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.

Many researchers have turned to machine learning to detect intrusions on the CAN bus, but relatively few have addressed the inherent limitations of applying machine learning in real-time and on limited hardware. For this project, I will aim to develop a machine learning pipeline using the Autoencoder architecture to train a model in near-real-time on a low-cost embedded microcontroller.

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.

Three papers are especially of interest for this project. [6] Outlines the use of autoencoders as One Class Classifiers (OCC), and the addition of a Batch-Wise Feature Weighting as a means of combating the autoencoders inherent skill at generalizing which reduces its accuracy in OCC applications. [7] Attempts to use an autoencoder for CAN bus intrusion detection, but with low accuracy and without investigating the impact of adding a BFW. In [8], researchers describe the methods used to accumulate the can-train-and-test dataset, which will be used in this project to train and test the model.

This project goes beyond previous work in a few ways. First, the work in [6] to define a BFW for use in autoencoders has not been applied to CAN bus intrusion detection. Further, although in [7], an autoencoder is used for CAN bus intrusion, and the model is run on embedded hardware, it is not run in real time, nor is an attempt made to train the model on the edge.

Project Deliverables

In this project, I will create a machine learning pipeline which uses an autoencoder to detect anomalies or intrusions of the CAN bus. I will compare the effects of using a Batch-Wise Feature Weighting as described in [6] and other tuning of hyperparameters to improve model recall and precision.

For this project I will develop the ML pipeline on my laptop, without needing to optimize for an embedded platform, before porting my architecture to the Arduino Uno Q, where I will replay the CAN message traces, confirming my model's performance in real-time detection. If time permits, I may also investigate training on the edge.

I hope to compare the performance of the ML model for this task with several feature extraction strategies, hyperparameter selections. I believe by using an autoencoder for one-class classification, effects of skewed training data will be minimal since the purpose of one-class classification is to produce a classifier with limited or non-existent data in the 'other' or 'anomalous' class.

Project Gantt Chart

Current progress report (Match):

I had 3 goals for this checkpoint, as per my Gantt Chart:

- Complete Programming of 'Non-Embedded' ML Model
- Analysis of 'Non-Embedded' ML Model
- Hardware Assembly

I made strides in each of these goals, but there was a caveat. I made the error of misplacing my ESP32 microcontroller which was to be part of my hardware assembly, so in the meantime before I get a replacement I am using my raspberry pi, which will suffice for a bit but ultimately I may want to switch over to the ESP32 as I had originally intended so I can broadcast CAN messages to a tighter timing specification and to emphasize the embedded nature of this project. Below is a picture of the current setup.

I am adding documentation and analysis of my autoencoder performance this week and it will be live on my github before grading.

Otherwise, I am pleased with the work I got done for this checkpoint as the model is working much better than before and I am ready to begin porting it to the Arduino Uno Q (after I catch up on documentation) :).

Over the next two weeks I will be porting my ML model onto the arduino uno Q. I will also be creating as much documentation and automated analysis as I can to make my final report as robust as possible.

Supporting Evidence (Factual):

- `https://github.com/finncg1234/RTES-2025`
- `https://github.com/finncg1234/RTES-2025/blob/main/README.md`
  README: Has instructions for how to run all of the code that I added this checkpoint
- `https://github.com/finncg1234/RTES-2025/tree/main/autoencoder_can`
  'Non-Embedded' Autoencoder model for CAN intrusion detection, analysis included

Skill Learning Report:

ML/AI: I have relatively little experience with machine learning, but for this checkpoint I implemented multiple versions of an autoencoder and assessed their performance based on metrics I saw used in other papers. I had to consider feature extraction and datasets.

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 had to learn about gradients and Lagrange Optimization.

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

Latex: For this checkpoint, I ported my report over to latex to make it easier to continually update it. I think this is a valuable skill.

Microcontrollers, MPUs: The Arduino Uno Q is a unique embedded system in that it has an MPU running a lightweight linux distro and an MCU (STM module) for real-time applications. The two need to talk to eachother and I am learning a lot about embedded system design through using this device.

Self-Evaluation:

Scope: 100
I continue to refine the scope as I read more and learn more about this problem. I had lofty goals at the beginning of this project of training the model in real-time on an embedded processor, but have realized this is unlikely to work and also likely undesirable. As such, I have focused on making the best possible autoencoder for one-class classification I can and then porting it to a unique embedded system (Arduino Uno Q) which has attributes desirable for this problem: real-time MCU processor for collecting CAN data and feature extraction, and MPU running linux with AI accelerators for running the model. I think my findings for this project would be useful to anyone working in the embedded systems space, particularly in the automotive space, and also to anyone considering working with the Arduino Uno Q as a showcase of its capabilities.

Match: 85
I am mostly on top of what I thought I would be at this point in the project. My inexperience with AI has slowed me down, and my mishap with the ESP32 is unfortunate. I still feel really good about where I am with this project.

Scope: 85
My documentation is a little bit below where I've had it in previous checkpoints, as I focused on optimizing my model, but overall, everything I claim is on my github and I'll be cleaning up documentation as my first task for next checkpoint.

References:

References

[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, ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2020.3046974. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9325944/.

[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, C. Chen, Ed., pp. 1–11, Jun. 27, 2022, ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2022/5827056. Accessed: Nov. 7, 2025. [Online]. Available: https://www.hindawi.com/journals/scn/2022/5827056/.

[3] J. Guidry, F. Sohrab, R. Gottumukkala, S. Katragadda, and M. Gabbouj, "One-Class Classification for Intrusion Detection on Vehicular Networks," in *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, Mexico City, Mexico: IEEE, Dec. 5, 2023, pp. 1176–1182, ISBN: 978-1-6654-3065-4. DOI: 10.1109/SSCI52147.2023.10371899. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/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 *2024 International Conference on Power, Energy and Innovations (ICPEI)*, Nakhon Ratchasima, Thailand: IEEE, Oct. 16, 2024, pp. 157–160, ISBN: 979-8-3503-5677-9. DOI: 10.1109/ICPEI61831.2024.10748816. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10748816/.

[5] L. Liang et al., "Intrusion Detection Model for In-vehicle CAN Bus Based on TPE-LightGBM Algorithm," in *2025 IEEE 34th Wireless and Optical Communications Conference (WOCC)*, Taipa, Macao: IEEE, May 20, 2025, pp. 419–423, ISBN: 979-8-3315-3928-3. DOI: 10.1109/WOCC63563.2025.11082193. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11082193/.

[6] Y. Liao and B. Yang, "To Generalize or Not to Generalize: Towards Autoencoders in One-Class Classification," in *2022 International Joint Conference on Neural Networks (IJCNN)*, Padua, Italy: IEEE, Jul. 18, 2022, pp. 1–8, ISBN: 978-1-7281-8671-9. DOI: 10.1109/IJCNN55064.2022.9892812. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9892812/.

[7] B. M. Tóth and A. Bánáti, "Autoencoder Based CAN BUS IDS System Architecture and Performance Evaluation," in *2025 IEEE 19th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, Romania: IEEE, May 19, 2025, pp. 000 099–000 104, ISBN: 979-8-3315-1547-8. DOI: 10.1109/SACI66288.2025.11030168. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11030168/.

[8] B. Lampe and W. Meng, "Can-train-and-test: A New CAN Intrusion Detection Dataset," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, Hong Kong, Hong Kong: IEEE, Oct. 10, 2023, pp. 1–7, ISBN: 979-8-3503-2928-5. DOI: 10.1109/VTC2023-Fall60731.2023.10333756. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10333756/.

[9] J. N. Brewer and G. Dimitoglou, "Evaluation of Attack Vectors and Risks in Automobiles and Road Infrastructure," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, Dec. 2019, pp. 84–89, ISBN: 978-1-7281-5584-5. DOI: 10.1109/CSCI49370.2019.00021. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9071073/.

[10] 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, Dec. 2013, ISSN: 00313203. DOI: 10.1016/j.patcog.2013.06.005. Accessed: Nov. 7, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0031320313002574.

[11] 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. 129 856–129 869, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3585956. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11071663/.

[12] F. Sohrab, J. Raitoharju, M. Gabbouj, and A. Iosifidis, "Subspace Support Vector Data Description," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Beijing: IEEE, Aug. 2018, pp. 722–727, ISBN: 978-1-5386-3788-3. DOI: 10.1109/ICPR.2018.8545819. Accessed: Nov. 7, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8545819/.