A Novel Dataset and Approach for Adversarial Attack Detection in Connected and Automated Vehicles

Tae Hoon Kim 1,*, Moez Krichen 2,*, Meznah A. Alamro 3 and Gabreil Avelino Sampedro 4

1 School of Information and Electronic Engineering, Zhejiang University of Science and Technology, No. 318, Hangzhou 310023, China
2 ReDCAD Laboratory, University of Sfax, Sfax 3038, Tunisia
3 Department of Information Technology, College of Computer & Information Science, Princess Nourah Bint Abdul Rahman University, Riyadh 11564, Saudi Arabia; meaalamro@pnu.edu.sa
4 School of Management and Information Technology, De La Salle—College of Saint Benilde, Manila 1004, Philippines; garsampedro@ieee.org
* Correspondence: 323020@zust.edu.cn (T.H.K.); k.moez@redcad.org (M.K.)

Abstract: Adversarial attacks have received much attention as communication network applications rise in popularity. Connected and Automated Vehicles (CAVs) must be protected against adversarial attacks to ensure passenger and vehicle safety on the road. Nevertheless, CAVs are susceptible to several types of attacks, such as those that target intra- and inter-vehicle networks. These harmful attacks not only cause user privacy and confidentiality to be lost, but they also have more grave repercussions, such as physical harm and death. It is critical to precisely and quickly identify adversarial attacks to protect CAVs. This research proposes (1) a new dataset comprising three adversarial attacks in the CAV network traffic and normal traffic, (2) a two-phased adversarial attack detection technique named TAAD-CAV, where in the first phase, an ensemble voting classifier having three machine learning classifiers and one separate deep learning classifier is trained, and the output is used in the next phase. In the second phase, a meta classifier (i.e., Decision Tree is used as a meta classifier) is trained on the combined predictions from the previous phase to detect adversarial attacks. We preprocess the dataset by cleaning data, removing missing values, and adjusting the Z-score normalization. Evaluation metrics such as accuracy, recall, precision, F1-score, and confusion matrix are employed to evaluate and compare the performance of the proposed model. Results reveal that TAAD-CAV achieves the highest accuracy with a value of 70% compared with individual ML and DL classifiers.

Keywords: adversarial attacks; connected and automated vehicles; deep learning; machine learning

1. Introduction

Globally, the number of automobiles has increased dramatically due to population growth and urbanization. This growth was undoubtedly one of the main factors contributing to millions of collisions on the roads each year, as the elevated vehicle density on the roads made for an irregular, clogged traffic situation that made driving more difficult and dangerous. Intelligent Transportation Systems (ITSs) have been established as a smart technical solution to encourage smart mobility and improve road safety, among other reasons [1,2]. ITS offers several benefits, such as enhanced safety services, user-oriented mobility services, smart traffic management and monitoring, etc. [3]. Additionally, they depend on a networked infrastructure through communication linkages between cars, road users, and road sensors, known as infrastructure (V2I) and inter-vehicle (V2V). They may report road conditions and receive real-time status updates from each other, which helps to prevent dangerous situations and enhance traffic flow management [4].

Thus, because of these noteworthy developments in ITS technology, autonomous driving systems are becoming more dependable, transforming the world’s transportation infrastructure. They comprise a range of parts integrating AI and IoT (Internet of...
Things) technology to lessen reliance on human interaction [5,6]. Autonomous vehicle (AV) adoption ushers in a new age of transportation with several benefits to traffic safety and transportation effectiveness [7]. Connected and Automated Vehicles (CAVs), equipped with sensors, wireless connections, and onboard computing, can perceive, communicate, and control driving duties [8]. It is continually connected to the Internet, other cars, and computers through interfaces like Dedicated Short-Range Communications (DSRCs) and Cellular Vehicle-to-Everything (C-V2X). With the help of those features, CAVs can become somewhat automated and gather the data necessary for making decisions. They are also expected to significantly increase accessibility for individuals with disabilities and the elderly, lower environmental pollution, and improve road safety and system efficiency.

However, despite all the actions produced in this sector, some significant obstacles still prevent the widespread use of this technology. Adversarial attacks pose a significant challenge to modern machine learning models, particularly in deep learning. These attacks concern presenting carefully prepared perturbations to input data, which can lead even well-trained models to make inaccurate predictions with high confidence [9]. Figure 1 presents the working of adversarial attacks on trained models. In the context of image classification, for example, an adversarial attack might involve adding a nearly imperceptible amount of noise to an image, causing the model to misclassify it. Such attacks can be used to evaluate machine learning models’ robustness and develop defenses against them. However, they also highlight the potential vulnerabilities of these models in security-sensitive applications, such as autonomous vehicles or facial recognition systems.

![Figure 1. Adversarial attack on trained models.](image)

However, autonomous driving systems need a refined structure to grab those anomalies and minimize their consequences due to sensor data uncertainties generated by adversarial attacks or external elements, like weather conditions, environmental anomalies, or malfunctioning sensors. Numerous objects with distinct profiles, particularly in urban settings, the poor precision of perception sensor readings, data loss during the fusion process, and other issues can all lead to outliers [10]. These anomalies can potentially distort sensor data, resulting in inaccurate navigational judgments and incidents involving fatalities. As a result, locating and identifying these outliers in the gathered data is critical, as they might seriously harm autonomous driving systems.

1.1. Research Motivation

Researchers are still investigating numerous strategies to counteract adversarial attacks, including as input preprocessing, model hardening, and robust training procedures, to assure the dependability and safety of machine learning systems. The research has looked at several strategies to decrease the influence of these anomalies on the operation of AVs in general and autonomous cars in particular. Numerous anomaly detection and classification methods have been studied in the literature; most of these methods depend on the concept of artificial intelligence [11–13]. The purpose of the research was to estimate the probability that cyberattacks will cause a crash [14]. However, these methods have
shortcomings. Specifically, when using deep learning techniques in CAVs, there needs to be more detection performance for both single and mixed attack types, and there needs to be a proper ensemble machine learning strategy to bolster the technique’s efficacy in identifying attacks that address limitations.

1.2. Research Contribution

To effectively detect different types of adversarial attacks in CAVs by handling the above-cited limitations, this research’s main contributions are listed below:

• Generated a novel dataset comprising three novel adversarial attacks in the CAV network traffic and normal traffic.
• Proposed a two-phased adversarial attack detection technique named TAAD-CAV, where in the first phase, an ensemble voting classifier having three machine learning classifiers and one separate deep learning classifier is trained, and the output is used in the next phase. In the second phase, a meta classifier (i.e., Decision Tree is used as a meta classifier) is trained on the combined predictions from the previous phase to detect adversarial attacks. We preprocess the dataset, such as data cleaning, removing missing values, and Z-score normalization.
• Results reveal that TAAD-CAV achieves the highest accuracy with a value of 70% compared with ML and DL classifiers.

1.3. Research Organization

This research work is further structured in multiple sections to help readers understand the study’s framework. Section 2 provides information related to previous techniques related to adversarial attacks. The proposed methodology, the dataset’s preliminaries, data preprocessing steps, and ML and DL models are all covered in detail in Section 3. The evaluation measurements, experimental results, and findings are presented in Section 4. The conclusion and future directions of the current work are also presented in this section.

2. Literature Review

The paper [15] presents a novel fuzzy decision-making model that considers several perspectives on fusion. Their methodology focuses on three steps and evaluates multi-ML classifiers for real-time adversarial attack prediction in VANETs. First, Dedicated Short-Range Communication (DSRC) data must be identified and preprocessed using conventional and fusion preprocessing techniques. Two DSRC datasets are produced due to considering two different communication scenarios: jammed and normal. Using normal preprocessing for dataset-1 and feature fusion preprocessing for dataset-2, they build multi-ML classifiers based on the DSRC datasets. At last, they assess the multi-ML classifiers using a fuzzy decision-making technique based on the fuzzy decision-making method by opinion score method (FDOSM) and an adversarial attack decision fusion matrix. The FDOSM’s External Fusion Decision (EFD) parameters choose the optimal model, give a unique rank, and handle individual ranking variation. The experiment’s findings show that the Random Forest (RF) model, used for dataset-2 using fusion preprocessing, has a score of 0.1819. In dataset-1, the K-Nearest Neighbors Algorithm (kNN) model accomplishes the highest score of 0.2048.

The article [16] presents an effective detector based on a controlled invariant subspace for detecting hidden attacks. This research is unique because it uses a real-time trajectory space generation approach and depends on the vehicle dynamics model for system state estimate. The suggested detector creates a trajectory subspace using an offline incremental-learning-based rapidly expanding random tree approach. Trajectories that deviate from this subspace are classified as anomalies. Additionally, taking into account the dynamic performance limits of the vehicle, this research offers a distance-measuring approach that integrates a radial basis function neural network with the linear quadratic regulation algorithm. This study contains rigorous arguments demonstrating the asymptotically optimal nature of the produced trajectory and the resilience of the stealthy attack.
detection approach based on controlled invariant subspace, which is used to evaluate the theoretical performance of the proposed detector. Simulation results show that the detector outperforms sophisticated methods in accuracy and the false-positive rate and can successfully identify stealthy assaults.

The author of [17] focuses on developing machine learning models using multiple techniques and evaluating them using specific criteria to find and suggest the best-suited model for detecting attacks in CAVs. Furthermore, other terminologies associated with CAVs are defined in this study, including CAV, CAV architecture, CAV cyber security, and various hazards and vulnerabilities inherent in the CAN bus. The potential assaults against CAVs, along with the accompanying mitigation and detection strategies, are then discussed in the study. Deep Reinforcement Learning (DRL) and Distributed Kalman Filtering (DKF) methods are used in [18] to reduce jamming interference and enhance anti-eavesdropping communication capability. With the help of neighboring nodes exchanging state estimations, the author has created a DKF technique allowing more precise attacker monitoring. Next, a design issue is defined to manage transmission power and choose a communication channel while meeting the approved vehicular user’s criteria for communication quality. A hierarchical Deep Q-Network (DQN)-based architecture is created to design the anti-eavesdropping power control and potential channel selection strategy because the eavesdropping and jamming model is unpredictable and dynamic. Without prior knowledge about the eavesdropping behavior, the ideal power control method may be rapidly achieved initially. When required, the system secrecy rate evaluation carries out the channel selection process. The results of simulations verify that, in comparison to existing methods, the proposed technique improves the secrecy and feasible communication speeds.

The research [19] proposed a federated learning framework, a machine learning technique that protects privacy, for CAV intrusion detection. Federated learning (FL), which leverages the combined capacity of several CAVs while maintaining data privacy, can enhance the detection capabilities and resilience of intrusion detection systems inside the CAV ecosystem. An in-depth study of FL tailoring for cooperative intrusion detection in CAVs, as well as potential directions for future research in this field, are presented in this work. The results of this study pave the way for the wider application of connected autonomous cars in the transportation sector by advancing the development of trustworthy and safe CAV systems. The author of [20] suggests three time-series DNN Adversarial Attacks for CAVs aware of misalignment. While the third technique introduces a perturbation for every movement that incorrectly classifies all inputs with all-time stamps as the beginning point, the first two approaches evaluate the target input’s start point. In comparison to the previous assault approach, MA 3 delivers up to 63.49% greater success rate, 56% less perturbation generation time, and 63.16% less perturbation quantity, according to observed evaluations based on actual driving data and genuine trials.

The author of [21] presents two new optimization-based black-box and clean-label data poisoning attack approaches. Poisoning perturbations are produced using a mix of particle swarm optimization, simulated annealing, and genetic algorithms. Experiments on a traffic sign recognition system using CAVs are used to assess the attacking strategies. The findings indicate that even with a tiny amount of poisoned data, the suggested approaches dramatically reduce the forecast accuracy of the global model. In [22] study, the author presents a CAV cyber security framework based on UML (Unified Modeling Language), which is used to categorize possible CAV system vulnerabilities. Benchmark data set KDD99 is used to create a new CAV communication cyber-attack data set (CAV-KDD). This dataset focuses on CAV cyberattacks that rely on communication. Two ML algorithms, Decision Tree and Naïve Bayes, are used as classification models. The proposed comparison and assessment will utilize these metrics, such as accuracy, precision, and runtime, of these two models. It has been discovered that the Decision Tree model is better suited for CAV communication attack detection and needs a shorter runtime.

The author of [23] suggests a unique way to deceive the ML model of CAV to attack it. The article shows how adversarial machine learning may provide impossible attacks
for existing ML classifiers for CAV misbehavior detection to identify using adversarial instances in CAVs. Firstly, typical attack engines create adversarial datasets, which CAV misbehavior detection ML algorithms may easily identify. Training and testing are the two stages in developing an attack machine learning model. Using the time-series data that were transformed from the adversarial datasets, Step I employs supervised learning to train the model. The next phase of model enhancement begins with a test of the model in Step 2. The Random Forest (RF) and KNN algorithms are used in the first round. The subsequent round employs Long Short-Term Memory (LSTM) of recurrent neural networks and Logistic Regression (LG) of neural networks, both directed by deep learning (DL) models. The outcomes in the form of receiver operating characteristic (ROC) and precision-recall (PR) curves verify that the suggested adversarial machine learning models are successful. The literature review is summarized in Table 1.

Table 1. Literature review summary.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>[15]</td>
<td>Presents a novel fuzzy decision-making model that considers several perspectives on fusion. The methodology focuses on three steps and evaluates multi-ML classifiers for real-time adversarial attack prediction in VANETs. The findings show that the Random Forest (RF) model used for the dataset with fusion preprocessing has the highest score.</td>
</tr>
<tr>
<td>[16]</td>
<td>Presents an effective detector based on a controlled invariant subspace for detecting hidden attacks. The research uses a real-time trajectory space generation approach and depends on the vehicle dynamics model for system state estimate. The proposed detector creates a trajectory subspace using an offline incremental-learning-based approach and can successfully identify stealthy attacks.</td>
</tr>
<tr>
<td>[17]</td>
<td>Focuses on developing machine learning models using multiple techniques and evaluating them to find and suggest the best-suited model for detecting attacks in CAVs. The study also defines various terminologies associated with CAVs and discusses potential attacks and mitigation strategies.</td>
</tr>
<tr>
<td>[18]</td>
<td>Uses Deep Reinforcement Learning (DRL) and Distributed Kalman Filtering (DKF) methods to reduce jamming interference and enhance anti-eavesdropping communication capability in CAVs. The proposed technique improves the secrecy and feasible communication speeds compared to existing methods.</td>
</tr>
<tr>
<td>[19]</td>
<td>Proposes a federated learning framework for CAV intrusion detection, which leverages the combined capacity of several CAVs while maintaining data privacy to enhance detection capabilities and resilience.</td>
</tr>
<tr>
<td>[20]</td>
<td>Suggests three time-series DNN adversarial attacks for CAVs aware of misalignment. The third technique introduces a perturbation for every movement that incorrectly classifies all inputs with all-time stamps as the beginning point.</td>
</tr>
<tr>
<td>[21]</td>
<td>Presents two new optimization-based black-box and clean-label data poisoning attack approaches for a traffic sign recognition system using CAVs. The findings indicate that even with a tiny amount of poisoned data, the suggested approaches dramatically reduce the forecast accuracy of the global model.</td>
</tr>
<tr>
<td>[22]</td>
<td>Presents a CAV cyber security framework based on UML, which is used to categorize possible CAV system vulnerabilities. The proposed comparison and assessment utilize metrics such as accuracy, precision, and runtime, and the Decision Tree model is found to be better suited for CAV communication attack detection.</td>
</tr>
<tr>
<td>[23]</td>
<td>Suggests a unique way to deceive the ML model of CAV to attack it, using adversarial machine learning. The outcomes in the form of ROC and PR curves verify that the suggested adversarial machine learning models are successful.</td>
</tr>
</tbody>
</table>

3. Proposed Framework

This section provided information on the proposed methodology, including all steps, such as dataset details, dataset preprocessing techniques, and network models (ML and DL classifiers) used for creating ensemble and hybrid models. Machine learning classifiers such
as MLP, RF, XGB, KNN, DT, and deep learning classifiers such as DNN are the foundational elements of our suggested techniques. The working of our proposed methodology is shown in Figure 2.

**Figure 2.** Proposed model based on ML and DL classifiers.

### 3.1. Dataset Preliminaries

The dataset was gathered while deploying the Safety Pilot Model (SPMD). Basic safety messages (BSMs), vehicle trajectories, different driver–vehicle interaction data, and contextual data that explain the conditions under which the Model Deployment data were gathered are the data types these entities supply. Almost all of the data in this context came from roadside units and onboard vehicle devices [24]. The data collection utilized in this investigation was first investigated for anomaly identification by [11]. The three characteristics in the dataset were as follows: (i) speed, which is determined by the vehicle speedometer and is represented by sensor 1; (ii) GPS speed, which is represented by sensor 2; and (iii) acceleration, which is determined by the vehicle speedometer and is represented by sensor 3. We used an original Safety Pilot Model Deployment (SPMD) dataset [24] for this study, which is devoid of abnormalities [11].

We generated the original dataset, which has the shape (203193, 3). This section details the first step of data preprocessing.

#### 3.1.1. Handling Missing Values

**Identification:** We first identified missing values in the dataset using descriptive statistics and visualization tools.

**Removal:** We then removed rows with missing values (NaN or None) from the DataFrame. This step resulted in a new data frame with the dimensions (30000, 3), indicating that the dataset now contains 30000 records and 3 features without any missing values.

#### 3.1.2. Adversarial Attacks

We divided the dataset into four parts to add different attacks, and every data frame had a shape of 7500. 3. Several adversarial attacks are available in the literature. Ref. [25] summarizes the taxonomy of intrusions or attacks in computer network systems, which encompass CAVs. Also, several faulty sensor behaviors are discussed in [26]. We used four types of adversarial attacks that are explained one by one below:

- **Adversarial Benign:** This attack represents the original, clean data without any modifications. In the context of machine learning, it refers to the normal, unperturbed instances that are typically used for training or testing models.
• **Adversarial Noise Injection**: In this type of attack, random Adversarial Noise Injection is added to the original data. The purpose of noise injection is to perturb the data so that it becomes more difficult for machine learning algorithms to classify or make predictions accurately. The added noise can disrupt the patterns present in the data, leading to incorrect model outputs.

• **Adversarial Outlier**: Outlier injection adds values to the original data. An outlier is a data point that significantly deviates from the rest of the data in the dataset. Outliers can skew statistical analyses and machine learning models, leading to inaccurate results. Outlier injection attacks aim to exploit vulnerabilities in models’ sensitivity to outliers.

• **Adversarial Perturbation**: Perturbation injection introduces deliberate changes or perturbations to the original data. These changes can be of various types, such as flipping the sign of feature values, adding or subtracting a small constant, or multiplying by a random factor. Perturbation injection attacks aim to deceive machine learning models by altering the input data in subtle ways that may not be immediately noticeable but can lead to incorrect predictions or classifications.

These four types of adversarial attacks demonstrate different strategies for manipulating data to deceive machine learning models, highlighting the importance of robustness and security in machine learning systems. Figure 3 presents the visualization of an adversarial attack on data.

![Figure 3. Visualization of an adversarial attack.](image)

Algorithm 1 outlines a process for generating a new dataset with adversarial attacks based on an original dataset without anomalies. The algorithm takes an original dataset without anomalies as input and aims to generate a new dataset containing adversarial attacks. The original dataset undergoes a data-cleaning process to prepare it for injecting adversarial attacks. This step might involve handling missing values. The cleaned dataset is divided into four parts to add different types of adversarial attacks. These attacks include adverse bait, noise, outlier, and perturbation. For each type of adversarial attack, a function is called to generate and add the corresponding adversarial instances to the divided parts of the dataset. After adding adversarial attacks to each part of the dataset, a function dataset merge is called to merge all datasets containing adversarial attacks into a new dataset. Finally, the algorithm returns the new dataset containing the data and adds adversarial attacks.
Algorithm 1: Pseudo Code for Adversarial Attacks.

1: **Input:** $D_s$ (Original Dataset without Anomalies)
2: **Output:** $\text{NewDataset}$ (Adversarial Attack Detection)
3: **Input:** $D_c$ Data Cleaning
4: **Output:** $\text{NewDataset}$ (Adversarial Attack Detection)
5: **function** Adding Adversarial Attacks
6: **function** $A_{\text{Benign}} \rightarrow \text{Adversarial Benign}$
7: **function** $A_{\text{Noise}} \rightarrow \text{Adversarial Noise Injection}$
8: **function** $A_{\text{Outlier}} \rightarrow \text{Adversarial Outlier}$
9: **function** $A_{\text{Perturbation}} \rightarrow \text{Adversarial Perturbation}$
10: **function** $D_{\text{merge}}$ Merge all dataset of all adversarial attacks
11: **return** $\text{NewDataset}$ (Generate New dataset)

3.1.3. Label Encoder

A Label Encoder is a tool used in machine learning to convert category labels into numerical values. Many machine learning methods require the input data to be in the numerical format. A Label Encoder gives each category a distinct number in a categorical feature. Since the transformation is carried out consistently, the same category will always have the same integer encoded. We only utilized one hot label encoder in this study. One-hot encoding is a method for converting each category into a binary vector representation. Every category is assigned a distinct column; for every data point, the category-corresponding column is labeled with a 1, while all other columns are labeled with 0s. Since it avoids imposing any artificial ordering, this strategy is helpful when there is no ordinal link between the categories [27].

Assume we want to encode a single sample $x$ with the value $x_i$, where $i$ is the category index in the categorical feature. We have a categorical feature with $n$ categories. For sample $x$, the one-hot encoded vector $v$ is expressed as follows: $v = [0, 0, \ldots, 1, \ldots, 0]$. The number of categories $n$ equals the length of vector $v$. The $x_i$ category is represented by vector $v$, whose $i$-th element is set to 1. Vector $v$ has all other elements set to 0.

3.1.4. Data Splitting

The dataset is divided into test, validation, and training sets to assess the interpretation of ML models. This prevents overfitting from happening and helps assess how generalizable the models are. First, we split the MI dataset into 80% training and 20% testing sets for this study. The mathematical formula used for the training and testing split is as follows:

**Training Set Size**

$$D_{\text{train}} = \text{round}(D \times \text{train\_ratio})$$

**Test Set Size**

$$D_{\text{test}} = D - D_{\text{train}}$$

The training and testing sizes are shown in Equations (1) and (2). The dataset’s total number of occurrences is $D$. The ratio of instances assigned to the training set is known as the train\_ratio. The ratio of instances assigned to the test set is the test\_ratio. Based on the given ratios, these formulae provide you with the sizes of the training and test sets. While the remaining samples implicitly determine the size of the test set after assigning to the training set, adjusting the train\_ratio will directly change the size of the training set.

3.1.5. Normalization

Normalization is a data preprocessing technique used to scale feature values to a common range. This is performed because similar-scale features yield superior results for many ML and DL classifiers. To rescale the dataset features between 0 and 1, this study uses a normalizing strategy [28], which makes normalization and assembly easier. In this study, the characteristics are transformed to have a mean of 0 and a standard deviation of 1 using...
Z-score scaling, commonly known as standardization. For Z-score scaling, the following mathematical Equation (3) is used:

$$Z_{\text{standardized}} = \frac{Z - \mu}{\sigma}$$

(3)

where $\sigma$ is the feature’s standard deviation, $\mu$ is the dataset’s mean, and $Z$ is the actual feature value.

3.2. Classification Models

This section explains the concepts of Multi-Layer Perceptron, Decision Tree, Random Forest, Extreme Gradient Boosting, and Deep Neural Network, which are the foundations of our suggested techniques. The classifiers are chosen according to their ability to handle various scenarios, including noisy, huge, and small datasets, and weak learning classifiers’ capacity to increase detection rates.

3.2.1. Multi-Layer Perceptron

An artificial neural network (ANN) called a Multilayer Perceptron (MLP) comprises several layers of nodes, including an input layer, one or more hidden layers, and an output layer. Every node in one layer is connected to every other node, and every connection has a weight attached. The first layer of the network receives the data, which are then sent through the hidden levels, where each layer modifies the data according to the connection weights. The output layer generates the predictions or classifications the network makes at the end. MLPs are usually trained using methods like backpropagation, in which the connections’ weights are iteratively changed to reduce the discrepancy between the output obtained and the anticipated output based on a specific loss function. MLPs are the building blocks of many deep learning models and are widely utilized in ML for various tasks, including pattern recognition, regression, and classification. However, they demand a substantial amount of processing power for training, particularly when dealing with huge datasets and intricate architectures, and they may experience overfitting if they are appropriately regularized. Figure 4 displays the basic architecture of the MLP classifier.

![Figure 4. Architecture of MLP classifier.](image)

3.2.2. Random Forest

A process known as Random Forest, which is an ensemble learning approach, creates a more accurate and dependable model by combining many decision trees [29]. Being a member of the tree-based model class, it is a well-liked model for regression and classi-
fications. Random Forest models, trained on bootstrapped dataset samples, independently construct several decision trees using random feature selection. It reduces variation because the model averages or uses majority voting to combine predictions from individual trees. It enhances generalization performance, making it less prone to over-fit than individual trees. Mathematically, the prediction of a Random Forest model can be represented in Equation (4) for classification:

\[
\hat{y} = \text{mode}(\{T_1(x), T_2(x), \ldots, T_{n_{\text{estimator}}}(x)\})
\]  

(4)

where \(T_i(x)\) is the prediction of the \(i\)-th decision tree, and \(\hat{y}\) is the final predicted class.

3.2.3. Decision Tree

Decision Trees are supervised learning algorithms that use the provided characteristic values to inform their decisions, resulting in the creation of regions within the feature area [30]. At every node, the tree optimizes a selected criterion (e.g., data collection, Gini impurity) to determine which qualities best segregate the data. This process iterates back and forth until an end condition is satisfied, like reaching a maximum depth or a minimum number of samples per leaf. Because of their simplicity of use and capacity to handle numerical and categorical data, decision trees are utilized extensively across several industries. If appropriate regularization processes are not followed, they may overfit and have trouble generalizing to new data in big, complicated datasets. The decision tree model can be mathematically represented for classification: Let us assume \(T_x\) denotes the decision tree model’s prediction for input \(x\). \(T_x\) can be represented as a function that recursively evaluates decision rules based on \(x\) features until reaching a leaf node. The predicted class for \(x\) is then determined by the majority class of the instances in the corresponding leaf node.

3.2.4. K-Nearest Neighbor

An effective supervised machine learning technique for regression and classification problems is K-Nearest Neighbors (K-NNs). It is straightforward yet highly effective. Instead of learning explicit patterns, the algorithm memorizes the training dataset in instance-based learning. In K-NN, every accessible example is stored, and new cases are categorized using a similarity metric. The algorithm determines the K-Nearest Neighbors for a given input sample from the training set and then designates the majority class among these neighbors as the input sample’s prediction. One hyperparameter that must be predetermined is the value of \(K\).

K-NN is a non-parametric algorithm, making no assumptions about the underlying data distribution. It is also called a lazy learning algorithm because it does not explicitly learn a model during training; instead, it waits until a prediction is required. While K-NN is comfortable comprehending and executing, it can be computationally costly, especially with extensive datasets, as it needs to compute distances for each new instance to all existing instances. Additionally, choosing the value of \(K\) and the appropriate distance metric can significantly impact the algorithm’s performance. The mathematical representation of the KNN algorithm is given in Equation (5) for classification:

\[
\hat{y} = \arg\max_{c} \left( \sum_{i=1}^{K} 1(y_i = c) \right)
\]  

(5)

where \(\hat{y}\) is the predicted class label for the input data point \(x\), \(K\) is the number of nearest neighbors, \(y_i\) represents the class labels of the \(i\)-th nearest neighbor, \(c\) represents each unique class label, and \(1\) is the indicator function.

3.2.5. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a powerful ensemble learning method based on gradient boosting. To optimize the weak learners—typically decision trees built one
after the other—gradient descent is employed [31]. XGBoost is well-known for its efficiency, scalability, and quick processing of structured and tabular data. It employs regularization strategies to prevent overfitting and has built-in capability for managing missing information. To maximize model performance, XGBoost also offers advanced features like cross-validation and early stopping.

In mathematical form, the prediction of XGBoost is represented in Equation (6):

$$\hat{y} = \sum_{i=1}^{N} f_i(x)$$

where $\hat{y}$ is the predicted output for the input data point $x$, $N$ is the total number of trees in the ensemble, and $f_i(x)$ represents the prediction of the $i$-th tree for the input $x$.

3.2.6. Deep Neural Network

An artificial neural network with many layers of deep architecture separating the input and output layers is called a Deep Neural Network (DNN). Layers of a DNN are composed of connected nodes or neurons. These networks are capable of identifying and expressing complex hierarchical patterns and features in data. Speech recognition, computer vision, natural language processing, and many other fields have shown great interest in deep learning, of which DNNs are a crucial component. Deep learning has also shown great potential in these and other disciplines. Deep Neural Networks (DNNs) have demonstrated remarkable efficacy in independently generating hierarchical representations from data [32]. This enables DNNs to capture intricate features and connections effectively. Figure 5 shows the architecture of the DNN model.

![Figure 5. Deep neural network architecture.](image)

DNN models predict the class labels (i.e., Normal, Adversarial Noise Injection, Adversarial Outliers, Adversarial Perturbation) based on the highest probability among the softmax probabilities generated for each class. After feeding the input data through the
neural network and all processes, such as weight assignment and activation function, the model produces a vector of probabilities, each representing the probability of belonging to a particular class. The class with the highest probability is then selected as the predicted class label. The model identifies this predicted class label based on the index corresponding to the highest probability in the probability vector.

\[
\hat{y} = \text{argmax}_i p(i)
\]

3.2.7. TAAD-CAV

This study proposes an innovative approach to enhance classification accuracy by combining traditional machine learning (ML) classifiers with a Deep Neural Network (DNN) model, augmented by a Decision Tree meta-classifier for final prediction aggregation. Initially, three distinct ML classifiers—Support Vector Classifier (SVC), Gaussian Naive Bayes (GaussianNB), and K-Nearest Neighbors (KNNs)—are individually trained on the dataset. These classifiers are integrated using a hard voting strategy within a voting classifier framework. Hard voting aggregates predictions from each base classifier, with the final prediction for each instance determined by the majority vote. This ensemble approach leverages the diverse strengths of each classifier to improve overall prediction robustness.

In parallel, a Deep Neural Network (DNN) model is trained on the same dataset to capture complex nonlinear relationships and patterns. Once trained, the DNN model independently generates predictions for the test data based on its learned representations and hierarchical feature extraction capabilities. This dual approach—combining ensemble classifiers with a deep learning model—aims to exploit the complementary advantages of both paradigms, enhancing classification performance across varied and complex datasets.

Following the generation of predictions from the ensemble classifier and the DNN model, these predictions are concatenated into a single array. This array forms a feature matrix where each row corresponds to an instance in the test set, and each column represents a prediction from either the ensemble classifiers or the DNN model. This structured data format allows for seamless integration and the comparison of predictions across different classifiers.

To consolidate and refine these diverse predictions, a Decision Tree classifier is introduced as a meta-classifier. The Decision Tree meta-classifier is trained using the concatenated predictions as features and the true labels of the training set as targets. The optimal hyperparameters, such as the maximum depth and the minimum number of samples per leaf for the Decision Tree meta-classifier, are determined through a systematic evaluation process. This involves grid search and cross-validation techniques, where the grid search systematically explores various hyperparameter combinations, and cross-validation assesses their impact on model performance. Specifically, cross-validation splits the dataset into multiple folds, trains the model on several combinations of these folds, and averages performance metrics like accuracy, precision, recall, and F1-score to ensure the robustness and generalization capability of the model. The chosen hyperparameter values maximize predictive accuracy and coherence in synthesizing outputs from diverse classifiers.

By learning to combine and weigh the outputs of the ensemble classifier and the DNN model effectively, the meta-classifier enhances prediction accuracy and generalization capabilities. This meta-learning approach ensures that the final predictions are based on a comprehensive synthesis of insights derived from both traditional ML and deep learning methodologies.

4. Experimental Result and Discussion

This section includes the evaluation measurements used to evaluate the model performance and explains the findings and results of the proposed model in detail. The efficiency of this study’s proposed model was evaluated utilizing different evaluation measurements, including accuracy, precision, recall, and F1-score and confusion matrix. Evaluation measurements were utilized to assess the prediction and classification problems.
4.1. Machine Learning Approach

Table 2 presents the performance results for the classification report for an RF classifier. Adversarial Benign had a precision of 0.61, which means 61% of instances predicted were correctly identified, and the recall value was 0.63, which means the classifier identified 63% of all actual instances of this class in the dataset. The F1-score, which balances precision and recall, for this class was 0.62. Adversarial Noise Injection had a precision of 0.73, which means 73% of instances were predicted correctly. The recall value was 0.68, which means the classifier identified 68% of all actual instances, and the F1-Score for this class was 0.70. Adversarial Outlier had a precision of 0.66, which means 66% of instances predicted as Adversarial Outlier were correct. The recall was 0.68 for this class, the classifier identified 68% of all actual instances, and the F1-score was 0.67. Adversarial Perturbation had a precision of 0.82. A total of 82% of instances predicted were correct, with a recall of 0.83 for all actual instances, and the F1-score for this class was 0.83. The Adversarial Perturbation class had the highest precision, recall, and F1 score. However, the performance for the Adversarial Benign class was the lowest among the four classes, with the lowest precision, recall, and F1 score. The model’s overall accuracy was 0.70, meaning that it correctly classified 70% of all instances across all classes. The macro-average values provide an average across all classes, with precision, recall, and F1-score being 0.71, 0.70, and 0.70, respectively. The weighted average values consider the class distribution, providing a more representative overall performance measure. In this case, the weighted average precision was 0.71, and the recall and F1-score had the same value of 0.70, reflecting the model’s performance while considering the varying numbers of instances in each class. In a classification report, “support” refers to the number of true instances for each label in the dataset. It indicates the number of occurrences of each class in the dataset that was used for evaluating the classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.61</td>
<td>0.63</td>
<td>0.62</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.73</td>
<td>0.68</td>
<td>0.70</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.66</td>
<td>0.68</td>
<td>0.67</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.82</td>
<td>0.83</td>
<td>0.83</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.70</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.70</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.70</td>
<td>9000</td>
</tr>
</tbody>
</table>

Table 3 presents the performance results for the classification report for the XGB classifier. Adversarial Benign had a precision, recall and F1-Score of 0.55, which means this classifier correctly identified 55% of instances predicted. Adversarial Noise Injection had a precision of 0.70, which means 70% of instances were predicted correctly. The recall value was 0.62, which means the classifier identified 62% of all actual instances, and the F1-Score for this class was 0.66. Adversarial Outlier had a precision of 0.62, which means 62% of instances predicted as Adversarial Outlier were correct. Recall was 0.71 for this class, the classifier identified 71% of all actual instances, and the F1-score was 0.66. Adversarial Perturbation had a precision of 0.82. A total 82% of the instances predicted were correct, with a recall of 0.78 for all actual instances, and the F1-score for this class was 0.80. The Adversarial Perturbation class had the highest precision, recall, and F1 score, and the Adversarial Benign class was the lowest among the four classes, with the lowest precision, recall, and F1 score. The classifier achieved an accuracy of 0.66, suggesting reasonable overall performance in classifying instances correctly. The macro-average values provide an average across all classes, with precision, recall, and F1-score being 0.67, 0.66 and 0.67,
respectively. The weighted average values consider the class distribution, providing a more representative overall performance measure. In this case, the weighted average precision was 0.67, and the recall and F1-score had the same value of 0.66, reflecting the model’s performance while considering the varying numbers of instances in each class.

Table 3. Classification report for XGB classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.70</td>
<td>0.62</td>
<td>0.66</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.62</td>
<td>0.71</td>
<td>0.66</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.82</td>
<td>0.78</td>
<td>0.80</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.66</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.67</td>
<td>0.66</td>
<td>0.67</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.67</td>
<td>0.66</td>
<td>0.66</td>
<td>9000</td>
</tr>
</tbody>
</table>

Table 4 presents the performance results for the classification report for the KNN classifier. The Adversarial Benign class achieves a precision of 0.50, a recall of 0.62, and an F1-score of 0.55. This suggests that the classifier identified a relatively high proportion of actual instances. For the Adversarial Noise Injection class, a precision of 0.66, a recall of 0.60, and an F1-score of 0.63 are achieved. This indicates a moderate performance in identifying instances of this class with a balance between precision and recall. The KNN classifier achieves a precision of 0.67, a recall of 0.59, and an F1-score of 0.63 for the Adversarial Outlier class. Similar to the Adversarial Noise Injection class, there is a moderate balance between precision and recall. The Adversarial Perturbation class achieves a precision of 0.80, recall of 0.76, and an F1-score of 0.78. This indicates that the classifier is quite effective in identifying Adversarial Perturbation. The Adversarial Benign class is the lowest among the four classes, with the lowest precision, recall, and F1-score. The overall accuracy of the K-Neighbors classifier is 0.64, suggesting that 64% of all instances are classified correctly across all defined classes. The macro-average precision, recall, and F1-score are 0.66, 0.64, and 0.65, respectively. The weighted average precision, recall, and F1-score are also 0.66, 0.64, and 0.65, respectively.

Table 4. Classification report for K-Neighbors classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.50</td>
<td>0.62</td>
<td>0.55</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.66</td>
<td>0.60</td>
<td>0.63</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.67</td>
<td>0.59</td>
<td>0.63</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.80</td>
<td>0.76</td>
<td>0.78</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.64</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.66</td>
<td>0.64</td>
<td>0.65</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.66</td>
<td>0.64</td>
<td>0.65</td>
<td>9000</td>
</tr>
</tbody>
</table>

Table 5 presents the performance results for the classification report for the DT classifier. The Adversarial has a precision of 0.60, meaning that out of all instances, 60% are predicted correctly. The recall and F1-score of 0.59 indicates that the model correctly identifies 59% of all actual instances. For the Adversarial Noise Injection class, the model performs better with a value of 0.68 for all evaluation metrics, including precision, recall and F1-score. The classifier achieves a precision of 0.64, a recall of 0.63, and an F1-score of 0.63 for the Adversarial Outlier class. In the Adversarial Perturbation class, the classifier demonstrates
relatively high performance with a precision of 0.78, a recall of 0.81, and an F1-score of 0.80. This indicates that the classifier is quite effective in identifying instances of Adversarial Perturbation. The model’s overall accuracy is 0.68, meaning that it correctly classifies 68% of all instances across all classes. The macro-average values provide an average across all classes, with a 0.68 precision, recall, and F1-score value. The weighted average precision, recall, and F1-score are also 0.67, 0.68, and 0.68, respectively. A weighted average considers the support for each class, giving more weight to classes with a larger number of instances.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.60</td>
<td>0.59</td>
<td>0.59</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.64</td>
<td>0.63</td>
<td>0.63</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.78</td>
<td>0.81</td>
<td>0.80</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.68</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.67</td>
<td>0.68</td>
<td>0.68</td>
<td>9000</td>
</tr>
</tbody>
</table>

Figure 6 visualizes the confusion matrix (CM) of the ML approaches. This visualization offers a high-level overview of how the classification algorithm executes. The performance of this approach appears to be superior, as indicated by fewer false positive and false negative outcomes, along with higher counts of true positive and true negative values. The CM illustrates instances where the algorithm misclassified records, while the diagonal elements represent correct predictions. Overall, the ML approach demonstrates improved accuracy and effectiveness in classification tasks compared to traditional methods.

![CM of RF Model](a) CM of RF Model ![CM of DT Model](b) CM of DT Model ![CM of KNN Model](c) CM of KNN Model ![CM of XGB Model](d) CM of XGB Model

Figure 6. Confusion matrix graphs of the machine learning approach.

4.2. Machine Learning Ensemble Approach

Table 6 presents the performance results for the classification report for the ensemble voting classifier. Adversarial Benign has a precision of 0.59, indicating that out of all instances predicted as Adversarial Benign by the classifier, only 59% are correct. It has a recall of 0.64, and the classifier identifies 64% of all instances of adversarial benign behavior in the dataset. F1-score is 0.62, which is the harmonic mean of precision and recall. Adversarial Noise Injection has a precision of 0.74, which means that 67% of instances predicted as Adversarial Noise Injection are correct, and recall with 0.68 and F1 score for this class is 0.70. Adversarial Outlier has a precision of 0.66, in which 66% of instances predicted as Adversarial Outlier are correct. The recall of this class is 0.69, in which the classifier identifies 69% of all actual instances of Adversarial Outliers in the dataset. The F1-score, which balances precision and recall for this class, is 0.68. Adversarial Perturbation has a precision of 0.84, which indicates that 84% of instances are predicted correctly, and recall has a 0.82 value, which the classifier identifies as 82% of all actual instances. The F1 score for this class is 0.83. The Adversarial Perturbation class has the highest precision, recall, and F1 score. However, the performance for the Adversarial Benign class is the lowest.
among the four classes, with the lowest precision, recall, and F1 score. The model’s overall accuracy is 0.70, meaning that it correctly classifies 70% of all instances across all classes. The macro-average values provide an average across all classes, with a precision and F1-score of 0.71 and recall of 0.70, respectively. The weighted average values consider the class distribution, providing a more representative overall performance measure. In this case, the weighted average precision is 0.71, and recall and F1-score are 0.70, reflecting the model’s performance while considering the varying numbers of instances in each class.

Table 6. Classification report for ensemble classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.59</td>
<td>0.64</td>
<td>0.62</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.74</td>
<td>0.67</td>
<td>0.70</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.66</td>
<td>0.69</td>
<td>0.68</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.84</td>
<td>0.82</td>
<td>0.83</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.70</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.71</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.70</td>
<td>9000</td>
</tr>
</tbody>
</table>

Figure 7 visualizes the CM of the ensemble voting approach. This visualization offers a high-level overview of how the classification algorithm executes. The performance of this approach appears to be superior, as indicated by fewer false positive and false negative outcomes, along with higher counts of true positive and true negative values. The CM illustrates instances where the algorithm misclassified records, while the diagonal elements represent correct predictions. Overall, the ensemble approach demonstrates improved accuracy and effectiveness in classification tasks compared to traditional methods.

4.3. Deep Learning Approach

Table 7 presents the performance results for the classification report for an MLP classifier. Adversarial Benign had a precision of 0.40, indicating that out of all instances predicted as Adversarial Benign by the classifier, only 40% were correct. It had a recall of
0.46, and the classifier identified 46% of all instances of adverse benign behavior in the dataset. F1-score was 0.43, which is the harmonic mean of precision and recall. For this class, it was 0.43, which balances both precision and recall. Adversarial Noise Injection had a precision of 0.67, which means that 67% of instances predicted as Adversarial Noise Injection were correct, and recall with 0.52 indicates the classifier identified 52% of all actual instances of Adversarial Noise Injection in the dataset. The F1 score for this class was 0.58. Adversarial Outlier had a precision of 0.58, in which 58% of instances predicted as Adversarial Outlier were correct. The recall of this class was 0.66, in which the classifier identified 66% of all actual instances of Adversarial Outliers in the dataset. Adversarial Perturbation had a precision of 0.79, which indicates that 79% of instances were predicted correctly, and recall had a 0.73 value, which the classifier identified as 73% of all actual instances. The F1 score for this class was 0.76. The Adversarial Perturbation class had the highest precision, recall, and F1 score. However, the performance for the Adversarial Benign class was the lowest among the four classes, with the lowest precision, recall, and F1 score. The overall accuracy of the model was 0.59, meaning that it correctly classified 59% of all instances across all classes. The macro-average values provided an average across all classes, with precision, recall, and F1-score being 0.61, 0.59, and 0.60, respectively. The weighted average values consider the class distribution, providing a more representative overall performance measure. In this case, the weighted average precision, recall, and F1-score were all 0.61, reflecting the model’s performance while considering the varying numbers of instances in each class. These metrics give equal weight to each class, regardless of its distribution in the dataset.

Table 7. Classification report for MLP classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.40</td>
<td>0.46</td>
<td>0.43</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.67</td>
<td>0.52</td>
<td>0.58</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.58</td>
<td>0.66</td>
<td>0.62</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.79</td>
<td>0.73</td>
<td>0.76</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.59</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.61</td>
<td>0.59</td>
<td>0.60</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.61</td>
<td>0.59</td>
<td>0.60</td>
<td>9000</td>
</tr>
</tbody>
</table>

Table 8 presents the performance results for the classification report for the DNN classifier. The DNN classifier achieves a precision of 0.39 and a recall of 0.44 for the Adversarial Benign class, resulting in an F1 score of 0.41. This indicates that the classifier correctly identifies 44% of instances belonging to the Adversarial Benign class out of all instances. For the Adversarial Noise Injection class, the classifier demonstrates a precision of 0.73, a recall of 0.49, and an F1-score of 0.58. This suggests that while the classifier achieves high precision, it may miss a significant portion of actual instances of Adversarial Noise Injection. The classifiers achieve a precision of 0.57, a recall of 0.70, and an F1-score of 0.63 for the Adversarial Outlier class. For the Adversarial Perturbation class, the classifiers demonstrate relatively high performance with a precision of 0.78, a recall of 0.74, and an F1-score of 0.76. This indicates that the classifier effectively identifies Adversarial Perturbation. The overall accuracy of the DNN classifier is 0.59, suggesting that 59% of all instances are classified correctly. The macro-average precision, recall and F1-score are 0.62, 0.59, and 0.60, respectively. The weighted average precision, recall, and F1-score are also 0.61, 0.59, and 0.59, respectively.
Table 8. Classification report for DNN classifier.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.39</td>
<td>0.44</td>
<td>0.41</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.73</td>
<td>0.49</td>
<td>0.58</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.57</td>
<td>0.70</td>
<td>0.63</td>
<td>2247</td>
</tr>
<tr>
<td>Adversarial Perturbation</td>
<td>0.78</td>
<td>0.74</td>
<td>0.76</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>0.59</td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.62</td>
<td>0.59</td>
<td>0.60</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.61</td>
<td>0.59</td>
<td>0.59</td>
<td>9000</td>
</tr>
</tbody>
</table>

Figure 8 visualizes the CM of the DL classifiers. This visualization offers a high-level overview of how the classification algorithm executes. The performance of this approach appears to be superior, as indicated by fewer false positive and false negative outcomes, along with higher counts of true positive and true negative values. The CM illustrates instances where the algorithm misclassified records, while the diagonal elements represent correct predictions. Overall, the MLP and DNN classifiers demonstrate improved accuracy and effectiveness in classification tasks compared to traditional methods.

![CM of MLP Model](image1)

![CM of DNN Model](image2)

Figure 8. Confusion Matrix Graphs of Deep Learning Approaches.

4.4. TAAD-CAV Results

Table 9 presents the performance results for the classification report for the meta classifier. The meta classifier achieves a precision of 0.59 and a recall of 0.64 for the Adversarial Benign class, resulting in an F1-score of 0.62. This indicates that 59% of instances predicted as Adversarial Benign are correct, and the classifier identifies 64% of all actual instances of Adversarial Benign in the dataset. For the Adversarial Noise Injection class, the meta classifier demonstrates a precision of 0.74, a recall of 0.67, and an F1-score of 0.70. This indicates that the classifier performs relatively well in identifying Adversarial Noise Injection instances. The classifier achieves a precision of 0.67, a recall of 0.69, and an F1-score of 0.68 for the Adversarial Outlier class. For the Perturbation Injection class, the classifier demonstrates a relatively high performance with a precision of 0.84, recall of 0.82, and an F1-score of 0.83. This indicates that the classifier is quite effective in identifying instances of Perturbation Injection. The macro-average precision, recall, and F1-score are 0.71, 0.70, and 0.71, respectively. This indicates the average performance across all classes without considering class imbalance. The weighted average precision, recall, and F1-score are also 0.71, 0.70, and 0.71, respectively. The weighted average considers the support for each class, giving more weight to classes with a larger number of instances. The meta classifier demonstrates relatively high performance across all classes, with balanced
precision and recall. The classifier achieves an overall accuracy of 70%, indicating its effectiveness in classifying instances correctly across all defined classes.

Table 9. Classification report for TAAD-CAV.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Benign</td>
<td>0.59</td>
<td>0.64</td>
<td>0.62</td>
<td>2335</td>
</tr>
<tr>
<td>Adversarial Noise Injection</td>
<td>0.74</td>
<td>0.67</td>
<td>0.70</td>
<td>2195</td>
</tr>
<tr>
<td>Adversarial Outlier</td>
<td>0.67</td>
<td>0.69</td>
<td>0.68</td>
<td>2247</td>
</tr>
<tr>
<td>Perturbation Injection</td>
<td>0.84</td>
<td>0.82</td>
<td>0.83</td>
<td>2223</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td>9000</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.71</td>
<td>9000</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.71</td>
<td>0.70</td>
<td>0.71</td>
<td>9000</td>
</tr>
</tbody>
</table>

Figure 9 visualizes the CM of the hybrid ML and DL approach, which we named the Meta model. This visualization offers a high-level overview of how the classification algorithm executes. The performance of this approach appears to be superior, as indicated by fewer false positive and false negative outcomes, along with higher counts of true positive and true negative values. Overall, the Meta model demonstrates improved accuracy and effectiveness in classification tasks compared to traditional methods.

Figure 9. Confusion matrix graphs of TAAD-CAV.

5. Deployment of TAAD-CAV

The proposed TAAD-CAV system is designed to be implemented on edge computing devices integrated within the CAV infrastructure. These edge devices, which are located either in the vehicles themselves or at roadside units, ensure minimal latency and quick response times by processing data locally. For efficient and real-time detection, we utilize powerful edge computing technologies that offer robust GPU capabilities suitable for running deep learning models. These devices provide the necessary computational power
to handle the demands of the TAAD-CAV system. To ensure scalability, maintainability, and ease of updates, we employ containerization technologies like Docker. This allows the system to be easily deployed and managed across different hardware setups. The edge devices utilize secure communication protocols to send critical data to a central server for further analysis and continuous model improvement. This ensures the system remains up-to-date with the latest threat patterns. The edge devices running the TAAD-CAV system require specific hardware and software configurations. They need multi-core ARM or x86 processors, at least 16 GB of RAM to handle real-time data processing and model execution, and an NVIDIA CUDA-compatible GPU with a minimum of 4 GB of VRAM to accelerate deep learning inference. Additionally, a storage capacity of at least 256 GB SSD is necessary for fast read/write operations and to store the necessary datasets and models. The software requirements include an operating system like Ubuntu 18.04 or later, Docker for containerization, TensorFlow or PyTorch for deep learning model implementation, and Python for scripting and algorithm execution. This configuration ensures the system can operate efficiently, providing timely detection of adversarial attacks while maintaining the integrity and safety of CAVs.

6. Conclusions

To maintain vehicle and passenger safety, Connected and Automated Vehicles (CAVs) must be secured against adversarial attacks. For the safety of CAVs, it is essential to accurately and promptly recognize attacks. Considering this problem, this study proposes an adversarial attack detection hybrid technique integrating machine (RF, DT, XGB) and deep (DNN) learning classifiers with meta classifiers (e.g., Decision tree) to generate a hybrid model. This research generates a dataset without anomalies and adds adversarial attacks as a label class to create a new dataset for prediction. The following actions are taken to preprocess the dataset: the data are cleaned, labeled, divided, and normalized using Z-score normalization techniques; missing values are eliminated; the one-hot encoding method is applied; and data are split. The performance of the proposed models is evaluated utilizing accuracy, recall, precision, the F1-score, and the confusion matrix. The meta classifier is a hybrid of ML and DL models that achieves the highest performance in all models with a value of 70% accuracy. To support the findings of our suggested framework, we want to develop and examine different methodologies in further work. In the future, we will also check the generalizability of newly created dataset-based adversarial attacks by applying different machine learning, deep learning, and feature optimization techniques.

Author Contributions: Conceptualization, T.H.K., G.A.S. and M.K.; Data curation, M.A.A.; Formal analysis, T.H.K. and G.A.S.; Funding acquisition; Investigation, M.A.A., M.K., T.H.K. and G.A.S.; Methodology, M.A.A., M.K., T.H.K. and G.A.S.; Project administration, M.A.A. Software, T.H.K. and G.A.S.; Supervision; Validation, M.A.A.; Visualization; Writing—original draft, T.H.K., G.A.S. and M.A.A.; Writing—review and editing, M.K., T.H.K., G.A.S. and M.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: Authors would like to acknowledge Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024RS03), Princess Nourah Bint Abdul Rahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors share no conflicts of interest.
References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.