient to run in a real-time scenario while attempting to preserve performance where possible.

I hope to show a reliable method for training an ML model on CAN bus traces and deploying the trained model on an embedded computing platform to detect anomalous behavior on the CAN bus. I hope to compare model performance, model size, and speed of various iterations of the chosen machine learning technique to better understand trade-offs in deploying machine learning for real-time monitoring of the CAN bus.

Schedule



RTES 2025 GANTT CHART

Date	9/25/2025		10/9/2025		10/23/2025		11/6/2025		11/20/2025		12/4/2025
Week	5	6	7	8	9	10	11	12	13	14	15
Ckpt	1		2		3		4		5		6
Tasks											
Background Reading	■										
Selection of Hardware		■									
Selection of Machine Learning Strategy			■								
Selection of Dataset(s)			■								
Programming Non-Embedded ML Model				■	■						
Analysis of Non-Embedded ML Model						■					
Hardware System Design			■								
Hardware Assembly						■					
Port ML Model to Embedded Platform							■	■			
Analysis of Embedded ML Model									■		
Final Report										■	

Current progress report (Match):

Checkpoint 2

In the last 2 weeks I have expanded on the reading I had already done for the first checkpoint. I read an additional 8 papers to help me narrow down the scope of my project and make decisions about the machine learning strategy, hardware, and dataset I will use for this project.

Machine Learning: Autoencoder for One-Class Classification (OCC)

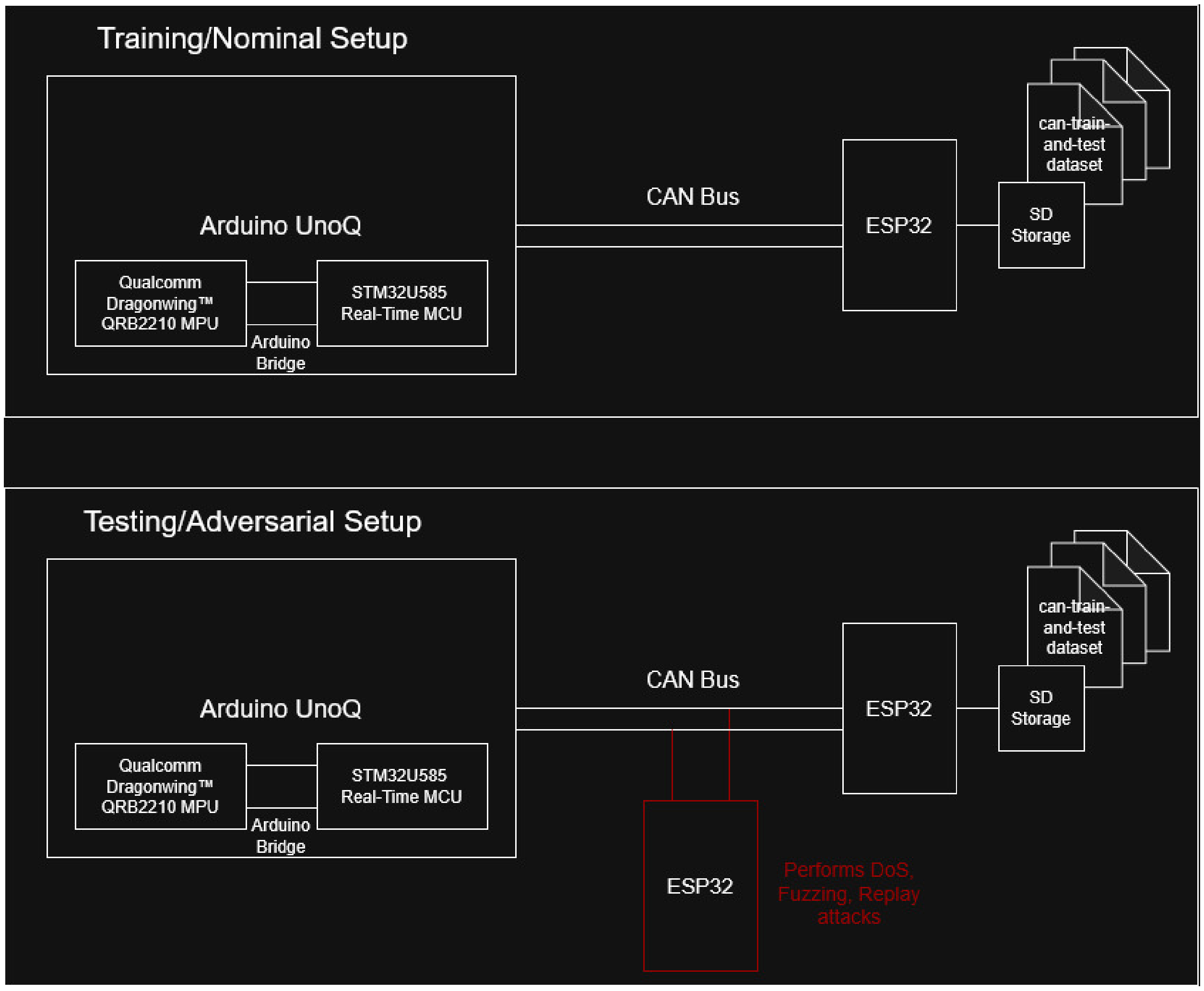
A one-class classifier seeks to distinguish between datapoints in a known and unknown group, also called a 'target' and 'other' group. This is desirable for detecting intrusions on the CAN bus since a model can be trained only on attack-free CAN bus traffic and anomalies can be detected when an intrusion occurs.

[7], [11] The autoencoder works by reconstructing the training data, minimizing the error between the data and the reconstructed data. For OCC, an autoencoder is trained to reconstruct nominal data and attempts to reconstruct unseen data. Low error for reconstructing new data implies it is similar to the training data, and thus belongs to the known or target group, while a high reconstruction error means the new data is anomalous.

For detecting intrusions on the CAN bus, I will develop a feature extraction from raw CAN bus traces that will serve as the input to an autoencoder. I will train the autoencoder on attack-free data and hope it will be able to distinguish anomalous behavior indicative of an intrusion.

Hardware Selection: Arduino Uno-Q and ESP32

I select the brand-new Arduino Uno-Q for its relatively low-cost (45$) for the AI/ML processing it advertises coupled with a real-time processor system. I will use ESP-32 dev boards to stream CAN data to the Arduino and may try several configurations of ESP-32s to simulate various adversarial situations that the model should be able to detect.


Training/Nominal Setup
Arduino UnoQ
Qualcomm Dragonwing™ QRB2210 MPU
STM32U585 Real-Time MCU
Arduino Bridge
CAN Bus
ESP32
SD Storage
can-train-and-test dataset
Testing/Adversarial Setup
Arduino UnoQ
Qualcomm Dragonwing™ QRB2210 MPU
STM32U585 Real-Time MCU
Arduino Bridge
CAN Bus
ESP32
SD Storage
can-train-and-test dataset
ESP32
Performs DoS, Fuzzing, Replay attacks

The STM microcontroller on the Arduino system will be used to read the CAN bus, extract features, and gather the data into batches which it will expose to the Dragonwing MPU through buffers. The MPU will have the sole responsibilities of training and/or running the model based on the data from the MCU.

It will be possible to stream data from the can-train-and-test dataset onto the bus from a single ESP32 through a SD card if extra memory is needed. CAN bus intrusions can be simulated either by streaming data from provided 'attack' datasets included in the can-train-and-test data, or by using a second ESP32 to simulate the attacks which would allow for trying even more scenarios than are included in the dataset.

Dataset: can-train-and-test
[10] I chose the can-train-and-test datasets provided by Lampe and Meng because of the abundance of attack-free CAN bus traces. These datasets are used in several of the other papers I have read, which gives me reasonable confidence that they will work well for this project.
https://bitbucket.org/brooke-lampe/can-dataset/src/master/
The datasets include data from several vehicles and attack scenarios.

Supporting Evidence (Factual): • https://github.com/finncg1234/RTES-2025

See the github link above for the project timeline (Gantt Chart) and annotated bibliography. The annotated bibliography shows citations for 13 papers I read and my thoughts on how I can use the results they had to inform my project.
I also started experimenting with an autoencoder architecture and a dataset on diabetes health indicators. My python script is available in the autoencoder_example/ subfolder, although it is not reproducible yet.

Skill Learning Report:

ML/AI: I have relatively little experience with machine learning, but as part of my research this checkpoint, I thoroughly investigated several machine learning architectures: Gaussian processes, Isolation Forests, Support Vector Machines, GRUs, and Autoencoders. I even created example programs for an SVM and an autoencoder in matlab and python.

Research: I have read 13 papers, doing my best to absorb the most relevant information to my project. I have honed in on machine learning models and their tradeoffs for real-time systems.

Mathematics: My investigations into ML architecture forced me to brush up on my linear algebra and multivariable calculus. I had to read and understand matrix equations and also had to learn about gradients and Lagrange Optimization.

Hardware System Design: I had to select hardware for this project and consider the clock speed and instruction time for the processors I will be using.

Technical Writing: This report is part of the work I did for this checkpoint.

Self-Evaluation:

Scope: 110%: I think the scope of this project is sufficient, maybe even ambitious. At this time I have well-defined my hardware and ML goals for the project. I don't know what the results of the project will be, but the scope may change if I discover limitations in the hardware and/or machine learning tools I have selected.

Match: 100%: In my original Gantt chart, I said I would select hardware, ML, and dataset by today. I have done that. I think I have done a good job clearing up many of the unknowns for this project.

Factual: 100%: I make no false claims in my match section, and by following the link to the github, I think you will agree that I did indeed read the papers I referenced and that I have a good understanding of how their findings fit into my project plan. The readme on the github includes much of the same information I have in this report.

References: (also available in annotated bibliography on github)

[1] F. Amato, L. Coppolino, F. Mercaldo, F. Moscato, R. Nardone, and A. Santone, "CAN-Bus Attack Detection With Deep Learning," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 8, pp. 5081–5090, Aug. 2021, doi: 10.1109/TITS.2020.3046974.

[2] H. Ma, J. Cao, B. Mi, D. Huang, Y. Liu, and S. Li, "A GRU-Based Lightweight System for CAN Intrusion Detection in Real Time," Security and Communication Networks, vol. 2022, Article ID 5827056, 11 pages, 2022, doi: 10.1155/2022/5827056.

[3] J. Guidry, F. Sohrab, R. Gottumukkala, S. Katragadda, and M. Gabbouj, "One-Class Classification for Intrusion Detection on Vehicular Networks," in Proc. 2023 IEEE Symposium Series on Computational Intelligence (SSCI), Mexico City, Mexico, 2023, pp. 1176–1182, doi: 10.1109/SSCI52147.2023.10371899.

[4] C. Chupong, N. Junhuathon, K. Kitwattana, T. Muankhaw, N. Ha-Upala, and M. Nawong, "Intrusion Detection in CAN Bus using the Entropy of Data and One-class Classification," in Proc. 2024 Int. Conf. on Power, Energy and Innovations (ICPEI), Nakhon Ratchasima, Thailand, 2024, pp. 157–160, doi: 10.1109/ICPEI61831.2024.10748816.

[5] L. Liang et al., "Intrusion Detection Model for In-vehicle CAN Bus Based on TPE-LightGBM Algorithm," in Proc. 2025 IEEE 34th Wireless and Optical Communications Conference (WOCC), Taipa, Macao, 2025, pp. 419–423, doi: 10.1109/WOCC63563.2025.11082193.

[6] J. N. Brewer and G. Dimitoglou, "Evaluation of Attack Vectors and Risks in Automobiles and Road Infrastructure," in Proc. 2019 Int. Conf. on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 84–89, doi: 10.1109/CSCI49370.2019.00021.

[7] Y. Liao and B. Yang, "To Generalize or Not to Generalize: Towards Autoencoders in One-Class Classification," in Proc. 2022 Int. Joint Conf. on Neural Networks (IJCNN), Padua, Italy, 2022, pp. 1–8, doi: 10.1109/IJCNN55064.2022.9892812.

[8] J. Lee, S. Park, S. Shin, H. Im, J. Lee, and S. Lee, "ASIC Design for Real-Time CAN-Bus Intrusion Detection and Prevention System Using Random Forest," IEEE Access, vol. 13, pp. 129856–129869, 2025, doi: 10.1109/ACCESS.2025.3585956.

[9] Z. Bi, G. Xu, G. Xu, M. Tian, R. Jiang, and S. Zhang, "Intrusion Detection Method for In-Vehicle CAN Bus Based on Message and Time Transfer Matrix," Security and Communication Networks, vol. 2022, Article ID 2554280, 19 pages, 2022, doi: 10.1155/2022/2554280.

[10] B. Lampe and W. Meng, "can-train-and-test: A New CAN Intrusion Detection Dataset," in Proc. 2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall), Hong Kong, 2023, pp. 1–7, doi: 10.1109/VTC2023-Fall60731.2023.10333756.

[11] B. M. Tóth and A. Bánáti, "Autoencoder Based CAN BUS IDS System Architecture and Performance Evaluation," in Proc. 2025 IEEE 19th Int. Symp. on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2025, pp. 000099–000104, doi: 10.1109/SACI66288.2025.11030168.

[12] M. Kemmler, E. Rodner, E.-S. Wacker, and J. Denzler, "One-class classification with Gaussian processes," Pattern Recognition, vol. 46, no. 12, pp. 3507–3518, 2013, doi: 10.1016/j.patcog.2013.06.005.

[13] F. Sohrab, J. Raitoharju, M. Gabbouj, and A. Iosifidis, "Subspace Support Vector Data Description," in Proc. 2018 24th Int. Conf. on Pattern Recognition (ICPR), Beijing, China, 2018, pp. 722–727, doi: 10.1109/ICPR.2018.8545819.