Scientific Research Publishing

Journal of Information Security, 2025, 16(3), 381-405
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

An Enhanced and Explainable Hybrid Ensemble Intrusion Detection System for Connected Vehicles

Xuemei Li[1], Rui Zhu[2], Huirong Fu[1]

[1]Department of Computer Science and Engineering, Oakland University, Rochester, MI, USA
[2]Department of Computer Science, Kettering University, Flint, MI, USA
Email: xuemeili@oakland.edu, rzhu@kettering.edu, fu@oakland.edu

How to cite this paper: Li, X.M., Zhu, R. and Fu, H.R. (2025) An Enhanced and Explainable Hybrid Ensemble Intrusion Detection System for Connected Vehicles. *Journal of Information Security*, **16**, 381-405.
https://doi.org/10.4236/jis.2025.163020

Received: May 6, 2025
Accepted: July 4, 2025
Published: July 7, 2025

Copyright © 2025 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution-NonCommercial International License (CC BY-NC 4.0).
http://creativecommons.org/licenses/by-nc/4.0/

Open Access

Abstract

Intelligent vehicles require strong cybersecurity. The 2021 UNECE WP.29 regulation mandates OEMs to establish Vehicle Security Operation Centers (VSOC), where Intrusion Detection Systems (IDS) are essential. Due to limited computing resources on in-vehicle units, offloading data to edge or cloud is preferred. This paper proposes a Hybrid Ensemble MLP IDS (HE-MLP IDS) combining auto-encoder-based feature extraction, ensemble neural networks, and weighted voting. Evaluated with varying architectures and explained using XAI, the model achieves 0.99 accuracy, recall, and F1-score, with a 7.8e−4 false alarm rate. It outperforms benchmarks in memory size and prediction speed, while effectively identifying attack-relevant features from CAN and network logs.

Keywords

Hybrid Ensemble Intrusion Detection System, MLP, Vehicle Networks, Cybersecurity, Explainable AI

1. Introduction

During the past decade, vehicle systems have been developed to incorporate electronic systems and wireless technologies. There is a growing trend in development of smart transportation infrastructure and autonomous driving vehicles to improve transportation efficiency and safety since 2014 [1]. Vehicles are high value assets to users. Connecting vehicles into networks opens opportunities for various threats and vulnerability exposures. **Figure 1** shows the general connected vehicle network architecture. It has Electronic Control Unit (ECU) buses that are connected to the gateway. It has Media Oriented Systems Transport (MOST), FlexRay,

and Local Interconnect Network (LIN) that are connected to the head unit. A TCU is connected to the head unit through Ethernet. The TCU has a SIM card for cellular network connection, radio systems, Bluetooth connections, and WiFi connections. An On-board Diagnostics (OBD-II) system has been developed for vehicle diagnostics, which can be accessed by connecting to the OBD-II port. Vehicle networks can connect to Internet cloud services and mobile services through the TCU.

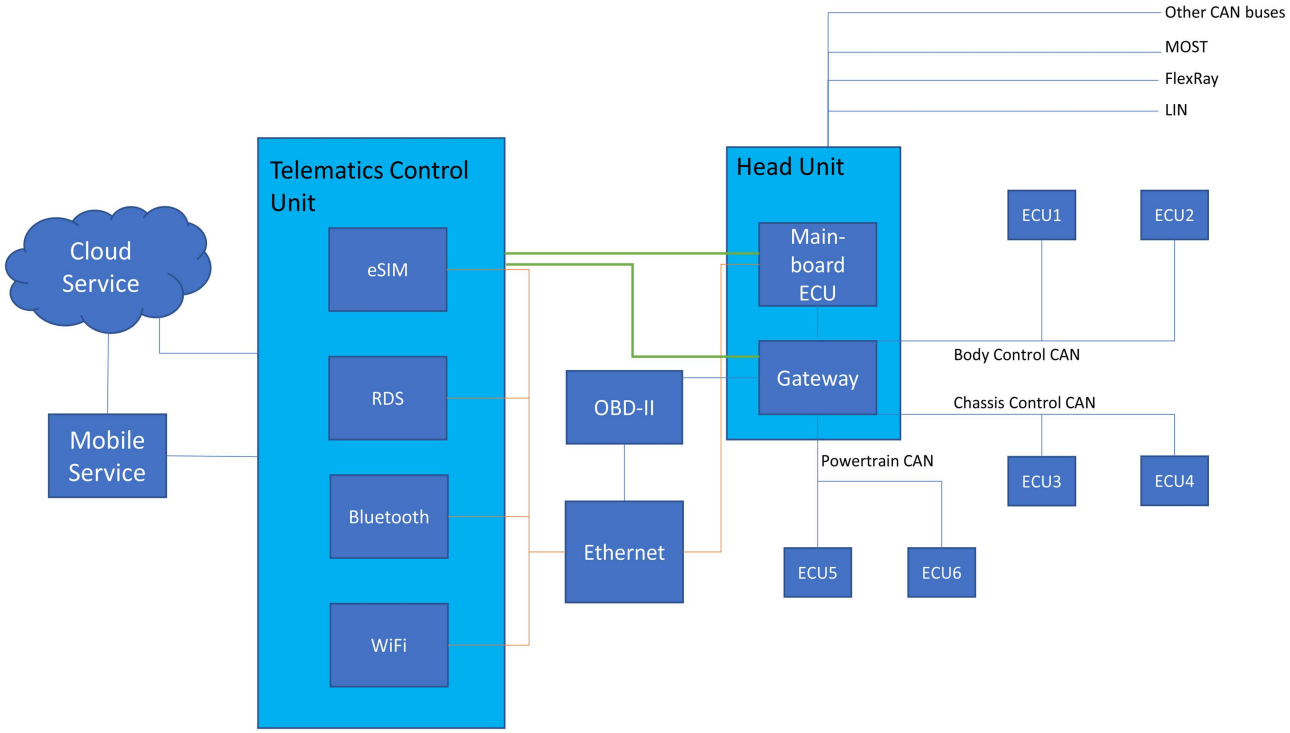

Figure 1. Common vehicle network architecture.

An IDS is a network security system that examines network traffic flow based on critical data features and patterns to detect and prevent network attacks. IDS is a passive system that will signal attacks when they are identified. There are multiple types of IDS. *Anomaly-based detection* predefines the baseline of normal cases, then new types of attacks can be identified once they are observed to carry abnormal information beyond the baseline. *Signature-based detection* stores various existing signatures of known attacks in a database to retrieve them later for making comparisons. Then, it detects the intrusion attack by comparing oncoming cases from the Internet of Vehicle (IoV) with existing signatures of known attacks in the store. *Hybrid intrusion detection* uses the combination of signature-based detection and anomaly-based detection to create multiple protection layers for a system. *Stateful protocol analysis* traces the protocol states to create a normal profile and detect any deviations. Traditional security systems encompass antivirus software, firmware utilizing both static and dynamic analysis, and cryptographic measure [2]. These methods heavily depend on domain knowledge. The

implementation of these methods requires creating rules and workflows in software. The capability for them to recognize a novel attack is limited due to the difficulty in catching up with the speed of growth of different kinds of malware and attacks. Consequently, they require cybersecurity experts to constantly monitor events happening throughout the networks and then develop new rules.

In 2000, Schultz *et al*. applied data mining methods and various Machine Learning (ML) models to detect new malicious executables [3]. Since then, various applications of ML on network security have been created, together with the development of ML and the leap to deep learning models. The most significant advantage of an ML model lies in its ability to learn from data, and its learning process can persist as long as new data is continuously fed to the model. Unsupervised learning models can cluster datasets, thus detecting unseen attacks. It has good parallel processing capability to take advantage of modern computational power increase.

There are several main challenges to developing a connected vehicle IDS based on ML models. First, compared to high end cloud or edge servers, a typical connected vehicle shown in **Figure 1** has limited computation power on its gateway or TCU. Second, a vehicle gateway has limited memory capacity for security functions. It can't fit large ML models. Third, the cybersecurity threat landscape changes daily. Every day, there are various new attack methods discovered. It requires continuous and frequently updated ML models to address unseen attacks. This requires a vehicle to upload those unseen attacks when needed. It also requires an Over The Air (OTA) update to the ML model once it is retrained frequently. The Line of Code update will cost around 15 million dollars per year [4], considering OTA update OEM has a fleet of 10 million vehicles, updating only the gateway, 100MB per update and 100 times per year. This is not realistic and practical for OEMs. Lastly, a gateway often handles real-time or near real-time applications where low latency is critical. ML models with long inference time will not suit to be deployed in vehicle gateway or TCU.

In this study, we aim to design an IDS that can address the main challenges. We develop a simple anomaly detection model based on autoencoder to filter normal vehicle logs. This allows the vehicle to upload only anomaly logs to edge or cloud servers. We create ensembled MLP model which is designed to be deployed in edge or cloud for fast and accurate attack classification. We utilize state-of-the-art explainable AI tools to understand the behavior and evaluate its efficiency in terms of capturing critical attack features.

Our work makes the following major contributions:

- We design a hybrid IDS and develop a general machine model and framework to protect both vehicle CAN bus communication and vehicle to internet communication.
- We investigate the detail relationship and feature patterns through various exploratory data analysis techniques to reveal the complexities of CAN and network traffic attack datasets.

- We propose to use an unsupervised autoencoder to detect unknown anomaly events in vehicles. The latent representation of the filtered anomaly data is sent to the cloud for threat classification. This could significantly reduce data needed to be uploaded to the OEM cloud.
- We create a fast and accurate ensemble MLP model for threat classification in the cloud.
- We benchmark with other common ML models and demonstrate our model that achieves similar accuracy while reducing CPU time and reducing memory size.
- We evaluate our proposed model by varying its hidden layer sizes and visualizing its prediction efficiency using explainable AI techniques.

The reminder of the paper is organized as follows: In Section II, we offer a comprehensive literature review. In Section III, we delve into our IDS architecture design, while in Section IV, we present our model pipeline. In Section V, we provide detailed information and exploratory data analysis about the datasets used in our research. We demonstrate our experimental results in Section VI. In Section VII, we discuss our proposed IDS capability in terms of capturing critical features using explainable AI. Lastly, we conclude our findings and outline our future works in Section VIII and Section IX.

2. Related Works

Anyanwu *et al*. [5] evaluated three ensemble learning methods using the BurST-ADMA dataset. This dataset includes false data injections in the position, speed, and heading of the basic safety messages for connected vehicles [6]. They proposed an algorithm based on optimizing the AdaBoost ensemble algorithm. They found ensembled algorithm can enhance the misbehavior detection accuracy. It outperformed other ensemble classifier models with accuracy of 98.92%. However, they didn't publish the results in terms of F1 score and Recall. They compared the prediction speed per second. They compared bagged trees, subspace KNNs, boosted trees, subspace discriminant, RUSBoosted trees, and optimized AdaBoost. They found bagged trees were the fastest with 52,000 observations per second and optimized AdaBoost was the slowest with 5400 observations per second but achieved the best accuracy.

Aloqaily *et al*. [7] proposed a new hybrid method called D2H-IDS for the network security of connected vehicles. It has 2 layers. The first layer is a deep belief network-based IDS. The deep belief network is used for data dimensionality reduction. The output from the first layer was passed directly to the ID3-based decision tree for feature selection and attack classification. The proposed solution achieved an overall accuracy of 99.43%, with 99.92% detection rate, 0.96% false positive, and a false negative rate of 1.53%. They studied a clustered network that consists of X clusters each containing Y vehicle nodes and Z trusted third party nodes. Each vehicle node forwards its data to its matching cluster head which collects data and forwards it to the corresponding service provider. This network as-

sumes that the cluster head vehicle will have enough bandwidth to send all data from the cluster to service providers in the cloud. Their approach considered having intrusion detection in the cloud only instead of in the vehicle. This design has the downside of considering the amount of data generated by the entire cluster of connected vehicles and the cost of data associated with uploading them into the cloud. The other challenge is that a vehicle is a safety related asset. It requires instant response when safety related intrusion is detected.

Kang *et al*. [8] proposed an IDS using a deep neural network based on a common architecture of IDS. It has a module for profiling and a module for monitoring new attack types. Their proposed IDS will be deployed on the CAN bus. It will classify CAN packets into Attack and Normal. It doesn't mention in the paper about the feasibility and where the IDSs can be deployed considering vehicle constraints. For a given packet, they output the probability of each class discriminating against normal and attack packets. They compared their methods to Artificial Neural Networks (ANN) and support vector machines (SVM). They achieved an accuracy of 98% with classification time being 3.78 ms. In their implementation, they considered the input layer and hidden layers without actuation layers. In the output layer, they applied softmax activation to output the probability of normal and attack packets. With the increase of hidden layers, the computational complexity, training, and testing time continued to increase. Their model is trained with labels. Its ability to detect unknown attack types was not discussed in the paper.

Song *et al*. [9] created an IDS by reducing the size of the Inception Resnet model. This IDS is based on analysis of time intervals of CAN messages for in-vehicle networks. Their model achieved best in class performance compared with LSTM, ANN, SVM, KNN, and DT models. They showed that there was a clear difference between time intervals of messages in the normal status and those of under-attack status. Time intervals of specific CAN ID in normal circumstances were about 0.1 seconds. In contrast, time intervals under injection attack status became short (almost 10% of the normal time interval). They compared it to message rate-based IDS and they reduced the delay of detection. The limitation of their work is that this model requires intensive hyperparameter tuning. They also acknowledged that it can't detect unknown attacks due to it being supervised learning.

Seo *et al*. [10] developed a Generative Adversarial Net (GAN) based IDS that can learn unknown attacks using only normal data and fake attack data. During feature engineering, they converted the raw CAN data into images after one hot vector encoding. They used a generative model G to capture the data distribution and a discriminative model D that estimated the probability of a sample coming from the training data rather than G. They combined the first discriminator for detecting known attack data and the second discriminator for detecting unknown attack data. Their model took 0.18 s to detect 1954 CAN messages. It is well beyond the message generated by vehicles per second. Since their approach needs to

transform CAN messages to images. It is not appropriate to be used in vehicle components with limited computation power and memory.

Yang *et al*. [11] developed a tree-based IDS for internet of vehicles for its network security and in vehicle network security. They trained 2 independent ensemble tree-based models to detect CAN injected message and network attacks. CAN and CIC-IDS 2017 dataset were used. It has a feature extraction using feature importance ranks that achieve high prediction accuracy, random forest, extra tree, and XGBoost. It applied Synthetic Minority Oversampling Technique (SMOTE) to oversample to reduce the effect of sample biases. It then stacked all the models together. They achieved an accuracy of 99.9%, detection rate of 99.9%, and false alarm rate at 0.0006%. However, their execution time was 325.6 s. Later, they designed a multi-tiered hybrid IDS [12]. They built signature-based IDS using tree-based models and added anomaly-based IDS using k-means clustering. The outputs were the attack types, normal or abnormal labels. They achieved 99.9% accuracy, and the proposed system can detect various types of unknown attacks with F1 score of 0.963 and 0.800 for the CAN and intrusion detection dataset. Both of their work has been focused on developing an IDS for vehicle deployment. They provided limited justification in terms of addressing the challenges of deploying large and multilayer models in vehicles.

Teng *et al*. [13] designed an adaptive collaboration intrusion detection method using KDD99 dataset. In their work, they applied environments classes, agents, roles, groups, and objects models. They used DT and SVM to build objects, roles and agents. The model has a multilayer structure. They trained their first layer SVM model with labeled data and filtered events into normal and suspicious. The other three SVMs in the following layers then classify the attacks into different types. Their approach performed better than the methods that used a set of single type SVM. Their approach works with labeled dataset and demonstrates effectiveness against network traffic dataset.

The majority of the state of art ML-based IDSs choose to deploy complex and large ML models in vehicles. There are 3 limitations of the existing works. First, there exists a scarcity of discussions regarding existing research on how to address unknown attack types. The high detection rate is based on supervised learning methods. Second, the feasibility to deploy these models in vehicles is not investigated in terms of model size, prediction speed. Third, most of the existing IDS models were trained using one type of attack data. It lacks generality. Lack of generality leads to the requirement to develop various individual IDSs for different data types such as CAN and inbound or outbound network traffic. This goes against the vehicle's limited computation power and memory space.

3. Intrusion Detection System Architecture Design

Connected vehicle networks security includes in vehicle communication security and vehicle to Internet and vehicle to vehicle security. As shown in **Figure 1**, gateway is the critical component that is responsible for facilitating the CAN bus com-

munication between vehicle ECUs and subsystems. For the vehicle to Internet and vehicle to vehicle security, a vehicle communicates to the Internet through its TCU. The response network packets are sent to the gateway and forwarded to the head unit. As a result, an IDS deployed on a gateway will be able to access the communication in vehicle and in/out of vehicle. However, the computation power and memory are limited in the gateway. The other challenge is that the continuous update and retraining ML models inside vehicles to detect unknown attacks is expensive.

In this work, our main focus is to propose a hybrid IDS to detect both CAN and network traffic. As shown in **Figure 2**, the overall proposed vehicle IDS architecture has 2 layers. The first layer is in the vehicle layer. The design constraints we applied for are limited computation power, limited memory, uploading a minimum amount of data to the cloud to save cost, requires minimal maintenance, able to detect unknown attacks, and need to be a general model to detect both CAN and network attacks. We design an unsupervised autoencoder to perform anomaly detection on the gateway. At this layer, the input to the model is CAN bus message, inbound and outbound network traffic. It can filter the data into normal data and anomaly data. The model will be trained with only normal CAN messages and normal network traffic data. It is trained to minimize log Mean Square Loss (MSE) of reconstructed normal data. It distinguishes anomaly data from normal if the loss is above a fine-tuned criterion. This model is designed to be light weight and requires minimum fine-tuning to ensure its simplicity. The second layer is located in the cloud. In this layer, we applied design constraints such as computation power and memory as demand requires the fastest detection speed to send alerts to VSOC for response. Ensemble learning methods such as bagging, stacking, and boosting have been applied across multiple IDS designs [5] [6] [10] [11]. The main advantages of using an ensemble model [13] [14] are improved performance by mitigating weak learners, increased stability by aggregating prediction, and better generalization for unseen data. Compared to traditional ML, ANN has self-learning capability, introducing non-linearity, continuous learning capability [15]. Because of these benefits, our study focus is to design an Ensemble Multilayer Perceptron Model (E-MLP) to perform threat classification in the cloud. Combining autoencoder and E-MLP, we develop a hybrid multilayer perceptron model (HE-MLP).

4. Model Pipeline

Our proposed model pipeline is shown in **Figure 3**. First, we perform data preprocessing. Data preprocessing is critical to ensure that we can have proper feature representation of data. We import CAN bus data and external network data. We add a new data type column to identify whether it's CAN or network. Then we concatenate them into one single dataframe. For rows, where it is CAN data, the network data related feature columns are filled with 0. For rows, where it is network data, the CAN data related feature columns are filled with 0. We perform data wrangling to remove NAN or infinity data points. We normalize the data

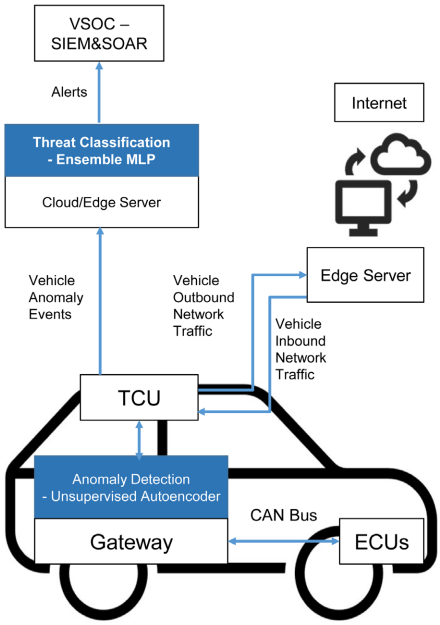

Figure 2. Proposed vehicle IDS architecture.

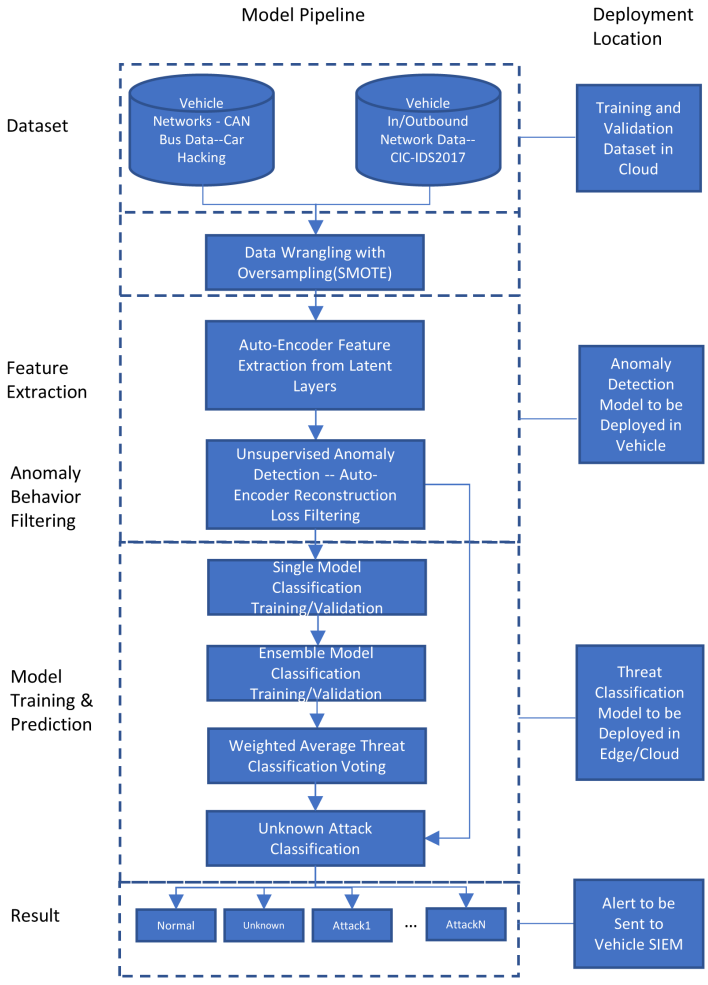

Figure 3. Proposed vehicle IDS framework.

feature column wise using equation (1), where x is the input data vector and x' is the normalized data vector that ranges from 0 to 1. We then split this dataset into 70% for training, 30% for testing. In order to resolve the dataset bias issue, we used SMOTE [16] to further balance the training dataset.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

In the training phase, after the data preprocess, we perform the FE and anomaly detection model training. We input the normal only dataset to train our autoencoder model without a label. This trained encoder of the autoencoder model is used to extract the latent representation of normal and anomaly combined dataset. The latent representation is sent to the decoder of the autoencoder to reconstruct the original input matrix. The losses are used to distinguish normal and anomaly labels. In the testing stage, only anomaly data will be sent to pretrained the ensemble MLP for threat classification. We then use the latent representation of the training dataset with its label to train individual MLP models for threat classification. Multiple single MLP models are ensembled into one model. The individual predictions from individual MLPs are passed into a learnable weighted layer. The layer is later trained with the individual MLP model's weight frozen. This represents the weighted average voting of the final prediction.

In the testing phase, the test data is passed into the autoencoder model. Autoencoder encoder outputs the latent representation. Then it classifies the data into normal and anomaly by checking the reconstructed loss between reconstructed input and the original input. The latent representation of anomaly data is sent to the ensemble MLP models for classification. The weight classification results are sent to unknown attack classification. If the predicted probability is greater than 0.9, then the label predicted by the threat classification model will be used. Otherwise, it will be assigned as an Unknown label. This means the autoencoder detected an anomaly, but the threat classification model can't confidently predict. Highly likely it's unknown data to the model.

4.1. Feature Extraction and Anomaly Detection Autoencoder

Feature Selection (FS) based on feature importance aims to identify subset of relevant features from the original set of features. The advantage is that it maintains the interpretability of the selected features. However, it is limited to the available features and may overlook potential interactions or intrinsic combination of features. FE involves transforming the original features into a new and lower dimensional representation in order to capture complex and intrinsic relations between features. In development of our proposed vehicle IDSs, we seek to upload a minimum amount of data to the cloud to reduce cost but maintain effectiveness. We compare the impact of FS and FE techniques by checking the threat classification performance using features generated by the two methods.

We propose to use an autoencoder as the FE model. There are two main rea-

sons. First, the model architecture has an encoder and a decoder. A typical autoencoder neural network is shown in Figure 4. The output of the encoder is the latent representation which is the extracted features. The output of the decoder is the reconstructed matrix. Its reconstructed loss can be calculated as MSE between input matrix and output matrix from the decoder. This loss can be used to check whether the input data is normal or abnormal. Normal data will show low reconstruction loss and anomaly data will show high reconstruction loss.

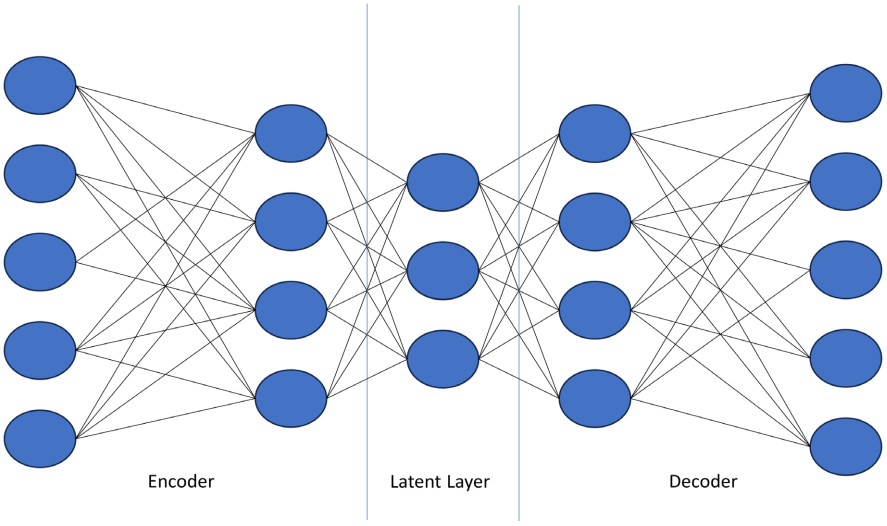

Figure 4. Autoencoder for feature extraction.

Let x be the input data feature vector in the dataset. Let $\boldsymbol{W}_e$ and $\boldsymbol{W}_d$ be the weight matrices of the encoder and decoder layers, respectively. Let $\boldsymbol{h}$ be the latent representation. Let $\hat{\boldsymbol{x}}$ be the reconstructed output. Let f and g be the activation functions used in the encoder and decoder, respectively. Let $\boldsymbol{b}_e$ and $\boldsymbol{b}_d$ be the bias vectors of the encoder and decoder layers, respectively. Let N be the number of samples in the dataset. Let $L(\boldsymbol{x}, \hat{\boldsymbol{x}})$ be the loss between input data and the reconstructed output. Typically, it uses MSE. We have the below formulas for an autoencoder.

$$\boldsymbol{h} = f\left(\boldsymbol{W}_e \boldsymbol{x} + \boldsymbol{b}_e\right) \tag{2}$$

$$\hat{\boldsymbol{x}} = g\left(\boldsymbol{W}_d \boldsymbol{h} + \boldsymbol{b}_d\right) \tag{3}$$

$$L\left(\boldsymbol{x}, \hat{\boldsymbol{x}}\right) = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\right\|^2 \tag{4}$$

In our work, we define loss function as log MSE as $\hat{L}$. The main purpose of defining a good loss function is to better separate the loss between normal and abnormal data. This loss function helps us to place more weight on large errors compared to MSE. It will be more sensitive to the outliers in the data compared to MSE. Normally, the outliers in CAN messages and network traffic are anomalies. When performing data visualization for individual point loss, we use negative log MSE to convert all loss to be positive value.

$$\hat{L}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \log\left(\frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\right\|^2\right) \tag{5}$$

4.2. Ensemble Model Classification Using MLP

A MLP model is a type of artificial neural network. The key components of it include an input layer, a hidden layer, an output layer, and an activation layer. An ensemble of MLP models combines the predictions of multiple individual MLP models to make a final prediction. In our implementation, we add one more layer which we train to decide the best weight assigned for each single MLP. The final prediction is combined through a weighted sum of each single MLP prediction. By doing this, the ensemble model assigns higher importance to more accurate models and lower importance to less accurate ones.

The ensemble MLP model takes input data feature vector as $\boldsymbol{x}$ and produces a predicted output $\hat{\boldsymbol{y}}$. Let m be the number of MLP models, denoted as $M = \{1, 2, \cdots, m\}$. For each MLP model $m \in M$, the model predicts an output $\hat{\boldsymbol{y}}_m$ based on $\boldsymbol{x}$. To obtain the final ensemble prediction $\hat{\boldsymbol{y}}_e$, each individual model prediction is weighted and combined using weights $\boldsymbol{w}_m$:

$$\hat{\boldsymbol{y}}_e = \sum_{m=1}^{M} \boldsymbol{w}_m \cdot \hat{\boldsymbol{y}}_m \tag{6}$$

The weights $\boldsymbol{w}_m$ are learned during training by minimizing the sparse categorical cross entropy loss function. Then the final output probabilities $\boldsymbol{y}_p$ are computed by applying a softmax activation using input $\boldsymbol{x}$, where $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{b}$ is the bias vector of the weights layer.

$$\boldsymbol{y}_p = \text{softmax}\left(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}\right) \tag{7}$$

4.3. Model Structures

In this study, we try to use the simplest structure design of an autoencoder and MLP. The purpose of this is to focus on the effectiveness of the model framework compared to benchmark models. A simple architecture is less prone to overfit and helps to establish the baseline performance. Figure 5 and Figure 6 are the autoencoder model structure and MLP model structure, respectively. To introduce non-linearity to the mapping functions, we employ ReLU and Sigmoid activation functions.

4.4. Validation Metrics

We use accuracy (BA), detection rate (DR)/recall, false alarm rate (FAR), F1 score, prediction time (PT), and model memory size (MEM) as the main metrics to evaluate the proposed IDS. The formula of them can be obtained from [17]. IDS dataset including attacks are normally imbalanced. This reflects the real-world fact that cyber-attacks are rare compared to normal data. Due to this nature, a high accuracy may not result in a high detection rate of attacks. Hence, it is important to check accuracy together with detection rate. FAR is critical as well. Falsely reported alarms lead to wasted investigation time of incidence triage. F1 score com-

bines accuracy and recall which reflects the overall model performance. Prediction time shows the speed of model prediction. This is measured using a timer in Jupyter notebook in units of seconds. Vehicle is a safety related product that requires instant detection and response. Model memory needs to be minimized in order to be deployed into vehicles. All models are exported into the same type of file after training. Its size will be reported as MEM. Thus, an ideal IDS for connected vehicles should have high DR for attacks, low FAR, low PT, and low MEM. Model training time (TT) is also reported to reflect whether models can be quickly retrained for new attacks.

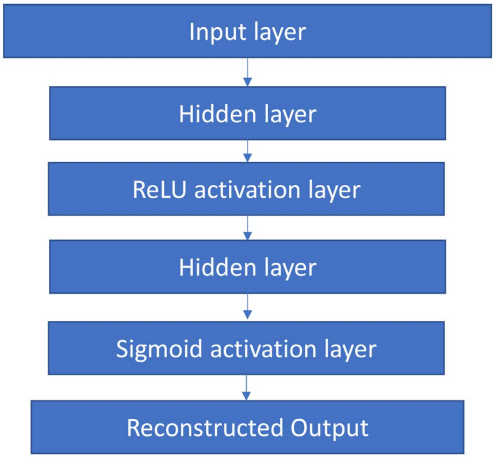

Figure 5. Autoencoder model structure.

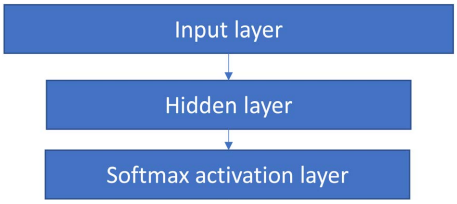

Figure 6. MLP model structure.

With C represents number of classes, TP as true positive, FP as false positive, TN as true negativ and FN as false negative, the main metrics can be calculated as following:

Balanced Accuracy (BA): BA is crucial for imbalanced datasets, as it calculates the average accuracy per class, offering a more equitable performance measure.

$$\mathrm{BA} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i} \tag{8}$$

Detection Rate (DR)/Recall: DR evaluates the model' ability to correctly identify true positives from all actual positives.

$$\mathrm{DR} = \frac{TP}{TP + FN} \tag{9}$$

False Alarm Rate (FAR): FAR is particularly pertinent in intrusion detection

systems (IDS), quantifying the rate of false positives in relation to all actual negatives. This metric has significant practical implications, especially considering the high volume of messages in vehicle systems.

$$\text{FAR} = \frac{FP}{FP + TN} \tag{10}$$

F1 score: The F1 score provides a balanced view of a model's precision and recall, useful for overall performance assessment.

$$\text{F1} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{11}$$

These metrics collectively offer a comprehensive understanding of the models' performance in real-world, imbalanced datasets, reflecting their practical utility and effectiveness in identifying genuine threats in vehicular systems.

4.5. Model Prediction Explaination

In order to understand how our proposed model works, we use SHAP [18] (SHapley Additive exPlanations) which is an explainable AI framework for understanding the output of ML models. SHAP value is based on Shapley values from cooperative game theory that allocates a value to a feature in a prediction based on their contribution to the outcome, considering all possible combinations of features.

With ϕ_i as SHAP value, N as the set of all features, S as a subset of features excluding feature i, $f(S)$ as the model prediction when considering subset features, S is the number of features in subset, N is the total number of features, The general formula of SHAP value can be expressed as following:

$$\phi_i(s) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S) \right] \tag{12}$$

For a given sample s with N features, the proposed model makes a prediction as $f(s)$. SHAP values are calculated for each feature using the formula above. These values represent the contribution of each feature to the model's prediction.The SHAP values provide an explanation of how each feature influences the prediction. Positive SHAP values indicate that a feature contributes positively to the prediction, while negative SHAP values indicate the opposite.

5. Datasets

In this study, our focus is to build an IDS for vehicles. Two datasets were critical to our study. First is the CAN bus data. It captures the communication within the vehicle. The second is the network traffic data. It represents vehicle communication with external networks.

5.1. CAN Dataset

The datasets we used are the same as Yang's paper [11]. We use the CAN dataset gathered by Song *et al.* [9]. This dataset covers three different types of attacks: DoS

attack, Fuzzy attack, and Spoofing attack. These datasets were constructed by logging CAN traffic via the OBD-II port from a real vehicle while message injection attacks were in process. Therefore, it could represent the real-world behavior of this vehicle. The data attributes include timestamp, CAN_id, DLC, data1, data2, data3, data4, data5, data6, data7, data8, Label. Table 1 shows the overview of the Car Hacking dataset.

Table 1. Car hacking dataset.

Attack Type	Messages	Normal Messages	Injected Messages
DoS Attack	3,665,771	3,078,250	587,521
Fuzzy Attack	3,838,860	3,347,013	491,847
Spoofing Drive Gear	4,443,142	3,845,890	597,252
Spoofing RPM Gauze	4,621,702	3,966,805	654,897
Attack-Free (Normal)	988,987	988,872	NA

We have performed in depth Exploratory Data Analysis (EDA) for the CAN dataset without considering timestamp features. Timestamp is an important feature which consists of sequence, frequency and interval time difference information between different messages. It can be captured by our ML model. Hence, we mainly focus on the messages themselves in our EDA. After removing duplicates, we discover that the DoS, RPM and Gear attack consists of one same message per attack type. The overall summary is shown in Table 2. Thus, there is only one sequence pattern for each attack. Although there are a lot of DoS, RPM and Gear attacks, these attacks will be easy for ML models to learn. It also needs to be noted that there are a lot of duplicated messages for each attack type per Table 1. These duplicated messages will help neural networks repeatedly update model parameters that can capture them. Therefore, it will cause neural networks performance to be biased towards these three patterns.

Table 2. Attack type and its attack message.

CAN ID	DLC	data1	data2	data3	data4	data5	data6	data7	data8	Label
0000	8	00	00	00	00	00	00	00	00	DoS
0316	8	45	29	24	ff	29	24	00	ff	RPM
043f	8	01	45	60	ff	6b	00	00	00	Gear

Fuzzy attack is a type of injection attack to find vulnerabilities in the victim components. Attackers send random data to the CAN network without knowing much about it. We find there are no duplicated messages among the fuzzy attack messages. There are 491,847 different fuzzy attack CAN messages in the dataset.

There are 14,237,958 benign messages. Among them, there are 186,614 unique benign messages. These messages represent various communication among different components in the CAN network.

With cor as pearson correlation coefficient, n as number of data point, x as one variable and y as one variable, the Pearson correlation between x and y is calculated in equation (13). The correlation matrix for CAN datatset is plotted in Figure 7 after dropping the timestamp column. It shows that CAN_ID, data1, data2, data5 and data8 have strong relationship with attack Label. It also can be seen that there are some correlations between some data channels to other data channels within a message. It means certain CAN messages have their own format having special meaning in terms of vehicle functions.

$$cor = \frac{n\sum xy - \sum x \sum y}{\sqrt{\left(n\sum x^2 - \left(\sum x\right)^2\right)\left(n\sum y^2 - \left(\sum y\right)^2\right)}} \tag{13}$$

Our EDA is essential for us to gain insight with the dataset features. It can also help us to verify model output and understand model behavior by cross-checking the data features related to different attack types.

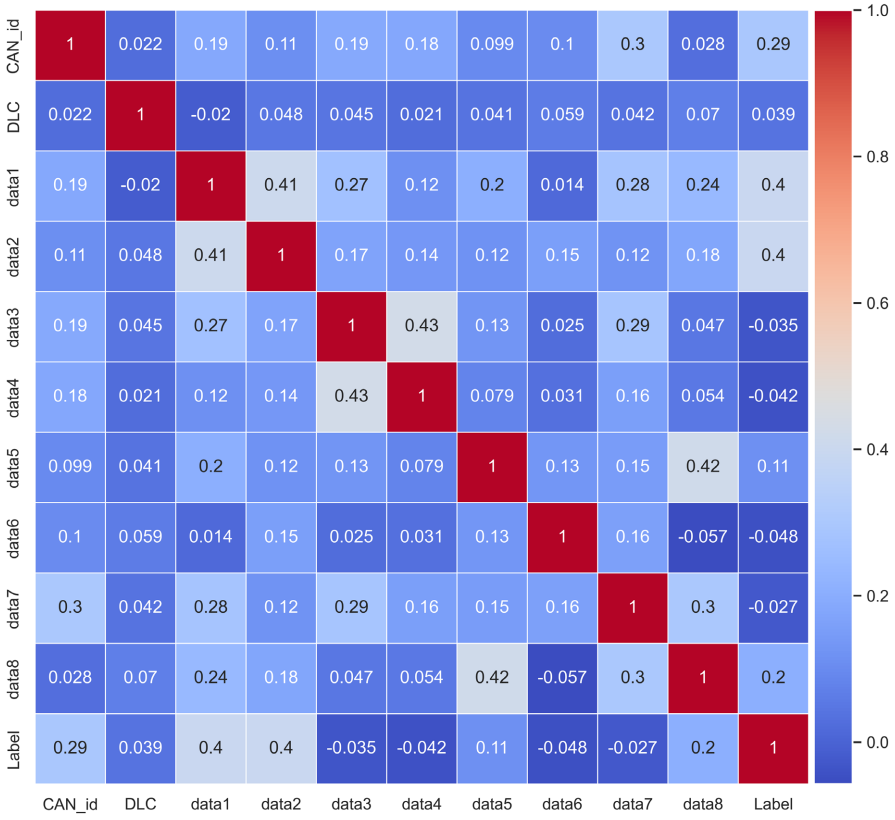

Figure 7. Correlation matrix between different CAN features and attack label.

5.2. Network Traffic Dataset

For the network traffic dataset, we use the "CIC-IDS2017" dataset from [19]. It

contains benign and 14 different attack packets and flow information which are captured through the packet capture tool (PCAP) and analyzed using CICFlow-Meter with labeled flows. We remove the duplications from the original dataset. Table 3 shows the overview of the "CIC-IDS2017" dataset.

Table 3. CIC-IDS2017.

Data Type	Num. of Samples	Num. of Samples after Removing Duplications
Benign	2,273,097	2,097,816
DoS Hulk	231,073	172,861
PortScan	158,930	90,819
DDoS	128,027	128,017
DoS GoldenEye	10,293	10,286
FTP-Patator	7938	5953
SSH-Patator	5897	3219
DoS Slowloris	5796	5385
DoS Slowhttptest	5499	5228
Bot	1966	1953
Web Attack - Brute Force	1507	1470
Web Attack - XSS	652	652
Infiltration	36	36
Web Attack - SQL Injection	21	21
Heartbleed	11	11

We perform correlation analysis to understand the relationship between different features. Before we perform the correlation study, we try to find any columns that have constant value. A constant value column will have no effect on the outcome of ML models. We found the following features contains only constant value: BwdPSHFlags, BwdURGFlags, FwdAvgBytes/Bulk, FwdAvgPackets/Bulk, FwdAvgBulkRate, BwdAvgBytes/Bulk, BwdAvgPackets/Bulk, BwdAvgBulkRate. After removing these columns, we obtain our correlationship heatmap in Figure 8. It can be observed that the network features that are closely related with attack labels are: DestinationPort, FlowDuration, FwdPacketLengthMin, BwdPacketLengthMax, BwdPacketLengthMin, BwdPacketLengthMean, BwdPacketLengthStd, FlowIATStd, FlowIATMax, FlowIATotal, FwdIATStd, FwdIATMax, MinPacketLength, MaxPacketLength, PacketLengthMean, PacketLengthStd, PacketLengthVariance, Down/UpRation, AveragePacketSize, AvgBwdSegmentSize, IdleMean, IdleMax, IdleMin.

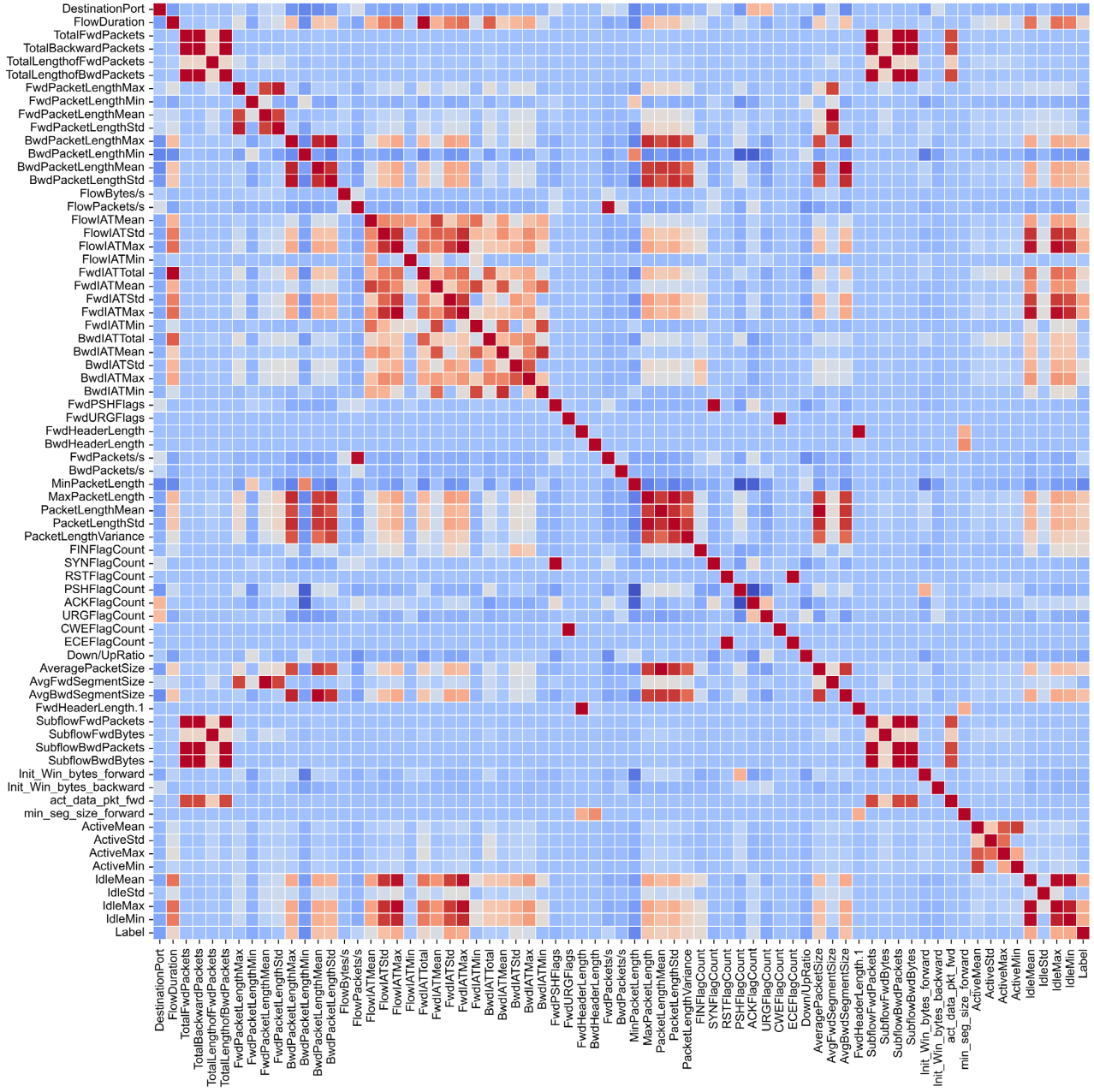

Figure 8. Correlation matrix between different network features and attack label.

Overall, it's obvious that the network traffic attack dataset contains more complex patterns than CAN attack dataset for each attack type.

6. Experimental Result

In the section, we present our proposed model performance compared to other benchmark models. For the benchmark models, we use Random Forest and XGBoost from [11]. We choose them since we use the same datasets and performances are more comparable. Our training machine is a desktop with AMD Ryzen 9 with 12 core processor with clock speed of 3.7GHz, 64GB RAM and

NVidia RTX 3090 with 24GB graphic RAM. Our experiments have 2 parts. First, we establish the base performance of our model and the 2 benchmark models. In the first experiment, all models are trained using a training dataset including autoencoder, ensemble MLP, random forest, and XGBoost. Due to the simplicity of the ensemble MLP and autoencoder model, we do not need to perform in depth tuning. We fine tune the number of ensemble models (3 to 10), hidden layer size (64, 128, 256, 512), batch size (32, 64, 128), learning rate (0.1, 0.01, 0.001, 0.0001), and training epochs to ensure convergence. For ensemble MLP, we have a learning rate of 0.001, batch size of 32, hidden layer size of 64, and number of models at 3. We have observed that our ensemble model has better performance than each individual model. We do not observe significant performance improvement by increasing the number of models. For autoencoders, we have a learning rate of 0.001, batch size of 64, and hidden layer size of 512. Table 4 is the summary of model performance. Figure 9 shows the visualization of autoencoder separation of normal and attack data in terms of its negative log MSE loss.

Table 4. Overall model performance benchmark -classification using all data features.

Model Type	BA	DR	FAR	F1	PT	MEM	TT
Random Forest	0.99	0.99	1e−5	0.99	8.6 s	11,824 KB	2 m
XGBoost	0.99	0.99	7e−5	0.99	137.0 s	3428 KB	1 d 22 h 46 m
E-MLP	0.99	0.99	1e−3	0.99	1.3 s	241 KB	3 h 23 m
Autoencoder	0.97	0.97	3e−2	0.97	11.8 s	572 KB	8 h 23 m

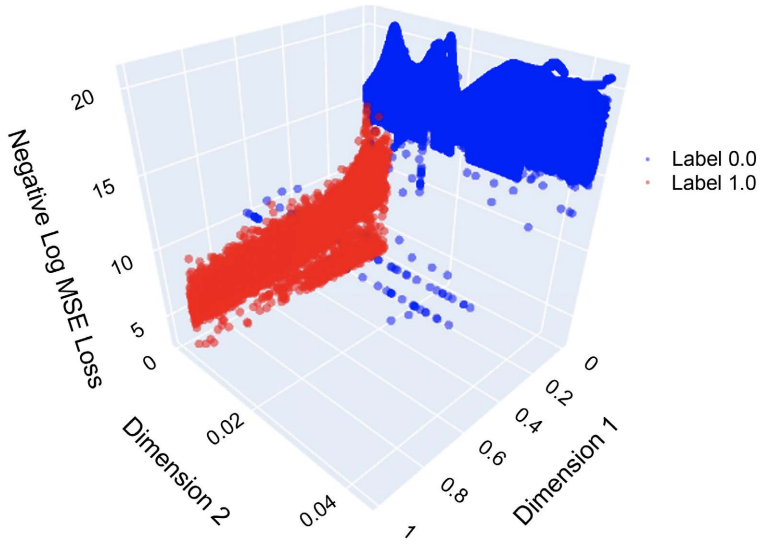

Figure 9. Negative log MSE loss.

The first experiment shows that the ensemble MLP is able to perform threat classification. It can achieve comparable performance as the benchmark models. It has significantly smaller model size as well as fastest prediction speed. It also

shows autoencoders are able to perform anomaly detection after unsupervised. It can classify normal and anomaly without knowing labels during training. This means autoencoders can be used to detect unknown attack type data. Figure 9 shows that the reconstructed loss of some normal data is very similar to abnormal data. This normal data will be falsely reported to the ensemble MLP for threat classification. The other observation is that an autoencoder is able to filter out normal data. In final, it will pass 99.7% of anomaly data and only 5.5% normal data were sent to classification. This is a significant reduction of data needed to be sent to the cloud. As the OEM connected vehicle fleet grows, the cost reduction impact will be further magnified.

The second experiment is to understand the model performance of FE compared with FS methods based on feature importance. After the first experiment, the feature importance was calculated by averaging random forest feature importance and XGBoost feature importance from the already trained models. The features are sorted based on their feature importance values. The top features which have total feature importance larger than 0.9 were kept. This is the FS dataset. The FE dataset from autoencoder is the latent representation output by the decoder of the trained autoencoder. We have tested the FE dataset at 17, 64 and 512 hidden layers. Table 5 shows the model performance of using FS dataset and FE dataset. It can be observed that the FE is slightly worse than the FS methods when the FE hidden layer size is at 17. However, when the FE hidden layer sizes increase to 64 and 512, E-MLP with FE dataset performance is comparable to the models that trained with FS dataset. The performance when the hidden layer is at 512 is the best. The main reason that more hidden layers lead to better performance is due to larger hidden layer sizes that can capture more complex patterns during the training process. We use loss value calculated from the autoencoder as a new feature for each row of data to store the information for the row of attached data. It also helps the model to perform attack detection and classification.

We can also observe that the classification performance for network traffic attack is worse than CAN attack. This is due to most network features are removed during FE and the attack pattern of network traffic dataset is more complex than the CAN attack as we observe in our EDA in Section V. It needs to point out that the training time (TT) and prediction time (PT) increase when FE hidden layer size increases. In summary, FE is superior since it is a by-product from autoencoder anomaly detection. This means there is no need to deploy a separate FS model in vehicles.

7. Discussion of Model Prediction

As shown in Figure 10, CAN_ID, loss computed from autoencoder, data2, data6, data8, timestamp and data4 have a positive effect on the prediction of CAN Benign message. Figures 11-18 show the feature contributions for different attacks. It can be clearly seen that the losses computed from the autoencoder are on the top 3 contributors to the prediction. As shown in Figure 13, for the feature contributions for

the Fuzzy attack, the features contributing to the prediction are very similar to CAN benign type messages. This is in line with the fact that Fuzzy attacks normally contain normal messages sent to devices in CAN to discover vulnerabilities.

Table 5. Overall model performance benchmark -classification using FS and FE.

Model Type	BA	DR	FAR	F1	PT	MEM	TT
Random Forest - FS	0.99	0.99	2.4e−5	0.99	7.4 s	10,642 KB	2 m
XGBoost - FS	0.99	0.99	9.0e−5	0.99	111 s	3017 KB	1 d 12 h 32 m
E-MLP - FS	0.99	0.99	3.0e−3	0.99	0.8 s	221 KB	3 h 15 m
Random Forest - FE17	0.98	0.97	6.0e−5	0.98	6.3 s	8277 KB	1 m 28 s
XGBoost - FE17	0.98	0.97	1.0e−4	0.98	137 s	2367 KB	1 d 04 h 12 m
E-MLP - FE17	0.97	0.97	2.0e−2	0.97	1.3 s	215 KB	2 h 58 m
Random Forest - FE64	0.99	0.99	8.0e−5	0.99	8.0 s	15,080 KB	3 m 40 s
XGBoost - FE64	0.99	0.99	8.7e−5	0.99	165 s	3654 KB	3 d 07 h 23 m
E-MLP - FE64	0.99	0.99	1.9e−3	0.99	3.5 s	215 KB	4 h 23 m
Random Forest - FE512	0.99	0.99	8.1e−5	0.99	8.6 s	15,399 KB	3 m 59 s
XGBoost - FE512	0.99	0.99	8.6e−5	0.99	175 s	3662 KB	3 d 08 h 50 m
E-MLP - FE512	0.99	0.99	7.8e−4	0.99	4.6 s	215 KB	6 h 05 m

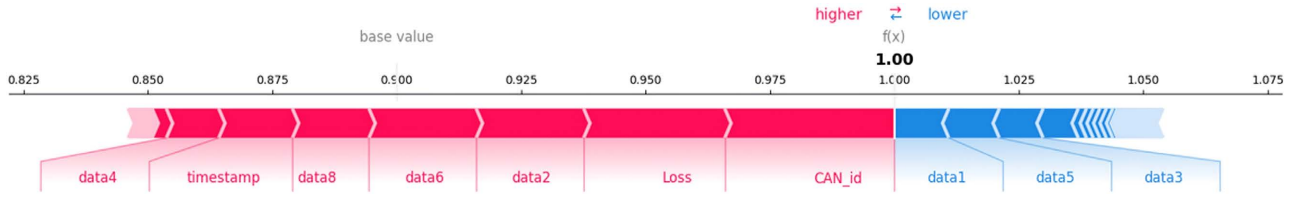

Figure 10. Force plot of CAN benign type.

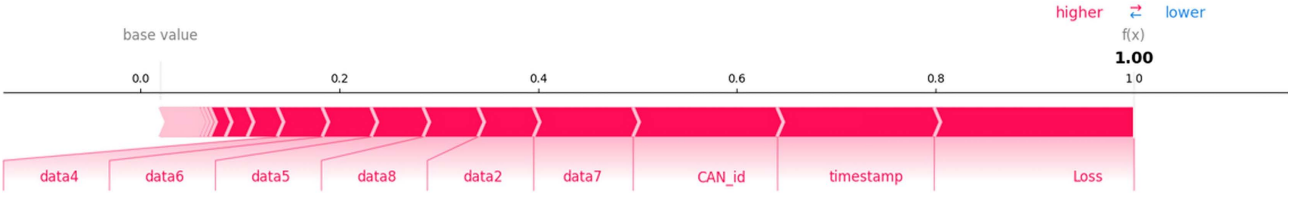

Figure 11. Force plot of CAN RPM spoofing attack type.

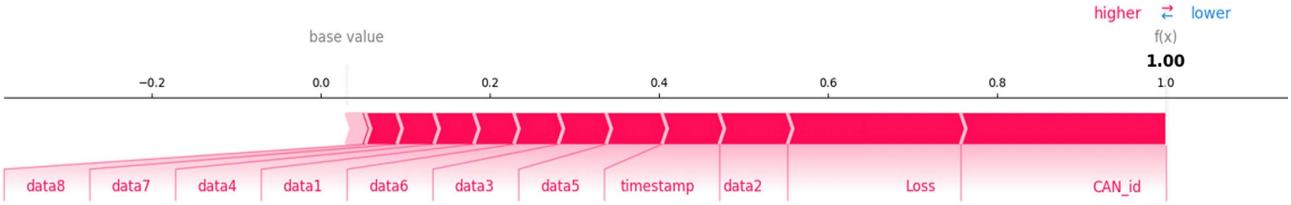

Figure 12. Force plot of CAN gear spoofing attack type.

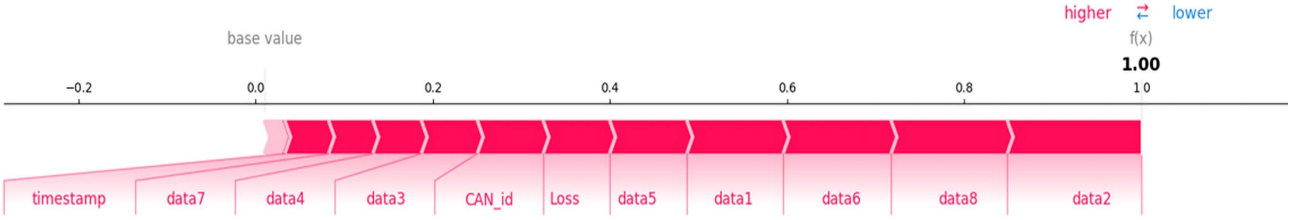

Figure 13. Force plot of CAN fuzzy attack type.

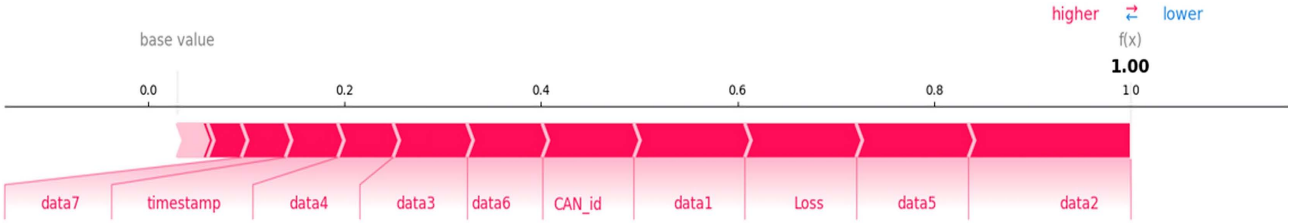

Figure 14. Force plot of CAN DoS type.

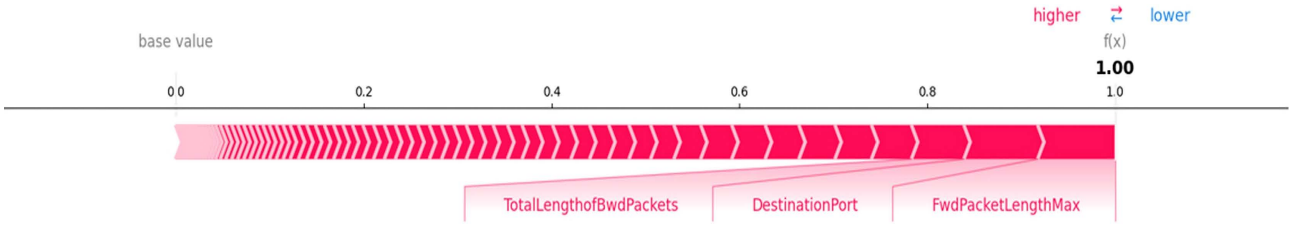

Figure 15. Force plot of network DosHulk type.

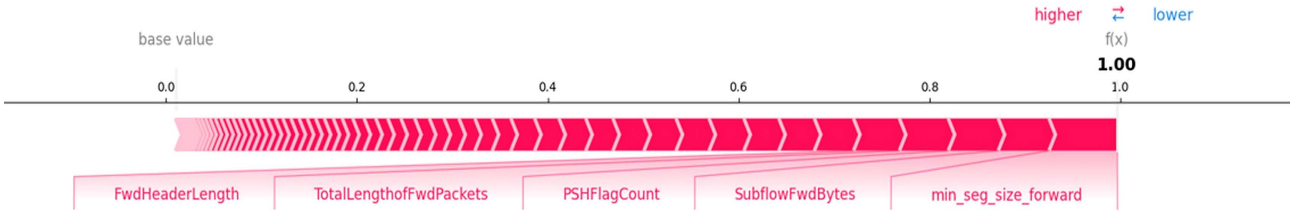

Figure 16. Force plot of network portscan type.

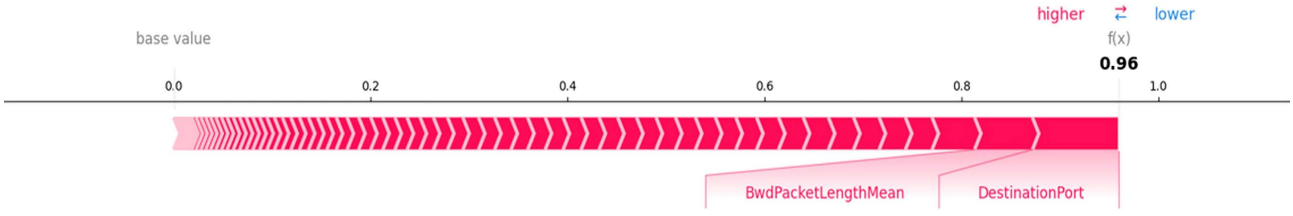

Figure 17. Force plot of network DDoS type.

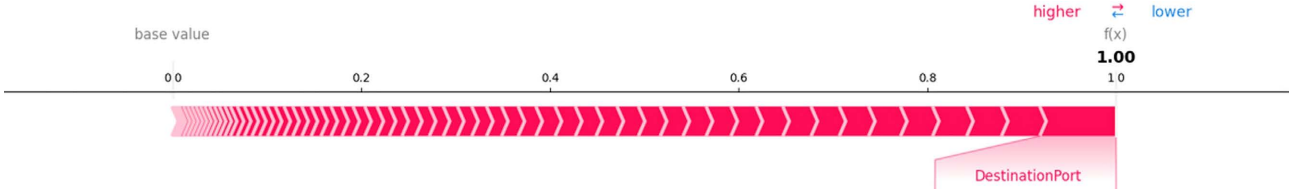

Figure 18. Force plot of network bot type.

For network attacks, we can clearly find out the main features that contribute to different types of attacks. For example, Figure 15 shows that the main contributors for the network DosHulk attack are TotalLengthofBwdPackets, DestinationPort and FwdPacketLengthMax. Figure 16 shows that the main contributors for Portscan attack are min_seg_size_forward, SubflowFwdBytes, PSHFlagCount, TotalLengthofFwdPackets and FwdHeaderLength. The common detection patterns for Portscan involve high port scanning rate through multiple ports, high rate of connection attempts to various ports, sequential or unusual port scanning, short connection duration. The features discovered by our models are closely related to these portscan network patterns. Figure 17 shows that the main contributors for DDoS attack are DestinationPort and BwdPacketLengthMean. DoS attack types often come with spike in traffic and unusual traffic patterns which are indicated by TotalLengthofBwdPackets, FwdPacketLengthMax, DestinationPort and BwdPacketLengthMean features. Figure 18 shows that the main contributor for Bot attack is DestinationPort. To detect a Botnet pattern, it's critical to understand where and how often the packets are sent. Thus, the DestinationPort used by our models to make predictions is critical.

Overall, all the features discovered by our models for each type of attack correlate well with the real-world behavior of these attacks.

8. Conclusions

In this paper, we propose a Hybrid Ensemble Multilayer Perceptron-based Intrusion Detection System (HE-MLP-IDS). An autoencoder is used to extract important features and detect anomalies. An ensemble MLP is developed to perform threat classification.

Our proposed HE-MLP IDS offers the following advantages when compared to existing approaches:

- The proposed IDS uses one model to protect both CAN bus and network at the same time.
- Using autoencoder anomaly detection to classify the data into normal and abnormal. It enables the ability to only send anomaly data into cloud servers for threat classification.
- Faster inference speed by leveraging GPU power, considering future deployment in cloud servers. The inference speed is 9 times faster than the next best random forest model.
- Since autoencoders distinguish normal and anomaly through negative log mean squared error, it is able to detect unknown attacks.
- Its ensemble model structure reduced the risk of model overfit by sacrificing small amounts of accuracy.
- The model size is around 10 times smaller than the next best XGBoost model.
- The proposed models have a very simple model structure, which requires very minimum model fine tuning.
- We integrate the explainable AI tool (SHAP) to show our model's capability

to use critical features from the CAN and network log to make accurate predictions.

9. Future Works

While the proposed HE-MLP IDS demonstrates strong performance in accuracy and efficiency, several important directions remain for future exploration. First, the current evaluation relies on traditional ML baselines such as Random Forest and XGBoost; benchmarking against state-of-the-art deep learning and graph-based IDS models (e.g., GNNs, transformers) would provide a more comprehensive assessment. We plan to include these comparisons in future benchmarking efforts, particularly at the cloud layer where computational resources are less constrained.

Second, our experiments were conducted in a simulated environment using public datasets. To validate real-world applicability, future work will involve deploying the IDS in actual connected vehicles or vehicle testbeds to assess performance under dynamic and noisy operational conditions.

Third, we intend to investigate adaptive complexity in IDS design—deploying lightweight anomaly detectors within the vehicle and enabling dynamic offloading of more complex threat classification tasks to the cloud. This layered architecture can help balance real-time responsiveness with high detection performance.

Additionally, we aim to develop a unified evaluation framework that supports cross-dataset validation to better assess generalizability and robustness across heterogeneous attack conditions. Such efforts will further improve the reliability and applicability of the proposed IDS in diverse and evolving connected vehicle scenarios.

Acknowledgements

This material is partially based upon work supported by the National Science Foundation under Grants NSF-2146280 and NSF-2327944. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Andrew, M. (2020) Business Insider Intelligence. https://www.businessinsider.com/iot-connected-smart-cars

[2] Dibaei, M., Zheng, X., Jiang, K., Abbas, R., Liu, S., Zhang, Y., *et al.* (2020) Attacks and Defences on Intelligent Connected Vehicles: A Survey. *Digital Communications and Networks*, **6**, 399-421. https://doi.org/10.1016/j.dcan.2020.04.007

[3] Schultz, M.G., Eskin, E., *et al.* (2001) Data Mining Methods for Detection of New

Malicious Executables. *Proceedings* 2001 *IEEE Symposium on Security and Privacy*, Oakland, 14-16 May 2000, 38-49.

[4] Auroralabs (2021) OTA Update Cost Considerations. Aurora Labs OTA Update Cost Considerations. https://www.automotive.auroralabs.com/ota-guide-viewer/

[5] Anyanwu, G.O., Nwakanma, C.I., Kim, J., Lee, J. and Kim, D. (2022) Misbehavior Detection in Connected Vehicles Using Burst-Adma Dataset. 2022 13*th International Conference on Information and Communication Technology Convergence*, Jeju Island, 19-21 October 2022, 874-878. https://doi.org/10.1109/ictc55196.2022.9952947

[6] Amanullah, M.A., Baruwal Chhetri, M., Loke, S.W. and Doss, R. (2022). Burst-Adma: Towards an Australian Dataset for Misbehaviour Detection in the Internet of Vehicles. 2022 *IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events* (*PerCom Workshops*), Pisa, 21-25 March 2022, 624-629. https://doi.org/10.1109/percomworkshops53856.2022.9767505

[7] Aloqaily, M., Otoum, S., Ridhawi, I.A. and Jararweh, Y. (2019) An Intrusion Detection System for Connected Vehicles in Smart Cities. *Ad Hoc Networks*, **90**, Article 101842. https://doi.org/10.1016/j.adhoc.2019.02.001

[8] Kang, M. and Kang, J. (2016) Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLOS ONE*, **11**, e0155781. https://doi.org/10.1371/journal.pone.0155781

[9] Song, H.M., Woo, J. and Kim, H.K. (2020) In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network. *Vehicular Communications*, **21**, Article 100198. https://doi.org/10.1016/j.vehcom.2019.100198

[10] Seo, E., Song, H.M. and Kim, H.K. (2018) GIDS: GAN Based Intrusion Detection System for In-Vehicle Network. 2018 16*th Annual Conference on Privacy, Security and Trust* (*PST*), Belfast, 28-30 August 2018, 1-6. https://doi.org/10.1109/pst.2018.8514157

[11] Yang, L., Moubayed, A., Hamieh, I. and Shami, A. (2019) Tree-Based Intelligent Intrusion Detection System in Internet of Vehicles. 2019 *IEEE Global Communications Conference* (*GLOBECOM*), Waikoloa, 9-13 December 2019, 1-6. https://doi.org/10.1109/globecom38437.2019.9013892

[12] Yang, L., Moubayed, A. and Shami, A. (2022) MTH-IDS: A Multitiered Hybrid Intrusion Detection System for Internet of Vehicles. *IEEE Internet of Things Journal*, **9**, 616-632. https://doi.org/10.1109/jiot.2021.3084796

[13] Teng, S., Wu, N., Zhu, H., Teng, L. and Zhang, W. (2018) SVM-DT-Based Adaptive and Collaborative Intrusion Detection. *IEEE/CAA Journal of Automatica Sinica*, **5**, 108-118. https://doi.org/10.1109/jas.2017.7510730

[14] Ganaie, M.A., Hu, M.H., *et al*. (2022) Ensemble Deep Learning: A Review. *Engineering Applications of Artificial Intelligence*, **115**, Article 105151.

[15] Shah, B. and H Trivedi, B. (2012) Artificial Neural Network Based Intrusion Detection System: A Survey. *International Journal of Computer Applications*, **39**, 13-18. https://doi.org/10.5120/4823-7074

[16] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*, **16**, 321-357. https://doi.org/10.1613/jair.953

[17] Salo, F., Injadat, M., *et al*. (2020) Data Mining with Big Data in Intrusion Detection Systems: A Systematic Literature Review.

[18] Lundberg, S.M. and Lee, S.-I. (2017) A Unified Approach to Interpreting Model Pre-

dictions. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 4768-4777.

[19] Sharafaldin, I., Habibi Lashkari, A. and Ghorbani, A.A. (2018) Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the* 4*th International Conference on Information Systems Security and Privacy*, Madeira, 22-24 January 2018, 108-116.