Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.035874
Article

Tech Science Press

Ensemble Voting-Based Anomaly Detection for a Smart Grid Communication Infrastructure

Hend Alshede[1,2,*], Laila Nassef[1], Nahed Alowidi[1] and Etimad Fadel[1]

[1]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia
[2]Self-Development Skills Department, Common First Year Deanship, King Saud University, Riyadh, 12211, Saudi Arabia
*Corresponding Author: Hend Alshede. Email: haalshede@stu.kau.edu.sa
Received: 08 September 2022; Accepted: 04 November 2022

Abstract: Advanced Metering Infrastructure (AMI) is the metering network of the smart grid that enables bidirectional communications between each consumer's premises and the provider's control center. The massive amount of data collected supports the real-time decision-making required for diverse applications. The communication infrastructure relies on different network types, including the Internet. This makes the infrastructure vulnerable to various attacks, which could compromise security or have devastating effects. However, traditional machine learning solutions cannot adapt to the increasing complexity and diversity of attacks. The objective of this paper is to develop an Anomaly Detection System (ADS) based on deep learning using the CIC-IDS2017 dataset. However, this dataset is highly imbalanced; thus, a two-step sampling technique: random under-sampling and the Synthetic Minority Oversampling Technique (SMOTE), is proposed to balance the dataset. The proposed system utilizes a multiple hidden layer Auto-encoder (AE) for feature extraction and dimensional reduction. In addition, an ensemble voting based on both Random Forest (RF) and Convolutional Neural Network (CNN) is developed to classify the multiclass attack categories. The proposed system is evaluated and compared with six different state-of-the-art machine learning and deep learning algorithms: Random Forest (RF), Light Gradient Boosting Machine (LightGBM), eXtreme Gradient Boosting (XGboost), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and bidirectional LSTM (biLSTM). Experimental results show that the proposed model enhances the detection for each attack class compared with the other machine learning and deep learning models with overall accuracy (98.29%), precision (99%), recall (98%), F_1 score (98%), and the UNDetection rate (UND) (8%).

Keywords: Advanced metering infrastructure; smart grid; cyberattack; ensemble voting; anomaly detection system; CICIDS2017

1 Introduction

The Advanced Metering Infrastructure (AMI) is a core component of the smart grid to enable bidirectional communication between a consumer's premises and the control center of the utility company [1]. The collected data supports various functionalities of the smart grid, such as energy consumption forecasting, energy theft detection, power quality monitoring, and dynamic energy pricing [2]. AMI communication architecture relies on three main types of communication networks: Home Area Network (HAN), Neighborhood Area Network (NAN), and Wide Area Network (WAN) [3]. Different communication technologies and protocols are used along with the Internet of Things (IoT), which has recently emerged as an enabling technology for smart grids [4]. The bidirectional end-to-end communication between smart meters and a utility company makes these networks highly vulnerable to cyberattacks [5]. Unethical customers, publicity seekers, and curious or motivated eavesdroppers might target the communication infrastructure for a variety of malicious reasons to cause local or regional consequences, such as customer information leakage, electricity theft, energy price manipulation, and system disruption, which could damage the economy or endanger human life [6].

For the smart grid, it is essential to secure the communication infrastructure to ensure the continuous and reliable flow of information and electricity [7]. Traditional security solutions such as encryption, firewalls, authorization, and authentication mechanisms are not enough to protect the AMI. Thus, Intrusion Detection Systems (IDSs) are vital for protecting AMI. IDSs are used to monitor the network traffic flow to detect various kinds of attacks [8]. IDSs can detect abnormal events and alert administrators to take appropriate measures [9].

IDSs are classified into two main types: Host-based (HIDS) or Network-based (NIDS). NIDSs analyze and investigate incoming network traffic. In contrast, HIDSs monitor node devices for malicious activity. NIDSs have three categories based on detection approaches: signature-based, anomaly-based, and hybrid-based. Signature-based detection systems use a known pattern of malicious behavior stored in a database that contains predefined malicious signatures. Anomaly-based Detection Systems (ADSs) identify attacks by looking for deviations in network traffic. The hybrid IDSs combine signature-based and anomaly-based approaches. Usually, ADSs are preferred over the signature-based one because the former can address new and undefined attacks [10]. Therefore, this paper proposes the development of ADS to detect multiple attack types against AMI communication infrastructure.

Machine learning and deep learning models are currently used to detect cyberattacks [11]. However, machine learning-based IDSs cannot process large amounts of nonlinear high-dimensional data, and it can be difficult to adapt the system to increasingly complex and diversified attacks [12]. Deep learning models do not suffer from these problems; therefore, they have been gradually applied to IDSs [13]. Considering the periodic, high-traffic, and nonlinear characteristics of AMI traffic, the objective of this paper is to develop a deep learning-based ADS to detect and classify multiple attack types that target the AMI communication infrastructure.

AMI has some distinguishing characteristics that are not parts of a conventional information technology environment, such as high-quantity and nonlinear high-dimensional data, imbalanced network traffic, and constant exposure to various attack methods. The CIC-IDS2017 dataset [14], represents the most valuable contemporary dataset satisfying the unique requirements of AMI and has accordingly been used in this paper to reflect real-world attack criteria and include the most recently updated network attacks. The main contributions of this paper are as follows:

C.1. The use of Auto-Encoder (AE) for feature extraction and dimensional reduction.

C.2. The use of two-step sampling to balance the high-class imbalance of the dataset. Random under-sampling and Synthetic Minority Oversampling Technique (SMOTE).

C.3 The development of ensemble voting Random Forest–Convolutional Neural Network (RF–CNN) to classify a wide range of attacks.

The rest of the paper is organized as follows. In Section 2, related works are reviewed. In Section 3, the proposed methodology is described. The results and discussion are presented in Section 4. Finally, the conclusion and future works are presented in Section 5.

2 Related Works

Subsection 2.1 briefly describes designing an ADS for different network environments. Subsection 2.2 discusses designing an ADS specifically for an AMI environment. In Subsection 2.3, different works discussing the CIC-IDS2017 dataset are considered.

2.1 Anomaly-based Intrusion Detection Systems (ADS) Related Works

ADSs can be classified into many types, such as statistics, data mining, machine learning, and deep learning. Some approaches combine two or more techniques [15]. This section reviews ADSs that are based on machine learning or deep learning.

Based on machine learning, Anwer et al. [16] developed a framework for detecting attacks in network flow. Their framework used three widespread machine learning classifiers: a Support Vector Machine (SVM), Gradient Boosted Decision Trees (GBDT), and RF. RF achieved the highest accuracy of 85.34%. The NSL-KDD dataset was used in the development of this framework. The authors validated their model based on accuracy only. Salman et al. [17] demonstrated the feasibility of supervised machine learning models for anomaly IDS and for classifying attacks using the UNSW dataset. In their experiments, RF had an accuracy of 99%. Othman et al. [18] used ChiSqSelector and an SVM classifier on an Apache Spark Big Data platform. They used the KDD Cup 99 dataset to train and test the model. The findings showed that the model had a very high accuracy of about 99%.

Based on deep learning, Hussain et al. [19] used CICDDoS2019 to simulate an IoT system. They developed an IDS with a CNN classifier to detect Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. It had an extremely high accuracy of about 99.99%. Otoum et al. [20] achieved an accuracy of 99.02% with an IDS based on a Stacked Deep Polynomial Network (SDPN), which was used to detect attacks in IoT systems using the NSL-KDD dataset.

By contrast, Ge et al. [21] studied an IoT environment. They developed an IDS based on a Feed-forward Neural Network (FNN) using the BoT-IoT dataset. These authors achieved an accuracy of 99.79%. Ge et al. [22] used the BoT-IoT dataset in another study. They designed an IDS that could detect DoS, DDoS, reconnaissance, and information theft in IoT systems. They collected the BoT-IoT dataset. Their FNN classifier had an accuracy of 99%. Derhab et al. [23] described a detection model for IoT networks based on a CNN classifier, which had an accuracy of 99.99% with the BoT-IoT dataset.

Ullah et al. [24] devised an IDS for detecting attacks in an IoT. They used a CNN classifier with the MQTT-IoT-IDS2020, BoT-IoT, and IoT-23 datasets. The results were excellent, with an accuracy of 99.97%. Moreover, Latif et al. [25] achieved an accuracy of 99.54%. They focused on the Industrial IoT (IIoT). The classifier was based on a Deep Random Neural Network (DRaNN) trained with the UNSW-NB15 dataset. Tufan et al. [26] developed an ADS based on a CNN model to protect institutional network systems. They trained the model on the UNSW-NB15 dataset. The system had a high accuracy of 99%. Table 1 summarizes the literature on ADSs.

Table 1: Summary of the ADS literature

Ref.	Year	Environment	Dataset	Classifier	Accuracy	Evaluation metrics
[16]	2021	IoT	NSL-KDD	RF	85	–
[17]	2017	Multi-cloud	UNSW-NB15	RF	99	–
[18]	2018	Big data	KDD Cup 99	SVM	99	Training time: 10.79 s
[19]	2020	IoT	CICDDoS2019	CNN	99	Precision: 87% Recall: 86% F_1 score: 86%
[20]	2022	IoT	NSL-KDD	SDPN	99	Precision: 99% Recall: 98% F_1 score: 99%
[21]	2021	IoT	BoT-IoT	FNN	99	–
[22]	2019	IoT	BoT-IoT	FNN	99	Precision: 99% Recall: 99% F_1 score: 99%
[23]	2020	IoT	BoT-IoT	CNN	99	Precision: 99% Recall: 97% F_1 score: 98%
[24]	2021	IoT	MQTT-IoT-IDS2020 and BoT-IoT	CNN	99	Precision: 99% Recall: 99% F_1 score: 99%
[25]	2020	IIoT	UNSW-NB15	DRaNN	99	Detection rate: 99%
[26]	2021	Institutional	UNSW-NB15	CNN	99	F_1 score: 99%

2.2 AMI Related Works

For the AMI environment, Rose et al. [27] presented a hybrid ADS to reduce the time complexity of an AMI environment. The proposed machine learning model was based on SVM and K-means clustering (KSVM means). This system was evaluated using the KDD Cup 99 dataset and achieved good results in terms of accuracy, precision, and the F_1 score (99%). Yao et al. [28] developed an ADS for a Smart Distribution Network (SDN). This IDS is based on a Light Gradient-Boosting Machine (LightGBM). Firstly, the system used borderline SMOTE to balance the dataset, and then it extracted features using an AE. The model was trained using the LightGBM machine learning approach. The authors used two datasets and achieved an accuracy of 99.90% with the KDD Cup 99 dataset and 99.70% with NSL-KDD.

Yao et al. [29] introduced an ADS trained with deep learning in an AMI environment. The proposed system was based on CNN and LSTM as the deep learning models. This combination increased the system's capability and performance. The accuracy results were 99.95% and 99.79%, respectively for the KDD Cup 99 and NSL-KDD datasets. Table 2 summarizes the AMI literature.

Table 2: Summary of the AMI literature

Ref.	Year	Environment	Dataset	Classifier	Accuracy	Evaluation metrics
[27]	2020	AMI	KDD Cup 99	KSVM means	99	Precision: 99% Recall: 99% F_1 score: 99%
[28]	2021	AMI	KDD Cup 99 and NSL-KDD	LightGBM	99	Time consumed: 0.005
[29]	2021	AMI	KDD Cup 99 and NSL-KDD	CNN-LSTM	99	–

The following are the main observations regarding the ADS literature reviewed for an AMI environment:

G.1 Most of the reviewed systems apply to datasets created a long time ago (KDD Cup99 in 1999 and NSL-KDD in 2009). However, the behavior of some attacks has changed since then, specifically becoming more akin to normal behavior, which negatively impacts the system performance due to their inability to detect these new attack forms.

G.2 As the number and types of attacks increase deep learning-based attack detection solutions are considered promising for enhancing performance in many application areas. However, little research (to the best of our knowledge) has been done on detecting attacks in an AMI environment.

2.3 CIC-IDS2017 Related Works

The CIC-IDS2017 dataset contains normal traffic and flows with 14 attack types. It has been used by many researchers to develop ADSs. Zhou et al. [30] used CIC-IDS2017 to design an IDS. Their model was based on feature extraction and ensemble algorithms. Firstly, they defined a dimensionality reduction algorithm called CFS-BA. This algorithm uses feature correlations to select the optimal subset. The authors used penalized attributes (PA) in an ensemble approach with two algorithms, RF and Forest, in a new model called Forest PA. Finally, the model used voting on the probability distributions to detect attacks. Their proposed CFS-BA-Forest PA model performed better than state-of-the-art models with an accuracy of 99.89% and an UN-Detected rate (UND) of 12%.

Choobdar et al. [31] also used the CIC-IDS2017 dataset to develop an IDS for Software Defined Networks (SDN). The proposed IDS was built in three stages. In the first stage, a sparse stacked AE was used for feature extraction. The second stage used the SoftMax algorithm to train and test the model. Finally, the system parameters were optimized. The system performance was evaluated with NSL-KDD as well as CIC-IDS2017. It had an average accuracy of 94.8%.

Sun et al. [32] trained an IDS using CIC-IDS2017. They used a hybrid model with a CNN and LSTM for feature extraction and classification. This deep learning-based IDS reached an overall accuracy of 98.67%.

Abdulhammed et al. [33] designed an IDS based on CIC-IDS2017. This work adopted two feature dimensionality-reduction methods: AE and Principal Component Analysis (PCA). They used Uniform Distribution Based Balancing (UDBB) to balance the high data distribution. The resulting subset from both techniques was then used to build the IDS with different algorithms: RF, Bayesian Network, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). The main contribution of this approach is that it reduced the number of dimensions in CIC-IDS2017 from 81 to 10, whilst achieving high accuracy of 98.8%.

Finally, Yulianto et al. [34] tackled the high-class imbalance of the CIC-IDS2017 dataset. They constructed an IDS based on SMOTE, PCA, and Ensemble Feature Selection (EFS). AdaBoost was used

to enhance performance. The proposed model had accuracy, precision, recall, and F_1 scores of 81.83%, 81.83%, 100%, and 90.01%, respectively. Table 3 summarizes the literature on the CIC-IDS2017 dataset.

Table 3: Summary of works using the CIC-IDS2017 dataset

Ref	Year	Feature selection	Sampling technique	No of classes	Classifier	Accuracy %	Evaluation metrics
[30]	2020	CFS-BA	–	6	Ensemble RF-forest	99.89	UND: 12%
[31]	2022	AE	–	15	SoftMax	94.8	–
[32]	2020	–	–	6	CNN-LSTM	98.6	F_1 score: 93.32%
[33]	2019	AE + PCA	UDBB	15	RF	98.8	Precision: 98.9%
[34]	2019	EFS	SMOTE	15	AdaBoost	81.8	F_1 score: 90.01% Precision: 81.8% Recall: 100%

The main observations about the papers reviewed on the CIC-IDS2017 dataset are as follows:

G.3 The dataset has highly imbalanced data. However, few works have been found that balance the data distribution.

G.4 Only one study [32] applied a deep learning model but without any feature selection or data sampling techniques. Moreover, this system classified only 6 classes.

G.5 Most of the studies have been concerned with detecting cyberattacks with high detection accuracy only. However, little work has focused on minimizing the UND, which measures how many attack flows are considered normal traffic. This work focuses on minimizing the UND due to the devastating physical effects of an attack in an AMI environment.

3 Proposed Methodology

This section presents the research methodology in two main subsections: the dataset description (which describes the training set in detail), and the proposed framework (which explains the proposed model in detail).

3.1 Dataset Description

CIC-IDS2017 was assembled by the Canadian Institute of Cybersecurity (CIC). It consists of the latest threats and features. It is one of the most important datasets since it includes more recent threat types that are not represented in the older datasets. Additionally, it has examples of the most up-to-date attack scenarios which fulfill all the criteria of real-world attacks. However, like real-world network data, it has a high degree of class imbalance. It contains 2,522,362 different network flows. Of these, 2,096,484 are benign (83.12% of the data). There are 425,878 attack flows (16.88% of the data) with 79 different features. It covers normal flow and 14 attack types. These attacks are briefly described in Table 4.

Table 4: Summary of attacks in CIC-IDS2017

Name	No of instances	Description
Benign	2,096,484	Normal traffic behavior.
DoS Hulk	172,849	DoS attack that targets a web server by creating a massive volume of obfuscated network flows.
DDoS	128,016	Multiple machines are used simultaneously to attack one target device.
PortScan	90,819	Collects information about the victim device, such as running services, by sending messages to different destination ports.
DoS GoldenEye	10,286	DoS attack using the goldeneye tool.
FTP Patator	5,933	Brute force attack to find a File Transfer Protocol (FTP) password using FTP Patator.
DoS Slowloris	5,385	DoS attack using the slowloris tool.
DoS Slowhttptest	5,228	Prevents access to a service by sending HTTP requests exceeding the server's allowed number of HTTP connections.
SSH Patator	3,219	Brute force attack to find the secure socket shell (SSH) login password using SSH patator.
Bot	1,953	Obtains remote control and manages several machines using a network of bots and Trojans.
Web Attack: Brute Force	1,470	Guesses privileged information, such as passwords, using trial and error.
Web Attack: XSS	652	Injects malicious scripts into a trusted website using a web application.
Infiltration	36	Gains unauthorized access using different infiltration tools and methods.
Web Attack: SQL Injection	21	Code injection method used with data-driven applications to insert structured query language (SQL) statements into an entry field for execution.
Heartbleed	11	Gains unauthorized access to information by adding malicious statements into OpenSSL memory.

3.2 Proposed Framework

The proposed framework consists of the following steps: (1) features are extracted with a deep learning approach, (2) the dataset is balanced in two steps (random under-sampling and SMOTE), and (3) a threat is detected by ensemble soft voting-based on RF and CNN to classify the data. The model can successfully identify a normal flow and 14 attack types. Fig. 1 illustrates the proposed model framework, which has the following phases:

- Dara pre-processing: This step transforms the raw data in the original dataset into a suitable input format for the machine learning and deep learning models.
- Normalization: The Min-Max method is proposed.
- Dataset splitting: Of the data, 70% is used for training and 30% is used for testing.
- Data distribution balancing: To enhance performance, the data distribution is balanced through under-sampling and SMOTE oversampling.
- Dimensionality reduction: To reduce the dataset's dimensionality, the most relevant features are selected with an AE. This step decreases the complexity and processing time for this high-dimensional dataset.
- Classifier training: To improve performance, the RF and CNN results undergo ensemble voting.

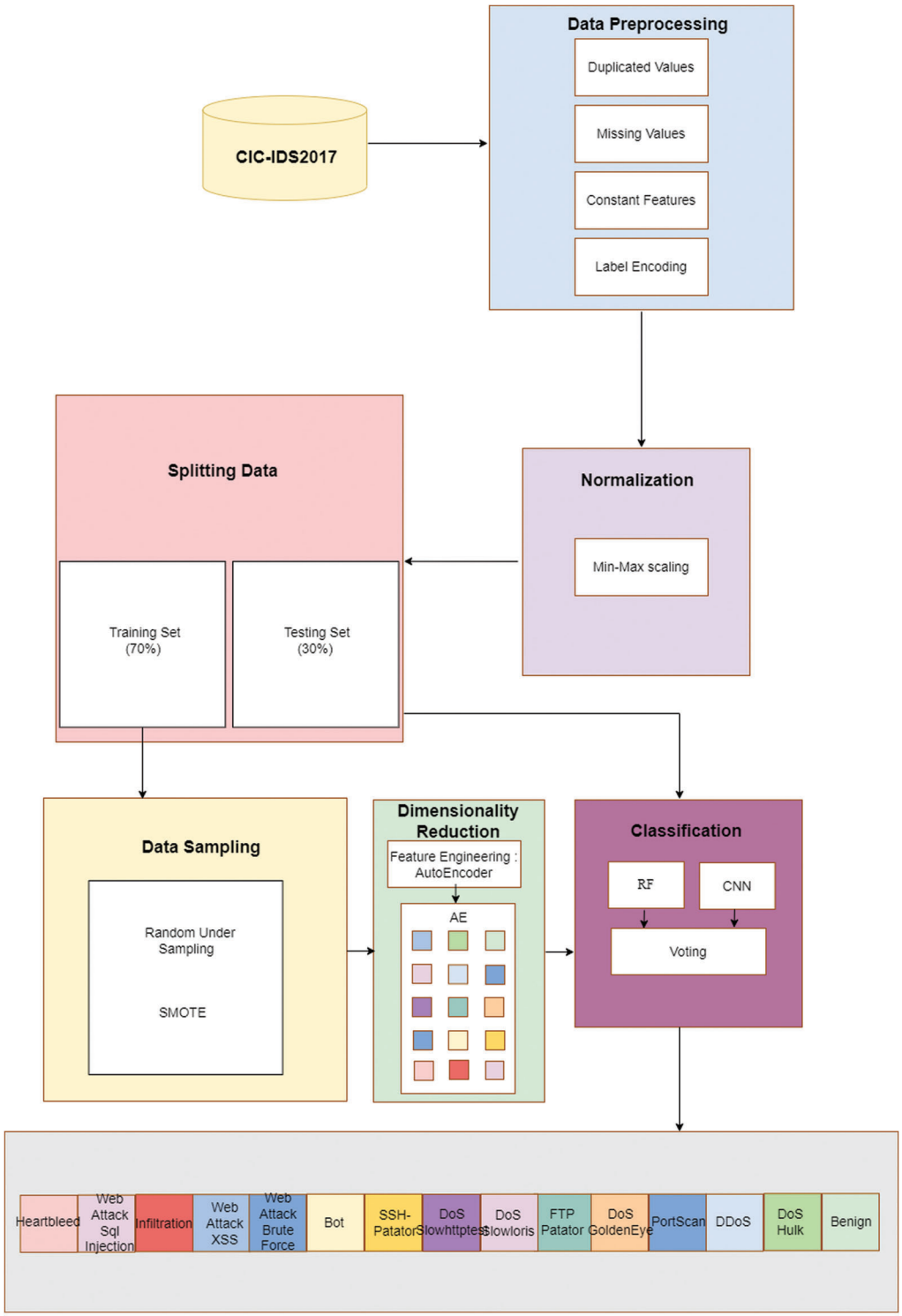


Figure 1: Proposed framework

3.2.1 Data Pre-Processing

The first step in building the model is to pre-process the data. This step addresses the noise and missing values in the data, both of which can degrade the model performance of the model. Four primary pre-processing steps are applied:

- Deleting Duplicate Rows: If a dataset has duplicate rows, the processing time when training or testing the model is increased owing to the redundant effort. Therefore, duplicates were removed to decrease the overhead. In this step, 308,381 duplicate rows were deleted from this dataset.
- Handling Missing Values: Many techniques have been proposed for filling lost or missing values. In this work, missing values were deleted to obtain a more accurate result.
- Removing Constant Features: Some features have the same values for all instances in the dataset. Eight such features were removed to decrease the training overhead: Bwd PSH Flags',' Bwd URG Flags','Fwd Avg Bytes/Bulk',' Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk',' Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate'.
- Encoding: The last step in the data pre-processing phase is to encode the text values as numbers because the classifier cannot work with text. In this dataset, only the label feature, which represents the attack type, had text values, which were encoded as in Table 5.

Table 5: CIC-IDS2017 attack codes

Name	Code	Name	Code
Benign	0	SSH patator	11
DoS Hulk	4	Bot	1
DDoS	2	Web attack: Brute force	12
PortScan	10	Web attack: XSS	14
DoS GoldenEye	3	Infiltration	9
FTP Patator	7	Web attack: SQL injection	13
DoS Slowloris	6	Heartbleed	8
DoS Slowhttptest	5		

3.2.2 Normalization

Normalization or standardization transforms the data, so it is contained in a unit sphere. Min-Max normalization was used to standardize all features values using the following formula [35]:

$$x_{i,j} = \frac{x_{i,j} - Min}{Max - Min} \tag{1}$$

where $x_{i,j}$ is the value of feature j of row i, Min is the minimum value of feature j in all rows, and Max is the maximum value of feature j in all rows.

3.2.3 Dataset Splitting

In this work, 70% of the samples were placed in the training set and 30% in the testing set. This distribution was done according to attack type, as listed in Table 6.

Table 6: Size of the training and testing set by attack type

Name	Training set	Testing set
Benign	1,467,539	628,945
DoS Hulk	120,994	51,855
DDoS	89,611	38,405
PortScan	63,573	27,246
DoS GoldenEye	7,200	3,086
FTP Patator	4,153	1,780
DoS Slowloris	3,770	1,615
DoS Slowhttptest	3,660	1,568
SSH Patator	2,253	966
Bot	1,367	586
Web Attack: Brute Force	1,029	441
Web Attack: XSS	456	196
Infiltration	25	11
Web Attack: SQL Injection	15	6
Heartbleed	8	3

3.2.4 Data Distribution Balancing

The proposed model uses two sampling steps. Firstly, a random under-sampling technique is applied only for class (0) (benign traffic). In this step, the samples were randomly deleted samples from the largest class. Secondly, SMOTE was run against all the other classes except class (4) (DoS Hulk). SMOTE was used to increase the size of these classes to that of class (4), which had 120,994 flows. SMOTE reduces the negative impact of both under and over-sampling by integrating both to generate a balanced dataset [36]. Fig. 2 represents the training dataset before and after applying the proposed data balancing technique, with 120,994 samples for each type of class. Algorithm 1 describes the proposed two-step sampling.

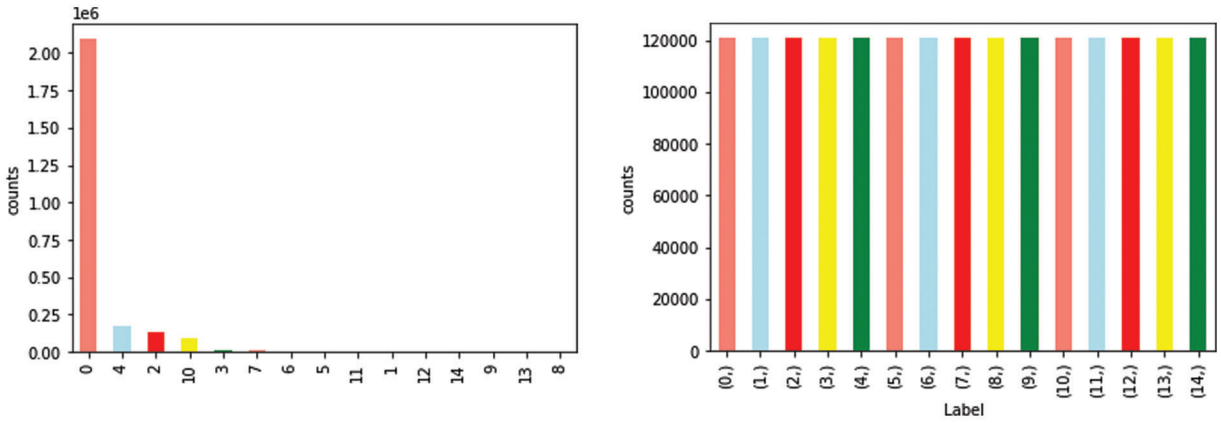
Algorithm 1: Proposed two-step data-balancing

1. Start
2. training_set, testing_set ← split [dataset, label, size = 0.3]
3. *x, y, z* ← split [training_set.class (0), training_set.class (4), training_set.drop (0, 4)]
4. *x* ← Random_Undersampling *(x, y)*, minority class *x*; the amount of undersampling *y*
5. For each *i* ← *x*
6. *R* = *i*. random delete
7. End for
8. If *R* > *y*
9. Repeat steps until *R* == *y*
10. End if

(Continued)

Algorithm 1 (continued)

Return R
$z \leftarrow$ SMOTE (z, y), minority class z; the amount of SMOTE y
For each $i \leftarrow z$
  R = Compute k nearest neighbors for i
End for
Increase z = Random select*(R)*
If $z < y$
  Repeat steps until $z == y$
End if
Return z
training_set $\leftarrow$ combine (x, y, z)
End



(a) (b)

Figure 2: Training dataset: (a) original distribution and (b) two-step balanced data

3.2.5 Dimensional Reduction and Feature Selection

CIC-IDS2017 is a high-dimensional dataset. In this phase, AE-based feature extraction, which is an unsupervised neural network is used for dimensional reduction. It has an encoder and a decoder. Fig. 3 shows the hierarchical structure of the AE. It selects and reconstructs the input features to extract the main features [37].

Let the information in the input layer be x, that in the hidden layer y, and that in the output layer z. The following formulae represent the encoding and decoding of the AE:

$$y = f(w_i x + b_i) \tag{2}$$

$$z = g(w_j y + b_j) \tag{3}$$

where w_i and w_j are the layer weights of the encoder and the decoder, respectively; b_i and b_j are their biases, and f and g are their activation functions.

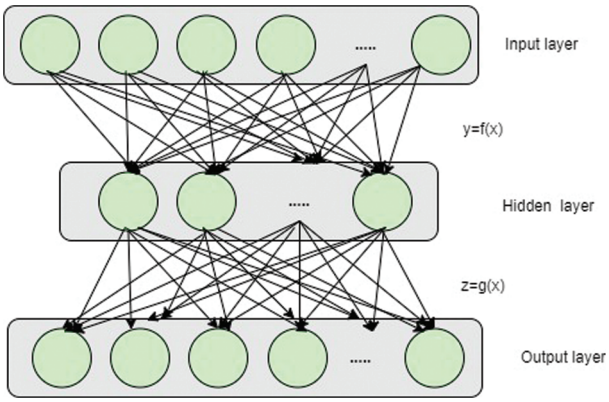


Figure 3: Hierarchical structure of the AE

3.2.6 Proposed Ensemble Voting RF–CNN Classification Model

Ensemble approaches combine the base classifiers in a final ensemble model for better results. There are different types of ensemble approaches. This work uses soft ensemble voting to assess the RF and CNN results. RF produces high results in terms of accuracy, precision, recall, and F_1 score, whereas a CNN is the best model for detecting a few sample attacks and has the lowest UND. In soft voting, the average probability of a class is the prediction output. Soft voting is described as [38]:

$$\hat{y} = \text{argmax} \sum_{N}^{j=1} \left(w_j P_j \right) \tag{4}$$

where $\hat{y}$ is the predicted class label, N is the number of classifiers, P_j is the probability for class j, and w_j is the weight assigned to the $_j$th classifier. The conceptual architecture of the proposed ensemble soft voting is shown in Fig. 4. Algorithm 2 shows the proposed voting RF–CNN model.

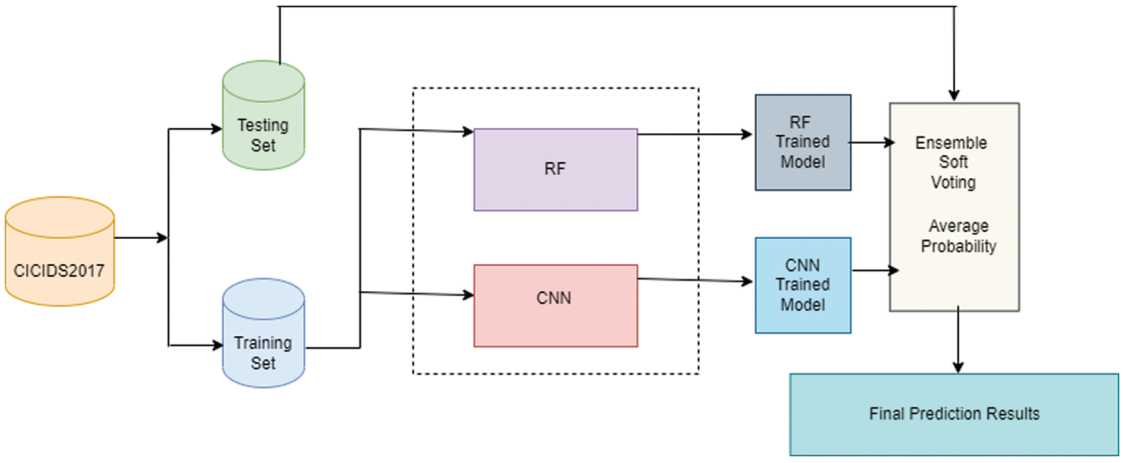


Figure 4: Conceptual architecture of the proposed ensemble soft voting RF–CNN

Algorithm 2: Proposed ensemble soft voting RF–CNN

```
1.  Start
2.  training_set, testing_set ← split [dataset, label, size = 0.3]
3.  M1 = RF (training_set, testing_set)
4.  M2 = CNN (training_set, testing_set)
5.  voting_Classifiers (M1, M2, soft)
6.  voting_Classifiers.fit (training_set)
7.  predictions = voting_Classifiers.Predict (testing_set)
8.  End
```

The proposed model uses RF and a CNN to classify attacks. The RF algorithm uses many Decision Trees (DT). It has higher accuracy compared to other traditional machine learning algorithms because it makes fewer classification errors. Other advantages of using RF include the following [39]:

(1) It can save generated forests for later use.
(2) It does not suffer from the overfitting problem.
(3) It automatically calculates accuracy and variable importance.

The final decision is the majority vote by all trees in the RF model. Fig. 5 shows the proposed architecture of the RF model.

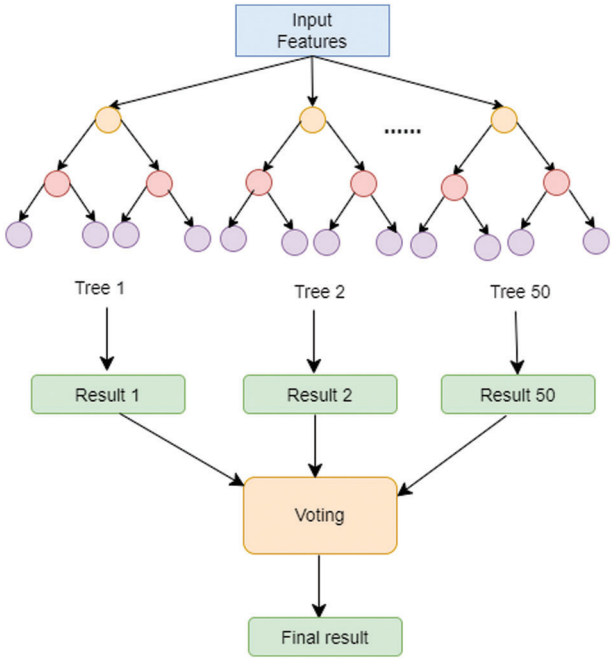


Figure 5: Architecture of the proposed RF model

Many experiments were run to find the best number of trees (k) with the highest classification accuracy. Each experiment used a random value of k in the range of (10, 100). When $k = 50$, the accuracy was highest at about 98.54%. For k above 50, the accuracy did not increase. The other RF parameters are shown in Table 7.

Table 7: Hyperparameters of the proposed RF model

Parameter	Value
max_depth	30
min_samples_leaf	1
criteria	gini
min_samples_split	2
n_jobs	5
random_state	123

CNNs are widely used deep learning algorithms. A CNN has the following types of layers: (1) convolution layers, which extract the main features, (2) pooling layers, which decrease the size of the convolved features to reduce the cost, (3) fully connected layers, which determine the class for the input layer, and (4) an output layer. A CNN has a feature learning stage and a classification stage. Each stage uses one or more layers. The feature learning stage usually combines convolutional layers and pooling layers. This stage extracts the main features from the data. These extracted features are fed into the classification stage, which has a fully connected layer to determine the class for the input layer. The architecture of a CNN with N-layers and input vector x_{n-1} is given as [40]:

$$x_n = f(w_n x_{n-1} + b_n) \tag{5}$$

An optimized set of weight vectors w_n and bias vectors b_n is used to reduce the error between the desired and the actual output, which is calculated as:

$$w_n = w_n - \eta \frac{\partial(x_n,\ x_{\text{desired}})}{\partial w_n} \tag{6}$$

$$b_n = b_n - \eta \frac{\partial e(x_n,\ x_{\text{desired}})}{\partial b_n} \tag{7}$$

Overfitting often occurs if all the features are connected to the fully connected layer. A dropout layer is used to overcome this problem. It drops a few neurons from the neural network during training to decrease the size of the model [40].

The proposed CNN model has four convolutional layers. The numbers of filters were 64, 128, 256, and 512, respectively. For each layer, the activation function is a Rectified Linear Unit (ReLU) with a kernel size of 3 and strides of 1. The ReLU is calculated as:

$$\text{ReLU} = \begin{cases} x_i, & x_i > 0 \\ 0, & x_i \leq 0 \end{cases} \tag{8}$$

Each convolutional layer is followed by a pooling layer with a pooling size of 2 and Max-pooling methods. The formula of the max-pooling layer is calculated as [41]:

$$x_t = f_{\max-\text{pooling}}\{\text{ReLU}(x_t * w_t + b_t)\} \tag{9}$$

where x_t is the output value after the max-pooling layer and $f_{\max-\text{pooling}}$ is the max-pooling operation. Each pooling layer is followed by a dropout layer with a 0.5 dropout rate. After these layers, a global pooling layer is used to reduce the dimensionality of the feature maps output by the last convolutional layer. Finally, two

fully connected layers are used. The first layer uses the ReLU activation function whereas the second one uses the SoftMax activation function, described as:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}} \tag{10}$$

where z is arbitrary with real values z_j and n is the vector size. The architecture of the proposed CNN model is shown in Fig. 6, and the hyperparameters are listed in Table 8.

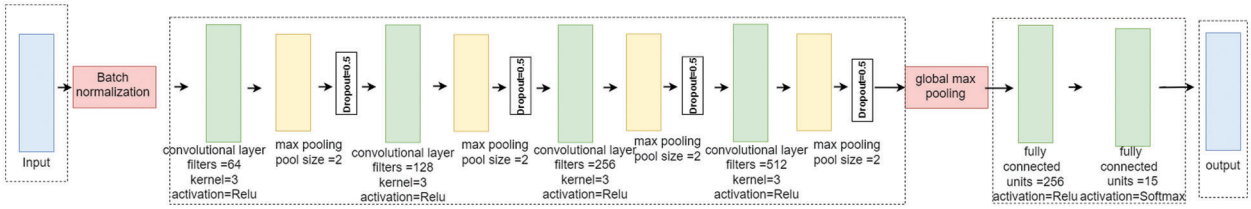


Figure 6: Architecture of the proposed CNN model

Table 8: Hyperparameters of the proposed CNN model

Parameter	Value
Batch size	512
Maximum epochs	100
Optimizer	Adam

4 Results and Discussion

As is evident from Subsection 3.1, the dataset in question is highly imbalanced. It has 2,522,362 instances of network traffic, 2,096,484 (83.12%) of which were normal traffic flows whilst 425,878 (16.88%) were instances of malicious traffic flows. Owing to the preponderance of normal traffic flows and the comparatively small number of attack flows, errors in detection would not significantly reduce the overall accuracy of the model. Accordingly, increasing the accuracy or other performance metrics in the model would produce misleading results thus suggesting an inability to detect attacks because of the imbalanced dataset, which in turn means the ADS may not be able to detect all attacks. Moreover, this research focused on enhancing the detection per class and the UND, which more accurately reflects the system's ability to identify attacks, as a low UND indicates the model can detect most incoming attacks.

Building on the related works described in Section 2 and examining the effect of the proposed two-step data distribution balancing algorithm, this work utilizes six state-of-the-art machine learning and deep learning algorithms: RF, LightGBM, XGboost, CNN, LSTM, and biLSTM. The efficacy of these algorithms is evaluated with the original dataset and dimensional reduction. The results are listed in Table 9. All six algorithms achieved high precision, recall, F_1 score, and accuracy due to a large number of normal samples compared to the attacks in the testing set.

Hence, although these models can classify most normal flows accurately, the high UND values suggest that many attacks were undetected because the models had not been trained correctly. This result is clear in Table 10, which shows the detection accuracy for each class for the six algorithms with the original dataset. Fig. 7 represents these results graphically.

Table 9: Performance (%) of the six state-of-the-art algorithms in the original dataset

Algorithm	Feature Eng.	Precision	Recall	F_1 _score	Acc	UND
RF	AE	99	99	99	98.71	18
XGboost	AE	99	99	99	98.57	28
LightGBM	AE	95	95	95	95.05	55
CNN	AE	98	97	97	97.25	40
LSTM	AE	99	99	98	98.50	32
biLSTM	AE	98	98	98	98.43	35

Table 10: Detection accuracy (%) for each class for each algorithm using the original CIC-IDS2017 dataset (values less than 50% in bold)

	RF	XGboost	LightGBM	CNN	LSTM	biLSTM
Benign	98.91	99.21	98.03	98.40	98.80	99.46
Bot	94.21	**37.36**	**0**	**35.88**	**38.87**	**37.93**
DDoS	99.98	97.76	84.32	96.18	98.89	99.46
DoS GoldenEye	99.12	92.93	90.41	96.79	98.93	96.76
DoS Hulk	99.96	98.17	86.59	94.23	98.84	98.97
DoS Slowhttptest	99.28	95.74	71.37	94.35	98.60	**46.14**
DoS Slowloris	99.20	91.22	79.30	89.14	86.07	95.80
FTP-Patator	99.46	98.52	**0**	99.23	99.11	98.83
Heartbleed	99.89	99.66	**0**	**0**	99.64	99.98
Infiltration	**0**	**0**	**0**	**0**	**0**	**0**
PortScan	99.28	88.91	68.79	99.65	92.25	92.76
SSH-Patator	98.89	90.95	91.00	91.43	91.32	91.40
Web Attack Brute Force	75.76	81.71	**0**	**10.09**	**11.32**	**11.43**
Web Attack Sql Injection	**32.66**	**0**	**0**	**0**	**0**	**0**
Web Attack XSS	**40.27**	**0**	**0**	**0**	**0**	**0**

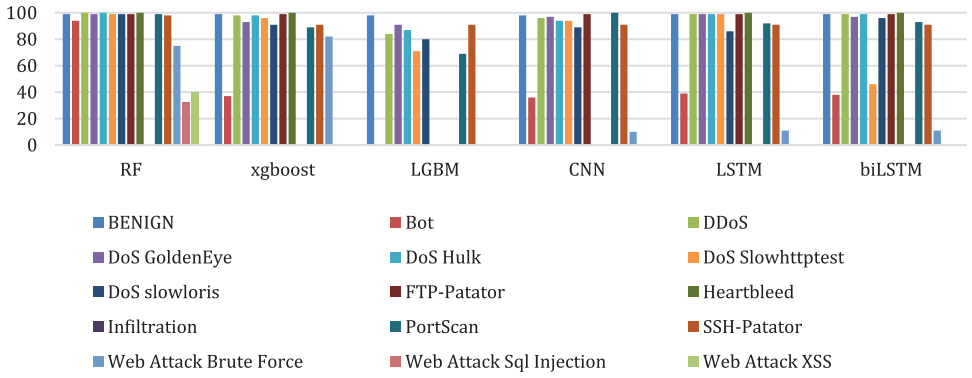


Figure 7: Detection accuracy for each class for each algorithm using the original CIC-IDS2017 dataset

The algorithms were next trained and evaluated after the proposed data distribution balancing, as shown in Table 11. The final row shows the results for the proposed method. The proposed two-step sampling leads to a noteworthy improvement in the UND with high accuracy score, indicating the improved ability of each model to detect attacks. Table 12 lists the accuracy per class for the algorithms after the two-step sampling. This data is represented graphically in Fig. 8.

Table 11: Performance (%) of the six state-of-the-art Algorithms after the two-step sampling

Algorithm	Feature Eng.	Precision	Recall	F_1 _score	Acc	UND
RF	AE	99	99	99	**98.54**	12
XGboost	AE	98	95	96	94.91	11
LightGBM	AE	99	98	98	98.05	14
CNN	AE	97	93	95	93.09	**8**
LSTM	AE	98	96	97	96.04	12
biLSTM	AE	98	95	96	94.89	10
Proposed Voting	**AE**	**99**	**98**	**98**	**98.29**	**8**

Table 12: Detection accuracy (%) for each class for each algorithm after the two-step sampling

	RF	XGboost	LightGBM	CNN	LSTM	biLSTM	Voting RF–CNN
Benign	98.34	93.72	98.00	92.23	94.82	93.96	**98.12**
Bot	97.77	99.31	98.78	99.88	99.24	99.48	**97.11**
DDoS	99.88	98.89	99.88	97.44	99.88	99.87	**99.03**
DoS GoldenEye	99.20	99.41	99.65	98.16	98.97	99.44	**99.29**
DoS Hulk	99.50	99.12	99.82	99.57	99.83	99.67	**99.89**
DoS Slowhttptest	99.48	99.17	99.92	98.85	99.42	99.48	**99.43**
DoS Slowloris	99.29	98.76	98.75	96.20	98.58	97.65	**99.45**
FTP-Patator	99.89	99.60	99.66	99.66	99.66	99.66	**99.99**
Heartbleed	99.52	67.32	67.35	99.99	67.08	99.93	**99.99**
Infiltration	54.53	91.22	63.63	82.40	55.03	63.72	**63.92**
PortScan	99.88	99.66	99.87	99.75	99.81	99.62	**99.87**
SSH-Patator	98.78	98.55	99.58	92.79	91.51	93.30	**99.34**
Web Attack Brute Force	62.14	53.92	63.71	38.32	49.16	68.26	**63.44**
Web Attack Sql Injection	67.26	50.16	33.33	99.73	83.33	83.33	**82.89**
Web Attack XSS	52.10	80.02	58.71	90.16	78.67	48.01	**52.17**

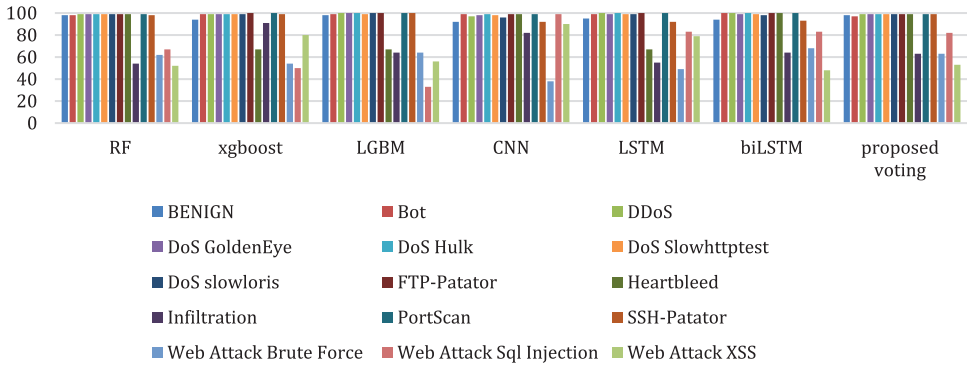


Figure 8: Detection accuracy for each class for each algorithm after the two-step sampling

Table 12 shows that, compared to RF, the proposed voting model had better detection performance against DoS GoldenEye, DoS Hulk, DoS Slowloris, FTP Patator, Heartbleed, Infiltration, SSH Patator, Web attack: Brute force, Web attack: XSS, and Web attack: SQL injection. Furthermore, the proposed voting model had better accuracy than did the CNN for benign flows and against DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris, FTP Patator, PortScan, SSH Patator, and Web attack: Brute force. The proposed voting model classified benign traffic with 98.12% accuracy, which was very close to models that are trained with all the benign flows in the original dataset, as shown in Table 10. Fig. 9 shows the confusion matrix for the proposed model. The proposed model can detect attacks better than both other machine learning and deep learning models.

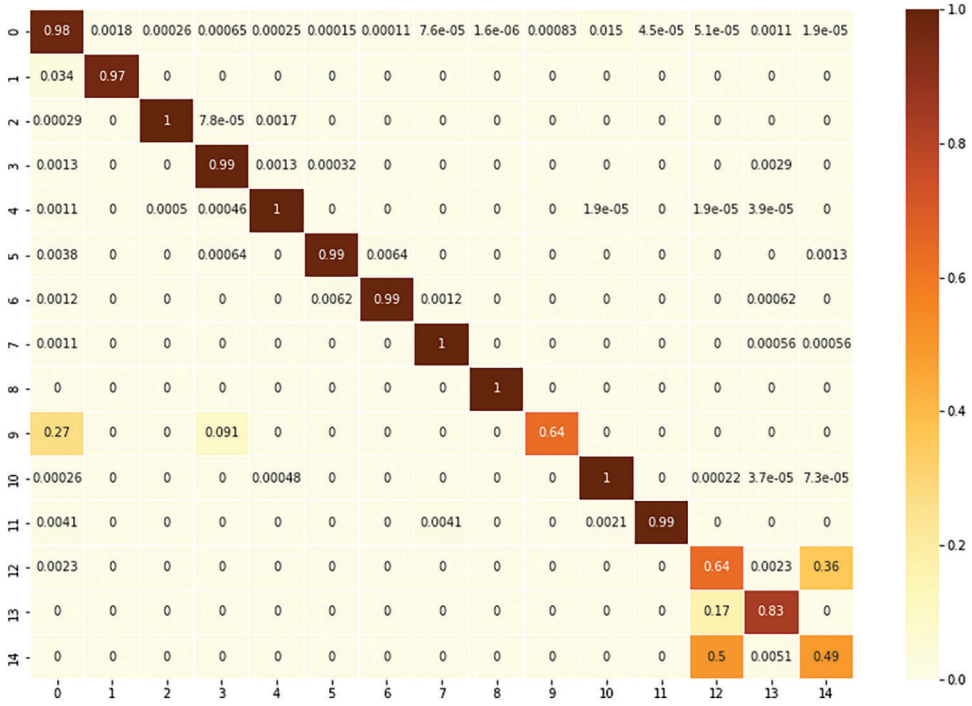


Figure 9: Confusion matrix for the proposed ensemble voting RF–CNN model

Table 13: Ensemble method performance (%) after two-step sampling

Algorithm	Precision	Recall	F1_score	Acc	UND
Stacked	99	99	99	98.23	16
Hard voting	99	98	98	98.14	16
Soft voting —Proposed	**99**	**98**	**98**	**98.29**	**8**

Three ensemble strategies were assessed in this work: soft voting, hard voting, and stacked ensemble. The results are displayed in Table 13. Soft voting was the most effective ensemble strategy.

Finally, Table 14 compares the proposed model with the other detection methods (listed in Table 3) tested with the same dataset. Each method's efficacy was assessed according to the number of attacks detected and classified. An ability to classify various attack types is reflective of a model's strength. Furthermore, although accuracy and UND are important metrics when designing an ADS for an AMI environment, accuracy can become skewed if the data are imbalanced. UND and accuracy per class, by contrast, reflect a system's ability to distinguish malicious traffic from normal flows, particularly in small-scale attacks that lack training. An AMI is a critical and sensitive system: indeed, an attack against it may cause serious damage, and the network may take a long time to recover. Therefore, AMIs must be very careful about which network flows they accept. Table 14 reveals that the proposed model performed better than the other methods, with one, minor exception. The method described in [33] had a higher accuracy but lower precision and the UND was not identified.

Table 14: Proposed model compared with other methods (best values in bold)

Ref.	Other methods	Ensemble soft voting RF–CNN
[30]	6 classes Imbalanced data UND: 12% **Accuracy: 99.89%**	**15 classes** **Balanced data** **UND: 8%** Accuracy: 98.29%
[31]	Imbalanced data UND not identified Accuracy: 94.8%	**Balanced data** **UND: 8%** **Accuracy: 98.29%**
[32]	6 classes Imbalanced data. F_1 score: 93.32% UND not identified **Accuracy: 98.6%**	**15 classes** **Balanced data** **F_1 score: 98%** **UND: 8%** Accuracy: 98.29%
[33]	UND not identified Precision: 98.9% **Accuracy: 98.8%**	**UND: 8%** **Precision: 99%** Accuracy: 98.29%
[34]	UND not identified Accuracy: 81.8%	**UND: 8%** **Accuracy: 98.29%**

5 Conclusion and Future Works

A smart grid is proposed to enhance the efficiency, reliability, stability, and security of traditional energy power systems. An AMI exchanges sensitive information through various types of communication networks with different technologies, which exposes it to different types of attacks. Detecting anomalies in network flow patterns is a difficult task, especially in the security management of a large network. The current IDSs have various drawbacks, such as high False Positive and False Negative, difficult feature engineering, and low efficiency when pre-processing the data. These problems arise because the range of possible normal behaviors of the multiple devices on the network is very large and dynamic. Additionally, many ADSs cannot process complex data and have poor classification performance.

To improve anomaly detection, this paper proposed ensemble voting based on deep learning algorithms to enhance system performance. The proposed system was developed in four stages: data preprocessing, dimensional reduction using AE-based feature engineering, data distribution balancing using under-sampling and SMOTE oversampling, and multiclass classification using the ensemble voting RF–CNN model. The findings show that the proposed model can detect 14 different attack types better than other machine learning and deep learning models. It had high performance in terms of accuracy, precision, recall, F_1 score, and UND. It could be used to build an efficient IDS for an AMI network. In future work, we recommend training the model on more attacks from other datasets. Moreover, we recommend enhancing the capability of the model to deal with zero-day attacks.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Tudor, M. Almgren and M. Papatriantafilou, "The influence of dataset characteristics on privacy preserving methods in the advanced metering infrastructure," *Computers & Security*, vol. 76, ISSN 0167-4048, pp. 178–196, 2018.

[2] J. L. Gallardo, M. A. Ahmed and N. Jara, "Clustering algorithm-based network planning for advanced metering infrastructure in smart grid," *IEEE Access*, vol. 9, pp. 48992–49006, 2021.

[3] W. Tong, L. Lu, Z. Li, J. Lin and X. Jin, "A survey on intrusion detection system for advanced metering infrastructure," in *Proc. 6th Int. Conf. on Instrumentation and Measurement, Computer, Communication and Control*, IMCCC, Harbin, China, pp. 33–37, 2016.

[4] W. Wisetsri, S. Qamar, G. Verma, D. Verma, V. Kakar *et al.,* "Electricity theft detection and localization in smart grids for industry 4.0," *Intelligent Automation & Soft Computing*, vol. 33, no. 3, pp. 1473–1483, 2022.

[5] G. Bendiab, K. P. Grammatikakis, I. Koufos, N. Kolokotronis and S. Shiaeles, "Advanced metering infrastructures: Security risks and mitigation," in *Proc. of the 15th Int. Conf. on Availability, Reliability and Security*, Dublin, Ireland, New York, NY, USA, ACM, Association for Computing Machinery, pp. 2–21, 2020.

[6] S. Tufail, I. Parvez, S. Batool and A. Sarwat, "A survey on cybersecurity challenges, detection, and mitigation techniques for the smart grid," *Energies*, vol. 14, no. 18, pp. 1–22, 2021.

[7] C. Bekara, "Security issues and challenges for the IoT-based smart grid," *Procedia Computer Science*, vol. 34, ISSN 1877-0509, pp. 532–537, 2014.

[8] J. Li, J. Huang, L. Tian and J. Wang, "Application of new active defense technology in power information network security," *IOP Conference Series: Materials Science and Engineering*, vol. 750, no. 1, pp. 012156, 2020.

[9] L. Wei, L. P. Rondon, A. Moghadasi and A. I. Sarwat, "Review of cyber-physical attacks and counter defense mechanisms for advanced metering infrastructure in smart grid," in *Proc. IEEE Power Engineering Society Transmission and Distribution Conf.*, Denver, USA, pp. 1–9, 2018.

[10] R. R. Chaudhari and S. P. Patil, "Intrusion detection system: Classification, techniques and datasets to implement," *International Research Journal of Engineering and Technology*, vol. 4, no. 2, pp. 1860–1866, 2017.

[11] S. M. Naser, Y. H. Ali and D. A. J. Obe, "Deep learning model for cyber-attacks detection method in wireless sensor networks," *Periodicals of Engineering and Natural Sciences*, vol. 10, no. 2, pp. 251–259, 2022.

[12] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *Proc. Int. Conf. on Big Data Analytics and Computational Intelligence*, Cirala, India, pp. 57–60, 2017.

[13] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, pp. 1–20, 2021.

[14] Intrusion detection evaluation dataset (CIC-IDS2017), Canadian Institute for Cybersecurity 2017. https://www.unb.ca/cic/datasets/ids-2017.html (accessed Feb. 26, 2022).

[15] M. Hachimi, G. Kaddoum, G. Gagnon and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5G cloud radio access networks," in *Proc. Int. Symp. on Networks, Computers and Communications*, Montreal, QC, USA, pp. 1–5, 2020.

[16] M. Anwer, S. M. Khan, M. U. Farooq and W. Waseemullah, "Attack detection in IoT using machine learning," *Engineering, Technology & Applied Science Research*, vol. 11, no. 3, pp. 7273–7278, 2021.

[17] T. Salman, D. Bhamare, A. Erbad, R. Jain and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Proc. 4th IEEE Int. Conf. on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE Int. Conf. of Scalable and Smart Cloud*, New York, NY, USA, pp. 97–103, 2017.

[18] S. M. Othman, F. M. Ba-Alwi, N. T. Alsohybe and A. Y. Al-Hashida, "Intrusion detection model using machine learning algorithm on big data environment," *Journal of Big Data*, vol. 5, no. 1, pp. 2–12, 2018.

[19] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad *et al.,* "IoT DoS and DDoS attack detection using ResNet," in *Proc. 3rd IEEE Int. Multi-Topic Conf., INMIC 2020*, Bahawalpur, Pakistan, pp. 1–6, 2020.

[20] Y. Otoum, D. Liu and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, pp. 2–16, 2022.

[21] M. Ge, N. F. Syed, X. Fu, Z. Baig and A. Robles-Kelly, "Towards a deep learning-driven intrusion detection approach for internet of things," *Computer Networks*, vol. 186, pp. 107784, 2021.

[22] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo *et al.,* "Deep learning-based intrusion detection for IoT networks," in *Proc. IEEE Pacific Rim Int. Symp. on Dependable Computing*, Kyoto, Japan, vol. 2, pp. 256–265, 2019.

[23] A. Derhab, A. Aldweesh, A. Z. Emam and F. A. Khan, "Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering," *Wireless Communications and Mobile Computing*, vol. 2020, no. 6689134, pp. 2–16, 2020.

[24] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[25] S. Latif, Z. Idrees, Z. Zou and J. Ahmad, "DRaNN: A deep random neural network model for intrusion detection in industrial IoT," in *Proc. Int. Conf. on UK-China Emerging Technologies*, Glasgow, UK, 20–21, pp. 1–4, 2020.

[26] E. Tufan, C. Tezcan and C. Acartürk, "Anomaly-based intrusion detection by machine learning: A case study on probing attacks to an institutional network," *IEEE Access*, vol. 9, pp. 50078–50092, 2021.

[27] T. Rose, K. Kifayat, S. Abbas and M. Asim, "A hybrid anomaly-based intrusion detection system to improve time complexity in the internet of energy environment," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 124–139, 2020.

[28] R. Yao, N. Wang, Z. Liu, P. Chen, D. Ma *et al.,* "Intrusion detection system in the smart distribution network: A feature engineering-based AE-LightGBM approach," *Energy Reports*, vol. 7, pp. 353–361, 2021.

[29] R. Yao, N. Wang, Z. Liu, P. Chen and X. Sheng, "Intrusion detection system in the advanced metering infrastructure: A cross-layer feature-fusion CNN-LSTM-based approach," *Sensors*, vol. 21, no. 2, pp. 1–17, 2021.

[30] Y. Zhou, G. Cheng, S. Jiang and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Computer Networks*, vol. 174, pp. 1–21, 2020.

[31] P. Choobdar, M. Naderan and M. Naderan, "Detection and multi-class classification of intrusion in software-defined networks using stacked auto-encoders and CICIDS2017 dataset," *Wireless Personal Communications*, vol. 123, no. 1, pp. 437–471, 2022.

[32] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu *et al.,* "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, ID 8890306, pp. 1–11, 2020.

[33] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, pp. 2–27, 2019.

[34] A. Yulianto, P. Sukarno and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CICIDS2017 dataset," *Journal of Physics: Conference Series*, vol. 1192, no. 1, pp. 2–9, 2019.

[35] M. Haggag, M. M. Tantawy and M. M. S. El-Soudani, "Implementing a deep learning model for intrusion detection on apache spark platform," *IEEE Access*, vol. 8, pp. 163660–163672, 2020.

[36] A. Aldegheishem, M. Anwar, N. Javaid, N. Alrajeh, M. Shafiq *et al.,* "Towards sustainable energy efficiency with intelligent electricity theft detection in smart grids emphasising enhanced neural networks," *IEEE Access*, vol. 9, pp. 25036–25061, 2021.

[37] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu *et al.,* "AE-MLP: A hybrid deep learning approach for DDoS detection and classification," *IEEE Access*, vol. 9, pp. 146810–146821, 2021.

[38] S. Karlos, G. Kostopoulos and S. Kotsiantis, "A soft-voting ensemble-based co-training scheme using static selection for binary classification problems," *Algorithms*, vol. 13, no. 1, pp. 26, 2020.

[39] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[40] C. E. Galván-Tejada, L. Calzada, A. Domínguez, R. Quintanar, H. García *et al.,* "Estimation of indoor location through magnetic field data: An approach based on convolutional neural networks," *ISPRS Int J Geoinf*, vol. 9, no. 4, pp. 226, 2020.

[41] R. Almarshdi, L. Nassef, E. Fadel and N. Alowidi, "Hybrid deep learning based attack detection for imbalanced data classification," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 297–320, 2023.