Few-Shot Learning for Classification of CAN Bus Attack Types

Bassel Samara (#200416110)
Department of Computer Science
North Carolina State University
Raleigh, USA
bmsamara@ncsu.edu

Caio Batista de Melo
Department of Computer Science
North Carolina State University
Raleigh, USA
cbatist4@ncsu.edu

***Abstract*—The purpose of this study is to classify cyberattacks on Controller Area Network (CAN) Bus systems, which are essential for automotive communications, using Few-Shot Learning (FSL) techniques, more especially Prototypical Networks. Because they are inherently weak in security, traditional CAN systems are vulnerable to attacks like replay, flooding, and spoofing. The requirement for large labeled datasets in traditional supervised learning techniques is a major drawback considering the infrequency and complexity of real-world attack scenarios. In order to overcome this difficulty, We used an FSL methodology that includes episodic training techniques that allow for efficient generalization from a small number of labeled examples. A precise CAN Bus data segmentation method using temporal thresholds, the extraction of informative statistical features (such as standard deviation, mean, median, and message count), and the restructuring of data into tensor formats appropriate for convolutional neural networks (CNNs) were among the custom implementations. Achieving 100% validation accuracy in a few epochs, the experimental results showed that the Prototypical Network achieved remarkable accuracy, confirming the method's practicality and efficacy in improving automotive cybersecurity.**

I. Research Elements

In automotive environments, Controller Area Network (CAN) Bus systems enable crucial communication between Electronic Control Units (ECUs), handling everything from braking systems to engine controls. These networks were first created without integrated security frameworks [1], making them extremely susceptible to cybersecurity risks like replay, flooding, and spoofing attacks. Vehicle safety, dependability, and operational integrity can all be seriously jeopardized by any of these attacks. Because actual attack events are prevalent, unpredictable, and dangerous, there is a critical need for a rapid and reliable way to identify these threats as they emerge. Classical supervised learning approaches are often too slow and data-hungry to adapt in real time or in data-scarce environments, making them impractical for timely in-vehicle deployment. By significantly lowering data dependency and attaining superior predictive performance, this study suggests Few-Shot Learning (FSL) approaches, particularly utilizing Prototypical Networks with episodic training [2], as an inventive remedy for these issues.

Report submitted for completion of CSC 499 during Spring 2025.

A. Problem Statement

The problem is that Controller Area Network (CAN) Bus systems are susceptible to cyberattacks, and it is difficult to classify these attacks using conventional supervised learning techniques, which necessitate large labeled datasets that are frequently unavailable because real-world attacks are unpredictable and infrequent. This calls for a real-time system that can reliably categorize attacks using a small amount of labeled data.

B. Short Literature Summary

A subfield of machine learning called Few-Shot Learning (FSL) was created to make it possible to classify novel or unknown categories using a small number of labeled examples [3]. This method works particularly well in fields like cybersecurity, where there are few examples of malicious events with labels. Episodic training [2], which simulates the limited availability of labeled data per class, is typically used in FSL methods. A query set (unseen samples for evaluation) and a support set (a small number of labeled samples) are included in every training episode.

Prototypical Networks [3] operate by calculating class prototypes from embedded support set features:

$$p_c = \frac{1}{|S_c|} \sum_{x_i \in S_c} f_\theta(x_i) \tag{1}$$

Here, p_c denotes the class prototype, S_c represents the support samples for class c, and f_θ is the embedding function, typically a CNN architecture. Classification decisions are made based on Euclidean distances between query instances and these prototypes:

$$d(x, p_c) = \sqrt{\sum_{i=1}^{D} (x_i - p_{c,i})^2} \tag{2}$$

where D specifies embedding dimensions.

C. Methodology

II. Methods and Materials

A. Data Preparation

Datasets `SynCanAttacks` and `CarHackingAttacks`, which included actual CAN Bus attack scenarios arranged by

attack type, were used in the study. The SynCanAttacks dataset yielded 13 features with 4 signals per message, while the Car-Hacking dataset [4] yielded 25 features with 8 signals per message. The datasets included 545 and 1200 attack blocks, respectively.

B. Feature Extraction and Tensor Transformation

In order to transform unstructured CAN bus messages into structured tensor representations appropriate for few-shot learning, the CANBus FSL Classifier put into practice a rigorous feature extraction pipeline. The system computed three statistical measures per signal—standard deviation (stdev), mean, and median—for every attack block found during data segmentation. These metrics were chosen because they were able to capture the central tendencies and volatility of attack patterns, which were important for differentiating between malicious and benign message sequences [5]. To provide more contextual information, a message count feature was added to measure the amount of traffic during each attack block.

With datasets that had N signals per message (where N = 4 for SynCanAttacks and N = 8 for Car-Hacking), this procedure produced $3N+1$ total features per block. Following that, the transformation pipeline converted these features into a tensor format that convolutional neural networks could use. The features were first dynamically rearranged into $N \times 3 \times 1$ matrices, with each row denoting a signal and the corresponding stdev, mean, and median values in the columns. To ensure consistent gradient behavior during backpropagation, these matrices were subjected to MinMax normalization, which scaled all values to the interval [0, 1]. The normalized tensors were replicated along the channel dimension to create final tensors of size $3 \times N \times 3$, which simulated RGB-like input channels anticipated by conventional CNN architectures. Despite being straightforward, this replication technique successfully enabled the model to use pretrained convolutional weights without requiring any changes to the architecture. The `CANBusFewShotDataset` class contained the entire pipeline, automating batch generation and guaranteeing compatibility with PyTorch's DataLoader tools.

C. Model Architecture

Initially, features were dynamically rearranged into $N \times 3 \times 1$ matrices, with each row denoting a signal and each column holding the corresponding median, mean, and standard deviation values. All of the values in these matrices were scaled to the interval [0, 1] using MinMax normalization, which guaranteed uniform gradient behavior throughout backpropagation. The final tensors were of size $3 \times N \times 3$ after the normalized tensors were replicated along the channel dimension to mimic RGB-like input channels that are expected by conventional CNN architectures. Even though this replication technique was straightforward, it successfully enabled the model to use pretrained convolutional weights without requiring changes to the architecture. The ResNet12 backbone, a compact residual network derived from the original ResNet architecture, served as the foundation for the convolutional encoder used in all experiments [6]. The entire pipeline was split into a 80/20 ratio with 80% of the data being used to train the model, and 20% being used to test, resulting in the Syncan and Carhacking datasets to include 436 and 960 attack blocks, respectively the other contained in the `CANBusFewShotDataset` class, which guaranteed compatibility with PyTorch's DataLoader tools and automated batch generation.

A crucial aspect of the implementation was the use of 4-way classification tasks in training, with five support examples and ten query examples per class per episode. The limitations in the real world, where new threat categories may have limited access to labeled attack data, were reflected in this setup. The model's input pipeline was designed to handle variable signal counts (N) using dynamic tensor reshaping, resulting in consistent performance across datasets with different message structures.

III. Model Training

A. Episodic Training

All models were implemented in PyTorch for CUDA acceleration, and training was done in an episodic manner on an NVIDIA GPU platform. Stochastic gradient descent (SGD) with a momentum of 0.9 and weight decay of 5e-4 was used in the optimizer configuration. The learning rate was set at 0.01 and decreased by a factor of 10 at epochs 120 and 160. Fifteen randomly selected few-shot tasks with balanced support and query sets were included in each training epoch.

By the first epoch, the model achieved a remarkable 100% validation accuracy on both datasets, which can be attributed to the effectiveness of the Prototypical Networks formulation and the discriminative power of the extracted statistical features. Consistent loss values below 1e-8 following the initial epochs demonstrated training stability, and despite the small episodic batch size, no overfitting was detected. The framework's suitability for real-time deployment scenarios was demonstrated by validation tasks that ran at 50–55 iterations per second.

B. Performance Metrics

Near-perfect classification performance was found across all datasets by quantitative evaluation. With inference times averaging less than 1 millisecond per task, the model achieved 100% accuracy in all validation episodes on the Car-Hacking dataset (960 attack blocks, 4 attack types). The approach's generalizability was confirmed by similar outcomes for the SynCanAttacks dataset (436 blocks, 4 attack types). The efficiency of the ResNet12 backbone was especially noteworthy; when serialized, the entire Prototypical Networks system used less than 6KB of memory, which is a crucial benefit for hardware deployments of automotive quality.

A comparison with conventional supervised methods demonstrated the data efficiency of the FSL method. The suggested system achieved maximum accuracy during training with just 5 examples per class, whereas traditional deep learning models may need thousands of labeled examples per attack type. The need for flexible intrusion detection systems

that can identify new threats with little retraining is directly met by this capability in the automotive sector.

C. Conclusion

This study showed that Few-Shot Learning with Prototypical Networks provides a workable way to classify CAN Bus attacks with perfect accuracy under strict computational limitations. A modified ResNet12 backbone optimized for few-shot automotive applications, an end-to-end training framework that converges quickly even with little labeled data, and a novel feature extraction pipeline that converts temporal CAN bus messages into spatially structured tensors were among the technical contributions. Future research might look into hybrid architectures that combine anomaly detection and few-shot learning to find previously undiscovered attack types. The system could then be ported to automotive-grade microcontrollers for in-car testing, and the temporal analysis window could be extended to assess performance on multi-stage attacks that develop over longer periods of time. These developments will help close the gap between scholarly research and automotive cybersecurity solutions that are ready for production.

D. References

References

[1] de Melo, C. B., & Dutt, N. (2023). LOCoCAT: Low-Overhead Classification of CAN Bus Attack Types. *IEEE Embedded Systems Letters*, 15(4), 178-181.

[2] Sicara. (2023). Easy Few-Shot Learning (EasyFSL) GitHub repository. https://github.com/sicara/easy-few-shot-learning

[3] Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

[4] Seo, E., Song, H. M., & Kim, H. K. (2018). GIDS: GAN based intrusion detection system for in-vehicle network. *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (pp. 1-6). IEEE.

[5] Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, *239*, 142-153. https://doi.org/10.1016/j.ins.2013.02.030

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

IV. Reflection Elements

A. Lessons Learned About Research

I learned a great deal about the research process and the real-world difficulties of using machine learning techniques to solve cybersecurity issues during this project. The project's scope was larger than I anticipated. Although the basic concept of applying Few-Shot Learning (FSL) to the classification of CAN Bus attacks appeared simple, careful data preprocessing, model adaptation, and iterative testing were necessary for its implementation. For significant results, it was essential to focus on prototypical networks and particular datasets (SynCanAttacks and CarHackingAttacks). Converting CAN Bus data into an FSL-compatible format was one of the main obstacles. By reorganizing the data into image-like tensors and making sure it was compatible with the ResNet12 backbone, the initial errors—such as tensor dimension mismatches—were fixed.

B. Project Evolution

From our initial meetings to the finished product, the project underwent significant change. Understanding FSL concepts and investigating current implementations, such as Prototypical Networks, were the main goals of the first phase (January–February 2025). Data formatting errors and model input dimensions were fixed by adapting the CAN Bus dataset to the FSL pipeline. The model was validated on the CarHacking dataset in the mid-phase (March 2025), revealing inconsistencies in data usage while achieving 100% accuracy. As the project grew to include the SynCAN dataset, variations in signal counts per message necessitated modifications to feature extraction and tensor reshaping. Results were consolidated and performance across both datasets was confirmed in the final phase (April 2025). The model's efficiency (e.g., less than 6KB memory usage) and real-time applicability were highlighted in the research report. According to our meeting notes, the project's success was guaranteed by the iterative process of testing, improving, and validating.

C. Scope Changes

There were two significant changes to the project's scope. At first, it was intended to use other datasets, such as Survival-IDS, but the small number of attack blocks (just 29) prevented useful analysis. SynCAN and CarHackingAttacks were given priority since they offered enough information for a thorough analysis. In order to compare the results, it was intended to test other FSL techniques (such as Matching Networks); however, it was already established that Prototypical Networks consistently outperformed them in terms of accuracy and training stability. As a result, the emphasis was reduced to Prototypical Network optimization. I learned from these adjustments how crucial it is to be flexible in research, adjusting to data limitations and concentrating on the most promising methods to produce useful outcomes.