DATASET EXPLORATION

Student Name:Ozan Duru
Instructor's Name:Mennan Guder
Course Title:Data Mining for Cybersecurity

Selected Dataset: CICIoV2024 - Controller Area Network - CAN bus in the context of cybersecurity (Canadian Institute for Cybersecurity - CIC)

IoV extended IoT into the automotive industry. It allows for real-time monitoring of vehicles and intercommunication of vehicles with external services. IoV enables various applications like predictive maintenance, autonomous driving, fleet management, road assistance, and more, which will enable improvement in safety, fuel emission reduction, and enhancement in traffic transportation.

Even with these advantages, IoT and IoV come up with a major security issue in cybersecurity. Nonavailability of encryption, poor authentication, and vulnerability to APTs and flood attacks impose serious threats. Attacks may sometimes affect confidentiality, integrity, and availability, which potentially causes safety hazards, privacy, or financial loss to IoV systems.

The IoV is getting more complex, and traditional security has been insufficient due to the core complications in traffic and network interaction. Machine Learning methods are one of these approaches that have provided efficient IoV systems, mitigating cyberattack detection and prevention. ML models leverage patterns in network traffic to identify suspicious activities, which represent potential threats. The main objective of this project is developing a realistic benchmark dataset named CICIoV2024 in order to fill the gap in the IoV security area. The dataset is created through experiments on a 2019 Ford vehicle's Electronic Control Units using five different attacks, including spoofing and Denial-of-Service attacks. The data is in various formats: binary, decimal, and hexadecimal representation, enabling multiple analyses.Related Works present the studies connected with IoV security datasets. The objective of the present research is to advance the current state-of-the-art in IoV security datasets so as to further assist and contribute to creating new solutions in the domain. It follows the analyses of the studies about the subject and their contributions.

IoV-Specific Datasets Using CAN Bus:

Dataset based on Hyundai YF Sonata CAN bus: Another work has proposed an IoV-specific dataset based on the CAN bus of an actual Hyundai YF Sonata. The researchers performed DoS, Fuzzy attack, RPM spoofing, and GEAR spoofing attacks. The data is in the form of hexadecimal images with detailed representation of network traffic during these attacks.

CAN Intrusion Dataset:

Authors of another work propose a dataset named CAN Intrusion, which has been generated in a KIA Soul vehicle testbed. They carried out various kinds of

attacks like DoS, Fuzzy, and Impersonation attacks, and presented the data collected in hexadecimal format.

IoV Network Traffic Dataset with Diverse Attacks:

Another research work presents a dataset for IoV network traffic concerning different types of attacks like accelerator attacks, correlated signal fabrication attacks, fuzzing attacks, max engine coolant temp fabrication, max speedometer fabrication, reverse light Off fabrication, and reverse light On fabrication. The data collected in this dataset is represented in a decimal form.

Employing General IoT and Security Datasets:

Many, due to the limited IoV-specific datasets, have used general IoT and security datasets to validate their solutions. Examples of such datasets include CICIDS2017 and ISCXIDS2012, covering a wide range of attacks that include brute force, DoS, DDoS, XSS, SQL injection, infiltration, botnet, and port scans.

Other popular ones include NSL-KDD, UNSW-NB15, ToNIoT, KDDCup99, IoT BotNet, KDD99, and AWID. These provide the best traffic data in detecting many types of attacks.

CICIoV2024 Dataset:

This work is based on these few efforts by providing the state-of-the-art IoV datasets in terms of expanding it with new attacks, adopting a comprehensive testbed architecture, and evaluating Machine Learning models against different methods of data representation.

Structure of a CAN (Controller Area Network) packet:

Start of Frame (SOF): A flag indicating the beginning of the data frame. It synchronizes all the nodes on the CAN bus for receipt of the message that is just about to arrive.

Arbitration Identifier (ID): 11 or 29 bits in length, depending on whether the frame is a standard format or extended format. It contains the identifier of the message and is utilized to prioritize the messages on the CAN bus. Messages with a lower ID enjoy higher priority.

RTR: It is a 1-bit field with various purposes: for differentiating a data frame from a remote frame. Its value is normally 0 for a data frame and 1 for a remote frame.

Control Field: A 6-bit field that contains control information, such as DLC, referring to the number of bytes of data in any given frame.

Data Field: This is up to 64 bits long and carries the actual payload or data being transmitted, such as sensor readings or control commands.

CRC: This is a 16-bit field utilized for detecting errors. It ensures that the integrity of the transmitted data is checked against any potential errors during transmission.

ACK: It is a field of 2-bit information for the acknowledgment to receive the message correctly. When the data is received correctly by at least one node, an acknowledgment is returned.

EOF: This is a 7-bit field indicating the end of the data frame, which simply shows the completion of transmission.

DoS Attacks on the CAN Bus

The basis for a Denial-of-Service attack in the CAN bus is an active exploitation of the mechanism of message arbitration. How this principle works and where the danger lies are explained here.

Manipulation of Arbitration:

Attackers can manipulate the arbitration values to grant their malicious packets higher priority. Flooding the CAN bus with a huge amount of misleading high-priority packets will prevent real communication from happening. This will make it tough or impossible for critical messages from the vehicle's Electronic Control Units to get transmitted.

Impact on Communication:

By manipulating the transmission rate of a particular flow of communication, the attackers prevent the ECUs from accessing the bus, an action that violates the system's availability. In other words, it is possible for an attacker to disturb the normal operation of the vehicle by interfering with critical systems related to brakes or the engine.

Authentication Weaknesses Exploitation:

Since authentication methods are not provided by the CAN bus itself, an attacker could send random CAN frames to the ECUs and monitor reactions of these units. Such a manipulation could finally lead to unpredictable behavior of the network, keeping the in-vehicle network unstable.

Arbitration ID Spoofing:

In a DoS attack, attackers may perform arbitration ID spoofing-continuously sending messages with the same ID of critical control messages. CAN bus uses arbitration IDs to determine the priority of messages. By casting lots of high-priority messages, attackers can interfere with regular communication and prevent key commands from reaching their destination on time.

Fuzzy Attack:

A fuzzy attack is the type of DoS attack where an attacker sends random or invalid data to disturb the networks. The difference is that this uses lower-priority nodes for creating the interference, which might make the interference a little unpredictable but all the same damaging to the general process of communications.

Spoofing Attacks on the CAN Bus

In the case of an impersonation attack by an attacker, it would be considered spoofing when unauthorized access is gained to the CAN bus network. Here is how those attacks work, and what the implications are.

Attack Mechanism:

The attackers intercept the traffic on the CAN bus to learn some key information, such as the CAN ID, payload range, and transmission rate. With this information at hand, they spoof the arbitration ID, masquerading themselves as legitimate nodes in order to manipulate the behavior of given ECUs.

For instance, an attacker may intercept information from the car's doors, change it, and inject malicious commands to unlock the doors while the car is moving. Other cases include modification of RPM data, which may potentially place passengers in danger due to unexpected actions of the vehicle during travel.

Spoofing Attacks:

To perform a spoofing attack, the attackers must have deep knowledge of the various details of the targeted ECUs. Indeed, due to privacy and security concerns, such detailed technical data regarding a vehicle's ECUs is not usually publicly available. If available, this kind of information typically originates from previous research or other documented hacking experiments.

These attacks, in this research, were carried out by relying on data from previous investigations in order to manipulate certain key parameters such as steering wheel position, RPM, speed, and gas pedal control. This sort of manipulation is shown in Table 3, which outlines how one can modify CAN packets in order to depict the specific action or behavior modification of the vehicle.

Spoofing vs. DoS Attack:

Both compromise vehicle security, but they differ in their execution and the way they affect a vehicle:

Spoofing is designed to masquerade as a genuine component and inject malicious data into the system so as to harm the integrity or functionality at large of the vehicle systems.

Denial-of-Service entails network flooding that overloads the system, disrupting communication such that the system becomes unavailable.

Data Collection and Organization:

Data collected from the CAN bus includes variations of different attack scenarios, namely: benign or normal traffic, DoS attacks, spoofing attacks on the steering wheel, RPM, gas pedal, and speed.

The data is recorded in the raw format in a TXT format and needs cleaning and pre-processing. The raw data is in hexadecimal format that is then converted into binary and decimal format for easier analysis.

Conversion Process:

This process of conversion from hexadecimal to binary increases the dimensionality because one hexadecimal character is represented using more bits. However, the decimal representation retains the same dimensionality as that of the original data.

Dataset Split:

Further, the dataset was divided into two parts: one for training the model and the other for testing the performance of the models trained:

Training Set: 80% utilizes training ML models.

Test Set: 20% utilizes the performance testing for the trained models.

This split ensures that the ML models are well trained and validated with different data.

Machine Learning Models

Logistic Regression:

This is a simple, interpretable model that is usually used for binary classification. It predicts the outcomes based on the relation of various input variables with the target variable. Although conceptually simple, it may be very effective in problems where classes are well separated.

Random Forest :

Ensemble technique that builds a large number of decision trees based on the different subsets of training data. Each of the trees makes a prediction; the final output is based on the majority vote of all these trees. This model is resistant to overfitting, and it is very good at capturing the complex relations in data. This will make them quite appropriate for diverse and imbalanced datasets, such as those obtained in CAN bus communications.

Adaboost:

Adaptive Boosting Ensemble: A method combining the predictions of many "weak" classifiers, usually simple decision trees, into one strong predictive model. In a series of steps, Adaboost adapts the weights on incorrectly classified instances to focus on hard-to-classify examples, therefore enhancing the performance of the overall model. It is also recognized for adapting to hyperparameter optimization and changing data patterns.

Deep Neural Networks:

DNNs are complex models stimulated by the neural structure of the human brain. They are characterized by several layers that consist of interconnected neurons with the ability to learn complicated and nonlinear data relations. Especially in cybersecurity, where detecting intricate patterns and subtle anomalies is fundamental, DNNs have been among the highly effective solutions. Their capability for handling vast amounts of information makes them a strong choice in high-dimensional data analysis, such as multi-feature CAN bus traffic.

Why These Models Have Been Chosen:

These ML models were chosen for their capability to process and analyze intricate data structures. Models would provide a good balance between simplicity and power, and some ensemble methods and deep learning techniques may work very well in those cases where relationships between data are difficult or nonlinear. Besides, variety enables performing a thorough assessment that would definitely ensure multiple kinds of cyber-attacks featured in the dataset are identified.

Evaluation Metrics:

True Positives(TP): Correctly Predicted Positive

True Negatives(TN): Correctly Predicted Negative

False Positives(FP): Incorrectly Predicted Positive

False Negatives(FN): Missed Positive

Results are provided in terms of several evaluation metrics in order to distinctly identify the performance of every model.

Parameters used for each ML model.

Model	Parameters
Random Forest (RF)	n_estimators = 100, criterion = 'gini', min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_features = 'sqrt', min_impurity_decrease = 0.0, bootstrap = True, oob_score = False, warm_start = False, ccp_alpha = 0.0
AdaBoost (AB)	estimator = DecisionTreeClassifier, n_estimators = 50, learning_rate = 1.0, algorithm = 'SAMME.R'
Logistic Regression (LR)	penalty = 'l2', dual = False, tol = 0.0001, C = 1.0, fit_intercept = True, intercept_scaling = 1, solver = 'lbfgs', max_iter = 100, multi_class = 'auto', warm_start = False
Deep Neural Network (DNN)	hidden_layer_sizes = (16,16,16,16), solver = 'adam', alpha = 0.0001, batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001, power_t = 0.5, max_iter = 200, shuffle = True, tol = 0.0001, warm_start = False, momentum = 0.9, nesterovs_momentum = True, early_stopping = False, validation_fraction = 0.1, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e−08, n_iter_no_change = 10, max_fun = 15000

Explanation of ML Model Parameters

This table shows the parameters used for different Machine Learning (ML) models: **Random Forest (RF)**, **AdaBoost (AB)**, **Logistic Regression (LR)**, and **Deep Neural Network (DNN)**. Each parameter influences how the model learns from the data and makes predictions.

1. Random Forest (RF) Parameters

- **n_estimators = 100**: The number of decision trees in the Random Forest. More trees generally lead to more robust and accurate predictions.
- **criterion = 'gini'**: The metric used to measure the quality of a split in the trees. 'Gini' refers to the Gini impurity, a measure of how often a randomly chosen element would be incorrectly identified.
- **min_samples_split = 2**: The minimum number of samples required to split an internal node. A smaller value means trees can grow deeper.
- **min_samples_leaf = 1**: The minimum number of samples required to be at a leaf node. A smaller number allows the tree to grow deeper.

- **max_features = 'sqrt'**: The number of features to consider when looking for the best split. 'sqrt' means the square root of the total number of features.
- **bootstrap = True**: Whether to use bootstrapped samples when building trees. If True, the model samples data with replacement.
- **oob_score = False**: Out-of-bag score. If set to True, it uses samples not included in the bootstrap sample to validate the model.
- **warm_start = False**: If True, it reuses the solution from the previous call to fit and adds more estimators to the ensemble.
- **ccp_alpha = 0.0**: The complexity parameter used for Minimal Cost-Complexity Pruning. A larger value leads to more pruning.

2. AdaBoost (AB) Parameters

- **estimator = DecisionTreeClassifier**: The base model used in boosting. In this case, a Decision Tree Classifier is used.
- **n_estimators = 50**: The maximum number of estimators or trees added sequentially in the model.
- **learning_rate = 1.0**: The rate at which the model adjusts its parameters. A smaller learning rate requires more estimators for good performance.
- **algorithm = 'SAMME.R'**: The boosting algorithm to use. 'SAMME.R' is a variant of AdaBoost for multi-class problems.

3. Logistic Regression (LR) Parameters

- **penalty = 'l2'**: The regularization method used to avoid overfitting. 'l2' refers to Ridge regularization.
- **dual = False**: Determines whether to use a dual or primal formulation. Usually, dual is used when the number of samples is greater than the number of features.
- **tol = 0.0001**: The tolerance for stopping criteria. A smaller value means more precise optimization.
- **C = 1.0**: The inverse of regularization strength. Smaller values mean stronger regularization.
- **fit_intercept = True**: If True, a constant (bias) is added to the decision function.
- **solver = 'lbfgs'**: The algorithm used to optimize the model. 'lbfgs' is an optimization algorithm suitable for small datasets.
- **max_iter = 100**: The maximum number of iterations taken for optimization.
- **multi_class = 'auto'**: Determines if the problem is treated as a binary or multi-class problem.
- **warm_start = False**: If True, it reuses the solution from the previous fit.

4. Deep Neural Network (DNN) Parameters

- **hidden_layer_sizes = (16,16,16)**: Defines the number of neurons in each hidden layer. In this case, there are three hidden layers, each with 16 neurons.
- **solver = 'adam'**: The optimization algorithm used for weight adjustment. 'adam' is a popular method for efficient optimization.
- **alpha = 0.0001**: The regularization term to prevent overfitting.
- **batch_size = 'auto'**: The size of the batch of data used in each iteration.
- **learning_rate = 'constant'**: The rate at which the model's weights are updated. 'constant' means the learning rate does not change during training.
- **learning_rate_init = 0.001**: The initial learning rate for weight updates.
- **power_t = 0.5**: Used in learning rate schedules when 'invscaling' is used. It adjusts how the learning rate decreases over time.
- **max_iter = 200**: The maximum number of iterations for training.
- **shuffle = True**: If True, the samples are shuffled after each iteration.
- **momentum = 0.9**: The momentum for gradient descent to speed up convergence.
- **nesterovs_momentum = True**: If True, it uses Nesterov's momentum, an accelerated form of gradient descent.
- **early_stopping = False**: If True, training stops early if there is no improvement on the validation set.
- **validation_fraction = 0.1**: The fraction of the training data used as a validation set.
- **beta_1 = 0.9, beta_2 = 0.999**: Parameters for the Adam optimizer, controlling the decay rates of the moving averages of the gradients.
- **epsilon = 1e-08**: A small value to avoid division by zero in the optimizer.
- **n_iter_no_change = 10**: The number of iterations with no improvement to stop training.
- **max_fun = 15000**: The maximum number of function calls for optimization.

Understanding the Parameters

Model-Specific Adjustments: The parameters differ from model to model, and their tuning is done in a way to optimize the learning for that particular model without getting into overfitting.

Regularization and Learning Rate: These are the parameters that control model complexity and how fast it picks up from the data.

Optimization and Iterations: These define how the model is going to adjust weights and when it should stop training.

Fine-tuning of these parameters will importantly lead to better performance, accuracy, and efficiency for anomaly detection in the CAN bus network using ML models.

Features extracted for the CICIoV2024 dataset.

Feature name	Description
ID	Arbitration — indicates the priority of the message and the type of data it carries.
DATA_0	Byte 0 of the data transmitted.
DATA_1	Byte 1 of the data transmitted.
DATA_2	Byte 2 of the data transmitted.
DATA_3	Byte 3 of the data transmitted.
DATA_4	Byte 4 of the data transmitted.
DATA_5	Byte 5 of the data transmitted.
DATA_6	Byte 6 of the data transmitted.
DATA_7	Byte 7 of the data transmitted.
label	The identification of benign or malicious traffic.
Category	The identification of the category to which the traffic belongs.
Specific_class	The identification of the specific class of the traffic.

Feature Explanation

ID: Refers to arbitration; it determines the priority of the message in CAN bus communication. The smaller the ID, the higher the priority of the message.

DATA_0 to DATA_7: This is the actual bytes of the data that are being transmitted, which contains vital information such as sensor values or control instructions.

Label, category, and specific_class:

label: Identifies whether the traffic is benign-meaning safe-or malicious-meaning an attack

category: Contains the type of attack, including but not limited to DoS and spoofing.

specific_class: It describes in detail the targeted information, which is under attack, be it RPM, speed, steering wheel, or gas.

Data Representations:

Decimal Representation: Data values are in bytes, and it computes descriptive statistics such as mean, standard deviation, minimum, percentiles for each feature .

Binary Representation: This data is represented by bits, giving a fine granularity in each feature and indicating the occurrence of each bit being 0 or 1.

Importance of Data Representation:

Threat detection is made simpler because the patterns in the data will be clearer.This enhances the performance of the machine learning using well-structured and clean features.Enables extensive processing by allowing the view of data at both high level- decimal as well as detailed- binary.The aim is to provide machine learning models with the capability for effective differentiation between normal and malicious traffic, hence offering enhanced security features for in-vehicle networks.

Data description for the decimal dataset.

	Mean	Std	Min	25%	50%	75%	Max
ID	537.207946	322.479994	65	357	516	578	1438
DATA_0	71.0865995	88.9771748	0	0	16	127	255
DATA_1	69.9892503	95.5837431	0	0	12	128	255
DATA_2	55.0112724	72.7658378	0	0	13	125	255
DATA_3	57.4536382	90.3207664	0	0	0	92	255
DATA_4	45.2851673	64.4583498	0	0	6	86	255
DATA_5	53.8826134	94.3361202	0	0	0	63	255
DATA_6	71.7491441	101.687183	0	0	0	138	255
DATA_7	60.2747691	99.9654672	0	0	0	80	255

Decimal Dataset Description

Summary statistics of features in the Decimal dataset. For each feature, it shows mean, standard deviation (std), minimum (min), percentiles (25%, 50%, 75%) and maximum (max) values:

1.ID:

Average: 537.21 - This is the average value of the IDs for CAN bus messages.

Standard Deviation: 322.48 - Pretty much describes the dispersion of the IDs.

Min and Max: 65 and 1438 – The variation in this range exists for the ID values :. This range defines message priority.

2.DATA_0 - DATA_7:

Each column of DATA_X defines a specific byte of data sent.

The mean values define usual data on those bytes. For instance, the mean of the DATA_0 is 71.09 while that of DATA_4 is 45.29.The Standard Deviation values define how much the data deviates.

Min - Max: Each DATA byte represents a range from 0 to 255 since each byte has the capacity to store up to 8 bits of data.

Data description for the binary dataset (Part I).

Feature	Mean	Std	Min	25%	50%	75%	Max	Feature	Mean	Std	Min	25%	50%	75%	Max
ID0	0.00	0.00	0	0	0	0	0	DATA_116	0.30	0.46	0	0	0	1	1
ID1	0.00	0.00	0	0	0	0	0	DATA_20	0.00	0.00	0	0	0	0	0
ID2	0.00	0.00	0	0	0	0	0	DATA_21	0.00	0.00	0	0	0	0	0
ID3	0.00	0.00	0	0	0	0	0	DATA_22	0.00	0.00	0	0	0	0	0
ID4	0.00	0.00	0	0	0	0	0	DATA_23	0.00	0.00	0	0	0	0	0
ID5	0.00	0.00	0	0	0	0	0	DATA_24	0.00	0.00	0	0	0	0	0
ID6	0.17	0.38	0	0	0	0	1	DATA_25	0.00	0.00	0	0	0	0	0
ID7	0.43	0.49	0	0	0	1	1	DATA_26	0.00	0.00	0	0	0	0	0
ID8	0.29	0.45	0	0	0	1	1	DATA_27	0.00	0.00	0	0	0	0	0
ID9	0.13	0.34	0	0	0	0	1	DATA_28	0.00	0.00	0	0	0	0	0
ID10	0.38	0.48	0	0	0	1	1	DATA_29	0.17	0.37	0	0	0	0	1
ID11	0.50	0.50	0	0	1	1	1	DATA_210	0.23	0.42	0	0	0	0	1
ID12	0.48	0.50	0	0	0	1	1	DATA_211	0.31	0.46	0	0	0	1	1
ID13	0.19	0.39	0	0	0	0	1	DATA_212	0.25	0.44	0	0	0	1	1
ID14	0.57	0.49	0	0	1	1	1	DATA_213	0.34	0.47	0	0	0	1	1
ID15	0.47	0.50	0	0	0	1	1	DATA_214	0.31	0.46	0	0	0	1	1
ID16	0.53	0.50	0	0	1	1	1	DATA_215	0.31	0.46	0	0	0	1	1
DATA_00	0.00	0.00	0	0	0	0	0	DATA_216	0.33	0.47	0	0	0	1	1
DATA_01	0.00	0.00	0	0	0	0	0	DATA_30	0.00	0.00	0	0	0	0	0
DATA_02	0.00	0.00	0	0	0	0	0	DATA_31	0.00	0.00	0	0	0	0	0
DATA_03	0.00	0.00	0	0	0	0	0	DATA_32	0.00	0.00	0	0	0	0	0
DATA_04	0.00	0.00	0	0	0	0	0	DATA_33	0.00	0.00	0	0	0	0	0
DATA_05	0.00	0.00	0	0	0	0	0	DATA_34	0.00	0.00	0	0	0	0	0
DATA_06	0.00	0.00	0	0	0	0	0	DATA_35	0.00	0.00	0	0	0	0	0
DATA_07	0.00	0.00	0	0	0	0	0	DATA_36	0.00	0.00	0	0	0	0	0
DATA_08	0.00	0.00	0	0	0	0	0	DATA_37	0.00	0.00	0	0	0	0	0
DATA_09	0.22	0.42	0	0	0	0	1	DATA_38	0.00	0.00	0	0	0	0	0
DATA_010	0.34	0.48	0	0	0	1	1	DATA_39	0.19	0.39	0	0	0	0	1
DATA_011	0.32	0.47	0	0	0	1	1	DATA_310	0.26	0.44	0	0	0	1	1
DATA_012	0.33	0.47	0	0	0	1	1	DATA_311	0.24	0.43	0	0	0	0	1
DATA_013	0.31	0.46	0	0	0	1	1	DATA_312	0.29	0.45	0	0	0	1	1
DATA_014	0.38	0.48	0	0	0	1	1	DATA_313	0.27	0.44	0	0	0	1	1
DATA_015	0.32	0.47	0	0	0	1	1	DATA_314	0.29	0.45	0	0	0	1	1
DATA_016	0.30	0.46	0	0	0	1	1	DATA_315	0.27	0.44	0	0	0	1	1
DATA_10	0.00	0.00	0	0	0	0	0	DATA_316	0.28	0.45	0	0	0	1	1
DATA_11	0.00	0.00	0	0	0	0	0	DATA_40	0.00	0.00	0	0	0	0	0
DATA_12	0.00	0.00	0	0	0	0	0	DATA_41	0.00	0.00	0	0	0	0	0
DATA_13	0.00	0.00	0	0	0	0	0	DATA_42	0.00	0.00	0	0	0	0	0
DATA_14	0.00	0.00	0	0	0	0	0	DATA_43	0.00	0.00	0	0	0	0	0
DATA_15	0.00	0.00	0	0	0	0	0	DATA_44	0.00	0.00	0	0	0	0	0
DATA_16	0.00	0.00	0	0	0	0	0	DATA_45	0.00	0.00	0	0	0	0	0
DATA_17	0.00	0.00	0	0	0	0	0	DATA_46	0.00	0.00	0	0	0	0	0
DATA_18	0.00	0.00	0	0	0	0	0	DATA_47	0.00	0.00	0	0	0	0	0
DATA_19	0.28	0.45	0	0	0	1	1	DATA_48	0.00	0.00	0	0	0	0	0
DATA_110	0.24	0.43	0	0	0	0	1	DATA_49	0.12	0.32	0	0	0	0	1
DATA_111	0.29	0.45	0	0	0	1	1	DATA_410	0.26	0.44	0	0	0	1	1
DATA_112	0.25	0.43	0	0	0	0	1	DATA_411	0.18	0.39	0	0	0	0	1
DATA_113	0.32	0.47	0	0	0	1	1	DATA_412	0.24	0.42	0	0	0	0	1
DATA_114	0.35	0.48	0	0	0	1	1	DATA_413	0.25	0.43	0	0	0	1	1
DATA_115	0.28	0.45	0	0	0	1	1	DATA_414	0.30	0.46	0	0	0	1	1

Description of Binary Dataset

This table presents the statistical summary of the features taken for binary dataset on a bit-wise basis. Each bit can take a value of either 0 or 1. The mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile and maximum values of the features are as follows:

ID0 - ID16:

These columns represent the bit-wise representation of the ID. For example, for ID7, the mean of 0.43 says this bit is 1 43% of the time. Features such as ID14 with a mean of 0.57 show this bit is 1 more often.

DATA_00 - DATA_016:

These are the bitwise representations of the DATA bytes. For example, for DATA_09, the mean of 0.22 indicates this bit is 1 22% of the time.

For DATA_015, the mean of 0.45 indicates this bit is 1 almost 45% of the time.

Number of instances for each class present in the CICIoV2024 dataset.

Label	Category	Class	Count
Benign	–	–	1 223 737
Attack	DoS	–	74 663
	Spoofing	GAS	9991
		Steering Wheel	19 977
		Speed	24 951
		RPM	54 900

Benign Traffic consists of 1,223,737 examples of normal, non-malicious CAN bus traffic. The base data will train machine learning models and give an understanding of what normal communication in the vehicle should look like. It would then be used for comparison to attack traffic.

Attack Traffic: The main categories involved in the analysis are DoS (Denial-of-Service) and Spoofing.

DoS: 74,663 attacks; attackers flood the CAN bus with data-overloading the system-starving out other valid communications, some normal functions of the vehicle impaired.

Spoofing Attacks: Some attacks on various vehicle components can be effectuated as follows in different counts and implications:

GAS: 9,991 instances. Spoof gas pedal position data which may affect vehicle acceleration.

Steer wheel: 19,977. The possible attacks to the steering information might cause undesired outputs of the steering commands that could be of great concern regarding control performances.

Speed: 24,951. Attacks obscure the real speed of the vehicle; it could be dangerous regarding safety concerns

RPM: 54,900. Attacks to information on RPMs affect the performance of the engine and, finally, the whole vehicle.

Results obtained in the Machine Learning (ML) evaluation.

		LogisticRegression	AdaBoost	DeepNeuralNetwork	RandomForest
Binary	Accuracy	0.95	0.87	**0.95**	**0.95**
	Recall	**0.68**	0.17	**0.68**	**0.68**
	Precision	**0.74**	0.14	**0.74**	0.60
	F1-score	**0.63**	0.15	**0.63**	0.62
Decimal	Accuracy	0.89	0.92	**0.96**	**0.96**
	Recall	0.50	0.66	**0.76**	**0.76**
	Precision	0.48	0.48	**0.83**	0.76
	F1-score	0.49	0.51	**0.78**	0.76

Performance Metrics:

1. **Accuracy**: The proportion of correct predictions among the total number of cases. Higher accuracy indicates better model performance.
2. **Recall**: The ability of the model to identify all relevant instances (true positives). High recall is crucial when detecting attacks to minimize missed threats.
3. **Precision**: The ability of the model to return only relevant instances (true positives out of all positive predictions). High precision reduces false alarms.
4. **F1-score**: The harmonic mean of precision and recall. It provides a balance between the two and is useful when dealing with imbalanced datasets.

Binary Representation Results:

- **Logistic Regression**:
  - **Accuracy**: 0.95 (High accuracy)
  - **Recall**: 0.68 (Moderate recall, indicating some attacks might be missed)
  - **Precision**: 0.74 (Reasonable precision, but some false alarms)
  - **F1-score**: 0.63 (Moderate overall performance)
- **AdaBoost**:
  - **Accuracy**: 0.87 (Lower accuracy compared to other models)
  - **Recall**: 0.17 (Very low recall, missing most attacks)
  - **Precision**: 0.14 (Low precision, high false alarm rate)
  - **F1-score**: 0.15 (Poor overall performance)
- **Deep Neural Network**:
  - **Accuracy**: 0.95 (High accuracy, similar to Logistic Regression and Random Forest)
  - **Recall**: 0.68 (Moderate recall)
  - **Precision**: 0.74 (Good precision)
  - **F1-score**: 0.63 (Balanced performance, similar to Logistic Regression)
- **Random Forest**:
  - **Accuracy**: 0.95 (High accuracy)
  - **Recall**: 0.68 (Moderate recall)
  - **Precision**: 0.60 (Slightly lower precision compared to Logistic Regression and Deep Neural Network)
  - **F1-score**: 0.62 (Moderate performance)

Decimal Representation Results:

- **Logistic Regression**:
  - **Accuracy**: 0.89 (Lower than other models)
  - **Recall**: 0.50 (Moderate recall)
  - **Precision**: 0.48 (Low precision)
  - **F1-score**: 0.49 (Lower overall performance)
- **AdaBoost**:
  - **Accuracy**: 0.92 (Higher than in the binary case but still lower than DNN and Random Forest)
  - **Recall**: 0.66 (Improved recall)
  - **Precision**: 0.48 (Low precision)
  - **F1-score**: 0.51 (Slightly better performance compared to binary representation)
- **Deep Neural Network**:
  - **Accuracy**: 0.96 (Highest accuracy)
  - **Recall**: 0.76 (High recall, better at detecting attacks)
  - **Precision**: 0.83 (Very good precision)
  - **F1-score**: 0.78 (Best overall performance among all models)
- **Random Forest**:
  - **Accuracy**: 0.96 (Same as Deep Neural Network)
  - **Recall**: 0.76 (High recall)
  - **Precision**: 0.76 (Good precision, slightly lower than Deep Neural Network)
  - **F1-score**: 0.76 (Strong performance)

Key Takeaways:

1. **Binary Representation**:
   - **Deep Neural Network (DNN)** and **Logistic Regression** perform similarly, both achieving high accuracy but with moderate recall and precision.
   - **AdaBoost** struggles significantly, with low recall and precision, making it ineffective in detecting attacks.
2. **Decimal Representation**:
   - **Deep Neural Network (DNN)** outperforms all other models, with the highest accuracy, recall, precision, and F1-score. This indicates that DNN is the most effective model for this dataset when using the decimal representation.
   - **Random Forest** also performs well, close to DNN in all metrics.
   - **Logistic Regression** has lower performance compared to DNN and Random Forest.

3. **Overall**:

    - **Decimal representation** provides better performance metrics across all models compared to binary.
    - **Deep Neural Network** is the most effective model for detecting threats, particularly in the decimal representation.

These results highlight the importance of model selection and data representation when building systems to detect and mitigate attacks on intra-vehicular communication networks

Explanation of the Results and Challenges:

From the results, it is clear that the performance of machine learning models is excellent in the case of simpler classification tasks such as binary classification, reaching an almost perfect score with some techniques. It follows that the particularity of most attack traffic bears a rather unique pattern that can be easily detected by any model. With growing complexity in the classification task, such as distinguishing between six classes, including different spoofing types like speed, RPM, and manipulation of the steering wheel, the performance metrics are slightly lower. Deep Neural Networks and Random Forest head the pack in this multi-class setting; the higher accuracy and F1-scores of these methods hint at their strength in the balance of recall and precision due to effective attack detection by keeping the false positives low. While DNN and RF performed better in both binary and decimal representations compared to LR and AB, AB performed abnormally poor to identify attack traffic in a binary classification task, which indicates the limitation of AB on this problem. Looking at the confusion matrix, it is observed that overlapping patterns among some spoofing attacks, such as manipulation of a steering wheel, speed, and RPM, turn out to be challenging since in most of these classes, the shapes of traffic patterns are somehow similar, raising difficulties for all models to distinguish these types. However, all models have classified DoS attacks correctly with more unique traffic patterns. With the increase in the complexity, these approaches face classification performance decrease in recall, precision, and F1-score with higher details in classification. Besides, real IoV datasets are normally imbalanced, with the normal traffic dominating in real IoV scenarios. This makes it more difficult for the model to perform efficiently in the recognition of less frequent attacks, such as speed spoofing. In the future, attempts will be made toward addressing these challenges by considering methodologies that handle class imbalance using techniques like data augmentation, oversampling of attack classes, or specialized algorithms. In all, DNN and RF are the best models in IoV complex attack detection while AdaBoost is relatively unsuitable, especially for binary tasks. Therefore, in a real-world IoV security application scenario, the key issue of model performance improvement lies in the solution to the class imbalance and similarity among some attack patterns.

Improving Data representation:

Data representation greatly enhances the performance of an ML model in IoV, in such a complex and security-sensitive context. Feature engineering usually works by deriving meaningful features through aggregation, interaction, or domain-driven transformations-for example, extracting features that can reveal abnormal activities or attacks relies on expertise from the automotive domain. This may also include augmentation strategies, like the generation of synthetic samples to balance datasets using SMOTE or the simulation of realistic variations in IoV communication to foster model generalization. While dimensionality reduction techniques like PCA reduce data with a view to keeping the most important variance, t-SNE or UMAP visualize high-dimensional data in a more articulate way. Normalization and scaling of data, such as Z-score normalization or Min-Max scaling, allows features to have equal proportions-a factor quite necessary in sensitive algorithms. Feature encoding techniques, like one-hot or frequency encoding, present categorical data in a more interpretable format to the model. These will be very important in addressing data imbalance, where techniques of resampling or assigning class weights may be used to focus on minority classes. Rich representations include time-series analysis that will extract temporal patterns in CAN bus traffic and graph-based features that might make use of the structure of node relationships. Other pre-processing steps include the proper encoding of hexadecimal values and data smoothing that might further refine data quality. The combination of different data formats, in this case, binary and decimal, with ensemble methods allows one to take advantage of the different strengths of both. Sequencing models, such as LSTM or GRU, are able to exploit temporal dependencies. Feature lagging enables tracking previous trends. Feature selection techniques, such as RFE or mutual information, select the most contributing attributes and further reduce overfitting. Moreover, data groupings or anomalies that are detected by clustering algorithms like K-means or DBSCAN can support feature creation and preprocessing, hence more structured data for cybersecurity analysis.

OTHER STUDIES AND COMPARISONS

GGNB: Graph-Based Gaussian Naive Bayes Intrusion Detection System for CAN Bus

1. Datasets Used

RawCAN Dataset:

It contains some attack variants like DoS, fuzzy, spoofing, and replay attacks. Another category of study is the mixed attacks that combine these attack variants. Attacked and benign - non-malicious - CAN bus traffic is included in this dataset.

The OpelAstra Dataset consists of the following attack categories:

DoS, diagnostic, fuzzing CAN ID, fuzzing payload, replay, and suspension attacks. It also considers mixed attack scenarios.

2. Attack Types and Description

DoS (Denial of Service) Attack: Where high-priority messages are used to flood the CAN bus in order to block regular communications.

Fuzzy Attack: Where invalid or random CAN IDs and their payloads are sent onto the system.

Spoofing Attack: Through spoofing, it sends fake commands or data by impersonating any existing CAN message. Replay Attack: For the disturbance in the system, replay the valid messages of CAN that are recorded previously. Diagnostic Attack: Inject messages within a certain range of CAN IDs using diagnostic tools. Suspension Attack: Disrupt or suspend the transmission of messages in an ECU. Mixed Attacks: Those scenarios where several types of attacks may occur simultaneously. 3. Structure of Dataset and Representation as Graph

Representation of Node and Edge: Each CAN message in this dataset has been represented as a node in a graph, while the interlink between messages is in the form of edges. Graph-based modeling helped to make the CAN bus traffic organized and structured.

Features: The graph-based features include measures of number of nodes, number of edges, maximum in-degree, minimum in-degree, maximum out-degree, minimum out-degree, and PageRank values.

4. Feature Extraction and Usage of PageRank

Features of PageRank: PageRank is the importance score assigned by PageRank to each node in a graph. In case of an attack, several of these nodes exhibit large variation in PageRank values with respect to normal traffic. These anomalies form a critical role in the detection of attacks.

Feature Importance: For example, PageRank for a particular ID will be unusually high in the case of a DoS attack, making the detection of that attack easier. Other features are important for the identification of fuzzy and spoofing attacks.

5. Dataset Statistics

RawCAN Dataset: Total - 18.5K benign, 37K DoS, 38K fuzzy, 42K spoofing, 19K replay, and 80K mixed attack graphs were generated.

This labeled OpelAstra Dataset consists of 13.5K benign graphs, 17.6K DoS, 17.5K diagnostic, 17.5K fuzzing CAN ID, 17.5K fuzzing payload, 17.5K replay, 17.5K suspension, and 37.7K mixed attack graphs.

6. Splitting the Dataset

Training and Testing Split: The data is split into a ratio of 67% for training and 33% for testing. This ensures that the model has been tested on its generalization capability.

7. Feature Utilization on the Dataset

Feature Selection: Maximum in-degree and out-degree feature were the most important features of attack detection. Other important features include median and maximum PageRank values.

Feature Reduction: Reducing the number of features optimized the performance of the model. Even with just four salient features, the model produced accuracy with very minimal degradation.

8. Data Processing for Efficiency

Graph Construction: Graph nodes represent CAN message IDs, while edges are built according to the sequence of messages. In anomaly detection, features such as nodes and edges will be obtained with a degree metric. The window size used in this analysis is ~23ms, which keeps one sample at 1Mbit/s CAN speed.

Algorithm 1 PageRank-related features extraction algorithm

Input: G, collection of graphs;
Output: The list of minimum, median, and maximum PR of each graph in G;

$graphList = []$ ▷ initializing the list of the PR of every vertex from each graph
$graphListWithMedianPR = []$ ▷ initializing the list of the median PR from each graph
$graphListWithMaximumPR = []$ ▷ initializing the list of the maximum PR from each graph
$graphListWithMinimumPR = []$ ▷ initializing the list of the minimum PR from each graph
for all $g_i \in G$: **do**
    $graphList[i] = PR(g_i)$ ▷ $PR(g_i)$ computes the PR of every vertex of g_i
    $graphListWithMedianPR[i] = median(graphList[g_i])$
    $graphListWithMaximumPR[i] = maximum(graphList[g_i])$
    $graphListWithMinimumPR[i] = minimum(graphList[g_i])$
end for

The GGNB study proposes a different approach that fuses graph-based methods with the GNB classifier for intrusion detection in CAN bus communications. In summary, GGNB represents CAN bus data as a graph where each message ID is a node, and messages received consecutively create edges between these nodes. Graph representation means the system can pick up important features indicative of anomalies, including PageRank values and in-degree or out-degree metrics. PageRank is an algorithm used by Google to rank web pages; here it is applied to give a ranking of the importance of nodes in the CAN bus graph. These scores are quite stable during normal conditions; however, in case of an attack-for instance, DoS-much variation may happen in the scores of some nodes, which can show anomaly. Gaussian Naive Bayes Classifier: It is a probabilistic model based on Bayes' Theorem; under the assumption of independence, all features become conditionally independent given the class label.

It assumes that the derived features from the graph are usually normally distributed, which is one of the crucial properties of the continuous data generated for PageRank scores and the degree metrics. GGNB computes a number of features from the graph: the number of nodes and edges in the graph, the maximum and minimum values in in and out-degree, and various statistical summaries of the PageRank scores, such as maximum, median, minimum. This places GGNB in a niche where the integration of graph theory with probabilistic classification makes it simultaneously efficient and effective in anomaly detection. GGNB differs from the classic machine learning models, such as SVM, Random Forest, and deep neural networks, as it detects anomalies in a graph-based manner.

Unlike these traditional models, which treat data as independent, flat feature vectors, GGNB captures the relationships and interactions between CAN messages, offering a holistic view of network behavior.

Traditional models typically work with tabular data and may not be effective at capturing the structural relationships that exist in CAN bus communications. Moreover, another novelty is the application of PageRank and graph-based features in GGNB. While most of the traditional algorithms rely on statistical properties or raw data representations, GGNB is able to highlight critical nodes whose behavior changes during an attack. The other fundamental difference of GGNB with respect to existing solutions concerns its efficiency in computational terms.

Regarding time and space complexity, GGNB is quite efficient-the training and testing times are drastically reduced compared to other models, such as SVM. GGNB is further optimized with regard to efficiency for hardware implementation on FPGA and hence suitable for automotive networks. In contrast, most of these models, especially those based on deep neural networks, need expensive computational resources, and turning toward real-time detection may be difficult in resource-constrained environments. GGNB has several advantages over other traditional models.

Algorithm 2 Anomaly Detection Algorithm

Input: Raw CAN data, WindowSize;
Output: Prediction; ▷ Attacked if Prediction is correct else benign

$G = CreateGraph(RawCANdata, WindowSize)$
$graphListWithMaximumIndegree = []$ ▷ initializing the list of maximum in-degree from each graph
$graphListWithMaximumOutdegree = []$ ▷ initializing the list of maximum out-degree from each graph
$graphListWithMinimumIndegree = []$ ▷ initializing the list of minimum in-degree from each graph
$graphListWithMinimumOutdegree = []$ ▷ initializing the list of minimum out-degree from each graph
$graphListWithNodesNumber = []$ ▷ initializing the list of number of nodes from each graph
$graphListWithEdgesNumber = []$ ▷ initializing the list of number of edges from each graph
for all $g_i \in G$: **do**
    $graphListWithNodesNumber[i] = numberOfNodes(g_i)$
    $graphListWithEdgesNumber[i] = numberOfEdges(g_i)$
    $graphListWithMaximumIndegree[i] = maximumIndegree(g_i)$
    $graphListWithMaximumOutdegree[i] = maximumOutdegree(g_i)$
    $graphListWithMinimumIndegree[i] = minimumIndegree(g_i)$
    $graphListWithMinimumOutdegree[i] = minimumOutdegree(g_i)$
end for
$\{PageRankFeaturesList\} = GraphListPageRank(G)$ ▷ extract PR-related feature using Algorithm 1
$\{data, labels\} = processData()$
$train, test, trainLabels, testLabels = trainTestSplit(data, labels)$ ▷ splitting the data and labels for training and testing
$model = gaussianNb(train, trainLabels)$ ▷ training data
$predictions = model.predict(test)$ ▷ making the predictions

It allows faster training and testing-up to 239× in training and 135× in testing time compared to SVM, which becomes crucial in real-time vehicular systems. It also uses much fewer hardware resources for its FPGA implementation, hence it will be appropriate for embedded systems and IoV environments. Besides, GGNB was designed for generalized attack detection and could handle a broad range of attacks, including mixed attack scenarios, without modification of the protocol. This generalization is something most of the traditional models often lack; they usually are optimized for certain types of attacks. Second, it utilizes a correlation matrix for feature reduction in GGNB. This, in turn, simplifies the model to reduce computational complexity while retaining high accuracy. In comparison, most of the traditional models, especially deep learning methods, may involve complex and high-dimensional feature sets that are more difficult to interpret. It follows that graph theory in work enables GGNB to capture relationships in structure that traditional methods may miss.

First, the employment of PageRank creates a new mechanism for anomaly detection, while its efficient hardware implementation makes GGNB suitable for real-time and embedded environments. In contrast with models that have expertise in detecting certain kinds of attack, GGNB represents a uniform, versatile approach to handling different and even mixed attack situations. The main contribution of this paper consists in the fact that GGNB represents one jump concerning CAN bus security, being highly accurate while maintaining computational efficiency, suitable for real-time intrusion detection in automotive networks.

Comparative Analysis: CICIoV2024 vs. GGNB Research Papers

Both studies focus on enhancing the security of Controller Area Network (CAN) by detecting various attack types. However, they approach the problem using distinct data processing and machine learning techniques. Here is a detailed comparison:

1. Core Approaches and Innovations

- **CICIoV2024 Paper**: This study uses several conventional and deep learning models to detect attacks like DoS and spoofing on a CAN bus. The CICIoV2024 dataset was collected from a real vehicle (2019 Ford) and includes binary and decimal representations of data to evaluate model performance.

- **GGNB Paper**: This research introduces a novel **Graph-based Gaussian Naive Bayes (GGNB)** algorithm that leverages graph properties and PageRank features to detect CAN bus attacks. The GGNB method demonstrates significant improvements in efficiency and performance, especially when implemented on hardware using Xilinx Zybo Z7 FPGA.

2. Machine Learning Models

- **CICIoV2024**:
  - Classical Models: Logistic Regression, Random Forest, AdaBoost.
  - Deep Learning: Deep Neural Networks (DNN).
  - Performance: DNN shows the best results in terms of accuracy and F1 score, while AdaBoost performs poorly.
- **GGNB**:
  - Uses a unique approach with **Graph-based Gaussian Naive Bayes (GGNB)**.
  - GGNB efficiently handles DoS, fuzzy, spoofing, replay, and mixed attacks, using graph-based features combined with Gaussian Naive Bayes.
  - The method significantly outperforms SVM models in terms of speed and resource efficiency.

3. Data Processing and Representation

- **CICIoV2024**: Data is represented in both binary and decimal formats, affecting model performance. Decimal representation generally provides better results in this context.
- **GGNB**: Utilizes a graph-based approach, where CAN bus messages are represented as graph nodes and edges. PageRank values help identify anomalies, making this a more structured and theoretically robust approach.

4. Performance Comparison

- **CICIoV2024**:
  - Best results from Deep Neural Networks: 96% accuracy with high F1 scores.
  - AdaBoost has the lowest performance, indicating inefficacy for this problem.
  - Decimal representation yields better outcomes compared to binary representation.
- **GGNB**:
  - Achieves up to 99% accuracy for various attack types, including mixed attacks.
  - The GGNB method is 239× faster in training and 135× faster in testing compared to SVM.
  - Efficiently implemented on FPGA, using significantly fewer hardware resources.

5. Hardware Implementation

- **CICIoV2024**: Focuses on software-based model performance without hardware optimizations.

- **GGNB**: Optimized for hardware using Xilinx Zybo Z7 FPGA, resulting in reduced resource usage (slices, LUTs, flip-flops, DSP units). This makes GGNB highly efficient for real-time applications.

6. Innovations and Differences

- **GGNB Innovations**:
  - Integrates graph theory and PageRank with Gaussian Naive Bayes for anomaly detection.
  - Efficient hardware implementation, crucial for real-time vehicular networks.
  - A generalized approach capable of detecting multiple attack types in a unified model.
- **CICIoV2024 Contributions**:
  - Expands IoV security research with a comprehensive dataset.
  - Evaluates a variety of machine learning models, providing insights into their effectiveness.
  - Emphasizes the impact of data representation on model performance.

7. Which Approach Is Better?

- **Performance**: GGNB is superior in handling mixed attacks and is much faster in both training and testing phases. It also uses fewer computational resources, making it suitable for real-world applications.
- **Applicability**: CICIoV2024 is more suitable for academic research due to its extensive dataset and performance comparisons. GGNB, on the other hand, is more practical for deployment in real-time environments.
- **Innovation**: GGNB stands out for its use of graph-based features and hardware optimization, offering significant advancements in efficiency.

GGNB provides a highly efficient, hardware-optimized solution for detecting a wide range of attacks, making it well-suited for real-time IoV systems. Its integration of graph properties with Gaussian Naive Bayes and FPGA implementation are groundbreaking.
CICIoV2024 offers valuable insights into model performance across various machine learning algorithms and emphasizes the role of data representation. It is an excellent resource for understanding how different approaches perform on vehicular network data.
Both studies make significant contributions to IoV security, but **GGNB** is more innovative and practical for high-performance, real-world scenarios.

Identify a Spoofing Attack on an In-Vehicle CAN Bus Based on the Deep Features of an ECU Fingerprint Signal

The three studies of Identify a Spoofing Attack on an In-Vehicle CAN Bus Based on the Deep Features of an ECU Fingerprint Signal, CICIoV2024 research, and the GGNB study present different yet very innovative solutions for securing Controller Area Network communications in intelligent vehicles. Each of these studies uses different methodologies for improving the security of the CAN bus by studying detection of the attacks but from different perspectives, such as one based on data representation, one on machine learning techniques, and finally one focused on computational efficiency.

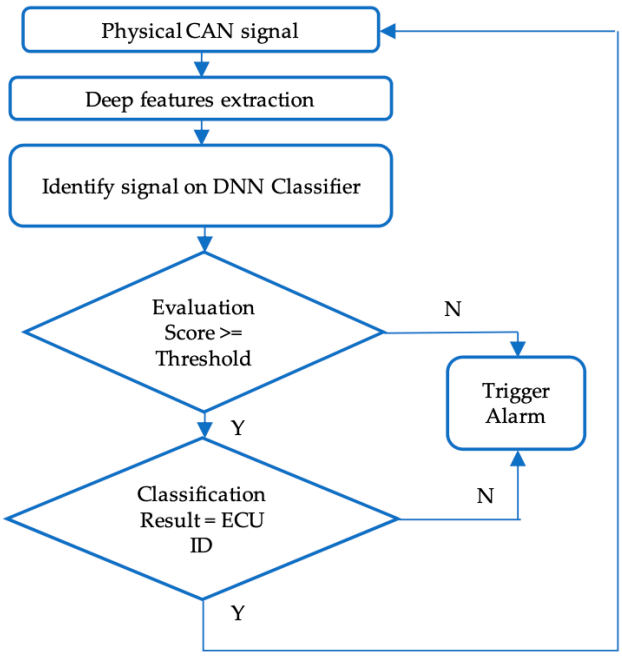

Figure 10. The simulation data based on the theoretical model of the CAN bus physical layer.

ECU Fingerprint Signal Study: A deep discussion about using analog signal features in authenticating and securing CAN bus communications. It is innovate-the research here presented has identified spoofing attacks with the use of a RNN-LSTM model, which grasps the signals of analog fingerprints of Electronic Control Units. This is a novelty, since it leverages unique time-domain and frequency-domain features for every ECU's transmission, that are hardly replicable by any attacker. For this study, the dataset is generated by the simulated ECU signals with Multisim-the

powerful simulation tool, modeling physical properties of the CAN bus and ECU transceivers by creating a rich set of signals with changing noise conditions to approximate how it will be in real life.

Table 1. The testing result of simulation data based on the CAN bus.

Machine Learning Model	Features	Average Accuracy (%)
BDT	Time (6 features), frequency (5 features)	92.00
NN	Time (6 features), frequency (5 features)	95.20
SVM (Linear kernel)	Time (6 features), frequency (5 features)	96.00
SVM (RBF kernel)	Time (6 features), frequency (5 features)	96.00
RNN-LSTM	deep features	98.76

Since temporal information could be buried in time series data, this paper chooses the RNN-LSTM algorithm because it has the potential for learning from time series data; it uses internal state to store important temporal information. Each LSTM unit consists of three kinds of gates-input, forget, and output-which enable the model to remember long-term dependencies. These are essential for a correct differentiation between real and spoofed ECU signals. With the goal of enhancing performance even further, the RNN-LSTM model is accelerated using FPGA technology. This hardware implementation overcomes the computational complexity of this model, which could be employed in real time. The original computationally expensive activation functions, like sigmoid and tanh, are replaced with their approximated variants optimized for execution on the FPGA, namely hard_sigmoid and hard_tanh. Further, this optimization allows for a better-executable version of the required matrix multiplications combined with nonlinear activations, yielding at least one order of magnitude speedups compared to pure software execution.

Experimentally, the ECU Fingerprint Signal Study shows excellent performance:. On the other side, the RNN-LSTM model detects tachycardia with a high accuracy of 98.76%, which is far away from traditional machine learning classifiers like BDT, NN, and SVM. In this study, six time-domain features were compared against five frequency-domain features such as minimum, maximum, mean, variance, skewness, and kurtosis combined with spectral skewness and spectral kurtosis. The RNN-LSTM model outperforms the classical methods with a clear margin, due to its deeper learning of temporal features. Besides, acceleration by FPGA increases the speed of the model tenfold compared to the same algorithm running on a more classic ARM Cortex-A9 processor, which underlines the suitability of the proposed model for real-time CAN bus security applications.

On the other hand, CICIoV2024 adopts a more traditional solution by testing various machine learning models on actual CAN bus data from a 2019 Ford vehicle. In fact, the dataset includes both binary and decimal versions of CAN messages, all examined for the evaluation of models like Logistic Regression, Random Forest, AdaBoost, and Deep Neural Networks-DNN. It underlined that the impact of data representation is confirmed in the CICIoV2024, generally showing better results when the data is decimal rather than binary. Among the models tested, DNNs are the best performers, capable of reaching an accuracy as high as ~96%, while the worst performing is AdaBoost. This is a good study, bringing insight into the strengths and limitations of different algorithms; hence, it is a comprehensive contribution to understanding various machine learning approaches against CAN bus attack detection. However, it doesn't include real-time efficiency and hardware optimization like ECU Fingerprint Signal Study does.

GGNB presents a graph-theoretic approach somewhat different from either of the other two studies. GGNB models CAN bus data as a graph where message IDs are nodes and sequential messages form edges; hence, for anomaly detection, graph properties and PageRank features will be exploited. The technique is especially powerful, as it catches structural relationships between the CAN messages-that is, something that the traditional models, such as SVMs and Random Forests, fail to do. PageRank, originally developed for the ranking of web pages, assigns a score of importance to nodes in graphs. In this case, this helps in the identification of message behavior deviations indicative of attacks. These are then classified using the Gaussian Naive Bayes classifier, considering data to be drawn from a normal distribution. GGNB happens to be extremely effective, with an accuracy as high as 99% regarding the attacks such as DoS, fuzzy, spoofing, replay, and mixed scenarios. Besides that, the FPGA implementation of GGNB is as efficient as up to 239× faster training and 135× faster testing as compared to SVM and hence suitable for real-time applications.

In this respect, the work is striking with hardware optimization-the resource usage on an FPGA is as minimal as possible, using much fewer slices, LUTs, and flip-flops when compared to classical neural networks. This makes the GGNB even more compatible with IoV applications where computational resources are at a premium. Unlike CICIoV2024, which is essentially a model evaluation using just software, GGNB blends well into

embedded systems, thus offering a practical and effective approach to intrusion detection within automotive networks.

Table 3. The field-programmable gate arrays' (FPGA) resource utilization of the RNN acceleration model.

Resource	Utilization	Available	Utilization (%)
LUT	8777	17,600	49.87
LUTRAM	601	6000	10.02
FF	9601	35,200	25.74
BRAM	9	60	15.00
DSP	25	80	31.25
BUFG	1	32	3.13

Comparing these three studies, there are differences that can be observed. Indeed, the ECU Fingerprint Signal Study goes a step further in detecting spoofing attacks by a novel use of deep learning on analog signal features. Since its reliance is on physical signal characteristics, it's pretty hard for attackers to replicate, and real-time performance is guaranteed by FPGA acceleration. However, the particular study of this research is for spoofing detection; it is probably not as generalizable compared to GGNB. Although CICIoV2024 is an excellent academic resource to understand how various algorithms perform on CAN bus data, with its diverse model evaluation, it lacks efficiency to be deployed in real-time systems. GGNB offered a comprehensive solution to the detection of multiple types of attacks due to its graph-based methodology combined with optimization on FPGA. Indeed, this has made it stand out.

Conclusion, each approach has its own strengths. The ECU Fingerprint Signal Study is a seminal effort toward leveraging deep learning from physical layer signals and achieving the spoofing detection goal with both high accuracy and real-time efficiency. CICIoV2024 provides a broad analysis of model performance; thus, it is more suitable for theoretical exploration than practical and real-world, real-time applications. GGNB represents an innovative application of graph theory and realizes its efficient hardware implementation, making it a versatile and practical solution for comprehensive CAN bus security. Which approach to apply would, therefore, depend on the immediate needs of any given application, be it a thorough analysis of model performance, specialized spoofing detection, or a highly generalized attack detection system that is efficient in real time.