ISSN: 0067-2904

Adapted CNN-SMOTE-BGMM Deep Learning Framework for Network Intrusion Detection using Unbalanced Dataset

Waad F. Kamil* , Imad J. Mohammed
Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq.

Received: 23/8/2022 Accepted: 31/10/2022 Published: 30/9/2023

Abstract
This paper proposes a method for improving network security by introducing an Intrusion Detection System (IDS) framework between end devices. It considers big data challenges and the difficulty of updating databases to uncover new threats to firewalls and detection systems. The proposed framework introduces a supervised network using CNN and the UNSW-NB15 dataset. Recursive Feature Elimination (RFE) and Extreme Gradient Boosting (XGB) were used to save time and resources. The Synthetic Minority Oversampling Technique (SMOTE) and Bayesian Gaussian Mixture Model (BGMM) reduce bias toward the majority class of the dataset. The results show that this model performs better than other methods, with 98.80% accuracy for binary classification and 96.49% for classification into multiple groups.

Keywords: Bayesian Gaussian Mixture Model, Convolutional Neural Network, Deep Learning, Extreme Gradient Boosting, Machine Learning, Intrusion Detection system, Recursive Feature Elimination, Synthetic Minority Oversampling Technique.

إطار عمل التعلم العميق CNN-SMOTE-BGMM لاكتشاف اختراق الشبكة باستخدام مجموعة بيانات غير متوازنة

وعد فلاح كامل * , عماد جاسم محمد
قسم علوم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة
تقترح هذه الورقة منهجية لتحسين أمن الشبكة من خلال تقديم إطار نظام لكشف التسلل (IDS) كطبقة وسطى بين الأجهزة الطرفية. ويأخذ في الاعتبار صعوبة تحديث قواعد البيانات للكشف عن التهديدات الجديدة التي تصيب جدران الحماية وأنظمة الكشف ، بالإضافة إلى تحديات البيانات الضخمة. يقدم الإطار المقترح شبكة IDS خاضعة للإشراف تعتمد على تقنية التعلم العميق للشبكات العصبية التلافيفية (CNN) باستخدام مجموعة بيانات(UNSW-NB15) يقوم بتنفيذ إلغاء الميزة (RFE) مع تعزيز التدرج (XGB) لتقليل استهلاك الموارد والوقت. تقليل التحيز تجاه الفئة الأكبرمن خلال الجمع بين تقنية الإفراط في أخذ عينات الأقلية (SMOTE) مع نموذج الخليط الغاوسي (BGMM) لحل مشكلة عدم توازن البيانات. توضح النتائج أن

*Email: waad.falaah1201a@sc.uobaghdad.edu.iq

هذا النموذج يتفوق بشكل كبيرعلى الأساليب الحالية ، ويحقق معدلات دقة في التصنيف الثنائي تصل إلى ٪98.80 و ٪96.49 في التصنيف المتعدد.

1. Introduction

The improved standard of living is attributable to technological advancements and remote network connectivity. However, these advancements introduce new threats. Therefore, active defense against attacks on computer networks are becoming more crucial than ever. A substantial analytical study needed to address these challenges was conducted on a network's safety can be guaranteed by using intrusion detection systems (IDS), which are separate systems that offer local network services due to their ability to manage the alarming rise in the number of unexpected attacks.

According to CyberEdge's research on cyber threats before 2022, attacks on worldwide networks have increased over the past five years (Fig. 1), with the Cyber Severity Threat Index revealing increased risk over previous years (Fig. 2). Denial of Service attacks and ransomware are examples of assaults that seek to steal or destroy data [1]. Network IDS (NIDS), host IDS (HIDS), and hybrid IDS (NIDS + HIDS) are three methods of network traffic analysis and monitoring used to safeguard a system against threats [2]. Machine-learning-based systems (supervised, unsupervised, and reinforced) are used to identify threats. Supervised learning produces a model based on label data that is then categorized by algorithms investigating the relationship between continuous variables (logistic regression).

Consequently, detecting new threats is challenging because the shallow structure, which comprises only one hidden layer, cannot extract features. As the amount of unbalanced data increases, detection accuracy degrades, and false alarm rates (FAR) rise. Recent breakthroughs have led to the design of a deep learning model, a subclass of machine learning that involves artificial neural networks with numerous hidden layers. The network evolves as further hidden layers are added to the artificial neural network architecture, producing the deep neural network (DNN). This means that convolutional neural networks (CNN) models have a greater detection and accuracy rate with fewer false positives. The extensive network architecture improves as data sets grow larger [3].

This paper examine the feasibility of essential feature extraction and data balancing CNN deep learning architectures for network security solutions that scan network traffic and alert when infractions are detected. This is accomplished by utilizing the UNSW-NB15 dataset, which incorporates real-world data and attack activity generated in a research laboratory.

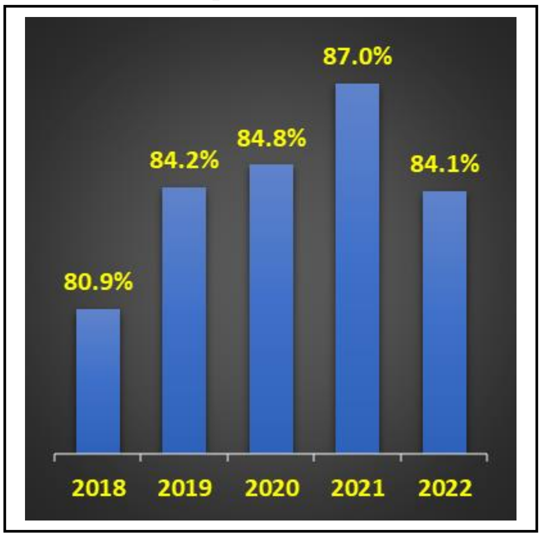


Figure -1 Percent of organizations affected by ransomware [1].

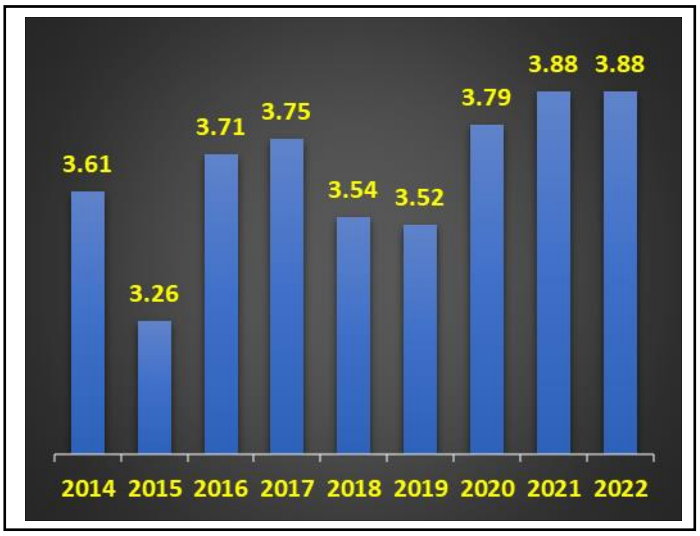


Figure -2 Security Concern Index and Cyber Threat [1].

paper is organized in the following order: Section 2 examines other work in this field and the theoretical background. In Section 3, the phases of the proposed system are explained. The empirical findings are discussed in Section 4, with conclusions offered in Section 5.

1. Related Work

Several analytical research papers on deep learning-based intrusion detection systems have recently been published. This field of study has increased in importance as the ability to learn and adapt makes it extremely effective in combatting the increased number of unforeseen attacks. Because it was developed using deep learning techniques, IDS performs more effectively and accurately.

Wang et al.[4] used Deep Multi-scale CNN (DMCNN) to create a NIDS. DMCNN extracts features from high-dimensional data using multiple convolution kernels. The batch normalization method is utilized in this strategy to optimize the network. Raw data was used to structure the learning rate and extract characteristics. The NSL-KDD dataset was used to demonstrate that their model has an accuracy rate of 94.65%, a high precision rate, and a low false positive rate (FPR).

Using the NSL-KDD dataset, Naseer et al.[5] created an IDS model with three distinct DNNs (CNN, autoencoder, and RNN). The experiment was run with three standard machine learning techniques (decision tree, support vector machine, and k-nearest neighbor) to serve as a benchmark. Using two test datasets (NSLKDDTest+ and NSLKDDTest21), the training performances were compared; LSTM demonstrated the highest level of accuracy, at 89%, compared to DCNN, which reached 85%. The accuracy of both the decision tree, SVM, and k-nearest neighbor was 82%.

Author Y. Jia et al.[6] outlines a method of creating a novel deep neural network (NDNN) model based on intrusion detection. NDNN with four hidden layers is utilized to capture and categorize the invasion features of the KDD99 and NSL-KDD training data. Experiments determine additional parameters. The NDNN-based IDS outperforms approaches based on standard machine learning algorithms with a 99.9% accuracy rate. However, there are too many restrictions and increasing nodes and layers increases computing time exponentially.

[7] discusses NIDS built on a CNN model that utilizes deep learning. The design features a convolutional layer that is double stacked, as well as Max Pooling and Dropout. The most recent simulated network traffic dataset, UNSW-NB15, was divided into training sets of 70% and testing sets of 30%. The models demonstrated that the proposed approach obtained an overall accuracy of 95.4%, with 95.6% achieved for the user-defined multiclass categorization. There is an opportunity for improvement through the application of feature reduction approaches and the balance of datasets.

Additionally, [8] proposed a deep learning strategy, a method for constructing an effective IDS using a one-dimensional convolutional neural network (1DCNN) with 1, 2, and 3 layers. The model was evaluated using the UNSW NB15 IDS dataset, and it was allowed to run for up to 200 epochs with a learning rate of 0.0001 on both imbalanced and balanced data. The suggested model was compared to two other models: Random Forest (RF) and Support Vector Machine (SVM). In contrast with the traditional machine learning classifiers, this alternative design demonstrated superior performance. The primary reason for this is that CNN can extract high-level features. The accuracy of the model's three hidden layers was

85.86% before the data was balanced, and it increased to 91.2% after the dataset was balanced.

In addition, A.Gumaei[9] asserts that NIDS is essential when it comes to shielding computer networks from intrusions and other forms of harmful activity.

This research proposes a novel two-stage deep learning (TSDL) model. The model is built on a stacked auto-encoder and uses a SoftMax classifier. Using a probability score to ascertain which stage makes the ultimate decision, it is established that the initial stage is responsible for determining whether network traffic is normal or abnormal. The experiments utilize the KDD99 and UNSW-NB15 datasets. The outcomes reveal that the suggested model is superior, with an accuracy of up to 99.996% on the KDD99 dataset and 89.134% on the UNSW-NB15 dataset.

1.1 Theoretical Background

1.1.1 Sigmoid and SoftMax Activation Function (AF)

- **Sigmoid:** A mathematical function with an 'S'-shaped curve is called a sigmoid curve for binary classification functions. The approach to representing all real numbers from 0 to 1 in a probability domain.

- **SoftMax:** unilized for multi-classification functions; sometimes called the normalized exponential function [10]. Normalizes the input into a probability distribution that sums to 1. The SoftMax and sigmoid functions take X Represent as their input vector. It has x elements from x classes, as computed below:

1.1.2 $$\text{SoftMax} = \frac{e^x}{sum(e^x)} \quad (1) \qquad \text{Sigmoid} = \frac{1}{1+e^{-x}} \quad (2)$$

The error or loss function trains the samples, calculates the output based on the training inputs, and then compares the output to the real label. Developing a function that evaluates errors during model training is essential. Determining the loss function for a DNN is dependent on the task and objectives, as different networks have different predictions based on their inputs. Calculating the likelihood of the model error is a common way to determine classification loss, which is equal to the percentage of incorrectly categorized inputs in the dataset[10][11]. The following loss functions are frequently employed in classification:

- **Binary cross-entropy (Bce):** used when dealing with two-class or binary classification problems. The output is a probability between 0 and 1.
- **Categorical cross-entropy (Cce):** a multiclass classification (more than two classes) problem that generates a one-hot array containing the most likely match for each category.
- **Sparse categorical cross-entropy (Scc)**: multiclass classification (more than two classes) problem that generates a category index for the most likely matching category.

1.1.3 Evaluation Metrics

The confusion matrix considering the calculated predicted class versus the actual class variables defines various performance metrics.

- True positive (TP): The number of harmful codes that have been accurately discovered.
- True negative (TN): The number of innocuous codes successfully identified.
- False-positive (FP): The number of times a detector incorrectly identifies a benign file as malware.

• False-negative (FN): The number of malicious codes detected by a detector incorrectly since the virus is new and no signature is yet accessible.

➢ **Accuracy:** A metric that measures the model's performance across all classes; it is particularly beneficial when all classes are of similar significance. Accuracy is calculated by dividing the number of accurate predictions by the number of incorrect ones. (3)

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

➢ **Precision:** described as the ratio between the number of positive samples correctly classified and the total number of samples classified as positive. The precision measures the model's accuracy in classifying a sample as positive.

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (4)$$

➢ **Recall:** calculated as the ratio between the number of positive samples correctly classified as positive and the total number of positive samples. The recall measures the model's ability to detect positive samples.

$$\text{Recall} = \frac{TP}{TP+FN} \qquad (5)$$

➢ **F1_Score:** considers the harmonic means between precision and recall to combine them into a single metric. This score utilizes both incorrectly classified positives and incorrectly classified negatives.

$$\mathbf{F1_{Score}} = \frac{2TP}{2TP+FP+FN} \qquad (6)$$

➢ **False Alarm Rate (FAR):** calculates the proportion of normal connection records marked as attacks relative to the total number of normal connection records. The FAR ranges between [0, 1], where a lower value indicates a superior machine learning model, It is defined as follows:

$$\mathbf{FAR} = \frac{FP}{FP+TN} \qquad (7)$$

It is common practice to measure classification performance using the confusion matrix. Accuracy is the most frequently observed quality. However, accuracy is only appropriate when the distribution of data is balanced. When the data is unbalanced, the F1 score is the most accurate minority representation statistical average. The rating quality of two or more categories can be measured using this method in machine learning. It combines precision and recall in a metric that calculates the harmonic mean between the two. The receiver operator characteristic (ROC) curve is a binary classification issue evaluation scale. The One vs. All classification can classify situations with many categories and plot the true positive rate (TPR) against the false positive rate (FPR). To summarize the ROC curve, the area under the curve (AUC) represents the classifier's ability to discriminate between different categories, as shown in Figure 3. The model differentiates between the positive and negative categories [12], [13].

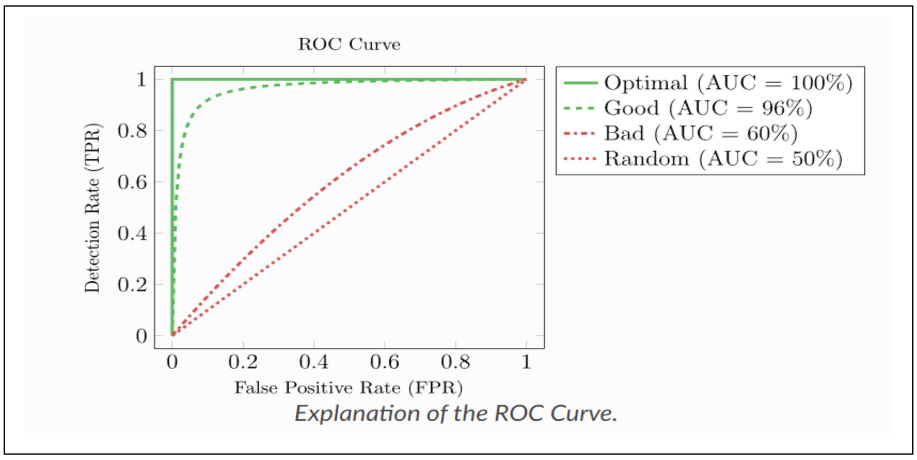


Figure 3: Explanation of the ROC Curve [13].

2. Proposed Methodology Framework

This section describes the structure of the IDS, which comprises five primary stages (illustrated in Fig. 4): Preprocessing, the method of selecting hybrid features, data segmentation and balancing, the classifier, and finally, the evaluation.

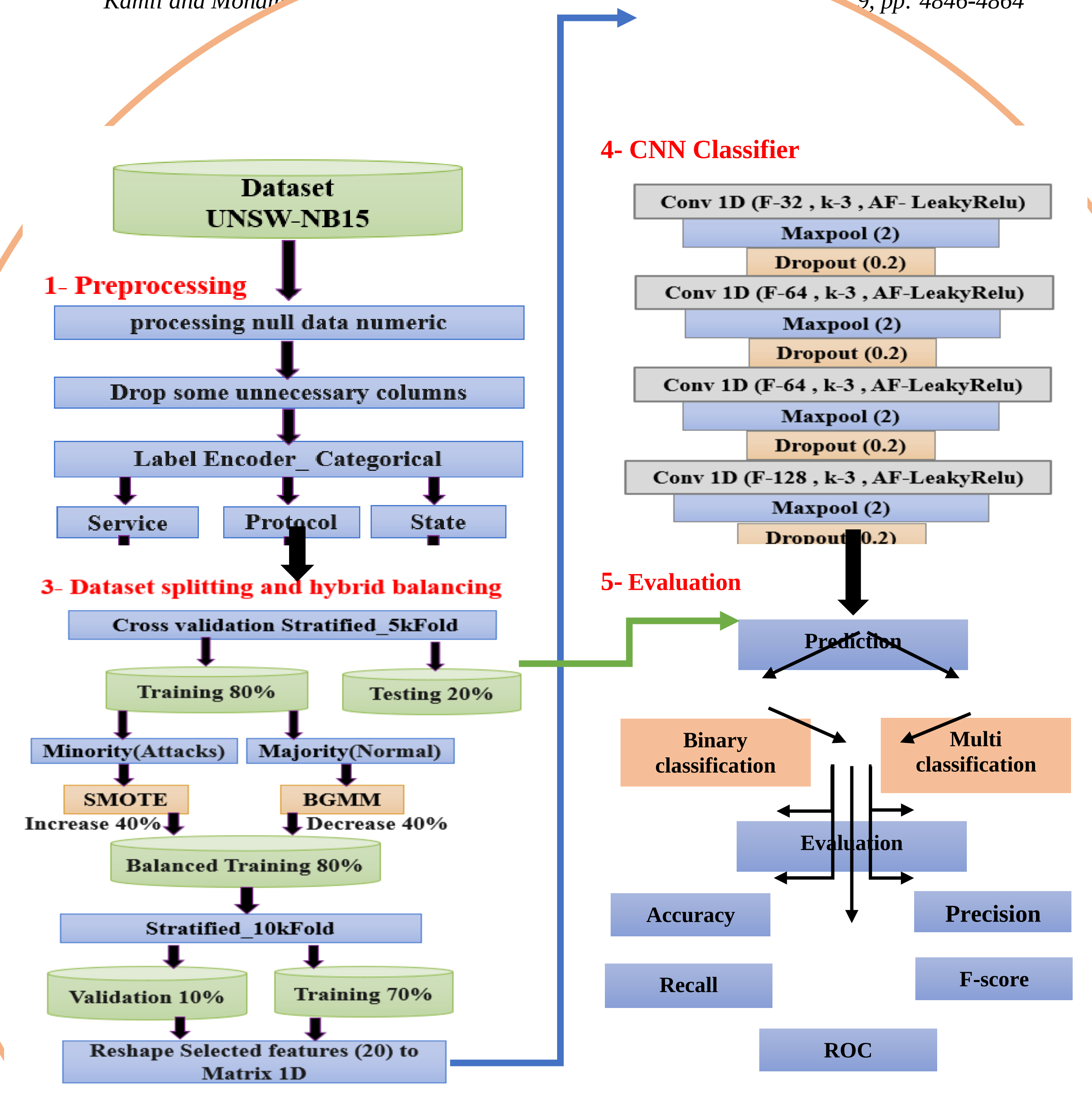


Figure 4 :The proposed IDS framework

Table 1- A synopsis of the factors considered for the proposed framework

stage	Problem	Solution
preprocessing and balancing the dataset	Distribution of samples between training and testing.	• Use only the first stage of the StratifiedKFold method to ensure a similar partitioning of the origin of the data and that no classes are left out.
	Data balancing and eliminating bias towards the majority class.	• SMOTE: Generate new samples of the same pattern to increase the minority. • BGMM: Reduce the majority and avoid neglecting or omitting information that can be useful for training.
Building a deep learning model	Overfitting & underfitting.	• Early Stopping: Stop training once the validation error is above the minimum. • Dropout: A regularisation method that approximates training. Some layer outputs are randomly ignored, forcing nodes within a layer to take on greater or lesser responsibility for the inputs probabilistically.
	Update the neural network weights and calculate the Cost function.	• Adam's optimizer combines gradient descent with momentum and Root Mean Square Prop algorithms. • Adam's optimizer updates the learning rate of every network-weight individual, reducing the error rate. • With faster runtime and lower memory requirements, It works well with large data sets and demands less fine-tuning than any competing optimization technique.
	Classification method and node perceptron death during training.	•In the hidden layers, a leaky ReLU activation function is used. It has the same form as the ReLU, except that positive values close enough to zero will leak to zero. Avoids the dead node issue and does not have a vanishing gradient. • In the dense layer, the Sigmoid activation function is used for binary problem methods. A sigmoid receives just one input and only outputs a single value. The softmax activation function is used for multiclass problems. Methods take in a vector of raw neural network outputs and return a vector of probability scores.

2.1 Dataset Pre-processing

For model evaluation, UNSW NB15 utilized datasets created by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to reflect a mix of the artificial world and simulated attack behaviors. The attack classes in the dataset have nine types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms [14]. Forty-nine features with the class label 'attack' and standard categories (0 if the activity is normal and 1 if it is attacked) are created, and the records are stored in four CSV files [15] [16]. Since symbolic and numeric values have different data types and redundant parts, preprocessing is essential in IDS. Additionally, data with large dimensionality must be processed before learning to reduce the influence on computing performance and detection accuracy.

2.1.1 Dataset merging

Four CSV files (UNSW-NB15_1, NB15_2, NB15_3 and NB15_4) are combined into a single CSV file.

2.1.2 Dataset cleaning

Unwanted values that waste computation time and make the algorithm challenging to process are removed. Null values are either replaced with 0 or with the arithmetic mean of the column, and labels are standardized. For example, 'backdoor' replaces 'backdoors. Attributes recorded during the acquisition of data streams that do not affect the model's results, such as Stime, Lime, dstip, scrip, and sports, are removed. As a result, the number of features is reduced from 49 to 42.

2.1.3 Attributes of dataset label encoder

Label encoding is a technique to preprocess categorical variables by assigning a unique integer to each label based on alphabetical ordering. These integers replace the variable in the same column. This is the method that was utilized in this study. An alternative method is One Hot Encoding, which adds a new column for each categorical variable. The additional variables make the data more complex. For example, there are 135 variables in the 'protocol' column, 13 in the 'service' column, and 16 in the 'status' column.

2.1.4 Min-max dataset normalization

The min-max technique normalizes data for each feature by converting the minimum value to decimal numbers between 0 and 1. This ensures the data can be more easily understood and reduces training time. It is required when attributes have different scales. The following equation is used:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (8)$$

Where Xmax and Xmin are the maximum and minimum feature values (X), resulting in output within the 0-1 range[5].

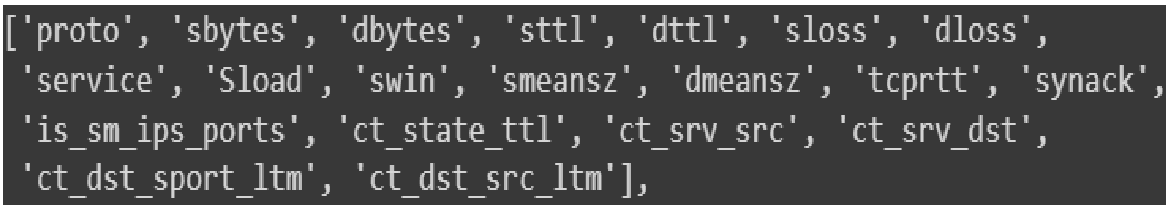
2.2 Hybrid Feature Selection Method

Feature selection in machine learning aims to discover the best mix of features for the supervised learning assignment. When developing an educational model, not all variables in the dataset will help build the model or add to the overall complexity. Thanks to feature selection algorithms, a more accurate model can be created at a lower cost. Feature selection methods are classified as either filter, wrapper, or embedded. The proposed method combines extreme gradient boosting (XGB) with a wrapper method for recursive feature elimination (RFE). XGB uses a superior regularisation technique to reduce over-fitting. As the model can better fit the training data, the tree reinforcement approach produces low and high variance bias results. RFE reduces the number of features [17] by repeatedly fitting the model at each step and deleting the least important attributes to select the best subset. This is demonstrated in Algorithm 1.

Algorithm 1- Feature selection based on XGB_RFE

```
Input: IDS dataset= {$A_1$, $A_2$,.... $A_n$}

Output: Feature Rank= {$R_{A1}$, $R_{A2}$, ....$R_{An}$}or Features selected
BEGIN
Step1: Set FR= {}                                      \\ FR: Feature Rank
Step2: Train the XGB classifier using the training IDS \\ XGB: Extreme gradient boosting
Step3: Calculate the XGB classifier performance (ACU)  \\ ACU: Accuracy
Step4: Determine the ranking of features
Step5: For each feature $A_i$ IN IDS
            $IDS^c$ = IDS -$A_i$                        \\ C: $complement$
            Train the XGB classifier using the $IDS^c$
            Recalculate the XGB classifier performance (ACU)
            Observe the accuracy reduction in the classifier due to the removal of $A_i$
        END
Step6: Create a loss profile for features
Step7: Based on the loss profile, determine the features' rank FR= {$R_{A1}$, $R_{A2}$, ....$R_{An}$}
END
```

By **combining the strengths of XGB and RFE**, the proposed method reduced the number of characteristics from 42 to 21 by selecting those most relevant, as illustrated in Fig. 6.



```
['proto', 'sbytes', 'dbytes', 'sttl', 'dttl', 'sloss', 'dloss',
 'service', 'Sload', 'swin', 'smeansz', 'dmeansz', 'tcprtt', 'synack',
 'is_sm_ips_ports', 'ct_state_ttl', 'ct_srv_src', 'ct_srv_dst',
 'ct_dst_sport_ltm', 'ct_dst_src_ltm'],
```

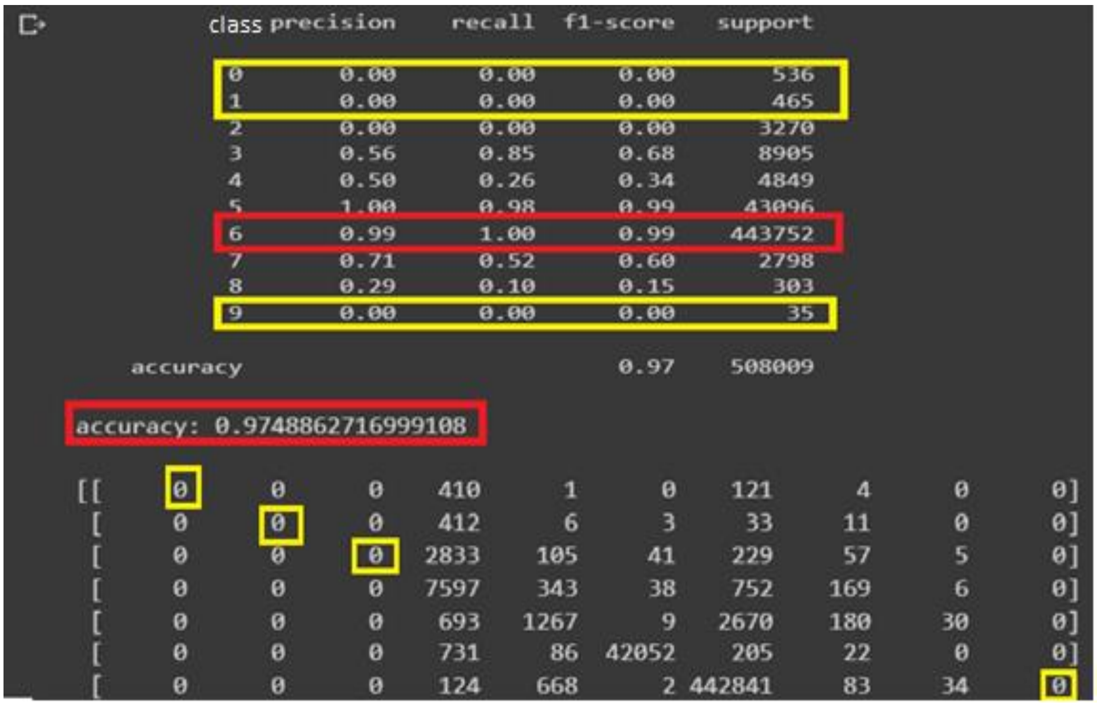
Figure 5: Sample results of the proposed hybrid XGR_RFE

2.3 Dataset Splitting and Balancing

The dataset was split into two parts: a 20% test dataset and an 80% balanced training dataset. Within the training dataset, 10% was used to assess the model's validation, while the remaining 70% was used for training.

2.3.1 Splitting the dataset

StratifiedKFold is a method of stratified data segmentation and an expansion of the cross-validation approach used to improve the classification. The class ratio remains constant throughout the K folds, as in the initial dataset [18] [19]. Rather than using synthetic data, the model was trained on real-world data. The result is an imbalance between the complete attack classes and the normal classes, and a class imbalance in the dataset. Therefore, the classifier tends to favor the majority classes while ignoring minority classes, leading to a lower true detection rate and a higher FAR. This effect occurs when numerous classes are employed. As seen in Fig. 6, 80% of training and 20% of testing are distributed data. The proposed method employs the first stage of the StratifiedKFold approach to ensure a fair distribution of samples from all groups, with ratios matching the original data set. To avoid classes with inadequate proportions, they may only be present in the training data when splitting. Dependency on the accuracy measure in unbalanced data is excluded due to the bias towards the majority category.



```
           class precision    recall  f1-score   support

           0       0.00      0.00      0.00       536
           1       0.00      0.00      0.00       465
           2       0.00      0.00      0.00      3270
           3       0.56      0.85      0.68      8905
           4       0.50      0.26      0.34      4849
           5       1.00      0.98      0.99     43096
           6       0.99      1.00      0.99    443752
           7       0.71      0.52      0.60      2798
           8       0.29      0.10      0.15       303
           9       0.00      0.00      0.00        35

    accuracy                           0.97    508009

accuracy: 0.9748862716999108

[[     0      0      0    410      1      0    121      4      0      0]
 [     0      0      0    412      6      3     33     11      0      0]
 [     0      0      0   2833    105     41    229     57      5      0]
 [     0      0      0   7597    343     38    752    169      6      0]
 [     0      0      0    693   1267      9   2670    180     30      0]
 [     0      0      0    731     86  42052    205     22      0      0]
 [     0      0      0    124    668      2 442841     83     34      0]
```

Figure 6: Determination between the classifier's bias towards majority color (red) and ignoring minority color (yellow)

2.3.2 Hybrid balancing and training of the dataset using SMOTE and BGMM

The goal of this balance-training phase is to correct the issue of class disparity in the training set. Two primary approaches can be applied either at data or computation levels (or planes). Data plane methods, such as over- and under-sampling or ADASYN, were employed due to their independence from algorithmic constraints. However, an increase in data volume raises the required processing time.

This study introduces a hybrid of the Synthetic Minority Oversampling Technique (SMOTE) to create new examples from existing data by adding minority class examples [20] [21]. Bayesian Gaussian mixture models (BGMM) constitute a form of unsupervised learning that assign weights equal to (or close to) zero to unnecessary clusters. Multi-classification data can help with tasks such as clustering, data compression, outlier detection, or classifiers. Typically, each Gaussian component is a multivariate Gaussian with a mean vector and covariance matrix [22]. **Dataset balance is necessary to reduce the algorithmic bias.** Class 9 accounts for less than 0.006% of total data, class 1 for less than 0.09%, and class 8 for less than 0.05%. Increasing the minority class from 257026 to 975377 using the **SMOTE** algorithm reduces the majority class by 35.4%, from 1775011 to 1047254. Using a smaller sampling based on the clustering of **BGMM**, **valuable information for constructing the model is preserved**, as demonstrated in Fig. 7.

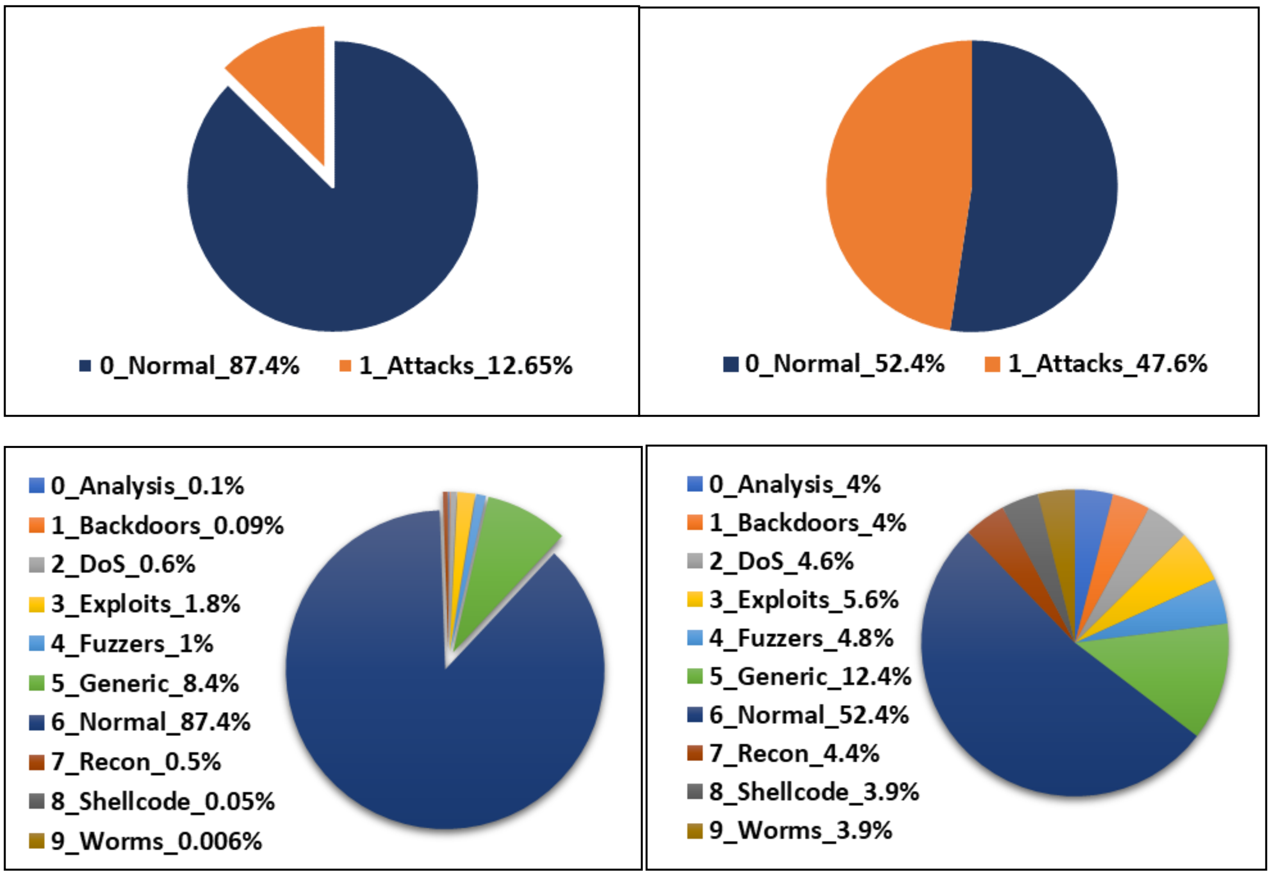


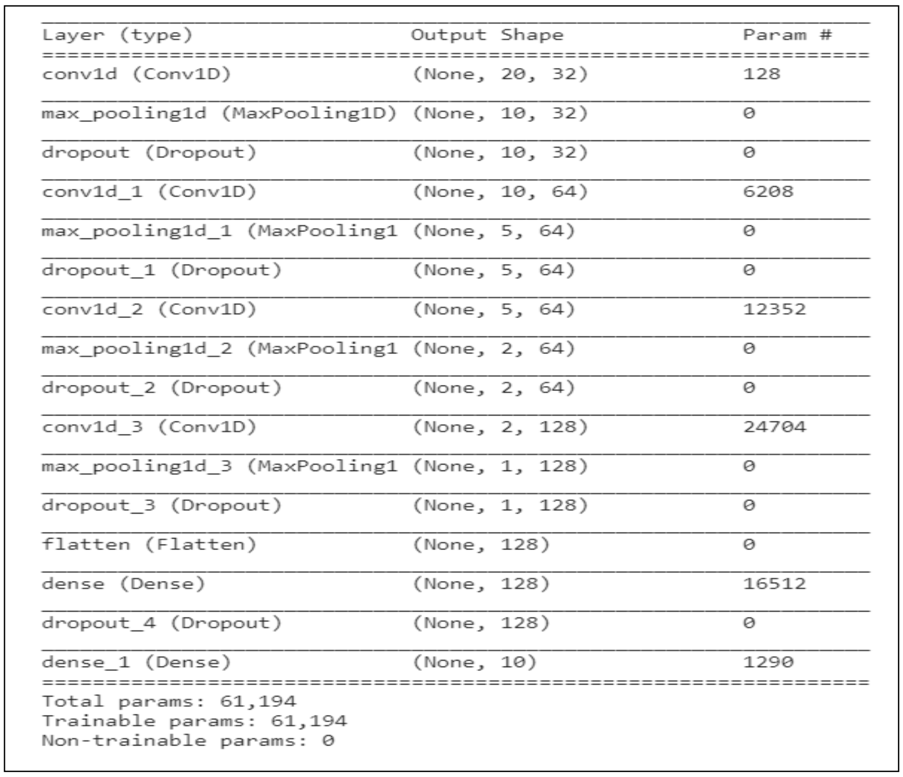
Figure 7: The results of Binary and multi-classification (Left) before balancing and (Right) after balancing

The **first step of the StratifiedKFold** is to split the 80% assigned to the balanced training dataset into 70% training and 10% validation. To ensure that samples from all groups are dispersed in proportions matching the original data set and the minority is not overlooked, the data is then converted to a one-dimensional matrix.

2.4 CNN Classifier

A data classifier is a type of algorithm that organizes collected information into distinct and labeled groups or categories. A CNN classifier consists of three primary layers: a convolution layer, a pooling layer, and a dense (fully connected) layer. The convolution layer is the core of CNN, where data processing begins. It extracts critical features by performing two steps: feature discovery (creating an array of the data to convert it into a feature map) and feature mapping (getting a small size of feature discovery). The pooling layer reduces dimensions using three primary methods: max-pooling, average-pooling, and sum. The dense layer, also called the fully connected layer, consists of neurons connected to every neuron in the preceding layer. It is a feed-forward neural network that uses aggregate properties for classification [23].

The proposed CNN classification model for the network structure shown in Fig. 4 and Fig. 8, contains four convolution layers with filter size (F) ordered by layer (32,64,64,128), in which kernel (K) denotes the number of processed together (K=3). Four max-pooling layers. There are five dropout layers and two fully connected layers (128,10) and one flattens layer. Dropout is used at a rate of 0.20 to prevent over-fitting. The primary purpose of Dropout is to improve the randomness of the parameters and reduce dependencies by neglecting some neurons during training. The LeakyRelU activation function (AF) was used within the CCN hidden layers, excluding the last layer, where Sigmoid was used for binary classification and SoftMax for multi-classification. The total training epoch was set to 300, the batch size is set to 1024, and the learning rate Adam was set to 0.00002. The Bce LF was activated in binary classification, while the Scc LF was used in multiple classifications. Another method of avoiding overfitting is to choose the number of training epochs and stop training when the model's performance on a holdout validation dataset stops improving. This is known as Early Stopping size (15).



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 20, 32)            128
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 10, 32)            0
_________________________________________________________________
dropout (Dropout)            (None, 10, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 10, 64)            6208
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 5, 64)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 64)             0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 5, 64)             12352
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 2, 64)             0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 64)             0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 2, 128)            24704
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 1, 128)            0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 128)            0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 61,194
Trainable params: 61,194
Non-trainable params: 0
```

Figure 8: The output of layers and the number of weights in each

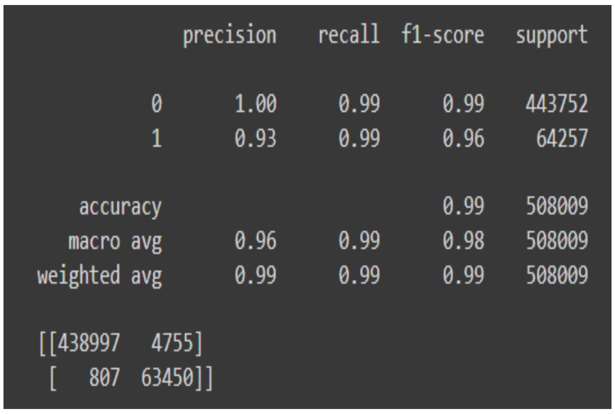
3 Results and Discussion

In the first track, the CNN model, was trained using the unbalanced UNSW NB15 dataset and achieved results in binary classification and multiple classifications with an accuracy of up to 98% due to the majority bias shown in Fig. 9. The confusion matrix directly influences the effectiveness of an IDS. The effectiveness of the proposed framework was evaluated primarily on the confusion matrix in addition to supported factors. As shown in Fig. 6, the classifier could not assign minority classifications to classes 0 or 1. Therefore, alternative measurements for accuracy (F-score, precision, ROC, and recall) were adopted. The focus was on multiple classifications as they successfully demonstrated the efficacy of a classifier after the data had been balanced.

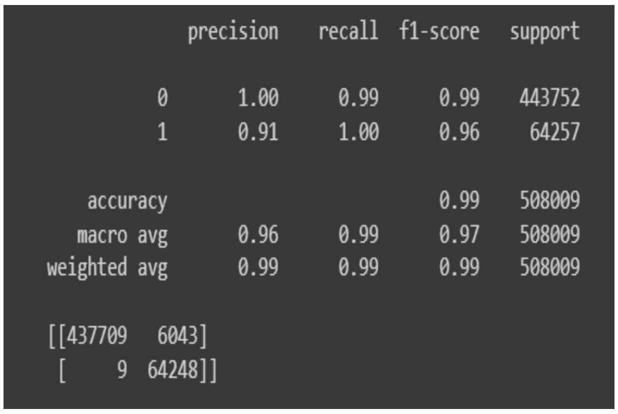
In the second track, the CNN model was trained using the UNSW NB15 dataset after applying the XGB_RFE hybrid feature selection method, followed by the hybrid balance algorithm. Initially, the first stage of the stratified k-fold algorithm was applied to the dataset. Then, it was divided into 80% training and 20% testing volumes. Balancing the training data using the SMOTE_BGMM hybrid yields an average class of 51.8% and an attack class of 48.2%, semi-balancing nine types of attacks.

The proposed model achieved the following: **First**, using all the data, rather than relying on previously divided data, provided the model with sufficient information for learning. **Second**, the number of features was decreased by 50% to reduce the number of calculations and the memory requirement. **Third**, as previously indicated, data balance and segmentation ensured that no class with a small fraction of the total data was excluded from the classification process. **Fourth**, dependence on a single metric due to imbalanced and biased data was avoided.

Fig. 9 and Fig. 10 exhibit results obtained using a binary and multiple classifications classifier on real-world data to demonstrate the impact considerations included in the proposed model.



```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    443752
           1       0.93      0.99      0.96     64257

    accuracy                           0.99    508009
   macro avg       0.96      0.99      0.98    508009
weighted avg       0.99      0.99      0.99    508009

[[438997   4755]
 [   807  63450]]
```

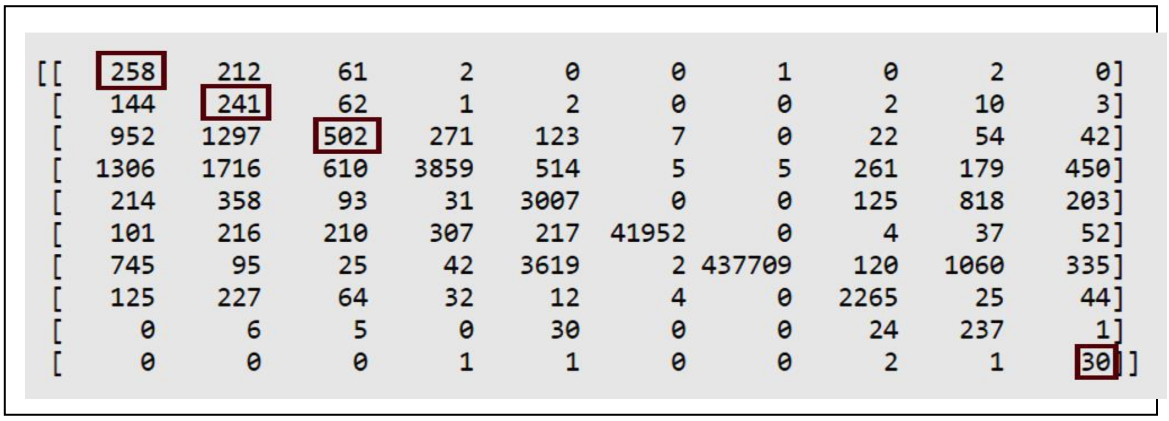


```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    443752
           1       0.91      1.00      0.96     64257

    accuracy                           0.99    508009
   macro avg       0.96      0.99      0.97    508009
weighted avg       0.99      0.99      0.99    508009

[[437709   6043]
 [     9  64248]]
```

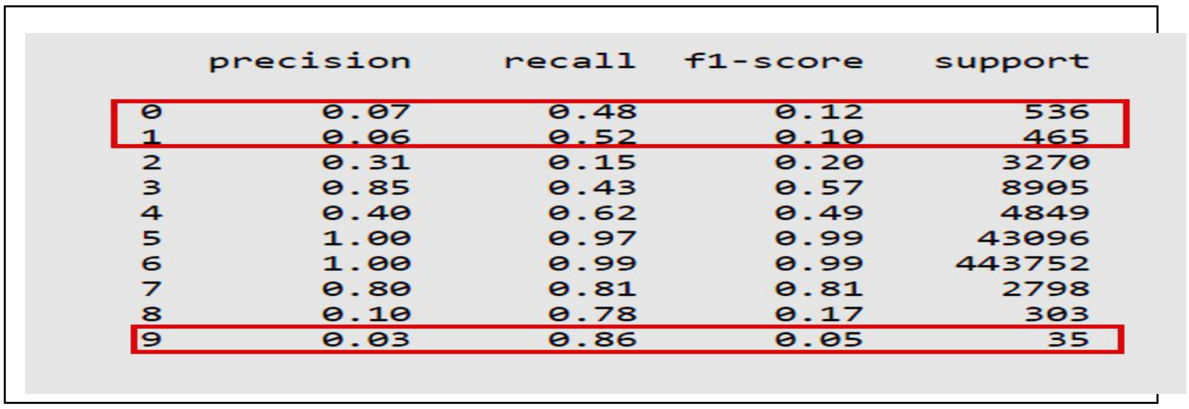
Figure 9: Binary classification report & confusion matrix (Left) before balancing and (Right) after balancing

The range is between [0, 1], where a lower value indicates superior machine learning via confusion matrix computation to binary classification, the found average value of FAR before and after balancing dataset (0.0116),(0.0068) in order and is lower than (0.1018) reported in the study [9].

When the classifier and confusion matrix for the classification of multi-categories after data balancing (Fig.10) are compared to those before data balancing (Fig.6), there is a notable improvement in the classification of the categories (0, 1, and 9) that were not classified at all before the balancing process. Adopting a ROC scale to measure the classifier's ability to distinguish between categories as the accuracy increases and decreases in the loss As the model's performance progresses, this is demonstrated in Figs. 11 and 12.



```
[[   258    212     61      2      0      0      1      0      2      0]
 [   144    241     62      1      2      0      0      2     10      3]
 [   952   1297    502    271    123      7      0     22     54     42]
 [  1306   1716    610   3859    514      5      5    261    179    450]
 [   214    358     93     31   3007      0      0    125    818    203]
 [   101    216    210    307    217  41952      0      4     37     52]
 [   745     95     25     42   3619      2 437709    120   1060    335]
 [   125    227     64     32     12      4      0   2265     25     44]
 [     0      6      5      0     30      0      0     24    237      1]
 [     0      0      0      1      1      0      0      2      1     30]]
```



```
             precision    recall  f1-score   support

           0       0.07      0.48      0.12       536
           1       0.06      0.52      0.10       465
           2       0.31      0.15      0.20      3270
           3       0.85      0.43      0.57      8905
           4       0.40      0.62      0.49      4849
           5       1.00      0.97      0.99     43096
           6       1.00      0.99      0.99    443752
           7       0.80      0.81      0.81      2798
           8       0.10      0.78      0.17       303
           9       0.03      0.86      0.05        35
```

Figure 10: Multi classification_report and confusion_matrix after dataset balancing

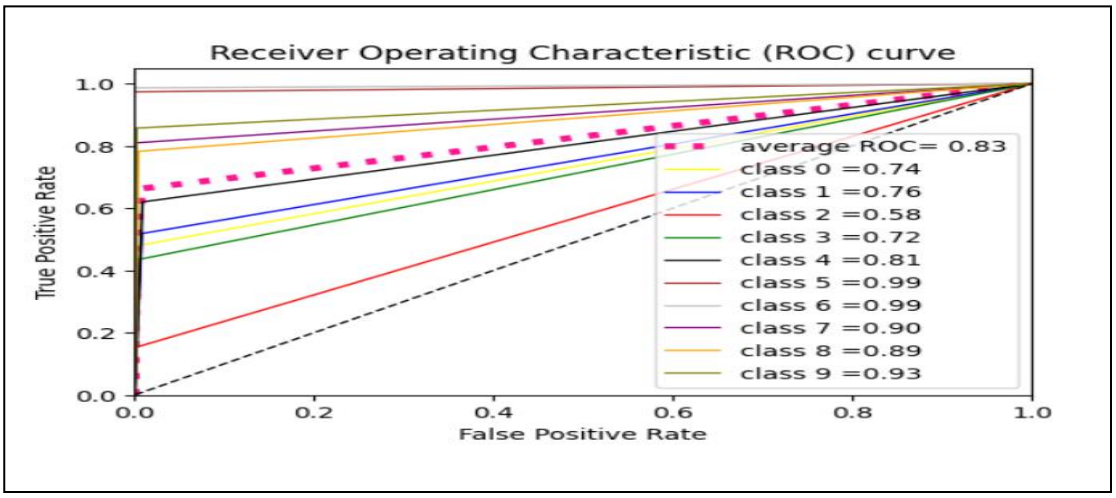


Figure 11: ROC curve (Above) before balancing and (Bottom) after dataset balancing

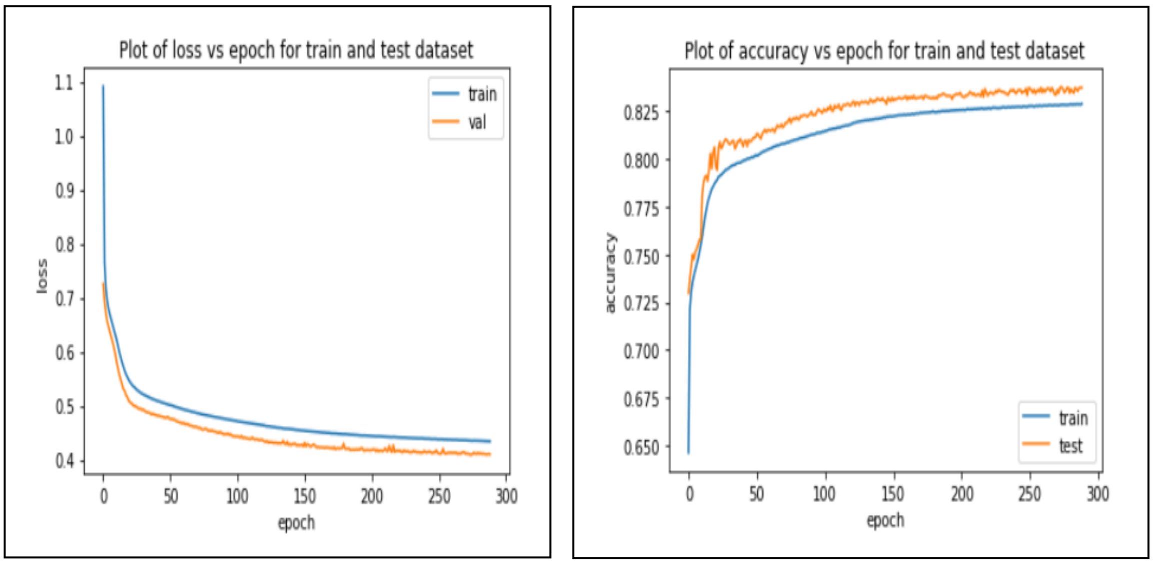


Figure 12: Loos and accuracy after balancing

Table 2 shows the summary results of this study in comparison to other stu, [8], and [9] that did not account for the following factors:

1. Large data volume for deep learning

Too little data is used when relying on the training and testing data that has been previously divided in size (257673) and represents less than 11% of the entire data volume. Successful deep learning requires significant amounts of data.

2. Adoption of pre-feature selection that reduces CNN complexity

Selecting the most influential characteristics is important, as reducing the number of features minimizes memory consumption and reduces the complexity of mathematical processes.

3. Balancing of data with minor and major class considerations

The data is balanced by increasing the proportion of the normal class, offsetting it with the sum of the attack classes while neglecting the minority attack classes, which can be an essential type of attack. Therefore, applying balancing to test data is important.

4. Variety of measurements that support minority-majority classes

Evaluation methods such as accuracy based on confusion matrix give too much weight to majority classes and ignore minority classes.

Table 2: Summary Results and Comparisons

IDS Schemes	Dataset used UNSW_NB15	Dataset balancing	Classification	No.of Features	Accuracy	precision	F1 Score	Average ROC
L. Ashiku [7]	Training &Testing CSV file	---	Multiclass	---	95.6	---	---	---
Han'guk [8]	Training &Testing CSV file	Balancing Training and testing	Binary	---	91.2	87.53	91.59	---
A. Gumaei [9]	Training &Testing CSV file	Balancing Training (increase normal)	Binary	10	89.71	89.74	89.79	---
			Multiclass	10	89.13	90.85	90.85	
Proposed	**All four CSV files**	**Balancing Training (decrease normal and increase attacks)**	**Binary**	**20**	**98.80**	**98.80**	**98.80**	**---**
			Multiclass	**20**	**96.49**	**98.31**	**97.18**	**83**

4. Conclusion

This study has attempted to improve network security systems and identify infiltration. It is built on a deep learning system that makes it difficult for firewall and anti-virus systems to deal with these changes and developing types of electronic attacks with the increase in data volume. The design of a supervised IDS system is based on CNN using the unbalanced UNSW-NB15 dataset with the number of actual practical features calculated by utilizing (XGB) and (RFE) merging to reduce features by half the original number to reduce resource consumption. The data balancing technique employs SMOTE and BGMM to reduce the majority category while preserving the important information and productively increasing the minority in proportions matching the normal distribution of the data.

It can be concluded that the proposed CNN classifier outperformed the deep models on the UNSW-NB15 dataset in terms of accuracy, precision, recall, and F1-score metrics. The results show that the model could classify traffic with up to 98% accuracy using binary classification and close to 96% accuracy using multi-classification.

However, issues remain that will need to be addressed in future work. First, the expansion in network structure will result in an exponential increase in processing time for too many hard-to-tune parameters, such as the number of nodes and layers. Several of the variables in this paper are experimentally determined. Parameter-adjusting approaches that aim to improve performance will need to be investigated in case it turns out not to be the best option. Second, the IDS should be tested in a real network environment since a high detection rate on the dataset does not necessarily translate to comparable performance detection in the real world. Third, it was discovered that a more evenly distributed dataset improved detection performance. The requirement of producing a more representative sample of data. It is recommended to use a model that incorporates both Convolutional Neural Networks (CNNs) and Operational Neural Networks (ONNs) to detect attacks on networks.

References

[1] CyberEdge, 2022. 2022 Cyberthreat Defense Report. https://go.illusivenetworks.com/ 2022-cyberthreat-defense-report.

[2] V. Sstla, V. K. K. Kolli, L. K. Voggu, R. Bhavanam, and S. Vallabhasoyula, "Predictive model for network intrusion detection system using deep learning," *Revue d'Intelligence Artificielle*, vol. 34, no. 3, pp. 323–330, Jun. 2020, doi: 10.18280/ria.340310.

[3] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences (Switzerland)*, vol. 9, no. 20. MDPI AG, Oct. 01, 2019. doi: 10.3390/app9204396.

[4] X. Wang, S. Yin, H. Li, J. Wang, and L. Teng, “A Network Intrusion Detection Method Based on Deep Multi-scale Convolutional Neural Network,” *Int J Wirel Inf Netw*, vol. 27, no. 4, pp. 503–517, Dec. 2020, doi: 10.1007/s10776-020-00495-3.

[5] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," IEEE Access, vol. 6, pp. 48231–48246, 2018, doi: 10.1109/ACCESS.2018.2863036.

[6] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," *IET Inf Secur*, vol. 13, no. 1, pp. 48–53, Jan. 2019, doi: 10.1049/iet-ifs.2018.5258.

[7] L. Ashiku and C. Dagli, "Network Intrusion Detection System using Deep Learning," in *Procedia Computer Science*, 2021, vol. 185, pp. 239–247. doi: 10.1016/j.procs.2021.05.025.

[8] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," in *2020 International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2020*, Feb. 2020, pp. 218–224. doi: 10.1109/ICAIIC48513.2020.9064976.

[9] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "TSDL: A Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019, doi: 10.1109/ACCESS.2019.2899721.

[10] M. Buzzelli and L. Segantin, "Revisiting the compcars dataset for hierarchical car classification: New annotations, experiments, and results," *Sensors (Switzerland)*, vol. 21, no. 2, pp. 1–18, Jan. 2021, doi: 10.3390/s21020596.

[11] Ş. Ozan, "Case studies on using natural language processing techniques in customer relationship management software," *J Intell Inf Syst*, vol. 56, no. 2, pp. 233–253, Apr. 2021, doi: 10.1007/s10844-020-00619-4.

[12] A. M. Carrington, D. G. Manuel, P. W. Fieguth, T. Ramsay, V. Osmani, B. Wernly, C. Bennett, S. Hawken, O. Magwood, Y. Sheikh, M. McInnes, and A. Holzinger,
"Deep ROC Analysis and AUC as Balanced Average Accuracy, for Improved Classifier Selection, Audit and Explanation," *IEEE Trans Pattern Anal Mach Intell*, 2022, doi: 10.1109/TPAMI.2022.3145392.

[13] D. Brzezinski and J. Stefanowski, "Prequential AUC: properties of the area under the ROC curve for data streams with concept drift," *Knowl Inf Syst*, vol. 52, no. 2, pp. 531–562, Aug. 2017, doi: 10.1007/s10115-017-1022-8.

[14] O. F. Rashid, "DNA encoding for misuse intrusion detection system based on UNSWNB15 data set," *Iraqi Journal of Science*, vol. 61, no. 12, pp. 3408–3416, Dec. 2020, doi: 10.24996/ijs.2020.61.12.29.

[15] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Comput*, vol. 23, no. 2, pp. 1397–1418, Jun. 2020, doi: 10.1007/s10586-019-03008-x.

[16] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems (UNSW-NB15 Network Data Set)." [Online]. Available: https://cve.mitre.org/.

[17] J. A. Faysal, S. T. Mostafa, J. S. Tamanna, K. M. Mumenin, M. M. Arifin, M. A. Awal, A. Shome, and S.S. Mostafa, "XGB-RF: A Hybrid Machine Learning Approach for IoT Intrusion Detection," *Telecom*, vol. 3, no. 1, pp. 52–69, Jan. 2022, doi: 10.3390/telecom3010003.

[18] E. G. Adagbasa, S. A. Adelabu, and T. W. Okello, "Application of deep learning with stratified K-fold for vegetation species discrimation in a protected mountainous region using Sentinel-2 image," *Geocarto Int*, vol. 37, no. 1, pp. 142–162, 2022, doi: 10.1080/10106049.2019.1704070.

[19] N. A. Diamantidis, D. Karlis, and E. A. Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation," *Artif Intell*, vol. 116, no. 1–2, pp. 1–16, Jan. 2000, doi: 10.1016/S0004-3702(99)00094-6.

[20] M. Mulyanto, M. Faisal, S. W. Prakosa, and J. S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry (Basel)*, vol. 13, no. 1, pp. 1–16, Jan. 2021, doi: 10.3390/sym13010004.

[21] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-020-00390-x.

[22] P. Zhang and Z. Obradovic, "Learning from inconsistent and unreliable annotators by a Gaussian mixture model and Bayesian information criterion," in Machine Learning and Knowledge Discovery in Databases. ECML PKDD (Lecture Notes in Computer Science), vol. 6913, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds. Berlin, Germany: Springer, 2011, doi: 10.1007/978-3-642-23808-6_36.

[23] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif Intell Rev*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020, doi: 10.1007/s10462-020-09825-6.