



## Deep Learning-Based Intrusion Detection System: Embracing Long Short-Term Memory (LSTM) and Roughly Balanced Bagging Synergies

Onuorah Martins Onyekwelu<sup>[1,2,\*]</sup>, Sun Yanxia<sup>[1]</sup>, Daniel Mashao<sup>[3]</sup>

<sup>[1]</sup> Department of Electrical & Electronic Engineering Science, University of Johannesburg, Auckland Park 2006, South Africa

<sup>[2]</sup> Department of Computer Science Kigali Independent University, Kigali Campus, P.O. Box 2280, Rwanda,

<sup>[3]</sup> Faculty of Engineering and Built Environment, University of Johannesburg, Auckland Park 2006, South Africa

\*<sup>[\*]</sup>martins.onuorah@ulk.ac.rw

**Abstract** This study introduces a novel approach to address class imbalance issues in network traffic datasets within a deep learning framework. We propose the implementation of roughly balanced bagging (RBB) in a long short-term memory (LSTM) architecture, using information gain (IG) to identify optimal features from an intrusion detection system (IDS) dataset exhibiting class imbalance. The approach begins with feature selection via information gain, applies RBB to create balanced subsets of the data, and then trains multiple LSTM models on these subsets to form an ensemble for improved classification of imbalanced network traffic data. Specifically, experimentation is conducted on subsets of features categorized into quartiles on the basis of their information gain, utilizing the CIC-IDS 2017 dataset. The minority class within each quartile is upsampled via the synthetic minority oversampling technique (SMOTE). Then, 10 roughly balanced bags are created from the upsampled data for classification by 10 long short-term memory (LSTM) models. This process is repeated across the first, second, and third quartiles, enabling a comprehensive analysis of feature importance and model performance across the different dataset subsets. Additionally, the dataset's 15 class labels were grouped into 7 classes on the basis of their characteristics, facilitating multiclassification tasks. Our methodology achieved an accuracy of 91.04%, precision of 91.04%, recall of 96.73%, AUC of 96.73%, and F1 score of 91.04% on binary classification using the first quartile (19) features. The performance of our methodology for multiclassification is measured by three metrics: recall, precision, and the F1 score. Class 2 has the highest recall of 98.00%, the F1 score of 92.00%, and class 3 has the highest precision of 97.00%.

**Keywords:** Intrusion detection, Deep learning, Class imbalance, Roughly balanced bagging, Feature selection

# 1 Introduction

The failure of traditional security mechanisms, such as firewalls, has led to the development of intrusion detection systems (IDSs) [1]. An intrusion detection system (IDS) is a software/hardware application that monitors network or system activities for malicious behaviours or policy violations [2]. An IDS can be based on machine learning, deep learning, or a hybrid [3]. It can be a signature-based method, which uses pattern-matching techniques to identify known attacks, or an anomaly-based method, which utilizes a computer system's expected behavior to detect both known and unknown attacks or intrusions [4].

One of the primary challenges encountered in intrusion detection is the presence of class imbalance within datasets [5]. Class imbalance refers to the unequal distribution of samples across different classes, where certain classes are significantly underrepresented compared to others [6]. In intrusion detection datasets, this imbalance often arises because of the rarity of certain types of network events or anomalies [7], such as security breaches or denial-of-service attacks, relative to normal network traffic. Class imbalance presents a significant challenge to developing and deploying effective predictive models and algorithms for detecting network intrusions [8]. Traditional machine learning-based intrusion detection systems do not work well with imbalanced datasets and tend to yield biased predictions [9].

Consequently, innovative methodologies are urgently needed to mitigate the adverse effects of class imbalance and improve the accuracy and robustness of predictive models in network traffic analysis [10].

In recent years, deep learning techniques have emerged as powerful tools for extracting meaningful patterns and representations from large-scale, high-dimensional data, including network traffic datasets [11]. Deep learning models, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, have demonstrated remarkable capabilities in capturing temporal dependencies and complex relationships inherent in sequential data, making them well-suited for tasks such as anomaly detection and classification in network traffic analysis [12].

Furthermore, ensemble learning techniques [13], such as bagging [14], have proven effective in enhancing predictive models' performance and robustness by aggregating multiple base learners' predictions. However, traditional bagging approaches may not adequately address the class imbalance, as they tend to produce bags heavily skewed toward the majority class [15].

## 1.1 Background

Deep learning, a subset of machine learning, has shown remarkable success in various domains, including image recognition, natural language processing, and, more recently, cybersecurity. In IDSs, deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been employed to automatically learn features from raw data and identify complex patterns indicative of malicious activity. Long short-term memory (LSTM) networks, a type of RNN, are particularly effective for sequence-based data, making them suitable for network traffic analysis.

A significant challenge in developing IDSs is the imbalance in datasets, where normal network traffic far outweighs malicious activities. This imbalance can lead to biased models that perform well on majority classes (normal traffic) but poorly on the minority classes (attacks). Addressing this imbalance is crucial for building robust and reliable IDSs. Researchers have employed various methods of oversampling, undersampling, and synthetic data generation to mitigate the effects of data imbalance.

## 1.2 Bagging techniques for imbalanced data

Bagging, or bootstrap aggregation, is an ensemble [16] learning technique that improves model stability and accuracy by combining the predictions of multiple base learners. For imbalanced datasets, roughly balanced bagging [17] involves creating multiple balanced subsets of the data, each used to train a base learner. This approach ensures that minority classes are adequately represented in the training process, hence enhancing the model's ability to detect rare events. The good performance of RBB comes from its ability to recognize unsafe types of minority examples better than other ensembles do [18].

## 1.3 Feature Selection with Information Gain

Feature selection is a critical preprocessing step in machine learning aimed at improving model performance by reducing dimensionality and removing irrelevant features [19]. Information gain, a

---

metric derived from information theory, measures the reduction in entropy or uncertainty a feature brings through feature weight calculation [20]. By selecting features with the highest information gain, models can focus on the most informative aspects of the data, leading to improved accuracy and efficiency. The formula for computing the information gained by a feature about the outcome variable is given in equations (1) to (3).

In light of these considerations, this research aims to propose and evaluate a novel approach that integrates roughly balanced bagging (RBB) within a deep learning framework for addressing class imbalance in intrusion detection datasets. By leveraging RBB [21] alongside feature selection techniques based on information gain and employing advanced deep learning architectures such as LSTM networks [22], we seek to enhance the effectiveness and reliability of predictive models for intrusion detection. Through comprehensive experimentation and evaluation of the CIC-IDS 2017 datasets, this research endeavors to advance the state-of-the-art in anomaly detection and classification, with implications for improving the security and efficiency of networked systems.

The proposed model has several significant contributions, including:

- Implementation of roughly balanced bagging in a deep learning environment.
- A mutual information classifier is used to select  $k$  features and print their information gain.
- Upsampling of the minority class via the synthetic minority oversampling technique (SMOTE).
- Experimentation on subsets of features categorized into first, second, and third quartiles on the basis of their information gain.
- The dataset's 15 class labels were consolidated into 7 groups on the basis of their characteristics.

The remainder of this paper is organized as follows. Section 2 begins with a thorough literature review, examines related studies in intrusion detection and class imbalance mitigation and thus identifies gaps in addressing class imbalance. Section 3 details our proposed approach, explaining the use of information gain for feature selection and the combined use of roughly balanced bagging (RBB) and SMOTE in deep learning architectures such as long short-term memory (LSTM) for mitigating class imbalance problems. Section 4 presents empirical results from experiments on the CIC-IDS 2017 dataset, demonstrating the efficacy of our method in improving predictive model accuracy. Finally, Section 5 discusses the findings, evaluates the approach's strengths and limitations, and suggests directions for future research, concluding with a summary of the study's contributions and implications.

## 2. Background and Review of Related Studies

### 2.1 Related studies

Numerous studies have explored the application of deep learning in IDSs. For example, [23] proposed a deep generative adversarial network called the conditional tabular generative adversarial network (CTGAN) model with common machine learning algorithms to construct more effective detection systems while addressing the imbalance issue. To assess the effectiveness of their proposed strategy, they combined the CTGAN with three machine learning algorithms: support vector machine (SVM), K-nearest neighbor (KNN), and decision tree (DT). The imbalanced NSL-KDD dataset was used, and several experiments were conducted. Similarly, [6] proposed a data resampling technique based on adaptive synthetic (ADASYN) and Tomek link algorithms in combination with different deep learning models to mitigate the class imbalance problem. The proposed model was evaluated on the benchmark NSL-KDD dataset via accuracy, precision, recall, and F1 score metrics. The experimental results show that in multiclass classification, the results outperform state-of-the-art models, with an accuracy of 99.98%. The two studies above were evaluated on the NSL-KDD created in 2009 to remove duplicate records in KDD'99; however, the features used in the dataset may not be sufficient for detecting modern attacks.

In [24], the authors applied a deep CNN model for detecting and classifying near-real-time network intrusion from an imbalanced cloud environment. They used a random forest for optimal feature selection, and their experiments were carried out on the CSE-CIC-IDS2018 dataset. The results show that the proposed CNN model achieved 97.07% testing accuracy with a 2.93% error rate. The

---

performance of the proposed model was also measured via precision, recall, and F1 scores, which were 98.11, 96.93, and 97.52%, respectively.

The study [11] aimed to improve IDSs by addressing the limitations of machine learning methods and the challenges posed by imbalanced datasets. The methodology involved employing data augmentation techniques on four prominent datasets—UNSW-NB15, 5G-NIDD, FLNET2023, and CIC-IDS-2017 and evaluating several deep learning architectures, including a simple CNN-based model. The results demonstrated that the CNN-based architecture achieved high accuracy in detecting network attacks, with up to 91% accuracy on the augmented CIC-IDS-2017 dataset.

The reference [25] aimed to address the limitations of existing NIDSs in detecting infrequent (minority) attacks and minimizing false alarms. The proposed CSE-IDS uses a three-layer approach: Layer 1 employs a cost-sensitive deep neural network to filter suspicious traffic; Layer 2 uses Xtreme gradient boosting to classify this traffic into normal, majority, and minority attack classes; and Layer 3 applies a random forest to classify minority attacks further. The CSE-IDS outperforms existing systems on the NSL-KDD, CIDDS-001, and CICIDS2017 datasets, achieving a high detection rate for both majority and minority attacks while reducing false alarms and confirming its real-world applicability.

To improve the detection rate for minority classes in anomaly based network intrusion detection systems (NIDSs) while maintaining efficiency, [26] proposed a hybrid approach that combines synthetic minority oversampling (SMOTE) and Tomek links to address dataset imbalance and reduce noise. Their chosen classifiers use long short-term memory (LSTM) and CNN models for intrusion detection. Their model achieved high performance on the NSL-KDD and CICIDS2017 datasets, with the LSTM model reaching 99.57% accuracy and 98.98% F score on the NSL-KDD dataset and 99.82% accuracy and 98.65% F score on the CICIDS2017 dataset. The CNN model achieved 99.70% accuracy and 99.27% F score on the NSL-KDD dataset and 99.85% accuracy and 98.98% F score on the CICIDS2017 dataset.

To authors [27] proposed an effective intrusion detection technique to identify and predict minority attacks in network traffic. The proposed technique employs a three-layer approach: Layer 1 uses a weighted deep neural network (WDNN) to identify suspicious traffic; Layer 2 uses a CNN and LSTM to classify traffic into normal, majority, and minority attacks; and Layer 3 applies the XGBoost algorithm to classify minority attack samples into their respective classes. The system also uses one-sided selection undersampling to remove noisy majority attack samples and an adaptive synthetic (ADASYN) oversampling algorithm to generate synthetic minority attack samples. The system was evaluated on the NSL-KDD, CICIDS-2017, and CIDDS-001 datasets and achieved overall accuracy rates of 97.94%, 98.3%, and 97.9%, respectively, demonstrating its effectiveness in detecting and predicting minority attacks.

The study [28] aims to improve the detection of underrepresented attacks in IDSs via a hybrid deep learning model. The proposed hybrid model, BLoCNet, combines a CNN and bidirectional long short-term memory (BLSTM) layers. The CNN quickly identifies patterns in network data, whereas the BLSTM layers use forward and backward propagation to detect malicious traffic. BLoCNet was evaluated on four datasets (CIC-IDS2017, IoT-23, and UNSW-NB15) and compared with five DL models and seven related studies. It achieved higher attack detection rates for CIC-IDS2017 and IoT-23, with accuracies of 98% and 99%, respectively, and an accuracy of 76.34% for UNSW-NB15, outperforming related methods.

Other recent studies that focused on the use of deep learning algorithms to address class imbalance in IDS datasets include [29], who proposed a system that includes a data preprocessing stage and four deep learning models: CNNs, bidirectional long short-term memory (BiLSTM), a bidirectional gate recurrent unit (BiGRU), and an attention mechanism. Preprocessing techniques and particle swarm optimization were used for feature selection, and focal loss addressed class imbalance. The BO-TPE algorithm was employed to optimize the model's hyperparameters. Their proposal was evaluated on the NSL-KDD dataset, and the models demonstrated high detection performance, effectively extracting spatiotemporal features of network traffic data. The system achieved accuracy rates of 0.999158 in binary classification and 0.999091 in multiclass classification, surpassing those of other state-of-the-art methods.

A study by [30] proposed a deep learning model for network intrusion detection (DLNID) that integrates an attention mechanism with a bidirectional long short-term memory (Bi-LSTM) network. It uses a CNN to extract sequence features, an attention mechanism to reassign weights, and Bi-LSTM to learn sequence features. To address data imbalance, the model employs ADASYN for minority class expansion and a modified stacked autoencoder for dimensionality reduction, forming a symmetric

---

dataset without the need for manual feature extraction. The proposed method was tested on the NSL-KDD dataset. The DLNID model achieved an accuracy of 90.73% and an F1 score of 89.65%, outperforming other comparison methods and demonstrating its effectiveness in network intrusion detection.

In contrast to resampling as a means of detecting attacks that are in minority classes, [31] applied a deep neural network (DNN) for intrusion detection, varying its parameters, and analyzed the detection performance of minority classes in imbalanced multiclass data. The model was trained and tested on the CICIDS-2017 dataset and evaluated on the CICIDS-2018 dataset. Two feature selection methods were applied to the preprocessed data to create different feature subsets. The analysis revealed that certain coarse-grained features are so significant that attacks with as few as three instances can be accurately detected. The study highlighted the importance of these feature characteristics for effectively detecting minority classes in network traffic.

**Table 1.** Summary of IDS addressing Class Imbalance

| Authors/year | Classifier                                 | Class imbalance strategy              | Feature selection | Ensemble                 |
|--------------|--|---------------------------------------|-------------------|--------------------------|
| [23]         | SVM, KNN, and DT                           | CTGAN                                 | x                 | x                        |
| [6]          | MLP, DNN, CNN, and CNN-BLSTM               | ADASYN and TomekLink                  | x                 | x                        |
| [24]         | CNN  |                                       | RF                | RF                       |
| [11]         | CNN, LSTM, and GRU                         | SMOTE                                 | x                 | Bagging, Boosting        |
| [25]         | XGBoost                                    | CSD                                   | x                 | Boosting                 |
| [26]         | CNN and LSTM                               | SMOTE, Tomek                          |                   |                          |
| [27]         | DNN, CNN, and LSTM                         | ADASYN                                | Chi-square        | Boosting                 |
| [28]         | CNN and BLSTM                              |                                       | x                 | x                        |
| [32]         | CNN, BiLSTM, BiGRU and Attention Mechanism | Universal Focal Loss (UFL) and BO-TPE | PSO               | x                        |
| [30]         | CNN and BiLSTM                             | ADASYN                                | x                 | x                        |
| [31]         | DNN  | Parameter variation                   | x                 | x                        |
| Our Approach | LSTM                                       | SMOTE                                 | IG                | Roughly balanced bagging |

The major limitation of the previous work in Table 1 is that the authors did not apply feature selection and relied on overly common ensemble techniques, potentially limiting the model's novelty and effectiveness. From Table 1 there remains a critical gap in effectively integrating deep learning techniques, specifically LSTM networks, with advanced methods for addressing imbalanced datasets and optimizing feature selection. Recent studies have not fully explored the synergistic potential of combining LSTM networks with techniques such as roughly balanced bagging (RBB) and information gain (IG) in the context of IDSs. This integration could potentially enhance the detection capabilities and overall performance of IDSs, particularly in handling the complex, high-dimensional, and often imbalanced nature of cybersecurity data. The lack of comprehensive studies in this area presents an opportunity to contribute novel insights to the field of cybersecurity, potentially leading to more robust and adaptive intrusion detection methodologies.

### 3 Overview of the proposed methodology

Experiments were conducted on subsets of features of the CIC-IDS 2017 dataset categorized into quartiles on the basis of their information gain. The minority in each quartile was upsampled via SMOTE, and 10 roughly balanced bags were created from the upsampled data for classification via 10 models of LSTM. This process was repeated for the first (19 features), second (38 features), and third quartiles (56 features) and the full set of 76 features, allowing for a comprehensive analysis of feature importance and model performance across different subsets of the dataset.

### 3.1 Dataset

The CIC-IDS 2017 dataset is a well-known dataset for cybersecurity research, specifically for IDS evaluation. It was created by the Canadian Institute for Cybersecurity (CIC) and includes a variety of modern network traffic scenarios that simulate real-world attacks and benign activities. A detailed description of the dataset is as follows:

The CIC-IDS 2017 dataset is designed to provide a comprehensive dataset for network intrusion detection and prevention system (IDS/IPS) research. It includes normal and malicious network traffic, allowing researchers to test and develop IDS/IPS models.

The dataset was generated in a controlled environment that mimics a real-world network. The data were collected over five days (Monday to Friday) and included a variety of traffic and attack scenarios. The network traffic was captured via common network sniffing tools, and the dataset includes detailed logs of all activities.

#### 3.1.1 Attack Scenarios

The dataset includes several contemporary attacks, each representing different categories of cyber threats: DoS (Denial of Service) Attack: Hulk, GoldenEye, Slowloris, and Slowhttptest. DDoS (Distributed Denial of Service) Attack: LOIC (Low-Orbit Ion Cannon), HOIC (High-Orbit Ion Cannon). Brute Force Attack: FTP-BruteForce, SSH-BruteForce. Web Attack: Brute Force, XSS (Cross-Site Scripting), SQL Injection, Botnet Attack, Infiltration Attack Heartbleed Attack, PortScan Attack, and Normal Traffic

In addition to malicious traffic, the dataset includes normal network traffic, which involves typical user activities such as browsing, emailing, chatting, file transfers, and other common network services.

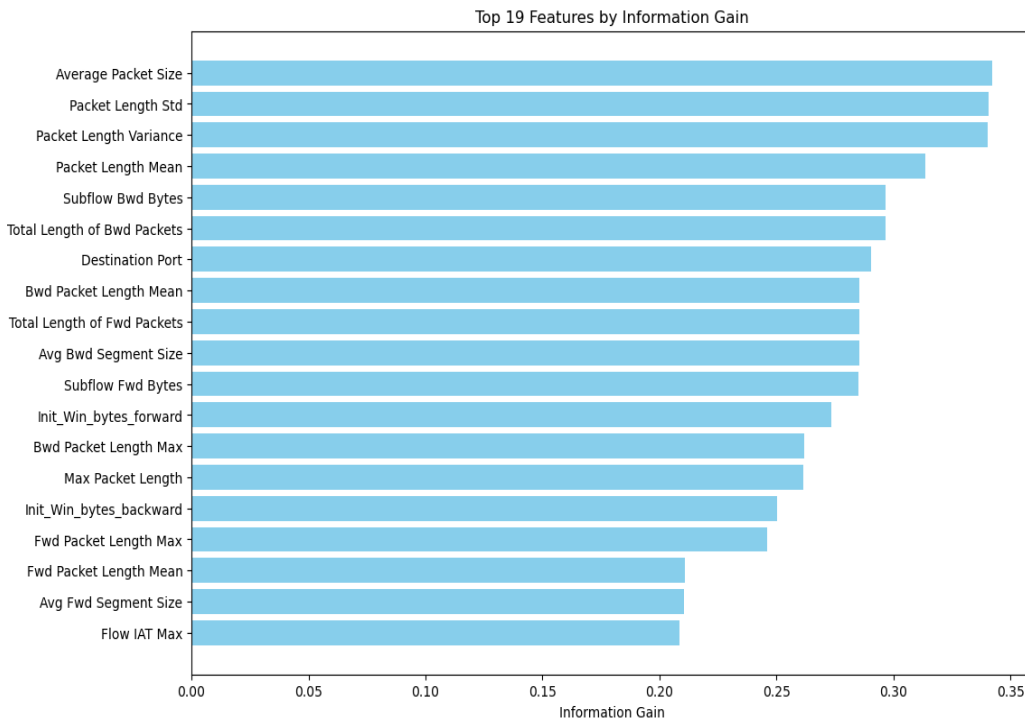
**3.1.2 Features** The dataset includes a wide range of features extracted from network traffic. These features can be broadly categorized into Flow-based Features: Information derived from network flow, including flow duration, flow bytes, and flow packets. Basic Features: Packet-level features such as source and destination IP addresses, source and destination ports, and protocol type. Content Features: Details derived from packet payloads, such as HTTP request headers and payload lengths. Time-based Features: Features related to timing, such as packet arrival times and flow durations. Behavioral Features: High-level features that describe the behavior of the network traffic, such as the number of connections per host and the number of failed connection attempts.

### 3.2 Data processing and feature selection

The CIC-IDS 2017 dataset comprises 79 features and 2.8 million records, but 2 features were removed because they contain infinity or NaN values, leaving 76 features. Thirty percent (30%) of the records were used, which were further divided into training, testing, and internal validation sets. To prepare the data, a min-max scaler was applied, and information gain was used to rank the features. The attacks were converted to attacks, and normal traffic was converted to benign traffic for binary classification. For multiclassification, the attacks were divided into 7 sets, namely, Benign = class 0, DDoS = class 1, PortScan = class 2, Patator = class 3, Bot = class 4, Web Attack = 5, and Infiltration = class 6. To check the imbalance in the data, SMOTE was used to upsample the minority class.

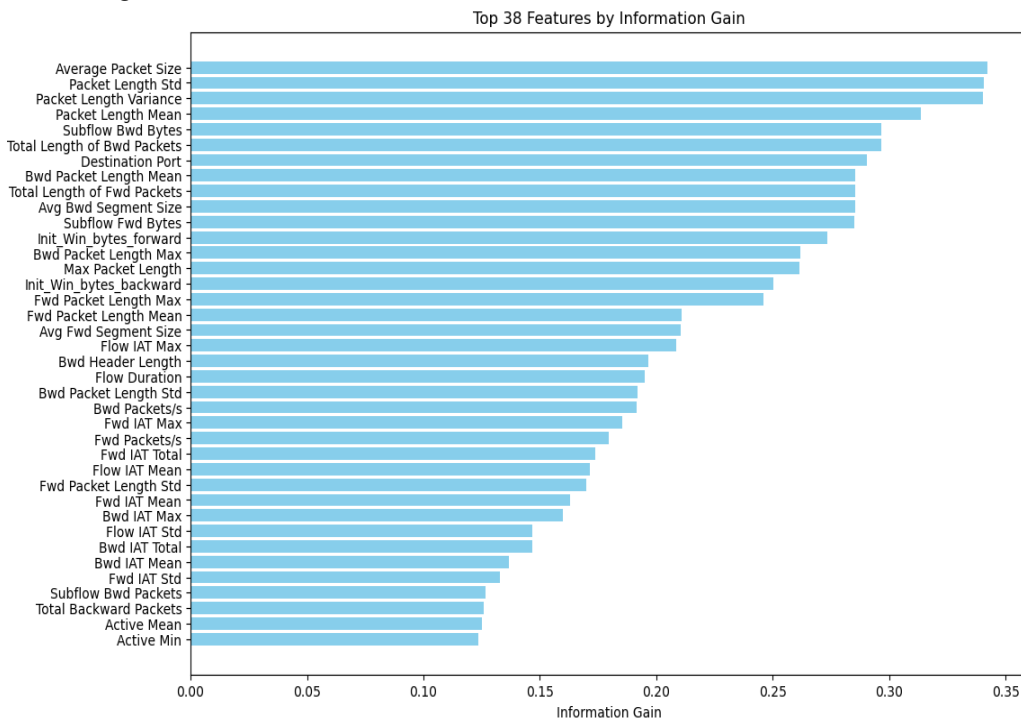
We select the top 19 features representing the 1st quartile features from the dataset on the basis of their information gain (mutual information) with the target variable  $y$ .

---

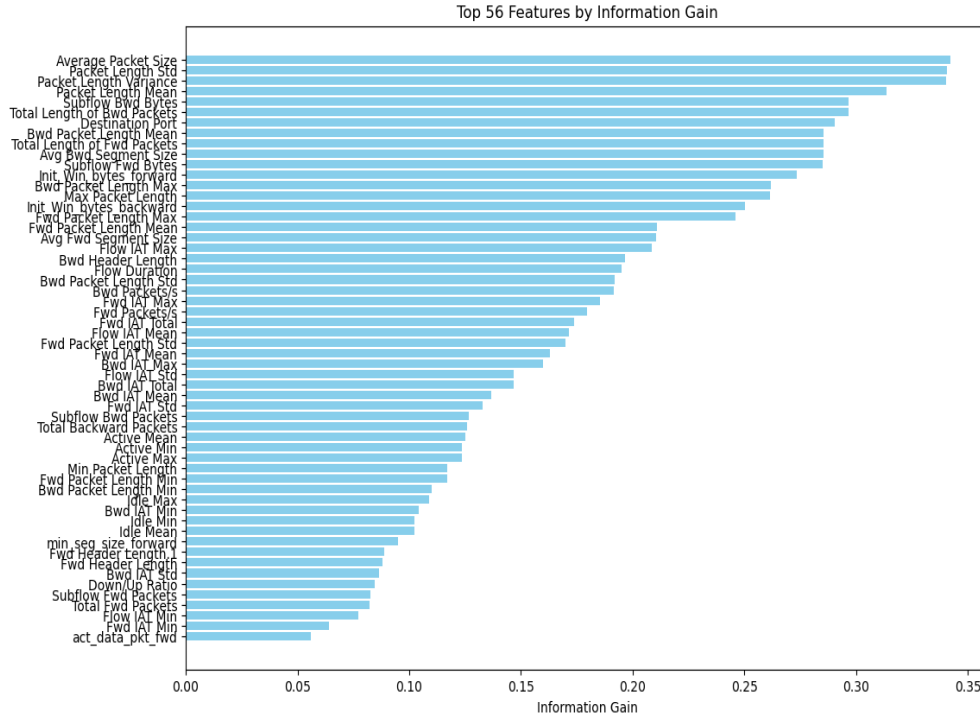


**Fig. 1:** Information gain values of the top 19 features in the dataset, ranked in descending order, with 'Average Packet Size' having the highest value (0.33), 'Flow IAT Max' the least (0.06), and 'Total Length of Fwd Packets' representing the middle value (0.15), highlighting their significance for feature selection.

The feature selection is used with mutual-info-classif as the scoring function, and the selected features are then transformed and stored. Finally, the names of the selected features are printed. This process was repeated for the 2nd quartile, 3rd quartile, and full feature sets, which were then used for data training.



**Fig. 2:** Information gain values of the top 38 features in the dataset, ranked in descending order, with 'Average Packet Size' having the highest value (0.33), 'Active Min' the least (approximately 0.02), and 'Max Packet Length' representing the middle value (around 0.15), highlighting their significance for feature selection.



**Fig. 3:** Information gain values of the top 56 features in the dataset, ranked in descending order, with 'Average Packet Size' having the highest value (0.33), 'act\_data\_pkt\_fwd' the least (approximately 0.01), and 'Max Packet Length' representing the middle value (around 0.15), highlighting their significance for feature selection.

Information gain is a measure used in machine learning to assess how effective an attribute is in classifying data. The reduction in entropy or uncertainty about the target variable is calculated after observing the attribute.

**Table 2.** Information Gain of the Features

| S/N | Feature                     | IG      | S/N | Feature                | IG      |
|-----|-----------------------------|---------|-----|------------------------|---------|
| 1   | Destination Port            | 0.29048 | 39  | Packet Length Mean     | 0.31369 |
| 2   | Flow Duration               | 0.19518 | 40  | Packet Length Std      | 0.34058 |
| 3   | Total Fwd Packets           | 0.08232 | 41  | Packet Length Variance | 0.34024 |
| 4   | Total Backward Packets      | 0.12583 | 42  | FIN Flag Count         | 0.01382 |
| 5   | Total Length of Fwd Packets | 0.28541 | 43  | SYN Flag Count         | 0.00664 |
| 6   | Total Length of Bwd Packets | 0.29640 | 44  | RST Flag Count         | 3.46135 |
| 7   | Fwd Packet Length Max       | 0.24596 | 45  | PSH Flag Count         | 0.04343 |
| 8   | Fwd Packet Length Min       | 0.11700 | 46  | ACK Flag Count         | 0.04154 |
| 9   | Fwd Packet Length Mean      | 0.21088 | 47  | URG Flag Count         | 0.02053 |
| 10  | Fwd Packet Length Std       | 0.16982 | 48  | CWE Flag Count         | 0.00025 |
| 11  | Bwd Packet Length Max       | 0.26192 | 49  | ECE Flag Count         | 0.0     |
| 12  | Bwd Packet Length Min       | 0.11017 | 50  | Down/Up Ratio          | 0.08443 |
| 13  | Bwd Packet Length Mean      | 0.28542 | 51  | Average Packet Size    | 0.34207 |
| 14  | Bwd Packet Length Std       | 0.19189 | 52  | Avg Fwd Segment Size   | 0.21058 |
| 15  | Flow IAT Mean               | 0.17152 | 53  | Avg Bwd Segment Size   | 0.28522 |
| 16  | Flow IAT Std                | 0.14673 | 54  | Fwd Header Length.1    | 0.08874 |
| 17  | Flow IAT Max                | 0.20848 | 55  | Fwd Avg Bytes/Bulk     | 0.0     |
| 18  | Flow IAT Min                | 0.07705 | 56  | Fwd Avg Packets/Bulk   | 0.00074 |

|    |                   |         |    |                         |         |
|----|-------------------|---------|----|-------------------------|---------|
| 19 | Fwd IAT Total     | 0.17369 | 57 | Fwd Avg Bulk Rate       | 0.0     |
| 20 | Fwd IAT Mean      | 0.16278 | 58 | Bwd Avg Bytes/Bulk      | 0.0     |
| 21 | Fwd IAT Std       | 0.13273 | 59 | Bwd Avg Packets/Bulk    | 0.00031 |
| 22 | Fwd IAT Max       | 0.18537 | 60 | Bwd Avg Bulk Rate       | 0.0     |
| 23 | Fwd IAT Min       | 0.06405 | 61 | Subflow Fwd Packets     | 0.08264 |
| 24 | Bwd IAT Total     | 0.14671 | 62 | Subflow Fwd Bytes       | 0.28499 |
| 25 | Bwd IAT Mean      | 0.13654 | 63 | Subflow Bwd Packets     | 0.12661 |
| 26 | Bwd IAT Std       | 0.08636 | 64 | Subflow Bwd Bytes       | 0.29664 |
| 27 | Bwd IAT Max       | 0.15971 | 65 | Init_Win_bytes_forward  | 0.27335 |
| 28 | Bwd IAT Min       | 0.10421 | 66 | Init_Win_bytes_backward | 0.25027 |
| 29 | Fwd PSH Flags     | 0.00680 | 67 | act_data_pkt_fwd        | 0.05596 |
| 30 | Bwd PSH Flags     | 0.0     | 68 | min_seg_size_forward    | 0.09511 |
| 31 | Fwd URG Flags     | 0.00019 | 69 | Active Mean             | 0.12497 |
| 32 | Bwd URG Flags     | 0.00040 | 70 | Active Std              | 0.01738 |
| 33 | Fwd Header Length | 0.08797 | 71 | Active Max              | 0.12359 |
| 34 | Bwd Header Length | 0.19640 | 72 | Active Min              | 0.12368 |
| 35 | Fwd Packets/s     | 0.17938 | 73 | Idle Mean               | 0.10230 |
| 36 | Bwd Packets/s     | 0.19154 | 74 | Idle Std                | 0.02251 |
| 37 | Min Packet Length | 0.11706 | 75 | Idle Max                | 0.10885 |
| 38 | Max Packet Length | 0.26131 | 76 | Idle Min                | 0.10239 |

The information gain (IG) of an attribute (feature) in a dataset  $T$  is given by:

$$IG(T, A) = H(T) - H(T/A) \quad (1)$$

The entropy  $H(T)$  of a dataset  $T$  is given by

$$H(T) = - \sum_i^n P(c_i) \log_2 P(c_i) \quad (2)$$

The conditional entropy  $H(T/A)$  of a dataset  $T$  for a given attribute  $A$  is given by:

$$H\left(\frac{T}{A}\right) = \sum_{v \in \text{Values}(A)} P(v) H(T_v). \quad (3)$$

where

$n$ : Number of classes in the dataset

$P(c_i)$ : Proportion of samples belonging to class  $c_i$

$\text{Values}(A)$ : Set of all possible values of attribute  $A$

$P(v)$ : The proportion of samples in dataset  $T$  that have value  $v$  or attribute  $A$

$T_v$ : The subset of the dataset  $T$  where attribute  $A$  has value  $v$

$H(T_v)$ : Entropy of the subset  $T_v$

### 3.3 Roughly balanced bagging

Roughly balanced bagging is an improved ensemble algorithm that uses a negative binomial distribution in the sampling process to ensure the utilization of all minority class data. For this sampling method, the number of samples in the largest and smallest classes are different, but they are effectively balanced when averaged over all of the subsets, which supports the approach of bagging more appropriately. The probability mass function of the negative binomial distribution is given by equation (4):

$$P(m/n) = \binom{m+n-1}{n} q^n (1-q)^m \quad (4)$$

where  $n$  is the number of successes,  $m$  is the number of failures in a Bernoulli trial, and  $q$  is the probability of success. The full algorithm of the roughly balanced bagging is presented below.

#### Roughly Balanced Algorithm

- Inputs:
  - $D$  is the training dataset
  - $L$  is the algorithm for the base learners
  - $K$  is the number of base learners

- $x_i$  is an example drawn from the test set
- Build a rough balanced bagging model  $(D, L, K)$ :
  - Divide  $D$  into a negative set  $D^{neg}$  and a positive set  $D^{pos}$
  - Set  $N^{pos}$  as the size of  $D^{pos}$  (i. e.  $|D^{neg}|$ )
  - For  $k = 1$  to  $K$ 
    - Draw  $N_k^{neg}$  from the negative binomial distribution equation (4):
    - with  $n = N^{pos}$  and  $q = 0.5$
    - Let  $D_k^{neg}$  be an  $N_k^{neg}$  example sampled from  $D^{neg}$  with or without replacement
    - Let  $D_k^{pos}$  be an  $N_k^{neg}$  example sampled from  $D^{pos}$  with or without replacement
    - Build a base learner model  $f^k(x)$  by  $L$  based on  $D_k^{neg} \cup D_k^{pos}$
    - Combine all  $f^k(x)$  into the aggregated model  $f^A(x)$
    - Return  $f^A(x)$
- Predict  $(f^A(x), x_i, y)$ :
  - Calculate  $p^A(y|x_i) = \frac{1}{K} \sum_{k=1}^K p^k(y|x_i)$  for all  $y$
  - Let  $\hat{y} = \underset{y \in L}{\operatorname{argmax}} p^A(y|x_i)$
  - Return  $\hat{y}$

### 3.4 The LSTM Architecture

The inability of the recurrent neural network (RNN) to store data for a long period, resulting in gradient vanishing, was identified and discussed by [33]. This problem was analysed further by [34], who proposed LSTM to address this challenge. It has wide application in a variety of areas such as social signal classification, speech recognition, and handwriting translation [34]. The LSTM's success lies in using cell gates for input, output, and forget. The structure of the LSTM is guided by equations (5) to (10):

LSTMs overcome the vanishing gradient problem through their unique architecture, which includes mechanisms for maintaining long-term dependencies. How LSTMs achieve this is as follows:

**Forget Gate:** Decides what portion of the cell state should be removed. It uses a sigmoid function to produce an output between 0 and 1, where 0 means "completely forget" and 1 means "completely keep."

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (8)$$

**Input Gate:** Decides what new information to add to the cell state. It consists of two parts: a sigmoid layer (input gate layer) that decides which values to update and a  $\tanh$  layer that creates a vector of new candidate values that can be added to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (6)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (7)$$

**Output Gate:** Decides what part of the cell state should be outputted. The output is based on the cell state, but it is a filtered version controlled by the output gate.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (8)$$

#### Cell State Update

The cell state is updated via the forget gate and input gate outputs. The forget gate determines how much of the previous cell state to retain, and the input gate determines how much of the new candidate values to add.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (9)$$

#### Output Calculation

Finally, the hidden state  $h_t$  is calculated on the basis of the cell state and the output gate. This hidden state is passed to the next time step and can be used as an output if needed.

$$h_t = o_t * \tanh(C_t) \quad (10)$$

#### 3.4.1 Model Architecture

The LSTM (long short-term memory) model architecture presented here is designed for binary classification tasks, such as an intrusion detection system (IDS). The model uses the Sequential API from Keras, a high-level neural network API in TensorFlow. The architecture begins with an LSTM layer containing 50 units and utilizes the 'tanh' activation function, which is ideal for capturing long-term dependencies in sequential data. This initial LSTM layer is configured to return sequences, enabling the next LSTM layer with a tanh activation function to process its output. A dropout layer with a dropout rate of 0.2 follows, acting as a regularization technique to prevent overfitting by randomly setting a fraction of the input units to zero during training. The second LSTM layer, which also has 50 units, processes the sequences from the previous layer. Finally, a dense output layer with a Softmax activation function produces the class probabilities, allowing the model to predict the likelihood of each class. The tanh and softmax activation functions are given by

$$\tanh(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \quad (11)$$

$$f(z) = \frac{e^{z_i}}{1 + e^{z_j}} \quad (12)$$

The model is compiled with the Adam optimizer, which is known for its efficiency in training deep learning models by adjusting the learning rate adaptively. The loss function used is binary cross-entropy, which is appropriate for binary classification tasks.

The binary cross-entropy, which measures the discrepancy between the true labels and predicted probabilities, is given by:

$$BCE = -\frac{1}{N} \sum_{i=1}^N [y_i \log(P_i) + (1 - y_i) \log(1 - P_i)] \quad (13)$$

where:

$N$  is the number of samples.

$y_i$  is the true label of the  $i^{\text{th}}$  sample (0 or 1).

$P_i$  is the predicted probability of the  $i^{\text{th}}$  sample being in class 1.

$\log$  is the natural logarithm.

The Adam optimization algorithm used in updating the model parameters is given below:

Given parameters  $\theta_t$  at time step  $t$ , gradients  $g_t$ , and hyperparameters  $\alpha$  (learning rate),  $\beta_1$  (decay rate for the first moment),  $\beta_2$  (decay rate for the second moment), and  $\epsilon$  (small constant to prevent division by zero):

#### Initialization

$m_0 = \mathbf{0}$  is the initial first-moment vector

$v_0 = \mathbf{0}$  is the initial second-moment vector

$t = 0$  is the time step.

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (14)$$

where

$g_t = \nabla_{\theta} f_t(\theta_{t-1})$  is the gradient of the loss function at time  $t$

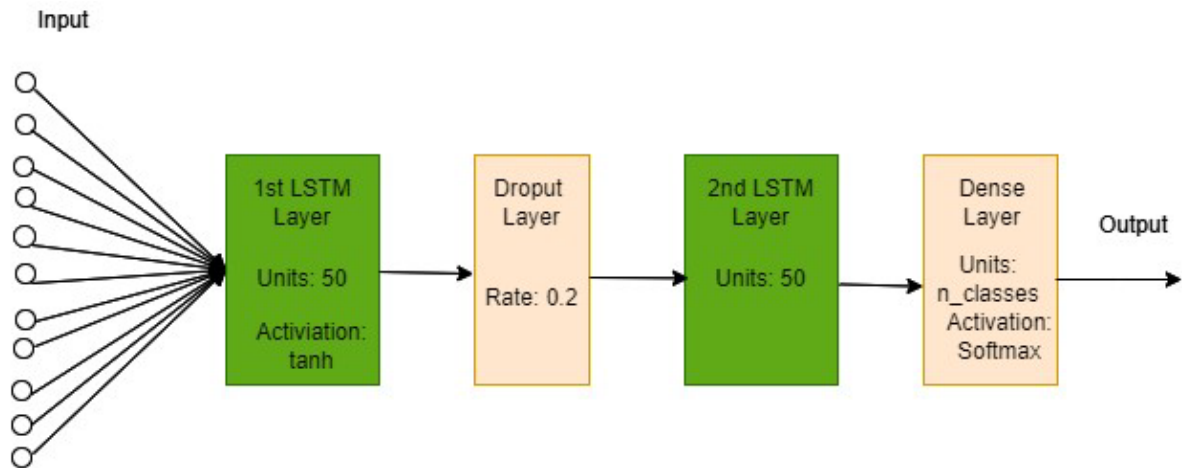
$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  is the biased first-moment estimate

$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  is the bias second-moment estimate

$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$  is the bias-corrected first-moment estimate

$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$  is the bias-corrected second-moment estimate

The training strategy involves creating roughly balanced bags of data through upsampling and splitting the dataset into training and test sets for each bag. This bagging approach helps improve model robustness by training it on diverse subsets of data, thus enhancing its generalization ability. Each bag is reshaped to fit the LSTM input requirements, and the target labels are one-hot encoded to match the output layer's format, ensuring seamless training and evaluation across multiple bags.



**Fig. 4** LSTM Model Architecture, showing the first layer, dropout layer, second layer and the dense layer. It also shows their units and activation functions.

## 4 Experimental results

### 4.1 Evaluation metrics

The evaluation matrix of the model includes several performance metrics: accuracy, precision, recall, area under the ROC curve (AUC), and a custom F1 score metric. These metrics provide a comprehensive evaluation of the model's performance, particularly in scenarios where data might be imbalanced, as they offer insights into both the model's ability to identify positive instances correctly and its overall classification capability.

**Loss:** This measures the difference between the values predicted by the model and the actual values. Lower loss values indicate better model performance. The loss function used in this study is the binary cross entropy given in equation (13).

**Accuracy:** The proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.

$$\text{Formula: Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

**Precision:** The proportion of true positive predictions compared with the total number of positive predictions. Formula:

$$\text{Precision} = \frac{TP}{TP+FP}$$

**Recall (also known as the sensitivity or true positive rate):** Definition: The proportion of actual positive cases that were correctly identified. Formula:

$$\text{Recall} = \frac{TP}{TP+FN}$$

**F1 score:** Definition: The harmonic mean of precision and recall, which provides a single score that balances both metrics. Formula:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

**Area under the curve (AUC):** Definition: A measure of the ability of a classifier to distinguish between classes. The AUC represents the area under the ROC curve. It ranges from 0 to 1, where 1 represents a perfect classifier and 0.5 represents a classifier that performs no better than random guessing.

**Receiver operating characteristic (ROC) curve:** Definition: A graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate (Recall) against the false positive rate at various threshold settings.

$$\text{False Positive Rate} = \frac{FP}{FP+TN}$$

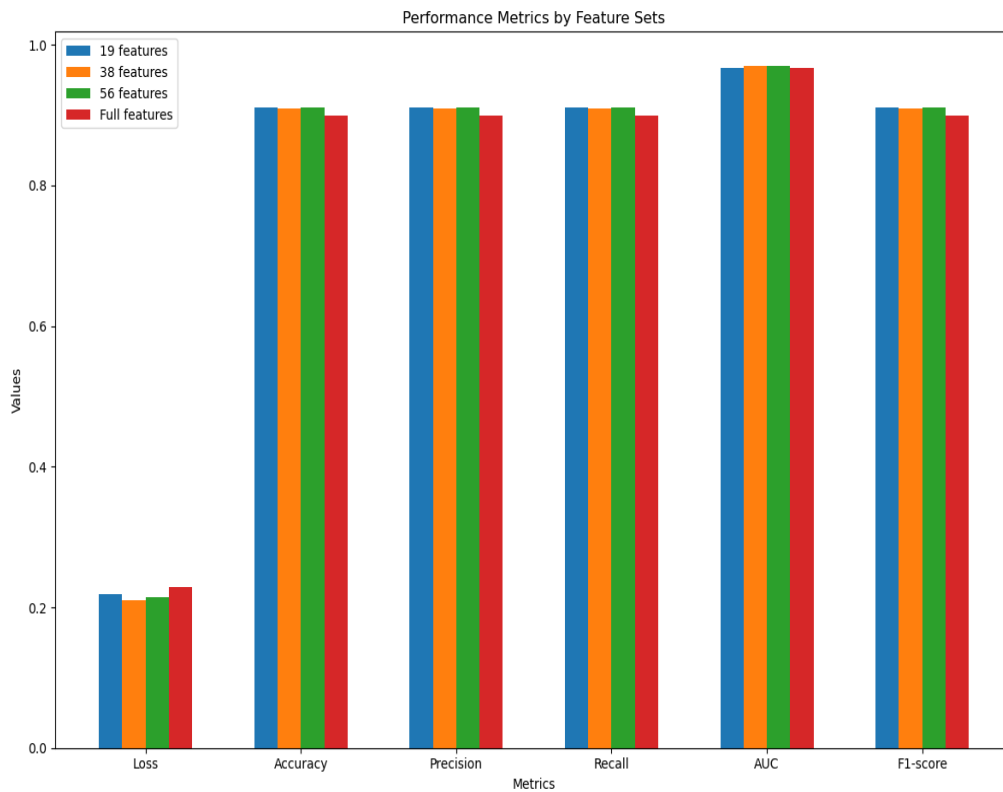
The AUC-ROC is calculated by measuring the entire two-dimensional area underneath the ROC curve.

## 4.2 Binary classification

Binary classification is a type of classification task that involves assigning one of two possible labels to each input instance. In the context of an intrusion detection system (IDS) using deep learning, binary classification typically involves distinguishing between normal (benign) network traffic and malicious (attack) traffic [12]. This approach leverages the capabilities of deep learning models, such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, to analyze and classify network data with high accuracy [35]. For binary classification, we used the following metrics: loss, accuracy, precision, recall, F1 score, and AUC, and their values are shown below.

**Table 3.** Binary classification metrics

|                  | 19 features     | 38 features | 56 features | Full features |
|------------------|-----------------|-------------|-------------|---------------|
| <b>Loss</b>      | <b>0.218662</b> | 0.209638    | 0.214828    | 0.228850      |
| <b>Accuracy</b>  | 0.910393        | 0.909257    | 0.911296    | 0.898664      |
| <b>Precision</b> | 0.910393        | 0.909257    | 0.911296    | 0.898664      |
| <b>Recall</b>    | 0.967310        | 0.909257    | 0.911296    | 0.898664      |
| <b>AUC</b>       | 0.967310        | 0.970047    | 0.969421    | 0.966805      |
| <b>F1-score</b>  | 0.910390        | 0.909256    | 0.911289    | 0.89866       |

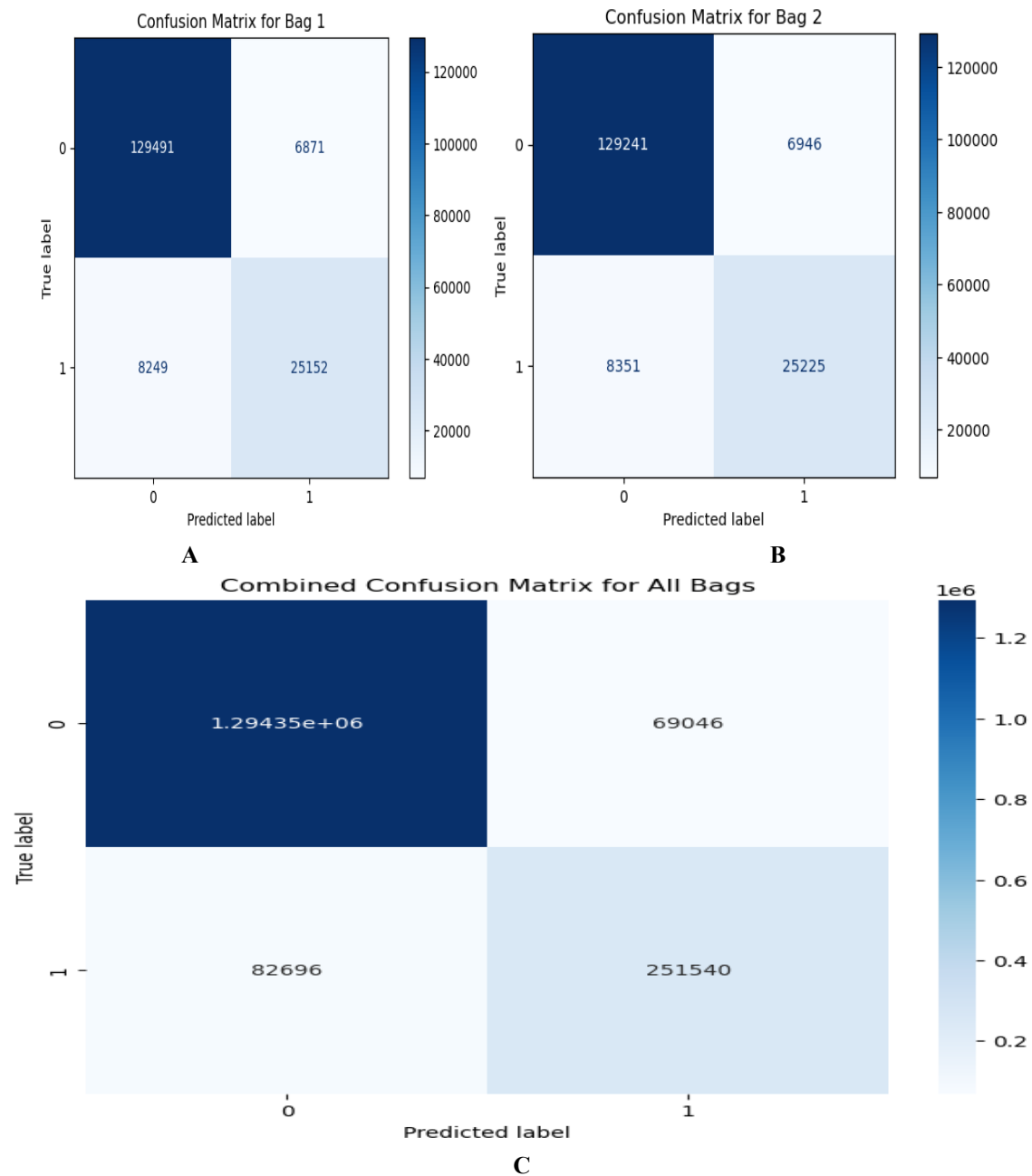


**Fig. 5:** Bar chart of the performance metrics for binary classification. The Fig resents a comparison of binary classification performance metrics across four feature sets: 19, 38, 56, and the full feature set. The metrics, including accuracy, precision, recall, AUC, and F1-score, remain consistently high (above 0.9) for all feature sets, demonstrating minimal performance variation regardless of the number of features used. However, the loss metric slightly increases as the feature set size grows, with the full feature set having the highest loss value, indicating potential overfitting or reduced efficiency with additional features.

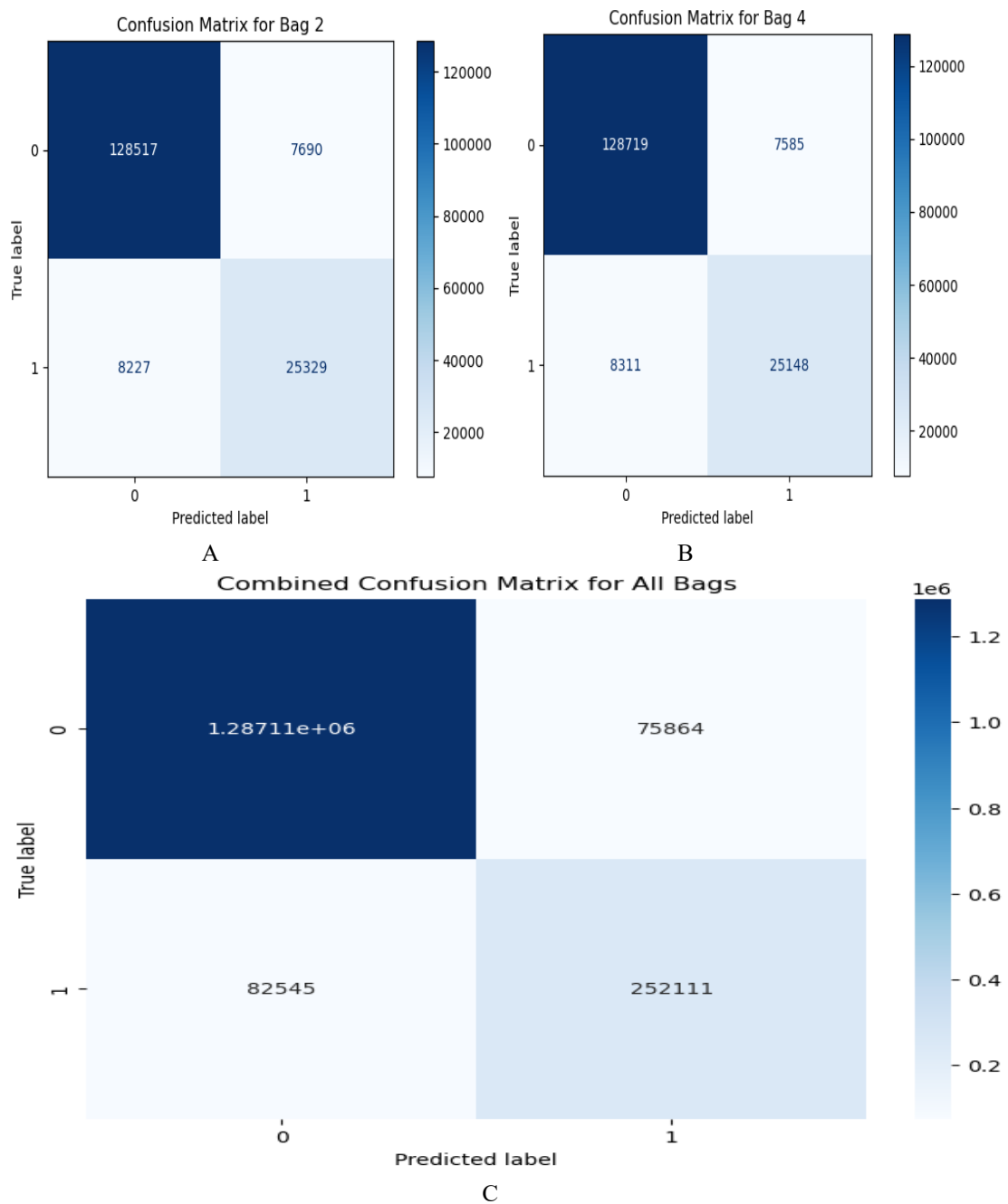
### 4.3 Confusion matrix for the binary classification

The confusion matrix for binary classification using four different features of our dataset is presented below. First, we randomly selected two confusion matrices from the classification results of the 10 bags. Then, we created an aggregate confusion matrix that represents the overall classification performance across all bags using these features.

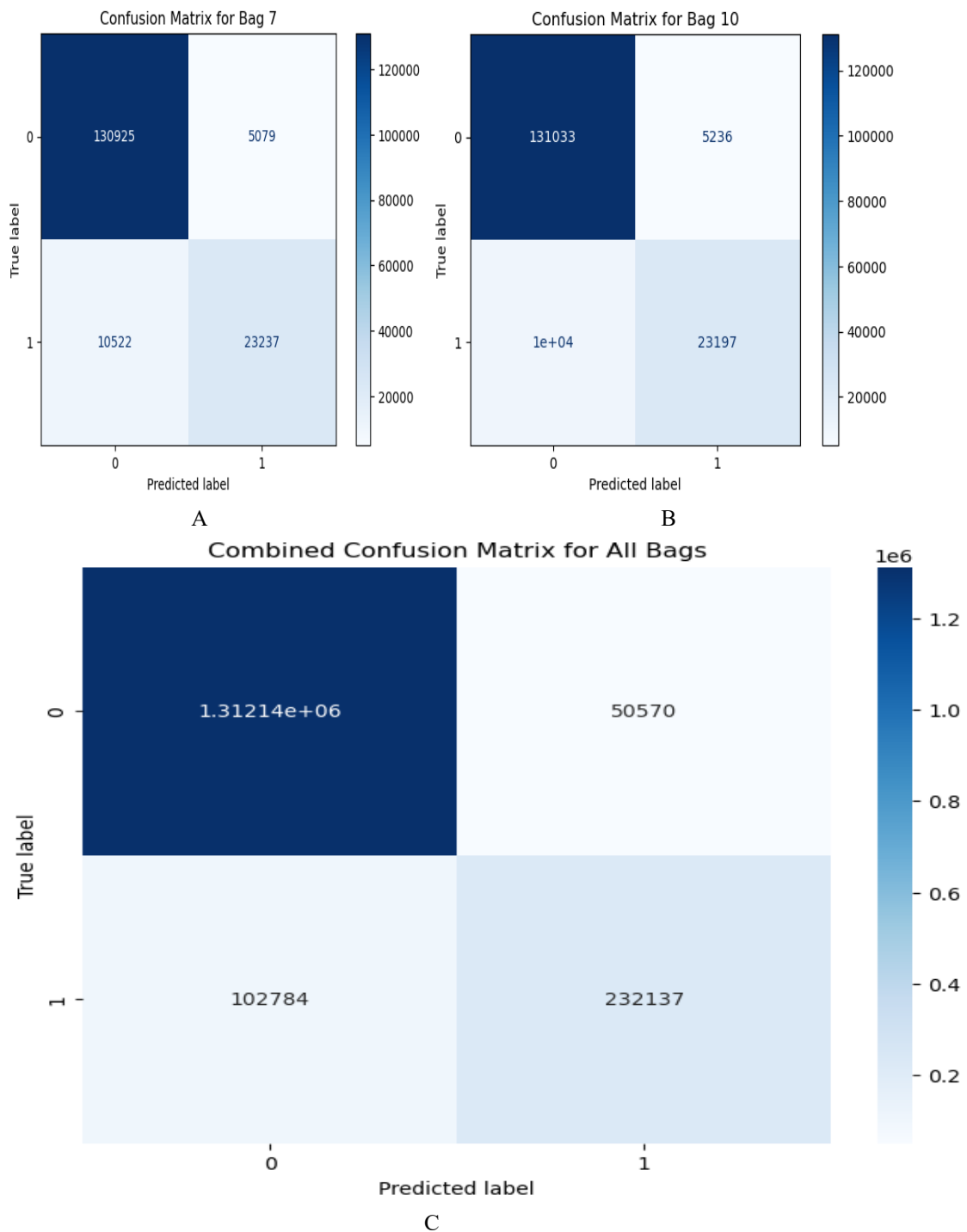
Analyzing individual confusion matrices from each bag allows us to understand how each bag's classification performs in isolation, capturing the variability and robustness of the model across different subsets of data. By aggregating these into a combined confusion matrix, we gain a comprehensive view of overall model performance, which is crucial for evaluating the consistency and stability of the model when applied to the entire dataset. This dual approach ensures that both the granular and holistic aspects of model accuracy are thoroughly assessed. The confusion matrices of the classification using the four subsets of the dataset are presented in Fig. 6, Fig. 7, Fig. 8, and Fig. 9



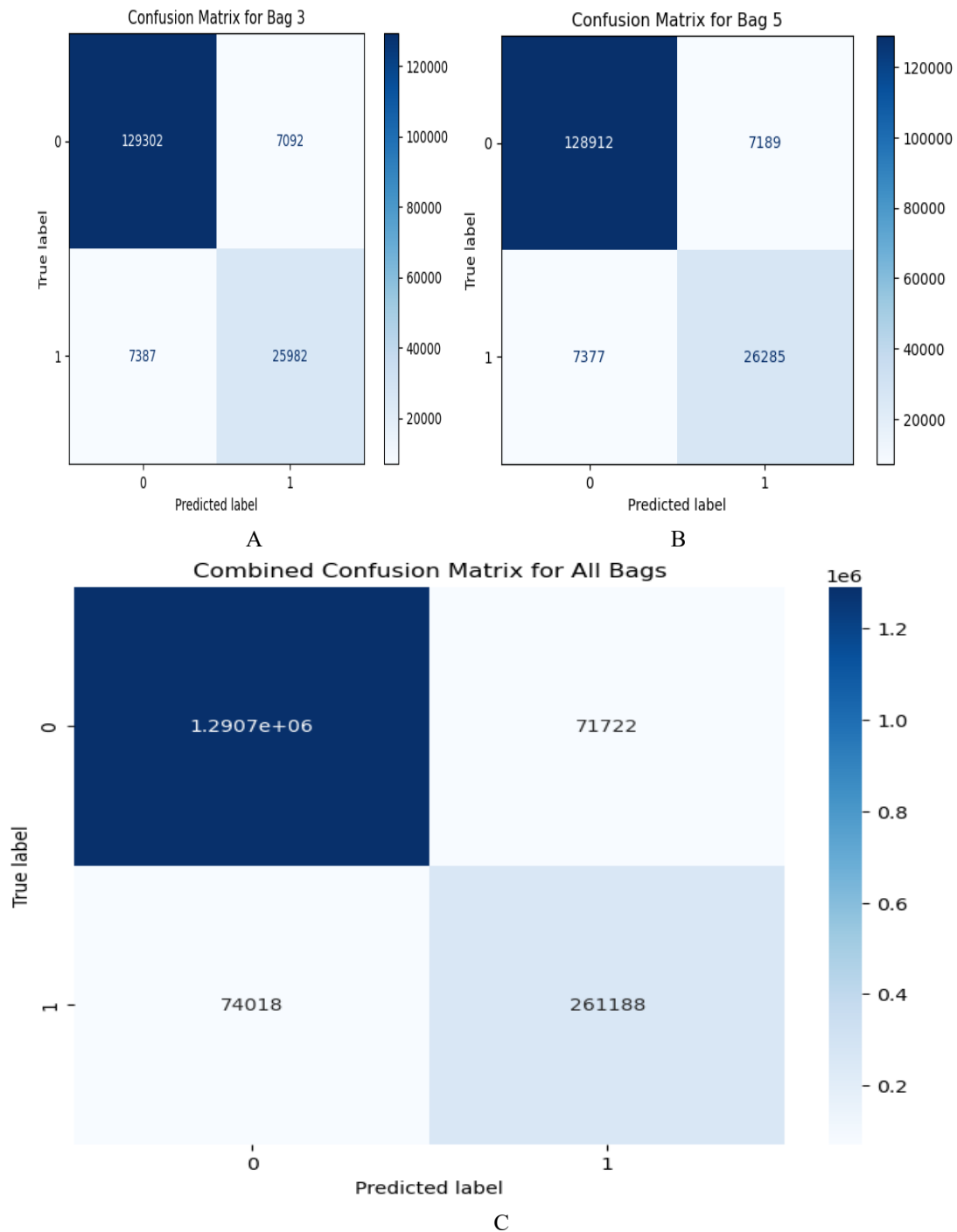
**Fig. 6:** (A) Confusion matrix for bag 1 using 19 dataset features shows 129,491 true negatives, 6,871 false positives, 8,249 false negatives, and 25,152 true positives. (B) The confusion matrix for another selected bag displays 129,241 true negatives, 6,946 false positives, 8,351 false negatives, and 25,225 true positives. (C) The aggregate confusion matrix across all bags reveals 1,294,350 true negatives, 69,040 false positives, 82,696 false negatives, and 251,540 true positives.



**Fig. 7:** (A) Confusion matrix for bag 2 using 38 dataset features shows 128,517 true negatives, 7,690 false positives, 8,227 false negatives, and 25,329 true positives. (B) The confusion matrix for another selected bag displays 128,719 true negatives, 7,585 false positives, 8,311 false negatives, and 25,148 true positives. (C) The aggregate confusion matrix across all bags reveals 1,287,110 true negatives, 75,864 false positives, 82,545 false negatives, and 252,111 true positives.



**Fig. 8:** (A) Confusion matrix for bag 2 using 56 dataset features shows 130,927 true negatives, 5,079 false positives, 10,522 false negatives, and 23,237 true positives. (B) The confusion matrix for another selected bag displays 131,033 true negatives, 5,236 false positives, 8,311 false negatives, and 25,148 true positives. (C) The aggregate confusion matrix across all bags reveals 1,287,110 true negatives, 75,864 false positives, 82,545 false negatives, and 252,111 true positives.



**Fig. 9:** (A) Confusion matrix for bag 3 using 76 dataset features shows 129,302 true negatives, 7,092 false positives, 7,387 false negatives, and 25,982 true positives. (B) The confusion matrix for another selected bag displays 128,912 true negatives, 7,189 false positives, 7,377 false negatives, and 26,285 true positives. (C) The aggregate confusion matrix across all bags reveals 1,290,700 true negatives, 71,722 false positives, 74,018 false negatives, and 260,188 true positives.

#### 4.4 Analysis of false negative and true negative

From the analysis of binary classification presented in Fig 6. Fig. 7, Fig. 8, and Fig. 9, we want to analyze in detail, the true and false negatives with the aim of identifying areas of improvement in the model. The analysis of false positives (FP) and false negatives (FN) from the study of the confusion matrices for binary classification using 19 features, Fig. 6 highlights critical aspects of the model's performance. In Bag 1, 6,871 benign instances were incorrectly flagged as malicious (FP), and 8,249 malicious instances were misclassified as benign (FN), while Bag 2 showed slightly higher counts of 6,946 FP and 8,351 FN. The aggregate confusion matrix across all bags revealed 69,040 FP and 82,696 FN, reflecting the model's overall trends. The high FP count indicates a tendency for false alarms, which can lead to unnecessary resource allocation and reduced trust in the IDS. Meanwhile, the substantial FN count points to missed detections of malicious traffic, posing significant security risks.

The analysis of the confusion matrices for binary classification using 38 features, Fig. 7 reveals important trends in model performance. In Bag 2 (A), the model correctly identified 128,517 benign instances as true negatives (TN) but misclassified 8,227 malicious instances as false negatives (FN). Similarly, in Bag 2 (B), the TN count increased slightly to 128,719, while the FN count rose to 8,311. Across all bags, the aggregate confusion matrix shows 1,287,110 TN and 82,545 FN, reflecting the overall trends. The high TN count indicates that the model effectively reduces false alarms, ensuring operational efficiency and reliability in benign traffic classification. However, the significant FN count highlights missed detections of malicious traffic, which could lead to security vulnerabilities.

The analysis of the confusion matrices for binary classification using 57 features Fig. 8 highlights key aspects of the model's performance. In Bag 2 (A), the model correctly identified 130,927 instances as true negatives (TN) and misclassified 10,522 malicious instances as false negatives (FN). In Bag 2 (B), the TN count increased to 131,033, while the FN count decreased significantly to 8,311. Across all bags, the aggregate confusion matrix shows 1,287,110 TN, 75,864 false positives (FP), 82,545 FN, and 252,111 true positives (TP). The high TN count across all configurations indicates that the model maintains strong accuracy in identifying benign traffic, minimizing false alarms. However, the FN count, especially the 10,522 in Bag 2 (A), reflects a challenge in detecting certain malicious traffic types, representing missed threats. These results suggest that while the model benefits from additional features, achieving high TN counts consistently, variability in FN counts across bags highlights potential inconsistencies in detecting malicious traffic.

The analysis of the confusion matrices for binary classification using 76 features Fig. 9, reveals the model's performance to detecting benign and malicious traffic. In Bag 3 (A), the model identified 129,302 instances as true negatives (TN) and misclassified 7,387 malicious instances as false negatives (FN). In Bag 3 (B), the TN count slightly decreased to 128,912, while the FN count was very similar at 7,377. The aggregate confusion matrix across all bags shows 1,290,700 TN, 71,722 false positives (FP), 74,018 FN, and 260,188 true positives (TP). The high TN count indicates the model's efficiency in accurately identifying benign traffic and minimizing false alarms. However, the FN count of 74,018 across all bags reveals a significant challenge in detecting malicious traffic, representing a substantial amount of undetected threats. While the increase in features to 76 slightly improved the TN and FN performance compared to previous configurations, the FN count remains high, suggesting that additional features alone may not be enough to address the issue of missed detections.

The solutions to the issue identified in this section form part of the suggested future research direction in the discussion section.

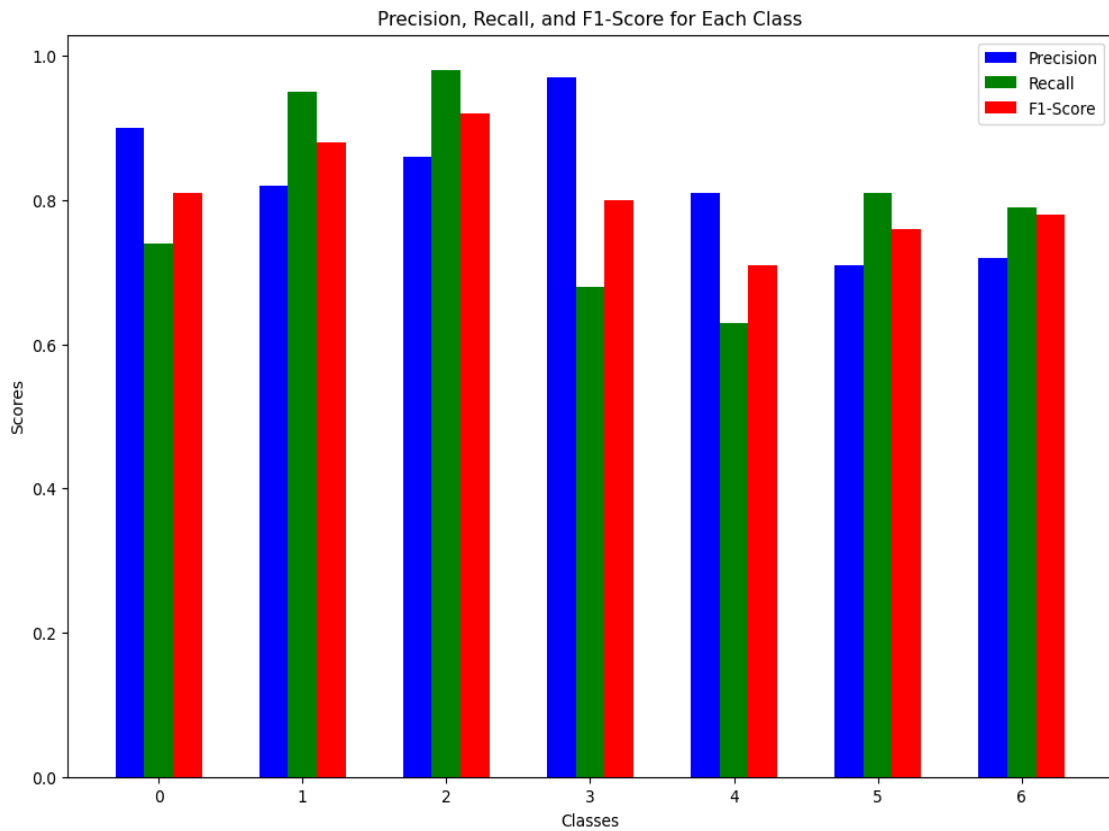
#### 4.5 Multiclassification

Multiclassification in intrusion detection systems (IDSs) extends beyond binary classification by categorizing network traffic into multiple types of attacks, which improves the granularity of threat detection [36]. The implementation of multiclassification enhances the ability of an IDS to respond to various cybersecurity threats more effectively, thereby improving overall network security [29].

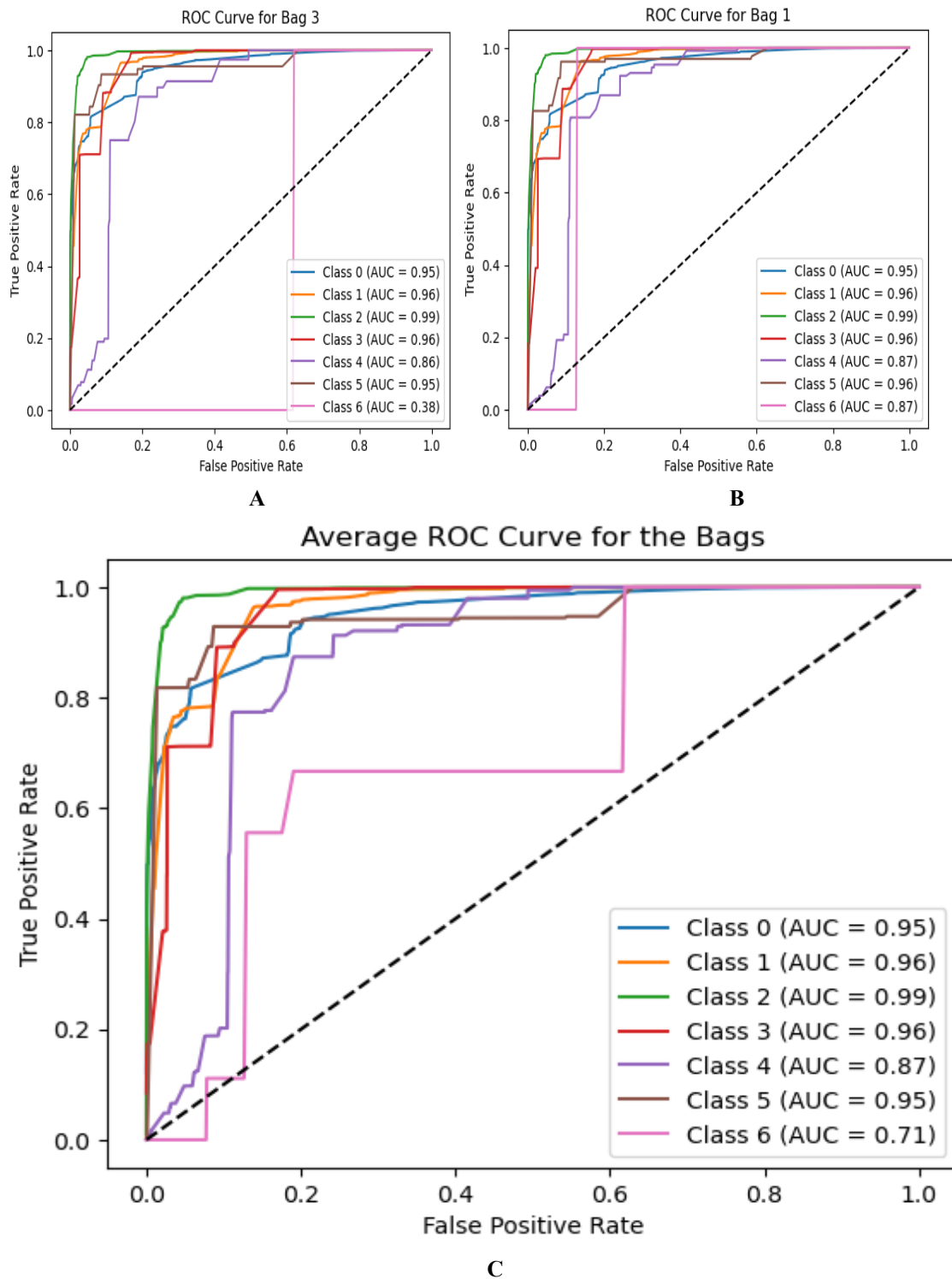
**Table 4.** Multiclassification metrics of the 7 classes

| Classes | Precision | Recall | F1-score |
|---------|-----------|--------|----------|
| 0       | 0.90      | 0.74   | 0.81     |
| 1       | 0.82      | 0.95   | 0.88     |
| 2       | 0.86      | 0.98   | 0.92     |
| 3       | 0.97      | 0.68   | 0.80     |

|   |      |      |      |
|---|------|------|------|
| 4 | 0.81 | 0.63 | 0.71 |
| 5 | 0.71 | 0.81 | 0.76 |
| 6 | 0.72 | 0.79 | 0.78 |



**Fig. 10:** Bar chart of multiclassification metrics of all the classes. The Fig. illustrates the precision, recall, and F1-score metrics for a multiclass classification problem across seven classes (labeled 0 to 6). The model demonstrates varying performance across the classes. Class 2 achieves the highest precision, recall, and F1-score, all near 1.0, indicating excellent accuracy and reliability in identifying instances of this class. In contrast, Class 0 shows relatively lower recall compared to its precision and F1-score, suggesting difficulties in capturing all instances and resulting in potential false negatives. For Class 3, recall is noticeably lower than precision and F1-score, highlighting a higher rate of missed predictions for this class. The metrics for the remaining classes (1, 4, 5, and 6) are well-balanced, with high and consistent values across precision, recall, and F1-score, reflecting reliable classification performance



**Fig. 11: Receiver Operating Characteristic (ROC) Curves for Different Bag Configurations**  
 This figure presents three subplots illustrating the ROC curves for various bag configurations. Subplot (A) shows the ROC curve for Bag 3, which illustrates the trade-off between sensitivity (true positive rate) and specificity (false positive rate) for different decision threshold values. Subplot (B) depicts the ROC curve for Bag 1, similarly revealing the classification performance and how well the model differentiates between positive and negative instances. Finally, subplot (C) presents the ROC curve for a

combined configuration of all bags, aggregating the results from Bags 1, 3, and potentially others. This combined curve provides an overall assessment of the model's classification ability across all configurations, allowing for a comparison with the individual bag curves to assess whether combining the bags improves model performance. The area under the curve (AUC) for each subplot quantifies the model's overall accuracy, with higher values indicating better performance.

## 5 Discussion and Conclusion

This study aims to develop an IDS to classify network traffic with underlying imbalances. The emphasis is on using deep learning algorithms owing to their higher performance over traditional machining. To leverage the wisdom of the masses, we adopted an ensemble technique called roughly balanced bagging (RBB). We used the mix max scaler to normalize our dataset so that the features had values between 0 and 1. This ensures that outliers are neutralized while preserving the shape of the distribution, which in turn propels convergence of the LSTM algorithm used as base learners. Before classifying the dataset, we used information gain to rank the features based on their importance to the outcome variables. We identified the top 19 features in Fig. 1, which represent the 1st quartile features from the dataset, based on their mutual information with the target variable. We repeated this process for the 2<sup>nd</sup> quartile Fig. 2, 3<sup>rd</sup> quartile Fig. 3, and the full feature set Fig. 4. We used SMOTE to select a subset of the dataset to be classified by the base learners. Finally, we used the RBB to select 10 bags of well-balanced datasets and presented them to 10 well-designed LSTM base learners for classification. The implementation of roughly balanced bagging in classification

For the binary classification metrics summarized in Table 3 and visualized in Fig. 5, the subset with 38 features achieved the lowest loss of 20.9%, while the subset with 56 features recorded the highest accuracy of 91.12%. The subset with 19 features delivered the highest precision of 91.12% and the highest recall of 96.76%. Additionally, the subset with 38 features attained the highest AUC (Area Under the Curve) of 97.00%, and the subset with 56 features achieved the highest F1 score of 91.12%. This emphasizes the significance of feature selection and the other innovations introduced in this study, aligning with prior research, such as [10], [27], and [31], which demonstrated the effectiveness of feature selection in optimizing the classification performance of network traffic.

We also presented the confusion matrices for our binary classification using 4 subsets of our dataset based on their information gain. Fig. 6 presents the confusion matrices for two selected bags and the combined confusion matrix for all bags using 19 features. Similarly, Fig. 7 shows the confusion matrices for two selected bags and the combined confusion matrix for all bags using 38 features, while Fig. 8 illustrates the confusion matrices for two selected bags and the combined confusion matrix for all bags using 57 features. Fig. 9 is the confusion matrix of 2 selected bags and the combined matrix for all the bags using 76 features. Analyzing individual confusion matrices from each bag allows us to understand how each bag's classification performs in isolation, capturing the variability and robustness of the model across different subsets of data. By aggregating these into a combined confusion matrix, we gain a comprehensive view of overall model performance

Our multiclassification metrics, Table 4, and its bar chart Fig. 10 were based on 7 categories (class labels) out of the 15 class labels in the dataset. Categorizing the outcome variable (class label) is also a common practice in network classification [11], [26], [27] when the number of attack types is similar to that in the case of our dataset. The performance of our methodology for multiclassification is measured by three metrics: recall, precision, and the F1 score. Receiver Operating Characteristic (ROC) Curves for our multiclassification are shown in Fig. 11. On average, class 2 Fig. (C) has the highest ROC of 99.00% and the highest recall of 98.00% Table 4 while Class 3 has the highest precision of 97.00% Table 4. Using different metrics for the evaluation of binary and multiclassification models is also a common practice [37]. Our results show that roughly balanced bagging can recognize minority classes with high accuracy compared to other ensembles, as opined by [38].

Our study has shown that with novel strategies such as the one presented here, the challenges of an imbalanced dataset will be averted to a greater extent.

Despite the strengths and contributions of this study, several challenges, limitations, and potential threats to validity must be acknowledged to provide a balanced interpretation of the findings. These challenges includes but not limited to:

**Imbalanced Dataset:** The CIC-IDS 2017 dataset had a significant imbalance between the number of records for certain attack types and normal traffic, making it challenging to build robust and unbiased models. This was addressed using the Synthetic Minority Oversampling Technique (SMOTE) to upsample the minority classes and create a more balanced dataset for classification.

**High Number of Class Labels:** The original dataset contained 15 distinct class labels, representing a wide variety of attacks and normal traffic. Managing and analyzing such many categories posed complexity in the classification task. To simplify the problem, these labels were grouped into 7 broader categories, ensuring a balance between granularity and model manageability.

**Large Volume of Data:** The dataset comprised 2.8 million records with 79 features, making data processing and feature selection computationally intensive. Preprocessing included removing features with infinity or NaN values, applying a min–max scaler, and ranking features based on information gain. These steps required careful execution to avoid errors and ensure data consistency.

**Feature Selection and Ranking:** Identifying the most relevant features for the classification models was critical to improving model performance and reducing computational load. This was achieved using information gain (mutual information), which required precision in ranking and selecting features for analysis.

**Binary and Multiclass Classification:** The study involved both binary classification (attacks vs. benign traffic) and multiclass classification (7 attack categories), necessitating separate preparation, modeling, and evaluation processes to cater to the nuances of each classification type.

**Challenges in Model Validation:** Ensuring the validity and reliability of the results required creating balanced subsets of the data using SMOTE and generating 10 independent bags for classification. This process increased computational demand and added complexity to the implementation.

#### **Future Research Directions to Improve the Model**

To address the challenges and limitations identified in the analysis of false negatives (FN) and false positives (FP) across various feature sets, the following areas of future research are recommended:

**Feature Engineering:** Adding features up to 76 slightly improved performance, particularly in reducing FN counts and increasing true negatives (TN). However, inconsistencies across feature sets and bags suggest further refinement. Future research should focus on enhancing the feature set by incorporating more discriminative and relevant features. This could involve domain-specific knowledge to identify critical indicators of malicious traffic.

**Model Optimization:** To balance precision, recall, and F1-scores, hyperparameter tuning should be explored. Techniques such as grid search or random search can help identify optimal settings for parameters like learning rate, regularization strength, and decision thresholds. Fine-tuning these parameters can help achieve a better trade-off between reducing false positives and minimizing false negatives.

**Error Analysis:** Detailed analysis of misclassifications, particularly the substantial FN counts observed across all feature sets, can provide valuable insights into the model's weaknesses. For example, understanding patterns in the 10,522 FN instances in Bag 2 (A) with 56 features, Fig. 8 or the 8,311 FN instances in Bag 2 (B) with 38 features, Fig 7 can guide targeted improvements in data preprocessing, feature selection, or model design.

**Advanced Training Strategies:** Incorporating advanced training techniques, such as adversarial training or transfer learning, could help the model adapt to subtle and evolving attack patterns. Additionally, training with balanced datasets or class-weighted loss functions could mitigate class imbalance and reduce FN rates for malicious traffic.

**Incorporating Real-world Scenarios:** Future research should evaluate the model's performance on real-world datasets to ensure its robustness in detecting malicious traffic under diverse conditions. This would provide practical insights into the model's application in operational environments.

#### **Declarations**

**Conflict of interests:** The authors have no financial or non-financial conflicting interest

**Acknowledgment:** This work was supported in part by the South African National Research Foundation under Grants numbers 141951, CHN231229201835,137951, and AJCR230704126719120106.

---

**Reference**

- [1] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review," *Procedia Comput Sci*, vol. 171, pp. 1251–1260, 2020. doi: [10.1016/j.procs.2020.04.133](https://doi.org/10.1016/j.procs.2020.04.133)
  - [2] M. Alkasassbeh and S. Al-Haj Baddar, "Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey," *Arab J Sci Eng*, vol. 48, no. 8, pp. 10021–10064, 2023. doi: [10.1007/s13369-022-07412-1](https://doi.org/10.1007/s13369-022-07412-1)
  - [3] E. Osa, P. E. Orukpe, and U. Iruansi, "Design and implementation of a deep neural network approach for intrusion detection systems," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 7, 2024. doi: [10.1016/j.prime.2024.100434](https://doi.org/10.1016/j.prime.2024.100434)
  - [4] M. Bhavsar, K. Roy, J. Kelly, and O. Olusola, "Anomaly-based intrusion detection system for IoT application," *Discover Internet of Things*, vol. 3, no. 1, 2023. doi: [10.1007/s43926-023-00034-5](https://doi.org/10.1007/s43926-023-00034-5)
  - [5] Z. Maasaoui, M. Merzouki, A. Bekri, A. Abane, A. Battou, and A. Lbath, "Design and Implementation of an Automated Network Traffic Analysis System using Elastic Stack," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, IEEE Computer Society, 2023. doi: [10.1109/AICCSA59173.2023.10479347](https://doi.org/10.1109/AICCSA59173.2023.10479347)
  - [6] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *Journal of Supercomputing*, vol. 79, no. 10, pp. 10611–10644, 2023. doi: [10.1007/s11227-023-05073-x](https://doi.org/10.1007/s11227-023-05073-x)
  - [7] H. Zhang, Z. Wu, Z. Wang, Z. Chen, and Y.G. Jiang, "Prototypical Residual Networks for Anomaly Detection and Localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16281–1629, 2023.
  - [8] P. Kaur and A. Gosain, "Issues and challenges of class imbalance problem in classification," *International Journal of Information Technology (Singapore)*, vol. 14, no. 1, pp. 539–545, 2022. doi: [10.1007/s41870-018-0251-8](https://doi.org/10.1007/s41870-018-0251-8)
  - [9] D. Dablain, B. Krawczyk, and N. V. Chawla, "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data," *IEEE Trans Neural Netw Learn Syst*, vol. 34, no. 9, pp. 6390–6404, 2023. doi: [10.1109/TNNLS.2021.3136503](https://doi.org/10.1109/TNNLS.2021.3136503)
  - [10] M. Temraz and M. T. Keane, "Solving the class imbalance problem using a counterfactual method for data augmentation," *Machine Learning with Applications*, vol. 9, no. 100375, pp. 1–16, 2022. doi: [10.1016/j.mlwa.2022.100375](https://doi.org/10.1016/j.mlwa.2022.100375)
  - [11] R. Mohammad, F. Saeed, A. A. Almazroi, F. S. Alsubaei, and A. A. Almazroi, "Enhancing Intrusion Detection Systems Using a Deep Learning and Data Augmentation Approach," *Systems*, vol. 12, no. 3, 2024. doi: [10.3390/systems12030079](https://doi.org/10.3390/systems12030079)
  - [12] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, "Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks," *Information (Switzerland)*, vol. 11, no. 5, 2020. doi: [10.3390/INFO11050243](https://doi.org/10.3390/INFO11050243).
  - [13] F. Demirkıran, A. Çayır, U. Ünal, and H. Dağ, "An ensemble of pre-trained transformer models for imbalanced multiclass malware classification," *Comput Secur*, vol. 121, Oct. 2022. doi: [10.1016/j.cose.2022.102846](https://doi.org/10.1016/j.cose.2022.102846)
  - [14] M. M. Rashid, J. Kamruzzaman, M. Ahmed, N. Islam, S. Wibowo, and S. Gordon, "Performance Enhancement of Intrusion Detection System Using Bagging Ensemble Technique with Feature Selection," in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2020*, Institute of Electrical and Electronics Engineers Inc., pp. 1–5, 2020. doi: [10.1109/CSDE50874.2020.9411608](https://doi.org/10.1109/CSDE50874.2020.9411608)
  - [15] H. P. Koapaha, "Bagging Based Ensemble Analysis In Handling Unbalanced Data On Classification Modeling," *Ananto Klabat Accounting Review*, vol. 2, no. 2, 2021.
  - [16] Z. H. Zhou, *Ensemble methods: foundations and algorithms.*, Illustrated. CRC Press., 2012.
  - [17] S. Hido, H. Kashima, and Y. Takahashi, "Roughly balanced Bagging for Imbalanced data," *Stat Anal Data Min*, vol. 2, no. 5–6, pp. 412–426, 2009. doi: [10.1002/sam.10061](https://doi.org/10.1002/sam.10061)
  - [18] M. Lango and J. Stefanowski, "SOUP-Bagging: a new approach for multi-class imbalanced data classification," Poznan, 2019.
  - [19] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer," *Expert Syst Appl*, vol. 148, no. 113249, 2020. doi: [10.1016/j.eswa.2020.113249](https://doi.org/10.1016/j.eswa.2020.113249)
-

- [20] S. D. Kurniabudi, M. Y. Darmawijoyo, M. B. Alwi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020. doi: [10.1109/ACCESS.2020.3009843](https://doi.org/10.1109/ACCESS.2020.3009843)
- [21] S. Hido, H. Kashima, and Y. Takahashi, "Roughly balanced Bagging for Imbalanced data," *Stat Anal Data Min*, vol. 2, no. 5–6, pp. 412–426, 2009. doi: [10.1002/sam.10061](https://doi.org/10.1002/sam.10061)
- [22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [23] A. A. Alqarni and E.-S. M. El-Alfy, "Improving Intrusion Detection for Imbalanced Network Traffic using Generative Deep Learning," *IJACSA International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, pp. 959–967, 2022.
- [24] A. D. Vibhute and V. Nakum, "Deep learning-based network anomaly detection and classification in an imbalanced cloud environment," in *Procedia Computer Science*, Elsevier B.V., pp. 1636–1645, 2024. doi: [10.1016/j.procs.2024.01.161](https://doi.org/10.1016/j.procs.2024.01.161)
- [25] N. Gupta, V. Jindal, and P. Bedi, "CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems," *Comput Secur*, vol. 112, 2022. doi: [10.1016/j.cose.2021.102499](https://doi.org/10.1016/j.cose.2021.102499)
- [26] M. Mbow, H. Koide, and K. Sakurai, "An Intrusion Detection System for Imbalanced Dataset Based on Deep Learning," in *Proceedings - 2021 9th International Symposium on Computing and Networking, CANDAR 2021*, Institute of Electrical and Electronics Engineers Inc., pp. 38–47, 2021. doi: [10.1109/CANDAR53791.2021.00013](https://doi.org/10.1109/CANDAR53791.2021.00013)
- [27] R. Harini, N. Maheswari, S. Ganapathy, and M. Sivagami, "An effective technique for detecting minority attacks in NIDS using deep learning and sampling approach," *Alexandria Engineering Journal*, vol. 78, pp. 469–482, 2023. doi: [10.1016/j.aej.2023.07.063](https://doi.org/10.1016/j.aej.2023.07.063)
- [28] B. Bowen, A. Chennamaneni, A. Goulart, and D. Lin, "BLoCNet: a hybrid, dataset-independent intrusion detection system using deep learning," *Int J Inf Secur*, vol. 22, no. 4, pp. 893–917, 2023, doi: [10.1007/s10207-023-00663-5](https://doi.org/10.1007/s10207-023-00663-5)
- [29] B. Xu, L. Sun, X. Mao, C. Liu, and Z. Ding, "Strengthening Network Security: Deep Learning Models for Intrusion Detection with Optimized Feature Subset and Effective Imbalance Handling," *Computers, Materials and Continua*, vol. 78, no. 2, pp. 1995–2022, 2024. doi: [10.32604/cmc.2023.046478](https://doi.org/10.32604/cmc.2023.046478)
- [30] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, "A Deep Learning Model for Network Intrusion Detection with Imbalanced Data," *Electronics (Switzerland)*, vol. 11, no. 6, 2022. doi: [10.3390/electronics11060898](https://doi.org/10.3390/electronics11060898)
- [31] M. S. Milosevic and V. M. Ciric, "Extreme minority class detection in imbalanced data for network intrusion," *Comput Secur*, vol. 123, 2022. doi: [10.1016/j.cose.2022.102940](https://doi.org/10.1016/j.cose.2022.102940)
- [32] B. Xu, L. Sun, X. Mao, C. Liu, and Z. Ding, "Strengthening Network Security: Deep Learning Models for Intrusion Detection with Optimized Feature Subset and Effective Imbalance Handling," *Computers, Materials and Continua*, vol. 78, no. 2, pp. 1995–2022, 2024. doi: [10.32604/cmc.2023.046478](https://doi.org/10.32604/cmc.2023.046478)
- [33] J. Hochreiter, "Hochreiter, J. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München. See <http://www7.informatik.tu-muenchen.de/~hochreit.>" Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München., 1991.
- [34] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [35] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans Emerg Top Comput Intell*, vol. 2, no. 1, pp. 41–50, 2018. doi: [10.1109/TETCI.2017.2772792](https://doi.org/10.1109/TETCI.2017.2772792)
- [36] D. Li, L. Yang, H. Zhang, X. Wang, and L. Ma, "Memory-Augmented Insider Threat Detection with Temporal-Spatial Fusion," *Security and Communication Networks*, vol. 2022. doi: [10.1155/2022/6418420](https://doi.org/10.1155/2022/6418420)
- [37] A. Alrefaei and M. Ilyas, "Using Machine Learning Multiclass Classification Technique to Detect IoT Attacks in Real Time," *Sensors*, vol. 24, no. 14, p. 4516, Jul. 2024. doi: [10.3390/s24144516](https://doi.org/10.3390/s24144516)
- [38] M. Lango and J. Stefanowski, "Applicability of Roughly Balanced Bagging for Complex Imbalanced Data," in *The 4th Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2015)*, C. Michelangelo, L. Corrado, M. Giuseppe, M. Elio, and W. R. Zbigniew, Eds., 2015, pp. 62–74.
-

