Models, Ensemble! Assessing Leader Class and Confidence Decision Ensemble with AdaBoost on ROAD Dataset

Bella White
Computer Science
Cal Poly SLO
San Luis Obispo, CA, United States
bwhite17@calpoly.edu

Andrew Cheung
Computer Science
Cal Poly SLO
San Luis Obispo, CA, United States
acheun29@calpoly.edu

Noa Kehle
Computer Science
Cal Poly SLO
San Luis Obispo, CA, United States
nkehle@calpoly.edu

***Abstract*—As Internet of Vehicles (IoV) become more widely adopted and Vehicle to Everything (V2X) communications increase, detecting malicious network traffic inside a vehicle and between vehicles is more important than ever. Ensembles of Gradient Boosting Decision Trees (GBDTs) have shown great potential for IDS in IoV contexts. This paper extends the work of Yang et. al and evaluates their proposed Intrusion Detection System (IDS) framework, Leader Class and Confidence Decision Ensemble (LCCDE). Our paper first successfully reproduces their work on the Car-Hacking dataset with negligible performance degradation, except for detecting Denial of Service attacks. We then evaluated LCCDE on a newer dataset, Real ORNL Automotive Dynamometer (ROAD) CAN Intrusion Dataset, and found that while LCCDE was not able to classify Masquerade or Fabrication attacks effectively. The overall LCCDE ensemble outperformed any individual base model, illustrating how the LCCDE model allows for more flexibility and robustness in the model to handle complex data. This paper also evaluated AdaBoost as a potential fourth base model and found AdaBoost does not perform well on the classification task for Car-Hacking or ROAD.**

***Index Terms*—Internet of Vehicles, Intrusion Detection, LCCDE, Ensemble Model**

I. Introduction

A. Background

The Internet of Vehicles (IoV) is a distributed network that transmits data between vehicles and between systems inside of vehicles. There are two main types of IoV communications: intra-vehicle and inter-vehicle communications [12]. Intra-vehicle communications occur between sensors and Electronic Control Units (ECUs). ECUs ingest data from sensors or other ECUs, process the data, and use actuators to control vehicle behavior. The CAN bus is responsible for carrying messages between the ECUs and is the most common network protocol in vehicles. Unfortunately, the CAN bus performs no authentication or encryption of the messages it carries, so it is known to be susceptible to attacks such as DoS, Fuzzing, and Spoofing [4]. Inter-vehicle communications, also known as Vehicle to Everything (V2X), are between vehicles and other vehicles, infrastructure, or pedestrians. The goal of inter-vehicle communication is to provide a safe and efficient system of transportation. Security vulnerabilities in V2X include data falsification, impersonation, and man-in-the-middle attacks [12]. With lives at stake, it is crucial to quickly and accurately detect attacks on IoV network systems. Research shows that machine learning can be an effective method for intrusion detection in vehicles [1].

B. Current and Related Work

One of the earliest publications using tree based ensemble models for IDS proposed applying an ensemble of Bayesian Networks and Classification and Regression Trees for IDS [2]. The 2005 paper showed that ensemble models were promising, but more research was required in developing better base models. More recently, Zhou et. al. effectively combined feature learning and a tree based (Random Forest, ForestPA) ensemble model system [17] on the NSL-KDD [13], AWID [7], and CICIDS2017 dataset. This paper demonstrated the effectiveness of tree-based ensemble models for network intrusion.

Elmasry et. al. utilized Particle Swarm Optimization for feature selection and a deep learning ensemble model composed of DNNs, RNNs and DBNs for IDS [5]. This paper illustrates the effectiveness of deep learning ensembles in network intrusion detection, however it is unclear if IoV systems can dedicate the computational resources required to use these models. This would be an interesting avenue for future work, however it is out of the scope of this paper.

Our work is a continuation on the paper *LCCDE: A Decision-Based Ensemble Framework for Intrusion Detection in The Internet of Vehicles* [16]. LCCDE, Leader Class and Confidence Decision Ensemble, is a ML model designed to identify intrusion attacks in IoV systems. The original ensemble model consists of three gradient boosting decision tree (GBDT) models, CatBoost, LightGBM, and XGBoost. GBDTs lend themselves to the task of intrusion detection as they are robust, can generate feature importance scores, require low computation complexity, and are typically generalizable [16]. Using ensemble models based on tree structures for IDS is not a novel idea, however their application to IoV network

intrusion detection is still emerging. The system was evaluated on the CICIDS2017 [11] and Car-Hacking [10] datasets, with the model performing extremely well on both datasets; its F1 scores were over 99% for nearly every class.

C. Motivation

As IoV systems are becoming more widespread, powerful IDSs are necessary to keep passengers safe from attackers. The LCCDE algorithm proposed by Yang et al. is robust, using multiple ML models to classify CAN attacks that may evade an individual model, and we wanted to evaluate its performance with a newer dataset that captures the modern threat landscape, which may have changed since the time of LCCDE's publication due to concept drift. In addition, we wanted to analyze the benefits of adding a fourth base learner, AdaBoost, to the ensemble. Although AdaBoost is an older algorithm, we wanted to find out if a simpler model could improve performance.

D. Contributions

We contribute the following to the scientific body of work:

- Reproduced results for LCCDE performance on Car-Hacking.
- Adding AdaBoost as a base learner does not improve predictive performance of LCCDE on Car-Hacking or ROAD.
- The LCCDE system has poor predictive performance when trained on the ROAD dataset.
- For a binary Malicious vs Benign sample problem, LCCDE trained on Car-Hacking cannot generalize to ROAD.
- Preprocessing code for ROAD and Car-Hacking datasets.
- Developed a more dynamic LCCDE implementation to accommodate the addition and removal of base learners.

II. Methods

A. Datasets

1) Car-Hacking: We first attempted to replicate Yang, et al.'s LCCDE results with the Car-Hacking dataset [10]. Published in 2018, the Car-Hacking dataset consists of five classes:

- Denial of Service (DoS): Flooding the network with malicious traffic to the point of being unusable.
- Fuzzy: Sending random and poorly formatted data to mess with the system.
- Spoofing the Drive Gear: An attempt to send a fake gear position to the system.
- Spoofing the RPM Gauge: An attempt to send a fake RPM reading to the system.
- Benign: Normal and non-malicious packets.

We ran various Python scripts on the original set of five csv files to get it into a more usable form, and then combined it into a single csv file that labeled the specific attack types. We had a strange issue with an NA value in Car-Hacking, but since it was a single value out of over 17 million, we decided to drop it and move on.

2) ROAD: We continued the work on LCCDE by evaluating its performance on the Real ORNL Automotive Dynamometer (ROAD) CAN Intrusion Dataset. Published in 2024, the ROAD dataset contributes five classes:

- Accelerator: An exploit that puts the vehicle in state where the driver has difficulty controlling the vehicle. Since this vulnerability is so dangerous, the ROAD authors only included CAN captures from after the attack was executed.
- Fabrication: False data being injected into the car for malicious purposes. Flam delivery was used to make the injections difficult to detect.
- Fuzzing: Sending random and poorly formatted data to mess with a system.
- Masquerade: The attacker impersonates another user or device to gain certain privileges and access. Due to technical limitations, this attack could not be carried out at the time of ROAD's publication. Instead, the masquerade attack was simulated through post-processing of the Fabrication captures.
- Benign: Normal and non-malicious packets.

The ROAD dataset is more modern in that it has these advanced accelerator and masquerade attacks, which are more subtle than previous CAN Intrusion datasets [14]. In addition, the researchers used a dynamometer to verify that their attacks had a physical effect on the vehicle. The ROAD dataset included CAN captures for 33 attacks (over about 30 minutes) and 12 ambient captures (over about 3 hours). However, the attack packets were not labeled as Malicious or Benign, but metadata was provided for users to label the data. We used the injection ID, injection data, and time intervals in the metadata to label all of the attacks, and published our code for processing on GitHub [1].

B. Training Methodology

We used most of the same training methodology as the original Yang, et al. paper [16]. We used the *sklearn* Python library to evaluate model performance and *SMOTE* to mitigate class imbalance. The primary difference was in the train/test split: Yang, et al. were able to take an 80/20 train/test split because they used samples of their Car-Hacking and CICIDS2017 datasets; we did not take samples, but instead used a smaller test size and larger training size. We split the data using stratification, taking 4% as the test size due to resource limitations. For Car-Hacking, we used 50% of the data for training and 2% for testing. For ROAD, we used 96% for training and 4% for testing. For extending Car-Hacking to ROAD (when we trained LCCDE on Car-Hacking and tested it on ROAD), we used 100% of Car-Hacking for training and 4% of ROAD for testing.

For the features, we used the eight bytes in a CAN bus packet. Since the timestamps were obfuscated, we did not use them as a feature. Unfortunately, due to our lack of familiarity with the IoV domain, we did not use the CAN ID as a feature,

[1] Our code is available at: https://github.com/bronte999/CSC429_LCCDE

which may have affected the base learners' performance on some classes.

C. Base Learner Models

The base learners in the original LCCDE paper were Gradient Boosting Decision Trees (GBDTs), specifically XGBoost, CatBoost, and LightGBM. GBDTs are ensemble models of decision trees that are iteratively improved upon based on the errors of the previous decision tree. In this way, GBDTs are able to create strong predictors from a group of weak learners with low computational resource requirements, making them a powerful tool in applications like IoV intrusion detection. GBDT libraries have different memory performances, regularization, and tree splitting algorithms that provide different benefits depending on the dataset and specific application environment.

Our paper adds a fourth base learner, AdaBoost, to compare its performance to more modern advanced GBDTs.

1) XGBoost: XGBoost is a distributed gradient tree boosting library that is often used in data science competitions due to its speed, flexibility, and memory efficiency [3]. XGBoost is well suited for training on large, sparse datasets on resource constrained devices, making it an effective base learner for an IoV ensemble model for intrusion detection.

2) CatBoost: CatBoost is an effective gradient tree boosting library that uses ordered boosting, prevents prediction shift, and is designed to handle categorical data efficiently [8]. Its performance rivals XGBoost and has been increasingly used for data science challenges due to its predictive performance and handling of categorical data. However, neither the Car-Hacking nor ROAD datasets used in this paper contain categorical values and CatBoost took much longer to train than the other base learners as shown in Figure XI.

3) LightGBM: LightGBM is a tree-based gradient boosting library that is designed to handle large sparse datasets with efficient memory usage, training speed, and high accuracy [6]. LightGBM uses histogram-based algorithms to reduce cost of tree splits and storage required and is generally very fast when training on large datasets. It is worth noting that this algorithm has a lower communication cost for distributed learning than algorithms like XGBoost that use pre-sorting algorithms for splits. Distributed learning can be used in IoV networks with each car being a node, so LightGBM is a base learner worth evaluating.

4) AdaBoost: Adaptive Boosting (AdaBoost) is a boosting ensemble model that starts with a weak learner, like a simple decision tree, and continuously improves upon its prediction performance by weighting misclassified samples as more important when training the next decision tree[9]. AdaBoost does not use gradient boosting, instead the final model is a weighted sum of its weak learners. This algorithm has been used since its introduction in 1997 for various classification problems and is not used as often anymore with the introduction of the base learners listed above. We included this model as a fourth base learner in the LCCDE system to determine whether a simpler model could perform well enough to contribute to this ensemble of more modern GBDTs.

D. LCCDE Implementation

The Yang, et al. paper contributed a Jupyter notebook that implemented LCCDE with LightGBM, XGBoost, and CatBoost as the base learners [2]. The notebook was set up to evaluate the ensemble's performance on the CICIDS2017 dataset. To make it easier for ourselves and others to work with LCCDE, we improved the original code in the following ways [1]:

1) We implemented the new *LCCDE_Model* class, which stores the model object (e.g. XGBClassifier()) and its evaluation metrics. We used this class to reduce code duplication in multiple places, such as the sections where the evaluation metrics were printed for each of the base learners.
2) The original LCCDE() function, which implements the ensemble prediction mechanism shown in [16], hard-coded the three base learners as the *m1*, *m2*, and *m3* parameters. We rewrote the function to take a list of *LCCDE_Model* objects instead, which makes it easier to add or remove models from the ensemble. In addition, we improved the Python implementation of the LCCDE logic, and wrote unit tests for this implementation.
3) We added global settings at the top of the notebook so that the user can easily select which dataset and which base models to use. The settings support the Car-Hacking and ROAD datasets; and the LightGBM, XGBoost, CatBoost, and AdaBoost base learners.

E. System Model

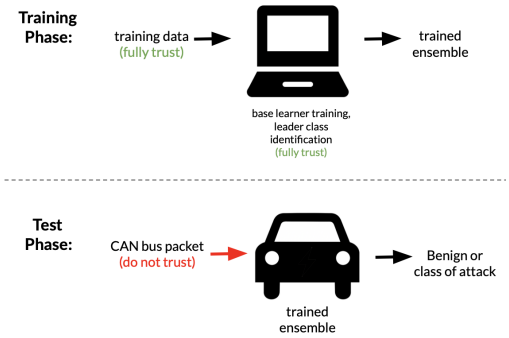

Fig. 1. System Model

Our work evaluates LCCDE's performance for detecting malicious packets in the CAN bus, using the Car-Hacking and ROAD datasets (Fig. 1). The system can be described as follows:

1) **System Model**: In the training phase, the LCCDE model is trained to classify CAN captures. As described in [16],

[2]Yang, et al.'s original code, which we adapted for this paper, is available at: https://github.com/Western-OC2-Lab/Intrusion-Detection-System-Using-Machine-Learning

this involves training the base learners in the ensemble and identifying the leading model for each class. In the test phase, the ensemble classifies CAN bus packets as Benign or a type of attack.

2) **Attacker**: The attacker's goal is to disrupt the vehicle's normal functions. The attacker is capable of injecting packets into the CAN bus, but the IDS is a black box (the attacker has no information about the training data or the IDS architecture).
3) **Trust Model**: In the training phase, the training data and compute are fully trusted. In the test phase, the CAN bus packets are not trusted.
4) **Threat Model**: We use LCCDE to detect intrusions in the intra-vehicle network through the CAN bus. Other attacks, such as data poisoning and membership inference, are not considered.

III. Simulation Results/Security Analysis

Note that *Reproduced LCCDE* refers to our simulation of LCCDE with Car-Hacking, using the LCCDE code from [16] after we improved the code, with the LightGBM, XGBoost, and CatBoost base learners. *LCCDE without AdaBoost* uses those same base learners, but on different datasets. *LCCDE with AdaBoost* refers to the same LCCDE setup, except with AdaBoost added on as a fourth base learner.

A. Reproducing LCCDE on Car-Hacking

We were able to reproduce the results of the original Yang et. al paper on the Car-Hacking dataset within a 2-3% margin for overall accuracy, precision, recall, and F1 score, as shown in Table II (note that the class numbers here start at 0 to match our code, while the class numbers in the original paper start at 1). There are two main reasons for our results 2-3% lower across all performance metrics. The first being that we could not run their code exactly as they did, as they did not provide their code for Car-Hacking or their sampling distribution for their training datasets. The other reason being that we were not able to reproduce their prediction performance for each class, specifically for Denial of Service attacks. We were only able to get a 66.04 F1 (%) score for DoS while the original LCCDE paper achieved 100% as shown in Table III. We believe this disparity may have occurred because Yang, et al. used CAN ID as a feature, but we did not.

It is also worth noting that our LCCDE selected XGBoost as the leader model for each attack while the original paper had a mixture of XGBoost and LightGBM with LightGBM being the leader model the majority of the time. The performances of the original three base models in the paper were so close that ties were broken by 0.005 differences in F1% scores, which we found to be true during our experimental runs as well. Due to rounding errors or a difference in training sets XGBoost was selected over the other two base models. It could be argued that for Car-Hacking, LCCDE may be "overkill" and only one base model is needed for accurate classification.

TABLE I
Methods for Reproducing LCCDE on Car-Hacking

Class	Total Samples	Training Samples	Testing Samples	Leader
0: Normal	15226820	7482710	299309	XGBoost
1: DoS	587521	293760	11750	XGBoost
2: Fuzzy	491847	245924	9837	XGBoost
3: Gear Spoofing	597252	298626	11945	XGBoost
4: RPM Spoofing	654897	327449	13098	XGBoost

TABLE II
LCCDE Performance Comparison on Car-Hacking

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
LCCDE [16]	99.9997	99.9997	99.9997	99.9997
Reproduced LCCDE	96.5066	98.2774	96.5066	97.0637
LCCDE with AdaBoost	96.5069	98.2777	96.5069	97.0639

B. Evaluating LCCDE on ROAD

LCCDE had a very mixed performance on the ROAD dataset. As shown in Table VI, LCCDE performed had a F1 score of 100% for Fuzzing attacks and 93% on identifying Benign samples. Prediction performance was poor for Accelerator, Fabrication, and Masquerade attacks, with the latter two having F1 scores of around 41%.

While ROAD is still a very new dataset and there aren't many papers to compare our results to, we found that Wang et. al applied a deep learning approach to classifying the Masquerade attacks in the ROAD dataset with similarly poor results [15]. The poor performance of more powerful models on the ROAD dataset suggests that LCCDE's poor performance is not an indication of the failure of the LCCDE approach to other network intrusion problems, but a testament to the difficulty of the ROAD masquerade classification problem.

CatBoost and LightGBM were chosen as leader models, outperforming XGBoost and AdaBoost. For ROAD, the LCCDE ensemble had a higher F1 score than the F1 any individual model as shown in Table VI, unlike Car-Hacking where all the leader models were XGBoost. The ensemble of leader models allows LCCDE to be more flexible and robust for complex datasets like ROAD.

AdaBoost was not selected to as a leader model due to its poor predictive performance as shown in Table VIII. However the inclusion of AdaBoost in did affect the performance of the LCCDE as shown in Table VII. For Masquerade attacks the F1 score without AdaBoost was 43.784% and dropped to 41.0802% when AdaBoost was included as a base learner. Interestingly the F1 score for Fabrication increased from 39.899% to 42.57% when AdaBoost was included as a base model despite AdaBoost having an F1 score of 39% for Fabrication. AdaBoost had the highest precision out of all the base learners for Fabrication attacks, so when it came to splitting ties, AdaBoost was able to correctly break the tie.

TABLE III
LCCDE PERFORMANCE COMPARISON ON CAR-HACKING BY CLASS

Class	LCCDE F1 (%) [16]	Reproduced LCCDE F1 (%)	LCCDE with AdaBoost F1 (%)	AdaBoost F1 (%)
0: Normal	99.9998	97.9396	97.9398	68.3028
1: DoS	100.0	**66.0428**	**66.0428**	**0.0**
2: Fuzzy	99.995	99.9898	99.9949	**11.4966**
3: Gear Spoofing	100.0	100.0	100.0	99.3760
4: RPM Spoofing	100.0	100.0	100.0	**0.0**

TABLE IV
ADABOOST CLASSIFICATION REPORT FOR CAR-HACKING

Class	Precision (%)	Recall (%)	F1 (%)	Support
0: Normal	93	54	68	299309
1: DoS	0	0	0	11750
2: Fuzzy	6	100	11	9837
3: Gear Spoofing	99	100	99	11945
4: RPM Spoofing	0	0	0	13098

C. Extending LCCDE from Car-Hacking to ROAD

To evaluate the generalization capabilities of the ensemble's learning, we used Car-Hacking data for training and ROAD data for testing. Since the attack types are not the same between the two datasets, we changed the label of all non-Benign captures to the Malicious class, turning this into a binary classification problem. Unfortunately, the F1 scores were extremely poor for both classes. When we added AdaBoost as a fourth base learner to the ensemble, it became the leader for the Benign class, but the F1 scores remained low. For comparison, Table X shows that the F1 scores for binary classification using Car-Hacking for both training and testing are much higher. Therefore, our findings show that learning to distinguish Benign and Malicious CAN captures on the Car-Hacking dataset does not transfer to the ROAD dataset. Future work could explore the performance of LCCDE when trained on both datasets.

D. AdaBoost Performance

We found that adding AdaBoost as a fourth base learning did not improve LCCDE results for Car-Hacking, however it did improve LCCDE's performance on identifying Fabrication attacks in the ROAD dataset. It was not selected as a leader model for Car-Hacking or ROAD because it vastly underperformed when compared to the other base learners. As shown in Table IV and VIII, AdaBoost was significantly worse for every metric for every attack class except for Fabrication where it had the highest precision. AdaBoost does not have the same capabilities as the other models when it comes to large, sparse, noisy data inherent to most network intrusion datasets. Table XI shows that it had close to the same execution time as XGBoost and LightGBM, however it did not achieve the same predictive performance. The only time AdaBoost was selected as leader was the attempt to transfer the ensemble trained on Car-Hacking to ROAD, but the F1 scores were still too low to be useful. The disappointing performance of AdaBoost shows us that LCCDE's performance power is from the combined extra capabilities of the original 3 base learners.

TABLE V
METHODS FOR EVALUATING LCCDE ON ROAD

Class	Total Samples	Training Samples	Testing Samples	Leader
0: Accelerator	828556	811985	16571	CatBoost
1: Benign	3203917	3139838	64079	CatBoost
2: Fabrication	30744	40000	615	LightGBM
3: Fuzzing	1059	10000	21	LightGBM
4: Masquerade	30744	40000	615	CatBoost

TABLE VI
EVALUATING BASE LEARNERS ON ROAD BY CLASS

Class	LightGBM F1 (%)	XGBoost F1 (%)	CatBoost F1 (%)	AdaBoost F1 (%)
0: Accelerator	64.336	68.4828	68.853	26.4826
1: Benign	93.3221	93.8409	93.8802	65.0286
2: Fabrication	41.4017	35.3765	41.221	28.8991
3: Fuzzing	100.0	100.0	100.0	0.0
4: Masquerade	41.5521	44.0242	45.8075	3.609

E. Training Performance

All training was performed on Google Colab free tier. Specifically we used the base CPU instance with 12.7GB of RAM and 107.7GB disk space, however we never came close to using all allocated memory or disk space. The LCCDE training process was not resource intensive. The training times for each model on the Car-Hacking and ROAD dataset are listed in Table XI.

IV. CHALLENGES AND FUTURE WORK

One of the main challenges we faced in this project was dealing with imbalanced data. In intrusion detection systems for Internet of Vehicles (IoV), attack events are relatively rare compared to normal traffic. This results in datasets where the number of benign instances far exceeds the number of malicious ones. With imbalanced datasets, we saw that our models were biased towards the majority class, with overall accuracy, precision, and recall being useless metrics. The overall F1 score of the model and the F1 scores for each class were more informative of the model's actual performance on identifying the network attacks. We attempted to address this issue by using data oversampling of the minority classes using SMOTE and undersampling of the majority class to balance the dataset. We still saw our models bias towards the majority class, so more work needs to be done on properly handling the imbalanced dataset, especially with ROAD.

In addition, the way we trained and tested our models was not ideal. The testing size was 2% for Car-Hacking and 4% for ROAD. Although the number of testing samples of each class was sizable for Car-Hacking (see Table I), our ROAD test samples only had 615 values for the Fabrication and

TABLE VII
EVALUATING LCCDE ON ROAD BY CLASS

Class	LCCDE Without AdaBoost F1 (%)	LCCDE With AdaBoost F1 (%)
0: Accelerator	68.853	68.853
1: Benign	93.8802	93.8802
2: Fabrication	39.899	42.5702
3: Fuzzing	100.0	100.0
4: Masquerade	43.784	41.0802

TABLE VIII
ADABOOST CLASSIFICATION REPORT FOR ROAD

Class	Precision (%)	Recall (%)	F1 (%)	Support
0: Accelerator	24	30	26	16571
1: Benign	81	54	65	64079
2: Fabrication	49	20	39	615
3: Fuzzing	0	0	0	21
4: Masquerade	2	56	34	615

TABLE IX
METHODS FOR EXTENDING LCCDE FROM CAR-HACKING TO ROAD

Class	Total Samples	Training Samples (from Car-Hacking)	Testing Samples (from ROAD)	Leader Without AdaBoost	Leader With AdaBoost
0: Benign	15226820	14965421	128157	LightGBM	AdaBoost
1: Malicious	2331517	2331517	35644	LightGBM	LightGBM

TABLE X
EXTENDING LCCDE FROM CAR-HACKING TO ROAD

Class	LCCDE Extension Without AdaBoost F1 (%)	LCCDE Extension With AdaBoost F1 (%)	LCCDE on Only Car-Hacking F1 (%)
0: Benign	56.2003	50.9339	98.0622
1: Malicious	33.9603	34.4742	85.4738

Masquerade classes, and 21 values for Fuzzing (see Table V). Furthermore, due to miscommunication, we did not use the same base learner parameters between runs with and without AdaBoost added to the ensemble (we redid the train/test split, SMOTE, and model training). This is likely the reason why evaluation metrics are a few ten-thousandths different between the Reproduced LCCDE and LCCDE with AdaBoost on Car-Hacking in Table II.

We made a mistake by dropping the CAN ID feature because of our inexperience with the IoV domain. This may be the reason why we couldn't replicate the Yang, et al. results for the DoS class in Car-Hacking. Since the Fabrication and Masquerade attacks in the ROAD dataset target specific IDs, LCCDE may have achieved better results on those classes if we had included the CAN ID feature.

To address our paper's limitations, future work could use our code to evaluate LCCDE's performance on ROAD with CAN ID included as a feature. This would give a better assessment of whether the ensemble can identify the Fabrication and Masquerade classes in ROAD. Furthermore, future work could explore LCCDE on ROAD with the 33 specific attack types labeled in a different way; we used the Accelerator, Benign, Fabrication, Fuzzing, and Masquerade labels, but the dataset could be split into more specific classes. Finally, future work could evaluate LCCDE's performance when trained on both the Car-Hacking and ROAD datasets. Since we found that learning did not transfer from Car-Hacking to ROAD, researchers could investigate LCCDE's ability to learn the attack types for a mixture of both datasets.

V. ETHICS STATEMENT

The goal of our research is to contribute to the security of Internet of Vehicles (IoV) systems by evaluating the effectiveness LCCDE on detecting CAN bus intrusions on a more modern dataset. The datasets used in our research are obtained from publicly available sources. All methods and code used in this study are documented and shared openly to maintain transparency and reproducibility of our findings. We also invite other researchers to verify our results and build upon them. The intrusion detection system used in this study is intended to be used for research to make IoV systems more secure.

VI. CONCLUSION

In our study, we reproduced most of the results for using the LCCDE from the previous Yang, et al. study on Car-Hacking, with the DoS class being the one result we were unable to reproduce. Once we applied this to a newer and more modern dataset, ROAD, we found that the attacks were harder to detect and thus produced lower F1 scores than the previous two datasets. When comparing with other similar paper's that used ROAD, our results appeared similar and even slightly better in a few categories, and specifically for the masquerade attack, we found that other researchers also struggled to classify it.

The introduction of AdaBoost as an additional base learner to the ensemble did not lead to any improvement in performance on the Car-Hacking dataset. AdaBoost had a similarly poor performance on the ROAD dataset, with the exception that it improved the F1 score of classification of Fabrication attacks by about 3%. We can conclude that the original three, XGBoost, CatBoost, and LightGBM are already well-optimized for this application; AdaBoost, a simpler model, does not add significant value to the ensemble.

For the binary classification task of distinguishing Malicious from Benign samples, the updated LCCDE model trained on the Car-Hacking dataset was not able to generalize to the ROAD dataset. This indicates that some attacks in ROAD, such as the Masquerade attack, may be very different from any of the attacks in Car-Hacking.

Our results come with some key limitations. We mistakenly excluded the CAN ID feature from the Car-Hacking and ROAD datasets, which may have decreased the performance of the base learners on some classes, such as Masquerade in

TABLE XI
TRAINING TIME PER BASE MODEL ON CAR-HACKING AND ROAD

Base Model	Training Time (Car-Hacking)	Training Time (ROAD)
XGBoost	12min	3min
CatBoost	3hr	41min
LightGBM	9min	3min
AdaBoost	12min	4min

the ROAD dataset. Due to resource limitations, we were only able to use 4% of ROAD as the test size, which had very few samples of the minority class, Fuzzing. We used SMOTE to mitigate some of the imbalance in the ROAD dataset, but LCCDE was still biased towards the majority class. Future work should consider undersampling the majority class in addition to oversampling the other classes.

To support our work, we developed preprocessing scripts for both the Car-Hacking and ROAD datasets. From this, we were able to conform the datasets into a more suitable form for model training and evaluation. We hope that this can streamline the process for continuing research in the future.

Lastly, we made significant improvements to the original LCCDE implementation. These enhancements include a more flexible model configuration particularly when working with additional models, reduced code duplication, better variable names, and overall making the system more adaptable and easier to use for future use.

REFERENCES

[1] Elmustafa Sayed et al. "Machine Learning Technologies for Secure Vehicular Communication in Internet of Vehicles: Recent Advances and Applications". In: *Security and Communication Networks* 2021 (2021). DOI: 10.1155/2021/8868355.

[2] Srilatha Chebrolu, Ajith Abraham, and Johnson P. Thomas. "Feature deduction and ensemble design of intrusion detection systems". In: *Computers Security* 24.4 (2005), pp. 295–307. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2004.09.008. URL: https://www.sciencedirect.com/science/article/pii/S016740480400238X.

[3] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[4] Abdelouahid Derhab et al. "Histogram-Based Intrusion Detection and Filtering Framework for Secure and Safe In-Vehicle Networks". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 2366–2379. DOI: 10.1109/TITS.2021.3088998.

[5] Wisam Elmasry, Akhan Akbulut, and Abdul Halim Zaim. "Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic". In: *Computer Networks* 168 (2020), p. 107042. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.107042. URL: https://www.sciencedirect.com/science/article/pii/S138912861930800X.

[6] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[7] Constantinos Kolias et al. "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset". In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 184–208.

[8] Liudmila Prokhorenkova et al. *CatBoost: unbiased boosting with categorical features*. 2019. arXiv: 1706.09516 [cs.LG].

[9] Anshul Saini. *AdaBoost Algorithm: Understand, Implement and Master AdaBoost*. 2024. URL: https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/#:~:text=AdaBoost%20algorithm%2C%20short%20for%20Adaptive,assigned%20to%20incorrectly%20classified%20instances. (visited on 06/13/2024).

[10] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. "GIDS: GAN based intrusion detection system for in-vehicle network". In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE. 2018, pp. 1–6.

[11] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." In: *ICISSp* 1 (2018), pp. 108–116.

[12] Hamideh Taslimasa et al. "Security issues in Internet of Vehicles (IoV): A comprehensive survey". In: *Internet of Things* 22 (2023), p. 100809. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2023.100809. URL: https://www.sciencedirect.com/science/article/pii/S2542660523001324.

[13] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee. 2009, pp. 1–6.

[14] Miki E Verma et al. "Addressing the lack of comparability & testing in CAN intrusion detection research: A comprehensive guide to CAN IDS data & introduction of the ROAD dataset". In: *arXiv preprint arXiv:2012.14600* (2020).

[15] Kai Wang et al. *Effective In-vehicle Intrusion Detection via Multi-view Statistical Graph Learning on CAN Messages*. 2023. arXiv: 2311.07056 [cs.NI].

[16] Li Yang et al. "LCCDE: A Decision-Based Ensemble Framework for Intrusion Detection in The Internet of Vehicles". In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 3545–3550. DOI: 10.1109/GLOBECOM48099.2022.10001280.

[17] Yuyang Zhou et al. “Building an efficient intrusion detection system based on feature selection and ensemble classifier”. In: *Computer Networks* 174 (2020), p. 107247. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107247. URL: https://www.sciencedirect.com/science/article/pii/S1389128619314203.