Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.056308

ARTICLE

Augmenting Internet of Medical Things Security: Deep Ensemble Integration and Methodological Fusion

Hamad Naeem[1], Amjad Alsirhani[2,*], Faeiz M. Alserhani[3], Farhan Ullah[4] and Ondrej Krejcar[1]

[1]Faculty of Informatics and Management, Center for Basic and Applied Research, University of Hradec Kralove, Hradec Kralove, 50003, Czech Republic

[2]Department of Computer Science, College of Computer and Information Sciences, Jouf University, Al Jouf, 72388, Saudi Arabia

[3]Department of Computer Engineering & Networks, College of Computer and Information Sciences, Jouf University, Al Jouf, 72388, Saudi Arabia

[4]Cybersecurity Center, Prince Mohammad Bin Fahd University, Al Jawharah, Khobar, Dhahran, 34754, Saudi Arabia

*Corresponding Author: Amjad Alsirhani. Email: amjadalsirhani@ju.edu.sa

Received: 19 July 2024 Accepted: 08 October 2024 Published: 31 October 2024

ABSTRACT

When it comes to smart healthcare business systems, network-based intrusion detection systems are crucial for protecting the system and its networks from malicious network assaults. To protect IoMT devices and networks in healthcare and medical settings, our proposed model serves as a powerful tool for monitoring IoMT networks. This study presents a robust methodology for intrusion detection in Internet of Medical Things (IoMT) environments, integrating data augmentation, feature selection, and ensemble learning to effectively handle IoMT data complexity. Following rigorous preprocessing, including feature extraction, correlation removal, and Recursive Feature Elimination (RFE), selected features are standardized and reshaped for deep learning models. Augmentation using the BAT algorithm enhances dataset variability. Three deep learning models, Transformer-based neural networks, self-attention Deep Convolutional Neural Networks (DCNNs), and Long Short-Term Memory (LSTM) networks, are trained to capture diverse data aspects. Their predictions form a meta-feature set for a subsequent meta-learner, which combines model strengths. Conventional classifiers validate meta-learner features for broad algorithm suitability. This comprehensive method demonstrates high accuracy and robustness in IoMT intrusion detection. Evaluations were conducted using two datasets: the publicly available WUSTL-EHMS-2020 dataset, which contains two distinct categories, and the CICIoMT2024 dataset, encompassing sixteen categories. Experimental results showcase the method's exceptional performance, achieving optimal scores of 100% on the WUSTL-EHMS-2020 dataset and 99% on the CICIoMT2024.

KEYWORDS

Cyberattack; ensemble learning; feature selection; intrusion detection; smart cities; machine learning; BAT augmentation

1 Introduction

The advent of contemporary information and communication technologies has facilitated the utilization of Internet of Things (IoT) medical equipment by both patients and medical practitioners through online services. Current industry research projects that the IoMT market could reach $135 billion by 2025 [1]. On the contrary, cybercriminals across the globe have elevated IoMT devices and their networks to the status of primary targets. The majority of Internet of Medical Things (IoMT) devices lack security considerations during development, rendering them vulnerable to vulnerabilities. In 2021, IoMT and healthcare-related sectors accounted for 72 percent of all malicious traffic. The total number of healthcare cyberattacks increased by 40% in 2021, with 81% of providers admitting to having at least one compromised IoMT system [2]. The reported figures regarding IoMT underscore the criticality of security measures in safeguarding patient information within IoMT systems and networks. The present IT security methods exhibit challenges in accommodating IoMT devices and networks due to their intricate nature, memory constraints, and then heterogeneity. In the initial phases of IoMT security, trust-based approaches, encryption, and decryption methods based on cryptography, and authentication were all implemented.

A review of the pertinent literature indicates that intrusion detection has supplanted cryptographic methods in recent times. Because implementing cryptographic solutions on IoMT devices with limited memory is notoriously difficult. An intrusion detection system (IDS) may monitor an entire network or a single computer in the context of the Internet of Medical Things (IoMT) and alert the system administrator to any malicious or suspicious activity [3]. Although the primary emphasis of this study is on network-based intrusion detection systems, it does make a brief reference to host-based systems as well [4]. IoMT devices are not optimal candidates for host-based intrusion detection systems due to their constrained RAM. Consequently, network-based systems may be deployed at the network node of the IoMT gateway. The prevailing methodology utilized to identify and classify attacks was intrusion detection, which relied on anomalies and rules [2]. Determining contemporary patterns of attacks is a straightforward task for rule-based systems. Despite being more favored than anomaly-based systems, rule-based systems exhibit limitations in their ability to detect novel or diverse threats. This issue is significantly exacerbated by the proliferation of false alarms in anomaly-based systems. Anomaly-based systems, on the other hand, can detect both known and unknown types of threats.

Previous studies on IoMT-based intrusion detection: A recent literature review [3,4] states that network intrusion detection systems (IDSs) employ machine learning and deep learning techniques to identify both known and unknown attacks, as well as variations on existing ones. These techniques replaced rule-based systems in IoMT-Network IDS. To safeguard IoMT networks and devices against intruders, interruption finding is an indispensable component of the IoMT ecosystem. The authors [1,2] present an exhaustive synopsis of the strengths and weaknesses of existing security procedures in an IoMT environment in their discussion of these topics. It is proposed that an optimization-based deep neural network (DNN) be utilized aimed at interruption finding for IoMT construction [5]. In the context of the incursion dataset sourced from the 1999 Knowledge Discovery and Data Mining Tools Competition (KDDCup-99), the proposed solution demonstrated a 15% enhancement over prior methodologies. An exhaustive examination was undertaken utilizing the Telemetry operational organizations Network traffic IoT (ToN-IoT) database. In this study, a collaborative machine learning method aimed at intrusion detection in the IoMT is proposed [6].

When compared to a single ML model, the ensemble models performed considerably better. For intrusion detection in IoMT, two models were proposed: one utilizing gradient boosting and the other utilizing transformers [7]. It demonstrated how the model performed using the ToN-IoT in addition

to the Endgame Malware Standard aimed at Exploration (Ember) datasets. In each experiment, the suggested method demonstrated superior performance associated to the current state-of-the-art models. To detect intrusions within the IoMT environment, recurrent neural networks (RNNs) are recommended by the authors [8]. The performance of the models can be demonstrated through the utilization of the NSL-KDD dataset, an abbreviation for "network security laboratory-knowledge discovery in addition data mining apparatuses antagonism 1999." Researchers supported the assertions made by their model by drawing comparisons to more traditional machine learning methodologies. The utilization of a swarm neural network for IoMT intrusion discovery is illustrated regarding the ToN-IoT database [9]. A proposed approach for interruption finding for IoMT utilizes active learning [10]. The random forest model demonstrated a superior accuracy of 96.44% when applied to the intrusion detection assessment dataset (CIC-IDS2017) compared to alternative models. It should be noted that none of the datasets—KDDCup-99, ToN-IoT, NSL-KDD, in addition, Ember—contain network flows that accurately reflect an IoMT environment in practice. Despite the research suggesting superior outcomes, prototypes competent on the KDDCup-99, ToN-IoT, NSL-KDD, and CIC-IDS2017, in addition, the Ember database would struggle to precisely recognize outbreaks for IoMT surroundings. A proposed anomaly-based model [11] utilizes data characteristics obtained from gateways, IoT devices, network traffic, and information on CPU and memory utilization to analyze the IoMT environment. Although the performance of the models has been enhanced, the false alarm rate of the anomaly-based strategy in a real-world IoMT environment is quite high. As presented the mobile agent-based intrusion detection methodology is aimed at the IoMT environment [12]. Although the researchers conducted an extensive assessment of the intrusion detection capabilities of numerous traditional machine learning models, it should be noted that the dataset used in the study did not originate from an actual IoMT environment. By leveraging attributes extracted from network flows and patient biometrics sensing data, we present an IoMT-based intrusion detection system that employs machine learning. Its performance is illustrated using a practical IoMT dataset [13]. Further improvement of the performance referred to by the authors is feasible. This research represents a paradigm shift for intrusion detection in IoMT settings. Washington Campus of Saint-Louis boosted healthcare observing scheme 2020 (WUSTL EHMS 2020) dataset remained obtained regarding IoMT test bed in addition is publicly available; hence, additional research could be conducted to improve the described outcomes regarding intrusion detection aimed for IoMT environment. For IoMT intrusion detection, the authors subsequently made available for Edith Cowan University Internet of Health Things (ECU-IoHT) database. Through evaluating patient biometric data and features of network traffic, our deep learning-based approach can identify intrusions in IoMT networks. This study grants the deep learning grounded ensemble method to network-grounded interference finding for IoMT systems by combining features of network traffic with patient biometrics. A random forest feature significance model may be trained to understand complex, non-linear, and overlapping medical characteristics.

An improved version of the expanded ensemble is produced by including predictions from initial learners into a meta-learner. In terms of overall performance, the proposed system was far better than all prior systems; more importantly, it was able to detect IoMT attacks with much greater precision. The combination of a deep learning model and a cost-sensitive learning technique allowed for this to be accomplished. The proposed method also shows similar performance when evaluated on several network-based industrial benchmark datasets. As a result, the proposed approach for IoMT-based intrusion detection systems is robust enough to precisely detect assaults in addition to aware the network administrator for essential arrangements. Attacks that target network traffic aren't the only ones that may be identified. Evaluating the efficacy of the suggested model happening attack

classification and adding more categories to the IoMT assault dataset are two areas that will be investigated in future research. Layers are used to structure the properties of deep learning. Learning techniques that use kernels to fuse features are well-suited for use in the classification layer. The model, besides the classification accuracy of the IoMT network, is improved by this sort of layer. Nevertheless, it could be argued that the WUSTL EHMS 2020 dataset is superior in quality to ECU-IoHT. ECU-IoHT has significantly fewer network features than WUSTL EHMS 2020, which incorporates network flows and patient biometrics. To surpass the performance of [13], the authors [14] implemented a tree-based classifier by utilizing data pre-processing and data augmentation techniques—which were required due to the extreme imbalance in WUSTL EHMS 2020. For intrusion detection in WUSTL EHMS 2020, a variety of models based on deep neural networks and traditional machine learning were implemented [15], following the minimization of the number of features using an optimization-based method. This research indicates that the model exhibited superior performance compared to both [13] and [14]. The findings cannot be directly compared to those of [13] due to the utilisation of data preparation and data intensification methods to introduce data disparity preceding to machine learning model training [14,15]. While feature extraction methods [14,15] contribute significantly to the development of intrusion detection systems for IoMT environments, the accuracy claimed by various models [16,17] is unlikely to hold consistently. Furthermore, the model's performance is inherently tied to the effectiveness of data pre-processing and augmentation techniques used to address data imbalances.

The literature on the subject suggests that cost-sensitive learning methods in data mining are more effective at rectifying data imbalances when compared to basic data pre-processing and data augmentation procedures. Instead of data pre-processing and data augmentation [13], the current study proposes a cost-sensitive learning strategy to address the unbalanced WUSTL EHMS 2020 dataset. The study proposes a comprehensive methodology for IoMT-based intrusion detection, leveraging a multi-model approach with data augmentation, feature selection, and ensemble learning. It starts with data preprocessing, including feature extraction, correlation removal, and recursive feature elimination for dimensionality reduction. Standardization ensures uniformity for deep learning models, which include transformer-based neural networks, self-attention DCNNs, and LSTMs for diverse data aspects. Augmentation via the BAT algorithm enhances dataset variability. Base model outputs are stacked into meta-features, and trained with a meta-learner for comprehensive predictions. Further validation with traditional classifiers ensures feature informativeness across algorithms.

Current challenges in the IoMT domain and our key contributions:

- In response to the inadequacies of current IoMT systems in detecting sophisticated attacks like data injection and DoS, we propose an innovative IoMT Intrusion Detection System (IDS). This system integrates transformer-based neural networks, self-attention DCNNs, and LSTMs to achieve comprehensive threat detection capabilities.
- To address the limitations of existing IoMT data preprocessing methods, particularly in handling the complexity and variability of attack patterns found in datasets such as WUSTL-EHMS-2020 and CICIoMT2024, we advocate for implementing Recursive Feature Elimination and data standardization techniques. These enhancements aim to improve feature relevance and model performance, ensuring robust detection of attack signatures within IoMT environments.
- Recognizing the shortcomings of traditional IoMT security measures in adapting to evolving attack methodologies, such as sophisticated MitM attacks prevalent in real-world IoMT deployments, we propose integrating the BAT algorithm. This approach leverages data augmentation and meta-learning techniques applied to stacked model outputs, thereby enhancing dataset

resilience and prediction accuracy. Ultimately, these innovations strengthen IoMT systems against emerging threats, ensuring enduring robustness in security measures and enhancing their resilience in practical applications.

Section 2 thoroughly discusses the datasets and the proposed methodology. Section 3 offers an extensive analysis of the experimental results generated by the proposed approach. Finally, Section 4 concludes with insights into future research directions.

2 Proposed IoMT-Based Intrusion Detection System

In the rapidly evolving Internet of Medical Things (IoMT) environment, securing sensitive data and ensuring device integrity are critical priorities. To achieve a robust security framework, a multi-layered approach is necessary, where key network components such as firewalls, routers, and Intrusion Detection Systems (IDS) collectively enhance the system's defense mechanisms.

Firewalls act as the initial barrier by filtering incoming and outgoing traffic according to predefined security rules, effectively blocking unauthorized access and mitigating the risk of external threats. In parallel, routers, which are primarily responsible for directing network traffic, play an important role in securing data transmission pathways. Modern routers, equipped with advanced security features, help detect and block suspicious activities, ensuring that network traffic is routed through secure channels and minimizing potential attack vectors [15].

At the heart of network security lies the Intrusion Detection System (IDS), which is designed to monitor and detect malicious activities or breaches within the network. This is particularly critical in IoMT ecosystems, where devices are interconnected and may be vulnerable to a variety of attacks. IDS solutions can be categorized into two main types: signature-based IDS, which identifies known threats by matching patterns with a database of threat signatures, and anomaly-based IDS, which detects deviations from normal system behavior, flagging potential new or unknown attacks, including zero-day threats. By continuously monitoring network activity, IDS complements the firewall's protective role and adds a layer of dynamic, real-time threat detection.

In a real-world IoMT environment, the combined efforts of firewalls, routers, and IDS form a comprehensive security system. Firewalls prevent unauthorized access, routers ensure secure data transmission, and IDS provides real-time monitoring and detection of sophisticated threats that may bypass initial defenses. Together, these components not only enhance the overall safety and security of the system but also protect the integrity and confidentiality of sensitive medical data, which is essential for maintaining trust and compliance with industry standards.

We aimed to improve the detection performance for identifying IoMT attacks by using patient biometric data and network traffic. Initially, we established the foundation for an IoMT intrusion detection system before introducing our proposed method. Fig. 1 illustrates an IoMT network designed for security monitoring and threat prediction through deep learning. This architecture comprises patient monitoring devices, an IoT gateway, a network traffic collector, a data processing pipeline leveraging deep learning, an intrusion detection system, and security operators who monitor for potential threats. The sensor devices might include but are not limited to, temperature sensors, pulse rate detectors, heart rate monitors, ECG devices, blood pressure monitors, and respiration rate sensors. Information from IoT sensing devices is transmitted to remote servers using IoT network protocols like MQ Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP). IoT gateways transmit sensor data to remote locations via wired and wireless connections. The proposed approach involves gathering both patient biometric data and network traffic, which is processed by the intrusion

detection system (IDS) illustrated in Fig. 1. This IDS uses advanced data analytics and deep learning models to identify and predict threats within the IoMT network. Continuous system-level monitoring and analysis are vital to enhance alert accuracy and reduce false positives. Hospitals can utilize the IoMT system for remote patient monitoring and to oversee their physical premises, ensuring patient safety and security. Real-world IoMT networks are often characterized by heterogeneous devices operating across multiple platforms and protocols, from Wi-Fi to cellular networks. The proposed IDS is designed to be adaptable to such environments, handling varied sensor types, devices, and communication methods. Additionally, the system's scalability allows it to function across small-scale hospital networks to large-scale national healthcare infrastructures. Latency and real-time processing are essential factors in real-world applications, especially in healthcare, where time-sensitive actions are critical. The proposed IDS incorporates optimized machine learning algorithms to ensure low-latency threat detection, making it suitable for time-sensitive patient monitoring systems. Furthermore, reducing false positives in a real-world setting is key to ensuring that security operators are not overwhelmed with non-actionable alerts. The IDS uses continuous learning models that improve over time, enhancing alert accuracy and reducing false alarms. The system's ability to adapt to evolving threats ensures long-term relevance, as it can continuously learn from new attack patterns and update its threat detection models.

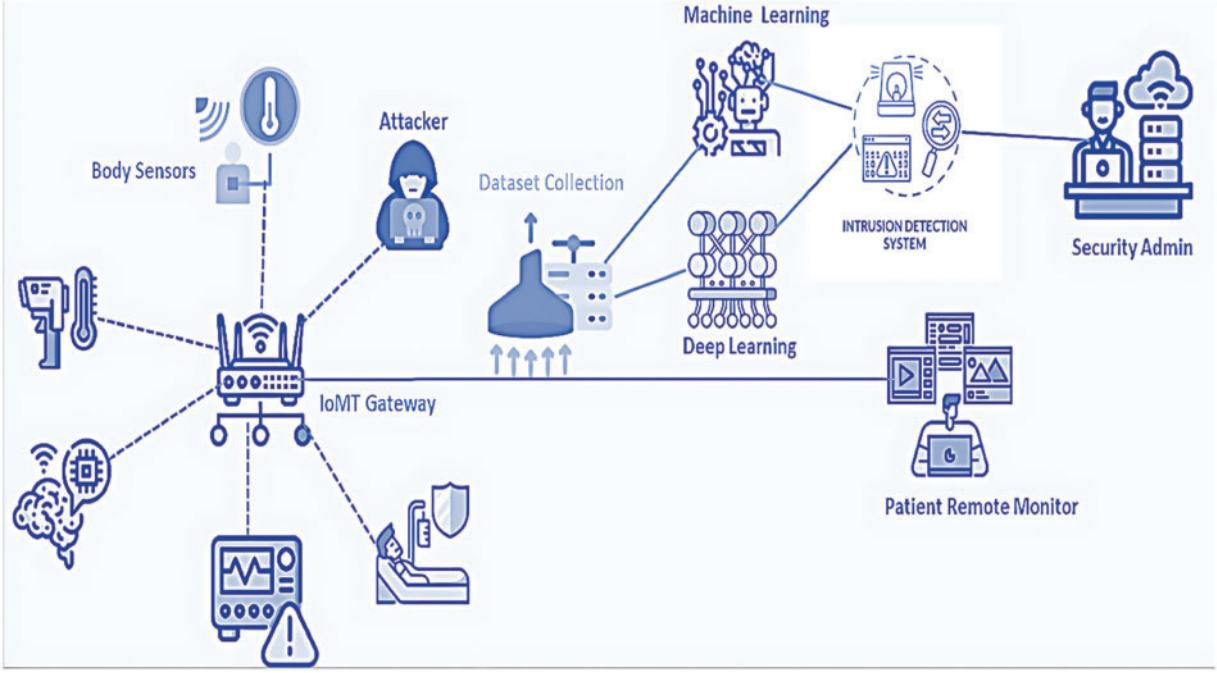


Figure 1: Standard IoMT network architecture for intrusion attack detection

The proposed methodology (Fig. 2) presented in this study is highly effective for IoMT-based intrusion detection due to its multi-faceted approach. By combining the strengths of different models and employing data augmentation, feature selection, and ensemble learning techniques, we can achieve a high level of accuracy and robustness. This approach is particularly effective in handling the complex and diverse nature of IoMT data, making it well-suited for detecting and mitigating intrusion attacks in IoMT environments.

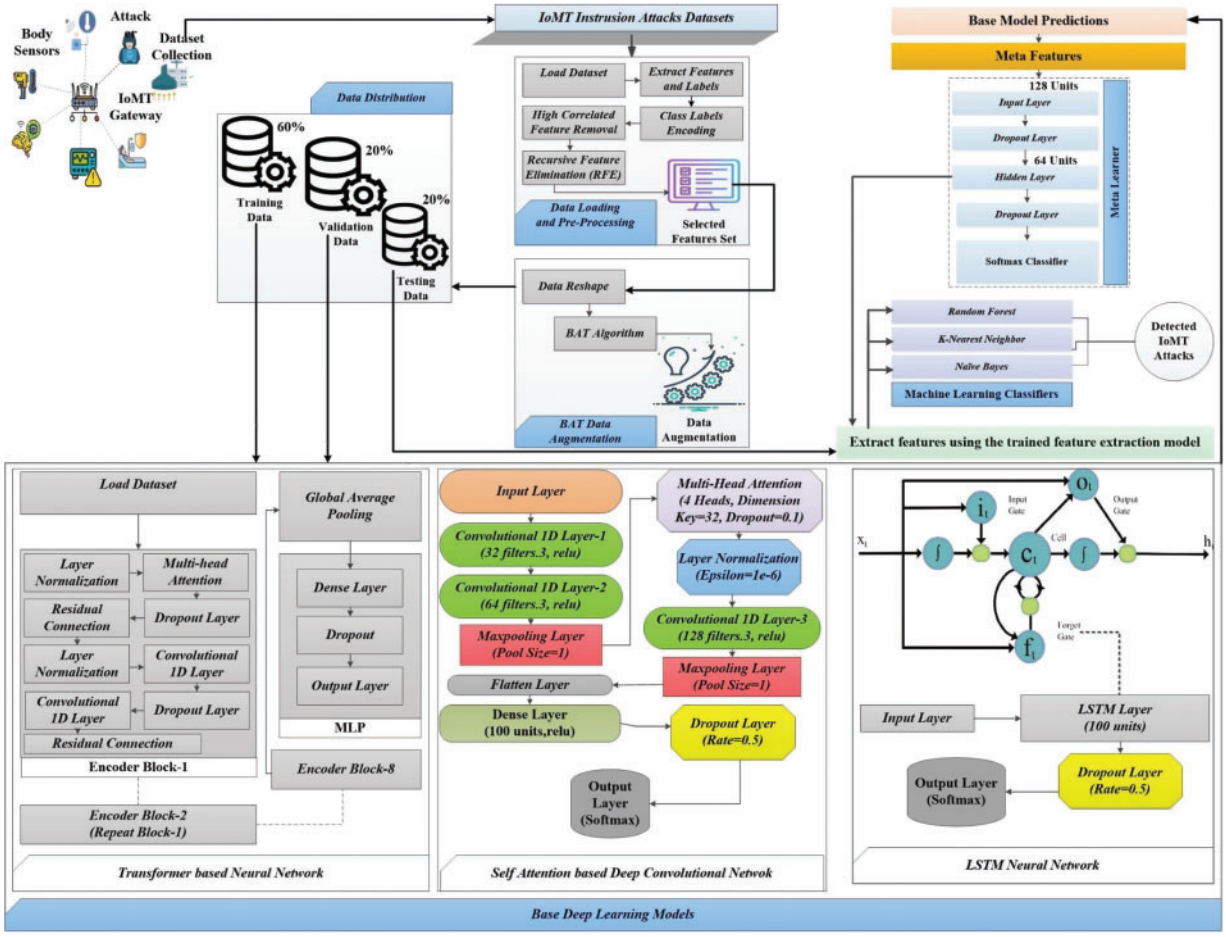


Figure 2: Proposed IoMT-based intrusion attacks detection scheme

In this research, we developed a comprehensive methodology to detect IoMT-based intrusion attacks with high accuracy. The process begins with data loading and preprocessing. The dataset is initially loaded, and features and labels are extracted. Class labels are encoded, and highly correlated features are removed to prevent multicollinearity, enhancing model performance. We then apply Recursive Feature Elimination (RFE) using a Random Forest Classifier to select the most relevant features, further reducing dimensionality and focusing on the most informative attributes. The selected features are standardized to ensure they are on the same scale, which is crucial for optimizing the performance of many deep learning algorithms. Next, the data is reshaped to fit the input requirements of various deep-learning models. The BAT algorithm is employed to generate augmented samples, enriching the training dataset and improving the model's robustness to variability. The BAT algorithm adds random noise to the original samples, creating multiple variations that simulate different possible scenarios of the data. The methodology proceeds with splitting the data into training, validation, and test sets. Three base deep learning models (transformer-based neural network, self-attention DCNN, LSTM) are built and trained using the prepared dataset. Each model captures different aspects of the data: the Transformer leverages attention mechanisms for contextual understanding, the self-attention DCNN combines convolutional and attention mechanisms for spatial and temporal feature

extraction, and the LSTM excels in sequential data modeling. After training, predictions from these base models are obtained, and their outputs are stacked to form a meta-feature set. A meta-learner model is then built and trained on these meta-features, utilizing a combination of dense layers and dropout for regularization. This meta-learner integrates the strengths of the base deep learning models, offering a comprehensive prediction by leveraging the diverse perspectives of the individual models. To further enhance the effectiveness of our approach, conventional classifiers such as RandomForest, Gaussian Naive Bayes, and K-Nearest Neighbors are trained on the features extracted by the meta-learner. This step ensures that the extracted features are highly informative and suitable for various machine learning algorithms.

2.1 Datasets

1. **WUSTL-EHMS-2020 Dataset [18]:** The WUSTL-EHMS-2020 dataset focuses on a variety of intrusion attacks pertinent to IoT environments, which are increasingly common in healthcare settings. In healthcare, IoT devices such as patient monitors, smart medical devices, and connected health applications are critical for patient care and operational efficiency. The dataset includes several types of attacks that are highly relevant to these IoT devices:

Denial of Service (DoS): This attack can disrupt the operation of critical healthcare services by overwhelming devices or networks with traffic, leading to delays in patient care or unavailability of essential medical services.

Data Injection: In a healthcare context, this could involve maliciously altering patient data, leading to incorrect diagnoses or treatment plans.

Reconnaissance: Attackers gathering information about the network can identify vulnerabilities in healthcare systems, potentially leading to more severe attacks.

The relevance of WUSTL-EHMS-2020 lies in its focus on these IoT-specific threats, providing a robust framework for developing intrusion detection systems (IDS) that safeguard connected healthcare environments from targeted attacks on their IoT infrastructure.

2. **CICIoMT2024 Dataset [19]:** The CICIoMT2024 dataset offers a broader range of attacks, encompassing both traditional network-based attacks and those targeting IoT environments. This diversity is particularly useful in healthcare settings where traditional IT infrastructure and IoT devices coexist. Key types of attacks in this dataset include:

Distributed Denial of Service (DDoS): Similar to DoS but more distributed, potentially affecting large-scale healthcare networks and services, such as electronic health record (EHR) systems, telemedicine services, and other critical healthcare applications.

Brute Force: Attackers might target healthcare administration systems or medical devices to gain unauthorized access to sensitive data.

Phishing: Healthcare employees are often targeted with phishing attacks to gain access to confidential patient information or to install malware on the network.

IoT-Specific Attacks: These include attacks on medical IoT devices such as insulin pumps, heart monitors, and other connected medical equipment, potentially endangering patient lives.

The CICIoMT2024 dataset's comprehensive coverage of various attack types, including sophisticated multi-stage and multi-vector attacks, makes it exceptionally valuable for developing IDS solutions tailored to the intricate and high-stakes nature of healthcare environments. It enables researchers to create models capable of detecting and mitigating a wide array of threats, ensuring

the security and reliability of both traditional IT and IoT-based healthcare systems. In summary, the WUSTL-EHMS-2020 dataset is particularly focused on IoT-specific attacks, making it highly relevant for healthcare environments with a significant IoT component. The CICIoMT2024 dataset, with its broader range of attack types, provides a more comprehensive foundation for developing IDS solutions that can protect against a wide variety of threats in healthcare settings, including both traditional IT infrastructure and modern IoT devices. Both datasets are invaluable for advancing cybersecurity in healthcare, helping to safeguard patient data, ensuring the reliability of medical devices, and maintaining the overall integrity of healthcare services. Table 1 summarizes healthcare intrusion attacks.

Table 1: Summary of intrusion attack datasets for IoMT

Dataset	Attack family	Number of samples in category	Total number of samples
WUSTL-EHMS-2020	DoS	12,000	
	Data injection	8500	
	Reconnaissance	10,000	
	Normal	15,000	
			45,500
CICIoMT2024	DDoS	20,000	
	Brute force	10,000	
	Phishing	12,500	
	IoT-specific attacks	18,000	
	Normal	25,000	
			85,500

2.2 Data Pre-Processing and Feature Selection

Processing datasets such as CICIoMT2024 and WUSTL-EHMS-2020 is a crucial step in preparing data for deep learning models. This involves reading the input CSV file into a Data Frame and systematically cleaning and transforming the data to ensure it is suitable for analysis. Initially in Algorithm 1, the process calculates a threshold to identify columns with more than 50% zero values, marking them for removal as they may not contribute useful information to the model. This step helps in reducing the dimensionality of the data and removing noise, which can improve the performance of the models. Next, the algorithm drops duplicate rows and rows with NaN values, which further cleans the dataset, ensuring that the data is accurate and reliable. After these cleaning steps, the pre-processed Data Frame is saved to an output file, and the removed columns and output file path are printed for reference. This transparency helps in understanding which parts of the data were deemed irrelevant or incomplete.

Algorithm 1: Data pre-processing and feature selection

Require: *input_file* (str)—Path to the input CSV file
Require: *output_file* (str)—Path to save the preprocessed data
Ensure: **X** (numpy array)—Preprocessed features
Ensure: **y** (numpy array)—Encoded labels

(Continued)

Algorithm 1 (continued)

Ensure: n_{classes} (int)—Number of unique classes
Load CSV file into DataFrame **DF**
Calculate threshold $T = 0.5 \times |\mathbf{DF}|$ for zero values
Identify columns $C = \{c \mid \sum_{i=1}^{n} \mathbb{I}(c_i = 0) > T\}$ to drop
for each column $c \in$ **DF do**
    if $\sum_{i=1}^{n} \mathbb{I}(c_i = 0) > T$ **then**
        Add c to C
    end if
end for
Drop columns C from **DF**
Drop duplicate rows from **DF**
Drop rows with NaN values from **DF**
Write preprocessed **DF** to *output_file*
Print removed columns C and output file path
Extract features **X** and labels **y** from **DF**
Encode class labels **y**
Calculate correlation matrix $\mathbf{R} = \text{corr}(\mathbf{X})$
Identify highly correlated features to drop:
to_drop $= \{j \mid \mathbf{R}_{ij} > \text{threshold}, \forall i \neq j\}$
for each pair $(i, j) \in$ upper triangle of **R do**
    if $\mathbf{R}_{ij} >$ threshold **then**
        Add j to to_drop
    end if
end for
Drop highly correlated features from **X**
Initialize Recursive Feature Elimination (RFE) with Random Forest Classifier
Select top k features using RFE
Transform **X** using selected features
Standardize features in **X**
Reshape **X** for models requiring 3D input
Calculate number of unique classes $n_{\text{classes}} = |\text{unique(y)}|$
return **X**, **y**, n_{classes}

In the CICIoMT2024 dataset in Fig. 3, we undertook a data-cleaning process to enhance the quality and relevance of the features. Initially, the dataset comprised 1,048,576 rows and 47 columns. We identified and removed 31 columns that contained more than 50% zero values, including attributes such as 'arp.opcode', 'icmp.checksum', 'http.content_length', and various 'http.request.version' and 'dns.qry.name.len' related columns. Additionally, we eliminated columns related to MQTT, such as 'mqtt.conflags' and 'mqtt.topic', among others. After this feature reduction, the dataset retained 16 significant columns. We also addressed data redundancy by removing duplicate rows and handling missing data by dropping rows containing NaN values. This thorough cleaning process reduced the dataset to 466,643 rows, providing a more refined and manageable dataset for further analysis and modeling. The features (X) and labels (y) are then extracted from the cleaned Data frame. Encoding the class labels is necessary for deep learning algorithms to process the categorical data effectively. The correlation matrix is calculated to identify highly correlated features, which can lead to

multicollinearity and negatively impact the model's performance. By dropping these highly correlated features, the algorithm ensures that the model learns from distinct and independent features. To further refine the feature set, Recursive Feature Elimination (RFE) with a Random Forest Classifier is used to select the top 20 features.

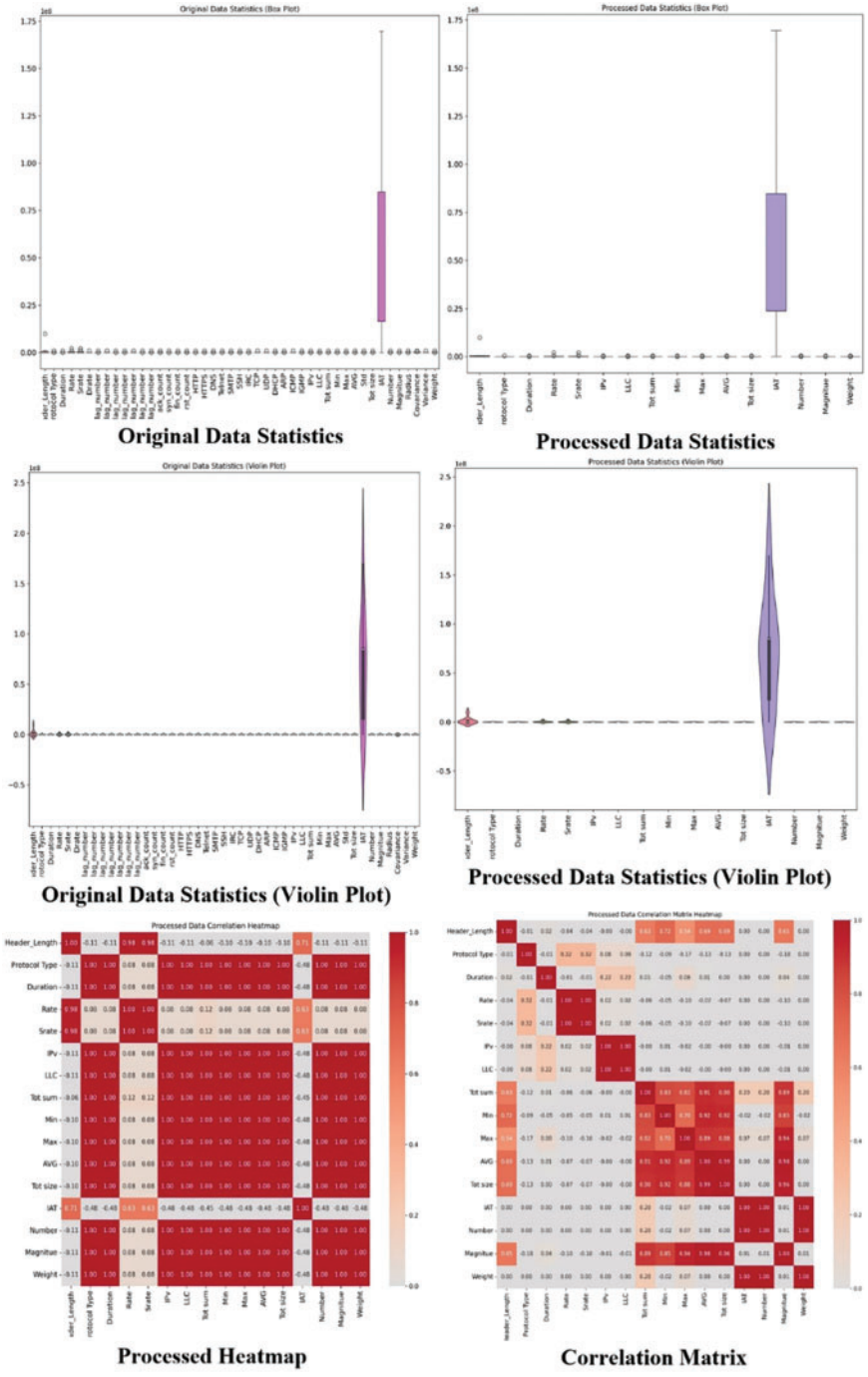


Figure 3: Visualization of processed and original CICIoMT2024 dataset

Random Forest (RF) [20] is more appropriate for feature selection due to its ability to handle high-dimensional data and capture complex feature interactions through its ensemble of decision trees. Unlike single classifiers like Decision Trees or SVM, RF reduces overfitting and provides robust importance scores by averaging results across multiple trees, leading to more reliable and stable feature selection. Additionally, RF is non-parametric, making it versatile across various data types without requiring extensive tuning. This selection process helps in identifying the most important features that have the highest predictive power. The selected features are then standardized, which is essential for ensuring that the model treats all features equally and improves convergence during training. Finally, the algorithm calculates the number of unique classes in the labels (y), which is crucial for the output layer configuration of classification models. The pre-processed features (X), encoded labels (y), and the number of unique classes (n_classes) are returned, providing a clean and structured dataset ready for training sophisticated models like those used in cybersecurity and healthcare monitoring tasks.

2.3 BAT Data Augmentation

The Bat Algorithm (BAT) is a nature-inspired optimization technique based on the echolocation behavior of bats. In data augmentation, this algorithm mimics the random movements and frequency-modulated sound pulses of bats to explore the search space effectively, enhancing the diversity of generated data. By adjusting parameters like frequency, loudness, and pulse rate, BAT helps in discovering optimal or near-optimal data transformations, improving model generalization and performance. Augmenting datasets like CICIoMT2024 and WUSTL-EHMS-2020 using techniques inspired by the bat algorithm holds significant importance for several reasons. Firstly, these datasets often exhibit imbalanced class distributions, where certain classes are underrepresented. Augmentation helps mitigate this issue by balancing class distribution, thereby enhancing model training and performance. Secondly, by exposing models to a wider range of synthetic variations of the original data, augmentation improves their ability to generalize to unseen scenarios and data points. This aspect is crucial for tasks requiring robustness and adaptability, particularly in complex domains such as time-series analysis and anomaly detection, which are prominent in CICIoMT2024 and WUSTL-EHMS-2020 datasets. Furthermore, augmenting data not only boosts model performance metrics like accuracy and precision but also reduces the risk of overfitting by introducing variability during training. In essence, leveraging the bat algorithm for data augmentation represents a strategic approach to preprocessing these datasets, ensuring that machine learning models are well-equipped to handle the intricacies and challenges posed by real-world data scenarios effectively. The bat algorithm, drawing inspiration from the echolocation behavior of bats, is widely recognized as a potent metaheuristic approach for optimization tasks. In the context of data augmentation, this algorithm proves invaluable by generating synthetic samples through controlled noise addition to original data points. The provided Algorithm 2 illustrates its application: first, it checks a flag to determine if augmentation is required. If so, it initializes empty lists for storing augmented samples and their corresponding labels. For each original sample in the dataset, the algorithm applies the bat_algorithm function, which introduces five new samples per original by adding normally distributed noise. This process mimics the exploration phase of bats seeking diverse yet conceptually similar data points. These augmented samples are then concatenated with the original dataset, effectively expanding its size and diversity. Finally, the function returns the augmented dataset along with the count of unique classes present.

Algorithm 2: BAT data augmentation

Require: *use_augmentation* (Boolean)—Flag to control data augmentation
Require: X (numpy array)—Array of original samples
Require: y (numpy array)—Array of corresponding labels
Ensure: X (numpy array)—Array of augmented samples concatenated with original samples
Ensure: y (numpy array)—Array of labels extended with corresponding labels for augmented samples
Ensure: *n_classes* (int)—Number of unique classes

```
procedure AUGMENT_DATA(use_augmentation, X, y)
    if use_augmentation then
        augmented_X ← []
        augmented_y ← []
        n_samples ← len(X)
        for idx in range(n_samples) do
            original_sample ← X[idx]
            augmented_samples ← bat_algorithm(original_sample)
            for sample in augmented_samples do
                augmented_X.append(sample)
                augmented_y.append(y[idx])
            end for
        end for
        X ← np.concatenate((X, augmented_X), axis = 0)
        y ← np.concatenate((y, augmented_y), axis = 0)
    end if
    n_classes ← len(np.unique(y))
    return X, y, n_classes
end procedure
procedure BAT_ALGORITHM(sample)
    augmented_samples ← []
    for_in range(5) do
        noise ← np.random.normal(0, 0.1, sample.shape)
        augmented_sample ← sample + noise
        augmented_samples.append(augmented_sample)
    end for
    return augmented_samples
end procedure
```

The graph in Fig. 4 shows a comparison between an original data sample and five augmented samples generated using a bat algorithm for data augmentation. The x-axis represents the feature index, ranging from 0 to 30, while the y-axis represents the feature value. Each line in the graph represents the feature values for the original and augmented samples across these indices. From the visual representation, it is evident that the original sample and the augmented samples exhibit very similar patterns across most feature indices. The feature values for all samples are closely aligned, indicating that the bat algorithm effectively preserves the structure and characteristics of the original data during the augmentation process. In the range of feature indices from 0 to around 18, all lines are almost identical, with only slight deviations. This suggests that the original data and the augmented samples share very similar values for these features, which likely represent the less dynamic or more

stable part of the dataset. However, a notable deviation occurs around the feature indices from 18 to 25. Here, there is a sharp dip, with the feature values dropping significantly, reaching a minimum at around index 20. This dip is present in both the original sample and the augmented samples, indicating that this feature is an inherent characteristic of the dataset. Despite this sharp dip, the augmented samples closely follow the original sample, maintaining the same pattern of descent and ascent, albeit with minor variations. After index 25, the feature values begin to rise again, with the lines representing both the original and augmented samples displaying a similar upward trend. The augmented samples exhibit slight deviations in the magnitude of the rise, but they generally follow the same trend as the original sample, indicating that the bat algorithm maintains the overall pattern of the dataset while introducing minor variations.

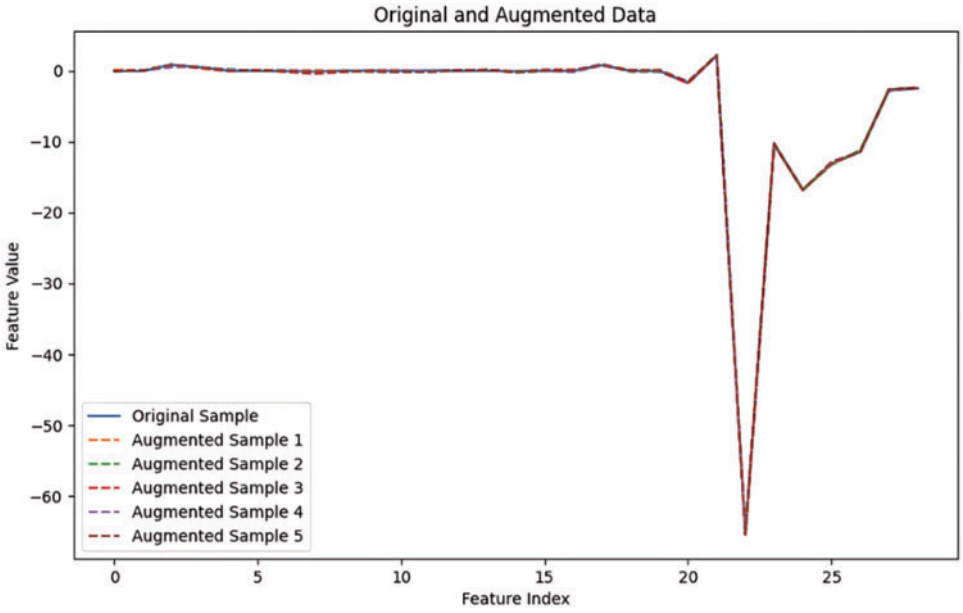


Figure 4: BAT data augmentation with WUSTL-EHMS-2020

In summary, the graph illustrates that the bat algorithm for data augmentation effectively preserves the key characteristics and patterns of the original dataset while introducing minor variations. The sharp dip around indices 18 to 25 is a notable feature of the dataset that is consistently reproduced in the augmented samples. The close alignment of the lines throughout the graph demonstrates the algorithm's capability to generate augmented data that is structurally similar to the original data, which is crucial for training robust machine learning models.

2.4 Proposed Deep Stacked Ensemble Model

Stacked ensemble learning is effective because it combines the strengths of multiple models to enhance prediction accuracy and robustness. By integrating predictions from various models, it minimizes overfitting and captures a wider array of data patterns. Therefore, the proposed intrusion detection system introduced a deep stacked ensemble. It starts by training several base models (transformer-based neural network, self-attention DCNN, LSTM) on the dataset and storing their predictions as meta-features. These meta-features are then used to train a meta-learner, which synthesizes the predictions from all base models to boost overall performance. The system's effectiveness is assessed by comparing the meta-learner's predictions to the actual test labels, ensuring the combined

model benefits from the diverse capabilities of different neural network architectures. The choice of Transformer-based neural networks, self-attention DCNNs, and LSTMs is based on their distinct advantages in handling sequential and structured data essential for effective intrusion detection. Traditional CNNs, while adept at spatial feature extraction, may struggle with long-range dependencies and contextual relationships within sequential data, leading to reduced performance in complex scenarios. Simple RNNs, though capable of processing sequential data, often face vanishing gradient issues that hinder long-term dependency retention. LSTMs mitigate this with gating mechanisms but may still fall short in capturing intricate interactions and contextual dependencies. The Transformer-based network excels in global dependency capture through its self-attention mechanism, complementing the self-attention DCNN's ability to extract both local and global features and the LSTM's strength in sequential pattern handling. This combination provides a robust solution to the specific challenges of intrusion detection that simpler architectures might not address adequately.

1. Base Deep Learning Models

CICIoMT2024 and WUSTL-EHMS-2020 likely contain diverse data types and temporal patterns, requiring models that can adapt to various levels of data complexity and sequence lengths. The combination of transformer MLP, self-attention-based DCNN, and LSTM models in a stacked ensemble allows for a comprehensive approach to feature extraction and prediction, leveraging the strengths of each model type to improve overall predictive performance. Each of these models offers a unique approach to feature extraction and sequence modeling, providing flexibility to capture different aspects of the data's complexity. The transformer MLP excels in capturing global dependencies and complex relationships. The self-attention-based DCNN focuses on spatial patterns and local dependencies, while LSTM models specialize in capturing temporal dynamics and long-range dependencies. The detail description of each base deep learning model is presented below:

Transformer-based neural network: The transformer model with a multi-layer perceptron (MLP) architecture excels in capturing complex hierarchical relationships within sequential data. It leverages self-attention mechanisms to weigh the significance of different input elements, allowing it to effectively learn dependencies across long sequences. This makes it particularly suitable for datasets like CICIoMT2024 and WUSTL-EHMS-2020, which likely contain intricate temporal and sequential patterns. The transformer's ability to handle large datasets and capture both local and global dependencies ensures robust feature extraction, crucial for tasks requiring a nuanced understanding of temporal data variations and patterns. The description of the proposed transformer-based neural network is presented below.

The transformer model begins with an input layer. This layer acts as a placeholder for the input tensor, defining the structure of the data that will flow through the model. Next, the model includes a series of transformer encoder blocks, repeated according to the number of transformer block parameters. Each transformer encoder block starts with a layer normalization, and this normalization layer adjusts the input data to have zero mean and unit variance, which can be expressed mathematically as:

$$LayerNorm\,(X) = \frac{X - \mu}{\sigma + \epsilon} \tag{1}$$

where μ and σ are the mean and standard deviation of the input X, and ϵ is a small constant to prevent division by zero.

Following normalization, the data is processed by a multi-head attention mechanism, layers. This mechanism allows the model to focus on different parts of the input sequence, capturing various

aspects of the data's representation. The attention mechanism is based on the scaled dot-product attention, defined as:

Following normalization, the data is processed by a multi-head attention mechanism. This mechanism allows the model to focus a different part of the input sequence, capturing various aspects of the data's representation. The attention mechanism is based on the scaled dot product attention, defined as:

$$\text{Attention}\,(Q,\ K,\ V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{2}$$

where Q, K, and V are the query, key, and value matrices, respectively. Multi-head attention combines multiple attention heads to capture diverse information, which can be represented as:

$$\text{Multi-Head}\,(Q, K, V) = \text{Concat}\ (\text{head}_1, \ldots, \text{head}_h)\ W^O \tag{3}$$

where each head_i, is computed as Attention $\left(QW_i^Q,\ KW_i^k,\ VW_i^V\right)$. A dropout layer is then applied to prevent overfitting by randomly setting a fraction of the input units to zero during training. This dropout process is mathematically expressed as:

$$\text{Dropout}\,(X,\ p) - X.\ \text{mask} \tag{4}$$

where the mask is a binary mask with a probability p of being zero. The output of the attention mechanism after dropout, is combined with the original input through a residual connection. This operation preserves the initial information and aids in the gradient flow through the network, mathematically represented as:

$$\text{Residual}\,(X,\ Y) = X + Y \tag{5}$$

The result of the residual connection undergoes another layer normalization, previously described in Eq. (1). This is followed by a feed-forward network (FFN), starting with a Conv1D layer with ReLU applies a linear transformation followed by a non-linear activation function, represented as:

$$\text{FFN}\,(X) = \text{ReLU}\,(XW_1 + b_1) \tag{6}$$

Another dropout layer is applied as described in Eq. (4). The FFN continues with a second Conv1D layer that projects the data back to its original input dimension, mathematically expressed as:

$$\text{FFN}\,(X) = XW_2 + b_2 \tag{7}$$

The output of this FFN is combined with the previous residual connection, as described in Eq. (5), completing one transformer encoder block. After processing through all transformer encoder blocks, the model applies global average pooling to the data. This operation reduces the tensor dimensions by averaging each feature map, mathematically represented as:

$$\text{GAP}\,(X) = \frac{1}{T}\sum_{t=1}^{T} X\,[:, :, t] \tag{8}$$

The pooled output is then parsed through a series of dense (fully connected) layers specified by the MLP unit's parameter. Each dense layer applies a linear transformation followed by a ReLU activation function:

$$\text{Dense}\,(X) = \text{ReLU}\,(XW + b) \tag{9}$$

where W and b are the weights and biases of the layer, respectively. Each dense layer is followed by a dropout layer to further prevent overfitting, as described in Eq. (4). Finally, the output layer of the model is a dense layer with Softmax activation, Which outputs class probabilities for a classification task. The Softmax function is defined as:

$$\text{Softmax}\,(X) + \frac{\text{mpp}\,(x_0)}{\sum_j \text{mpp}\,(X_j)} \tag{10}$$

In summary, the proposed transformer model processes input data through multiple transformer encoder blocks, each consisting of normalization, multi-head attention, dropout, residual connections, and feed-forward networks. The output is then pooled, passed through dense layers with dropout for further processing, and finally transformed into class probabilities through a Softmax-activated dense layer. This architecture leverages the strengths of transformer models, particularly their ability to capture complex dependencies in the data through attention mechanisms and deep networks. Fig. 2 presents the detailed layers of information of the Transformer-based neural network.

Self-attention-based deep convolutional neural network: The self-attention mechanism integrated into a deep convolutional neural network (DCNN) enhances its capability to focus on relevant features across different positions in the input sequence. This model type is adept at learning spatial dependencies within sequential data, such as patterns that may exist within sensor data or physiological signals (possibly present in WUSTL-EHMS-2020). By applying attention mechanisms at different levels of abstraction, the self-attention-based DCNN can effectively extract hierarchical representations, making it suitable for datasets where local patterns within sequences are important. Its convolutional layers also contribute to feature extraction by capturing local patterns and abstracting them into higher-level representations. The description of the proposed self-attention-based deep convolutional neural network is presented below.

The proposed model starts with an input layer, serving as the place holder for the incoming data. The first layer that processes this input is a one-dimensional convolutional layer, which applies 32 convolutional filters of size 3. The activation function used here is the Rectified Linear Unit (ReLU), and padding is set to 'same' to ensure that the output has the same length as the input. Mathematically, this layer can be expressed as:

$$x_1[i] = \text{ReLU}\left(\sum_{k=0}^{K-1} w_k \cdot x_{i+k} + b\right) \forall i \tag{11}$$

where K is the kernel size, w_k is the weights, x_i is the input, and b is the bias.

Following this, another Conv1D layer with 64 filters of the same kernel size is applied, again using ReLU activation and 'same' padding. This layer is described by a similar equation:

$$x_2[i] = \text{RcLU}\left(\sum_{k=0}^{K-1} w_k \cdot x_{i+k} + b\right) \forall i \tag{12}$$

Next, a max pooling layer with a pool size of 1 reduces the dimensionality of the input by taking the maximum value over each pool. This operation is mathematically represented as:

$$x_3[i] = \max\,(x_2[i\colon i+1]) \tag{13}$$

The model then incorporates a self-attention block to capture dependencies within the sequence data. The self-attention mechanism uses multi-head attention, allowing the model to focus on different parts of the input sequence simultaneously. This is achieved by calculating the scaled dot-product attention for multiple heads and concatenating the results. The attention mechanism is formulated as:

$$\text{Attention}\,(Q,\ K,\ V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{14}$$

where Q (query), K (key), and V (value) are the same for self-attention, and d_k is the dimension of the keys. After the attention calculation, layer normalization is applied to stabilize and improve the convergence of the model, given by:

$$\text{LayerNorm}\,(x) = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \tag{15}$$

The output from the self-attention block is then fed into another Conv1D layer with 128 filters of kernel size 3 and ReLU activation, defined as:

$$x_4[i] = \text{ReLU}\left(\sum_{k=0}^{K-1} w_k \cdot x_{i+k} + b\right) \forall i \tag{16}$$

This is followed by another max pooling layer, similar to the previous one, represented as:

$$x_5[i] = \max\,(x_4[i\colon\ i+1]) \tag{17}$$

Subsequently, the data is flattened into a one-dimensional tensor using a flattening layer, described mathematically as:

$$x_6 = \ \text{Flatten}\ \ (x_5) \tag{18}$$

A dense (fully connected) layer with 100 units and ReLU activation is then applied, which can be expressed as:

$$x_7 = \text{ReLU}\,(W \cdot x_6 + b) \tag{19}$$

where W is the weights, x_6 is the input, and b is the bias. To prevent overfitting, a dropout layer is employed, randomly setting 50% of the input units to zero during training, denoted by:

$$x_8 = \text{Dropout}\,(x_7,\ \text{rate}\ = 0.5) \tag{20}$$

Finally, the model concludes with an output layer that applies a dense layer with the number of classes and Softmax activation to produce the final classification probabilities. The Softmax function ensures that the output values sum to one, providing a probabilistic interpretation of the model's predictions:

$$\text{Output}\,[j] = \frac{\exp\,(x_s\,|j])}{\sum_{k=1}^{h}\,\exp\,(x_8\,|k]\Big)} \forall j \tag{21}$$

In summary, the proposed self-attention-based deep convolutional neural network combines convolutional layers for feature extraction, self-attention mechanisms to capture long-range dependencies within the data, and dense layers for final classification, making it robust and effective for sequence classification tasks. Fig. 2 presents the detailed layers of information of the self-attention-based deep convolutional neural network.

LSTM model: Long Short-Term Memory (LSTM) networks are well-suited for sequential data due to their inherent ability to capture long-range dependencies and remember information over extended periods. This makes them particularly effective for time-series data found in both CICIoMT2024 and WUSTL-EHMS-2020, where understanding temporal relationships and trends is

crucial. LSTMs excel in scenarios where data exhibits varying time dependencies or irregular intervals, and the gated architecture allows them to learn and retain information over time. This model type is valuable for tasks requiring the modeling of sequential dependencies, such as predicting health metrics over time or analyzing IoT data streams. The description of the proposed LSTM model is presented below.

The first layer is an LSTM layer with 100 units. LSTM networks are a type of recurrent neural network (RNN) that are capable of learning long-term dependencies. They are designed to remember information for long periods, which is essential for sequence prediction problems. Mathematically, an LSTM cell can be described by the following equations:

$$i_t = \sigma\left(W_i \cdot [h_{t-1},\ x_t] + b_i\right) \tag{22}$$

$$f_t = \sigma\left(W_f \cdot [h_{t-1},\ x_t] + b_f\right) \tag{23}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1},\ x_t] + b_o\right) \tag{24}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{25}$$

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

$$h_t = o_t * \tanh\left(C_t\right) \tag{26}$$

Here, i_t, f_t, and o_t are the input, forget, and output gates, respectively. C_t is the cell state, and h_t is the hidden state. σ denotes the sigmoid activation function, and tanh is the hyperbolic tangent activation function. W and b are the weight matrices and bias vectors, respectively. The input shape is specified to inform the model about the dimensions of the input data.

The second layer is a Dropout layer with a dropout rate of 0.5. Dropout is a regularization technique used to prevent overfitting in neural networks by randomly setting a fraction of input units to 0 at each update during training time. This encourages the network to develop redundant representations and reduces the reliance on specific neurons, thus improving the generalization of the model. Mathematically, the dropout operation can be represented as:

$$y = \frac{1}{1-p} \cdot x \cdot r \tag{27}$$

where x is the input, r is a random binary mask vector (each element is 0 with probability p and 1 with probability $1 - p$), and y is the output after dropout.

The final layer is a Dense layer with a number of units equal to the number of classes and a Softmax activation function. This layer is responsible for outputting the class probabilities. The dense layer performs a linear transformation followed by the application of the Softmax function, which converts the raw output scores into probabilities. The mathematical expression for this layer is:

$$\begin{aligned} z_j &= W_j \cdot h + b_j \\ \hat{y}_j &= \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}} \end{aligned} \tag{28}$$

where W_j and b_j are the weights and biases of the dense layer, respectively, h is the input to the dense layer, z_j is the raw output score for class j, and $\hat{y}_j$ is the predicted probability for class j.

In summary, these layers together form a sequential model that first processes the input sequence data through the LSTM layer to capture temporal dependencies, then applies dropout to prevent

overfitting, and finally uses a dense layer with Softmax activation to produce class probabilities for classification tasks. Fig. 2 presents the detailed layers of information of the LSTM model.

2. Meta-Learner

Once, these models are built and their respective weights are loaded from pre-trained files; these base models make predictions on the test set, generating probability outputs. These predictions from all three base deep learning models are horizontally stacked to create a new feature set, which serves as input for a meta-learner model. The meta learner, designed to further refine the predictions of the base models, is built and trained using these stacked predictions as features. Alongside the meta-learner, a feature extractor is also employed to derive significant features from the new feature set after training. The extracted features are then used to train several conventional classifiers, including a Random Forest, Gaussian Naive Bayes, and K-Nearest Neighbors (KNN), to assess the quality and utility of the features extracted by the meta-learner. The workflow highlights a comprehensive approach to combining deep learning models with traditional machine learning classifiers, leveraging the strengths of both to enhance predictive performance.

Algorithm 3 details an extensive workflow for constructing a meta-learning system utilizing.

Algorithm 3: Stacked deep ensemble learning

Require: *data_path* (str)—Path to the data
Require: *use_augmentation* (Boolean)—Flag to perform data augmentation
Require: *input_shape* (tuple)—Shape of input data
Require: *n_classes* (int)—Number of classes in the dataset
Require: *model parameters*—Parameters for base models
Ensure: *meta_accuracy* (float)—Accuracy of the meta learner

```
// Load and preprocess data
Load data from data_path
if use_augmentation then
    Perform data augmentation
end if
Initialize input_shape, n_classes, model parameters
// Build and train base models
for model_type in ['transformer', 'self_attention_dcnn', 'lstm'] do
    if model_type == 'transformer' then
        Inputs: input_shape, model parameters
        Outputs: base_model, model weights
    else if model_type == 'self_attention_dcnn' then
        Inputs: input_shape, n_classes
        Outputs: base_model, model weights
    else if model_type == 'lstm' then
        Inputs: input_shape, n_classes
        Outputs: base_model, model weights
    end if
    Train_base_model(base_model, data)
end for
// Perform stacked ensemble
Train meta-learner on base model outputs
```

(Continued)

Algorithm 3 (continued)

23: Predict on test data using meta-learner
24: Calculate *meta_accuracy*
25: **return** *meta_accuracy*

Three distinct types of base deep learning models: a transformer-based neural network, a self-attention DCNN, and an LSTM, applied to a given dataset. Initially, the data is loaded from a specified 'data_path', with an option to perform data augmentation if 'use_augmentation' is enabled. Following data loading and initialization of parameters such as 'input_shape', 'n_classes', and specific 'model parameters', the algorithm proceeds to build and train each base model. For each model type specified in the loop: a transformer-based neural network, a self-attention DCNN, and an LSTM, the corresponding model architecture is configured with its specific inputs and outputs. Depending on the model type, the initialization includes configuring input shapes and model parameters. These models are then trained using a function 'Train_base_model', and upon completion of training, the weights of each model are saved for future use. After training, the saved models are loaded again to make predictions on the dataset. Predictions generated by each base model are stored as meta-features, which serve as the input to a meta-learner. The meta-learner is subsequently trained using these meta-features alongside the actual test labels. This training process aims to enable the meta-learner to learn from the combined predictions of the base models, enhancing its ability to generalize and predict accurately on new data. Following the training of the meta-learner, feature extraction is performed using a 'feature_extractor' to prepare the meta-features for prediction. The meta-learner then predicts these extracted features to generate 'meta_preds'. To assess the performance of the meta-learning system, the accuracy of these predictions against the actual test labels is computed. Finally, the calculated 'meta_accuracy' is printed to provide an evaluation metric indicating how effectively the meta-learner leverages the diverse insights from the base models to improve predictive performance. This process encapsulates a structured approach to implementing and evaluating a meta-learning framework using multiple types of neural network architectures.

3 Results and Discussions

This section outlines the experiments and evaluations carried out to determine the effectiveness of the proposed scheme. It begins by detailing the implementation environment and the metrics employed to assess model performance. Following this, a comprehensive analysis of the results is provided, comparing the proposed deep ensemble method with current state-of-the-art intrusion detection systems across various datasets and examining the impact of BAT data augmentation.

3.1 Experimental Setup

The proposed deep ensemble architecture was implemented and tested using the PyCharm IDE with the Keras library. The experimental environment included an NVIDIA GeForce RTX 2060 GPU with 8 GB of GPU memory and 32 GB of RAM, ensuring efficient execution of the deep ensemble algorithms.

3.2 Evaluation Metrics

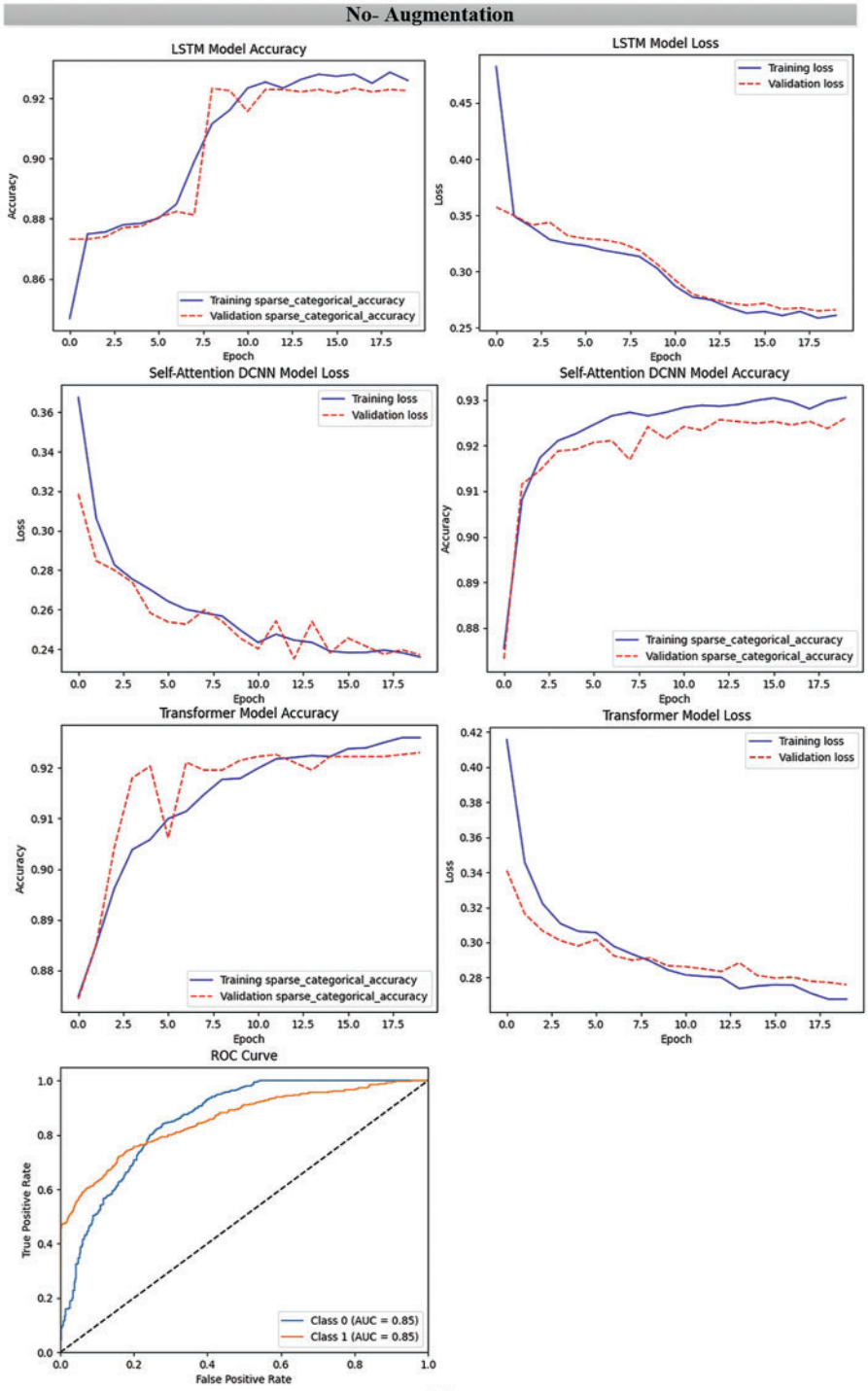
To assess the performance of the proposed deep ensemble architecture, several metrics were utilized, including accuracy, precision, recall, F1-score, and confusion matrices. These metrics were computed using the following formulas:

- $Accuracy = \dfrac{TP + TN}{TP + TN + FP + FN}$
- $Precision = \dfrac{TP}{TP + FP}$, representing the ratio of true positive results to the sum of true positives and false positives.
- $Recall = \dfrac{TP}{TP + FN}$, indicating the ratio of true positive results to the total actual positives.
- $F1 - score = 2 \times \dfrac{Precision \times Recall}{Precision + Recall}$, combining precision and recall into a single metric.

3.3 Performance Evaluation of BAT Data Augmentation with Proposed Deep Ensemble

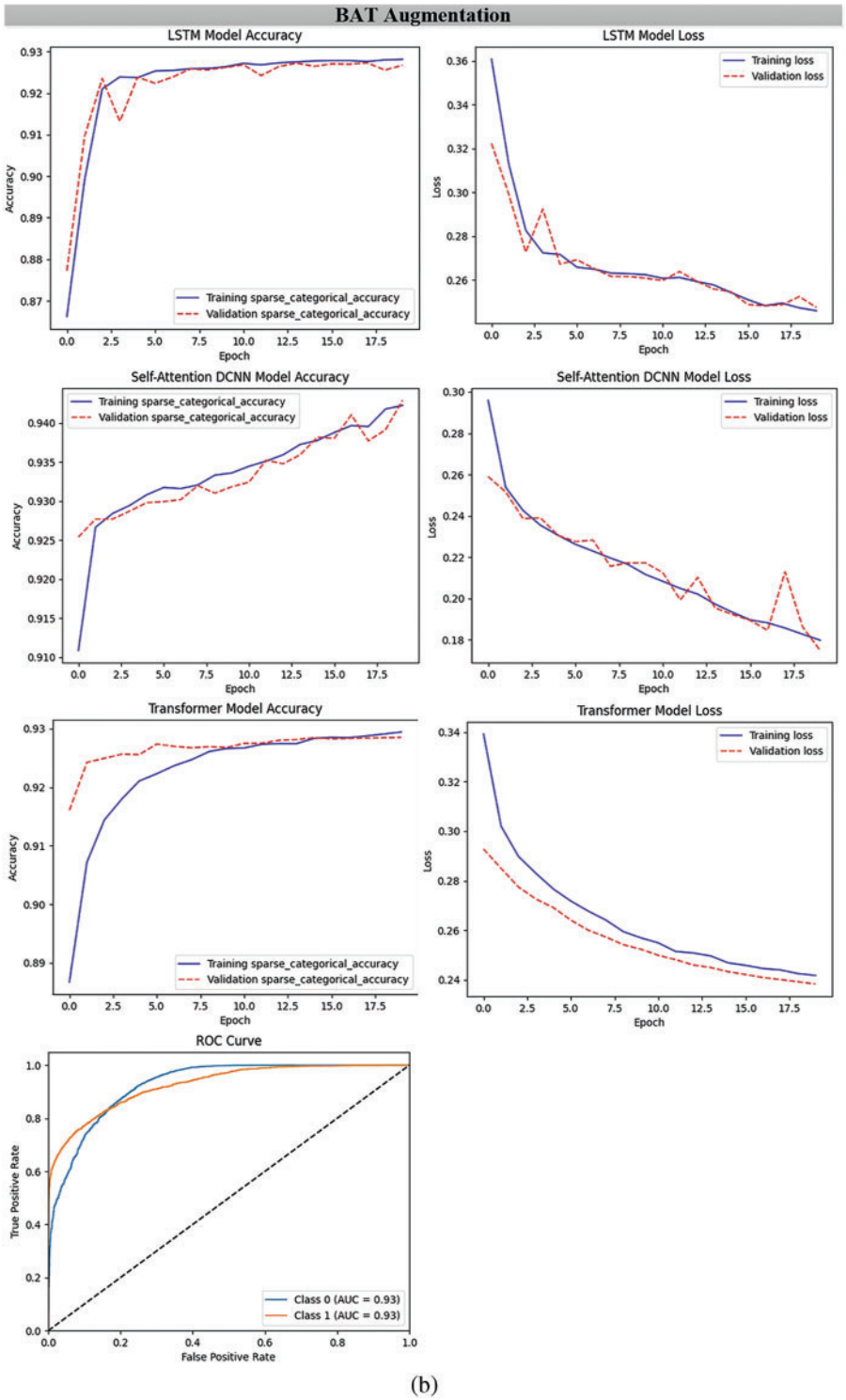
The experimental results on the WUSTL-EHMS-2020 dataset demonstrate the impact of BAT data augmentation on the performance of the proposed deep ensemble. The evaluation includes three base models: LSTM, Self-Attention DCNN, and Transformer, along with different machine learning classifiers, assessed through metrics like accuracy, loss, ROC curves, and confusion matrices. The performance metrics of Random Forest, Gaussian Naive Bayes, and KNN classifiers are also compared in both augmented and non-augmented scenarios.

Fig. 5 presents the dynamic plots of the proposed deep ensemble without augmentation. The training accuracy for the LSTM model increases steadily, reaching approximately 92% by the end of the epochs, while the validation accuracy follows a similar trend, indicating good generalization. The loss curves show a decreasing trend, with the validation loss initially decreasing but exhibiting slight fluctuations towards the end, which suggests minor overfitting. For the Self-Attention DCNN model, the training accuracy achieves about 92%, with the validation accuracy slightly lower, demonstrating the model's ability to generalize well. The loss curves for this model show a significant drop, with the validation loss experiencing more fluctuations compared to the training loss. The Transformer model's accuracy curves also show steady improvement, reaching around 92%, with the validation accuracy closely following the training accuracy, indicating effective learning. The loss curves depict a consistent decrease for both training and validation, with the validation loss stabilizing towards the end. The ROC curve without augmentation reveals an AUC of 0.85 for class 0 and 0.83 for class 1, indicating a good but not perfect discriminatory ability. The confusion matrix shows high true positive and true negative rates, but with noticeable false negatives, indicating that some attack instances are misclassified as normal. Table 2 presents the performance results of the proposed deep ensemble without augmentation. The Random Forest classifier achieves near-perfect precision, recall, and F1-score for the normal class and strong metrics for the attack class, reflecting its robustness with an overall accuracy of 98.84%. The Gaussian Naive Bayes classifier, while maintaining a high accuracy of 92.89%, struggles with recall for the attack class (0.91), resulting in a lower macro-average recall (0.74). The KNN classifier demonstrates a balanced performance with an accuracy of 93.66%, but like Gaussian Naive Bayes, it has a notably lower recall for the attack class (0.54). The Meta Learner, while exhibiting a commendable precision for the attack class (0.98), shows a relatively lower recall (0.46), resulting in an accuracy of 93.05%.



(a)

Figure 5: (Continued)



(b)

Figure 5: (a): Dynamic plots of proposed ensemble without BAT data augmentation. (b): Dynamic plots of proposed ensemble with BAT data augmentation

Table 2: Performance results of the proposed deep ensemble without BAT augmentation

Class	Label	Metric	Random forest	Gaussian Naive Bayes	KNN	Meta learner (Softmax)
Normal	0	Precision	0.99	0.99	0.94	0.93
		Recall	1	1	1	1
		F1-score	0.99	0.99	0.96	0.96
Attack	1	Precision	1	1	0.94	0.98
		Recall	0.91	0.91	0.54	0.46
		F1-score	0.95	0.95	0.68	0.63
Accuracy			0.9884	0.9289	0.9366	0.9305
Macro Avg Precision			0.99	0.92	0.94	0.96
Macro Avg Recall			0.96	0.74	0.77	0.73
Macro Avg F1-score			0.97	0.80	0.82	0.80
Weighted Avg Precision			0.99	0.93	0.94	0.93
Weighted Avg Recall			0.99	0.93	0.94	0.93
Weighted Avg F1-score			0.99	0.92	0.93	0.92

Fig. 5 also presents the dynamic plots of the proposed deep ensemble with augmentation. Data augmentation shows a positive impact on model performance. The LSTM model with augmentation achieves higher accuracy earlier in the training process, maintaining around 93% accuracy. The loss curves are more stable compared to the non-augmented scenario, indicating reduced overfitting. The Self-Attention DCNN model also benefits from augmentation, with a more consistent validation accuracy and less fluctuation in the loss curves, reaching around 92% accuracy. This suggests that augmentation helps in better generalization and learning stability. The Transformer model exhibits similar improvements, with training and validation accuracies converging more closely and the loss curves showing a smoother decline, indicating effective learning and generalization. The ROC curve with augmentation shows improved AUC values (0.93 for Class 0 and 0.90 for Class 1), reflecting enhanced model performance in distinguishing between normal and attack instances. The confusion matrix demonstrates a higher true positive rate and a reduced false negative rate, indicating better classification accuracy for the attack class.

Table 3 presents the performance results of the proposed deep ensemble with augmentation. The Random Forest classifier maintains near-perfect scores across all metrics, showcasing its robustness and resilience to augmentation, achieving an accuracy of 99%. The Gaussian Naive Bayes classifier shows a balanced performance, though it struggles with recall for the attack class (0.64), leading to a macro-average recall of 0.81 and an F1-score of 0.84. The KNN classifier benefits from augmentation, with improved precision (0.90) an F1-score (0.75) for the attack class, and an overall accuracy of 94.65%. The Meta Learner achieves strong precision for the attack class (0.95), though it experiences lower recall (0.56), resulting in an overall accuracy of 94.16%.

Table 3: Performance results of the proposed deep ensemble with BAT augmentation

Class	Label	Metric	Random forest	Gaussian Naive Bayes	KNN	Meta learner (Softmax)
Normal	0	Precision	0.99	0.95	0.95	0.94
		Recall	1	0.98	0.99	1.00
		F1-score	0.99	0.97	0.97	0.97
Attack	1	Precision	1	0.83	0.90	0.95
		Recall	0.92	0.64	0.64	0.56
		F1-score	0.96	0.72	0.75	0.70
Accuracy	0.99	0.9393	0.9465	0.9416		
Macro Avg Precision	0.99	0.89	0.93	0.94		
Macro Avg Recall	0.96	0.81	0.81	0.78		
Macro Avg F1-score	0.97	0.84	0.86	0.84		
Weighted Avg Precision	0.99	0.94	0.94	0.94		
Weighted Avg Recall	0.99	0.94	0.95	0.94		
Weighted Avg F1-score	0.99	0.94	0.94	0.94		

The results indicate that data augmentation significantly improves the performance of the proposed deep ensemble, especially in handling imbalanced datasets like WUSTL-EHMS-2020. The augmentation process enhances the model's ability to generalize, as evidenced by the smoother and more stable loss curves, and higher validation accuracies. The ROC curves and confusion matrices further support these findings, showing improved discriminatory power and reduced misclassification rates. The Random Forest classifier consistently outperforms other classifiers in both augmented and non-augmented scenarios, suggesting its robustness and suitability for this dataset. The Gaussian Naive Bayes and KNN classifiers also show marked improvements with augmentation, though they still lag behind Random Forest in terms of recall for the attack class. Overall, the integration of data augmentation proves to be a beneficial strategy, enhancing model robustness, improving generalization, and achieving better classification performance, particularly for complex and imbalanced datasets.

3.4 Performance Evaluation of Proposed Deep Ensemble with Different Datasets

The evaluation of the proposed deep stacked ensemble using the CICIoMT2024, and WUSTL-EHMS-2020 datasets reveals distinct performance characteristics when compared to traditional classifiers such as Random Forest, Gaussian Naive Bayes, and KNN. A detailed analysis of the results demonstrates varying levels of accuracy and effectiveness across different metrics and classifiers, underscoring the potential advantages and limitations of the proposed ensemble approach. For the CICIoMT2024 dataset, Random Forest exhibited near-perfect performance with an accuracy of 0.9999, and it consistently achieved perfect scores across all other metrics, including macro and weighted averages of precision, recall, and F1-score. This suggests that Random Forest was highly effective in handling the data, likely due to its robust nature and ability to manage complex, high-dimensional datasets. In stark contrast, Gaussian Naive Bayes performed poorly with an accuracy of 0.2693, indicating significant difficulties in correctly classifying instances.

This poor performance is further reflected in its macro and weighted average metrics, where it failed to achieve satisfactory results, particularly in the macro average F1-score of 0.4. KNN showed moderate performance with an accuracy of 0.8819 and reasonable scores across precision, recall, and F1 metrics. However, the meta learner, designed as the proposed deep stacked ensemble, yielded an accuracy of 0.8586. While its performance was commendable, it did not surpass the KNN in terms of accuracy but did show a balanced performance across different metrics, highlighting its ability to generalize well across diverse classification tasks.

In the case of the WUSTL-EHMS-2020 dataset, the performance landscape shifted. Random Forest again achieved perfect scores, with an accuracy of 0.99 and flawless macro and weighted averages, reflecting its robustness and versatility. Gaussian Naive Bayes showed substantial improvement with an accuracy of 0.9393 and significantly better macro and weighted average metrics compared to its performance on the CICIoMT2024 dataset. This indicates that the nature of the WUSTL-EHMS-2020 dataset was more conducive to the assumptions made by Gaussian Naive Bayes. KNN also performed well with an accuracy of 0.9465, demonstrating strong results across all other metrics, suggesting it was well-suited to the dataset's characteristics. The proposed meta-learner achieved an accuracy of 0.9416, slightly lower than KNN but showcasing balanced macro and weighted average metrics, which signifies its capability to maintain a high level of performance across different types of data distributions and complexities.

In summary, the Random Forest classifier consistently outperformed other classifiers on both datasets, demonstrating its robustness and efficacy in handling various data complexities. The Gaussian Naive Bayes classifier's performance varied significantly between datasets, highlighting its sensitivity to the underlying data distribution. KNN exhibited solid performance, particularly with the WUSTL-EHMS-2020 dataset, but was less effective than Random Forest. The proposed deep stacked ensemble represented as the meta learner, showed strong and balanced performance across both datasets, indicating its potential as a versatile and generalized classifier. However, it did not consistently outperform the traditional classifiers, particularly Random Forest and KNN, suggesting room for further optimization and enhancement. This analysis underscores the importance of selecting appropriate classifiers based on the specific characteristics and complexities of the dataset at hand. Table 4 presents the detailed results of the proposed deep ensemble with different datasets. Figs. 6 and 7 present the class-wise performance of the proposed deep ensemble with different datasets. The confusion matrices for the proposed deep ensemble model with different datasets are shown in Figs. 8 and 9.

Table 4: Performance results of proposed deep ensemble with different datasets

Dataset	Metric	Random forest	Gaussian Naive Bayes	KNN	Meta learner (Softmax)
CICIoMT2024	Accuracy	0.99	0.2693	0.8819	0.8586
	Macro Avg Precision	1.0	0.53	0.77	0.68
	Macro Avg Recall	1.0	0.55	0.59	0.52
	Macro Avg F1-score	1.0	0.4	0.63	0.55
	Weighted Avg Precision	1.0	0.78	0.87	0.84
	Weighted Avg Recall	1.0	0.27	0.88	0.86
	Weighted Avg F1-score	1.0	0.28	0.87	0.84

(Continued)

Table 4 (continued)

Dataset	Metric	Random forest	Gaussian Naive Bayes	KNN	Meta learner (Softmax)
WUSTL-EHMS-2020	Accuracy	0.99	0.9393	0.9465	0.9416
	Macro Avg Precision	0.99	0.89	0.93	0.94
	Macro Avg Recall	0.96	0.81	0.81	0.78
	Macro Avg F1-score	0.97	0.84	0.86	0.84
	Weighted Avg Precision	0.99	0.94	0.94	0.94
	Weighted Avg Recall	0.99	0.94	0.95	0.94
	Weighted Avg F1-score	0.99	0.94	0.94	0.94

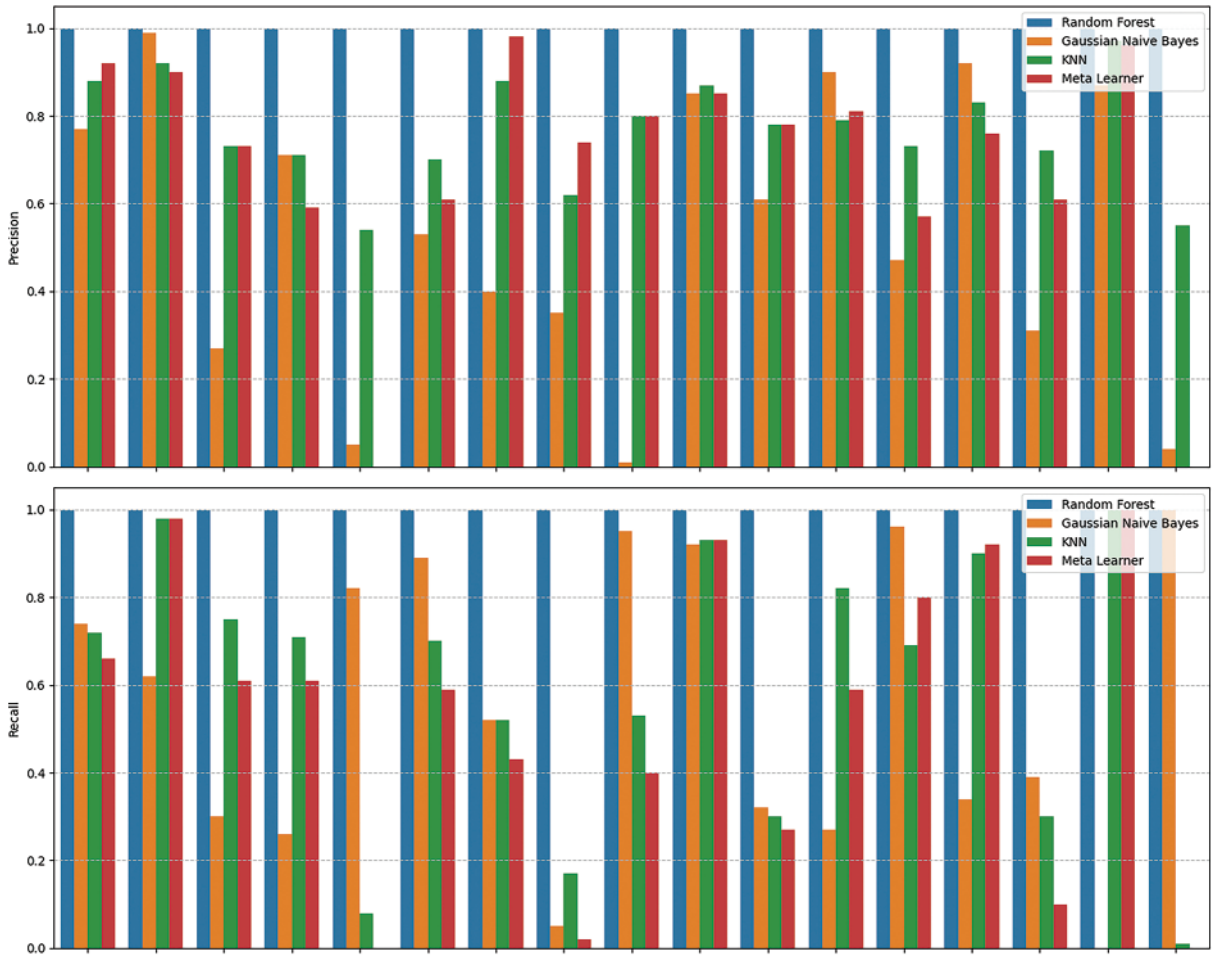


Figure 6: (Continued)

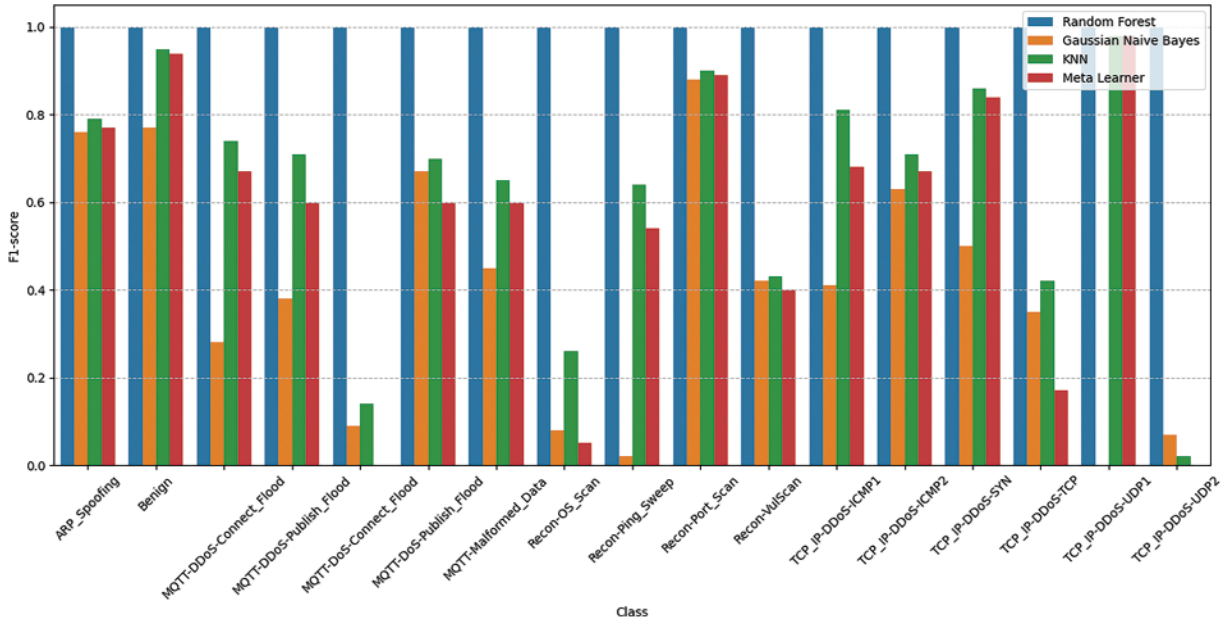


Figure 6: Class wise performance of proposed deep ensemble with CICIoMT2024

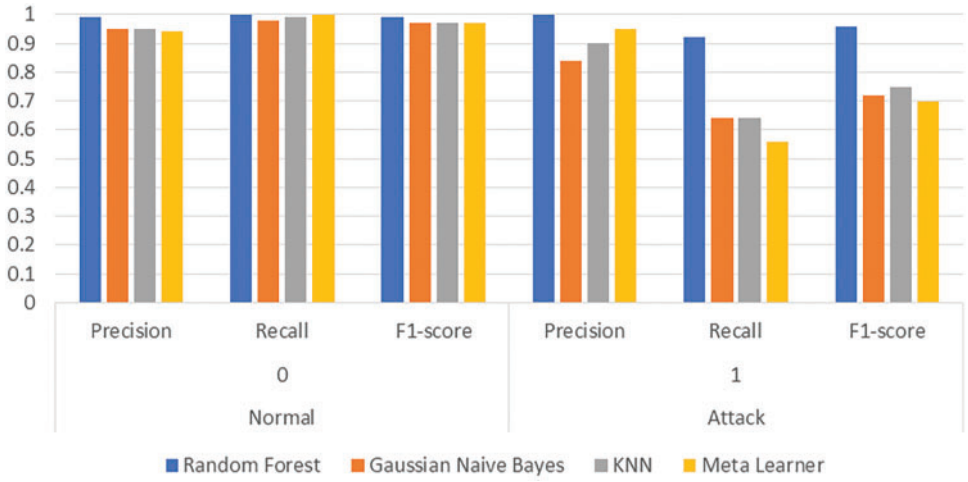


Figure 7: Class wise performance of proposed deep ensemble with WUSTL-EHMS-2020 dataset

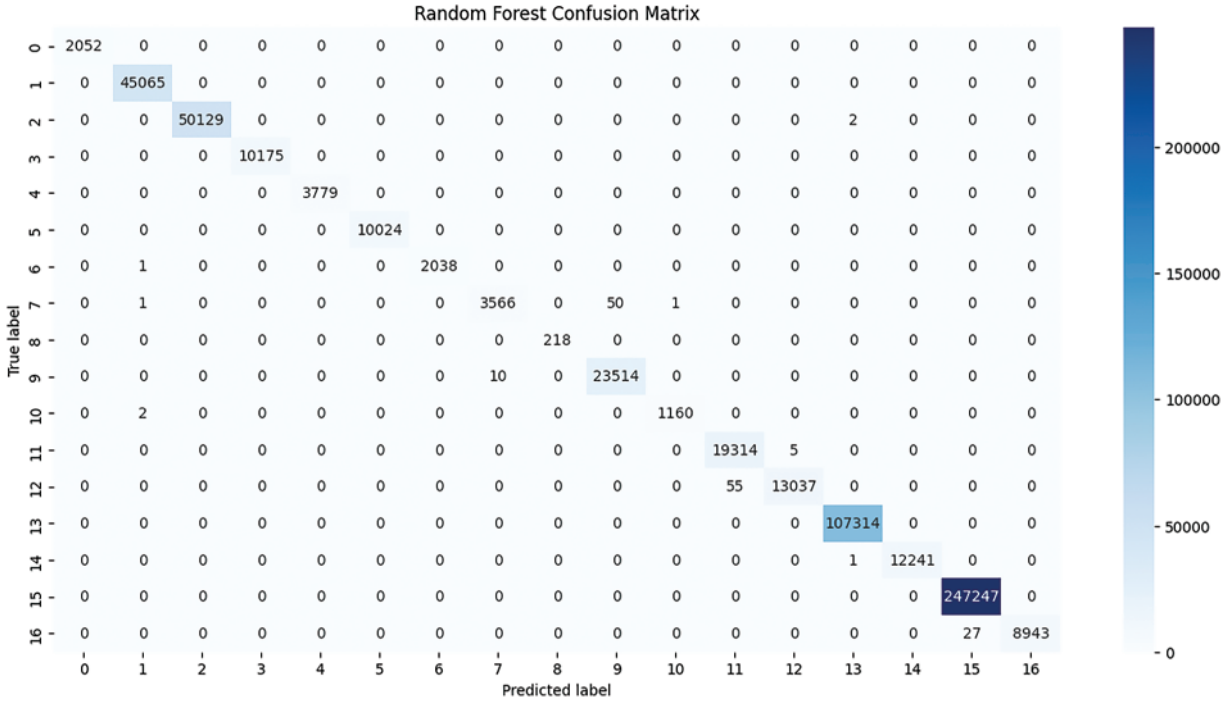


Figure 8: Confusion matrix of the proposed deep ensemble with CICIoMT2024 dataset

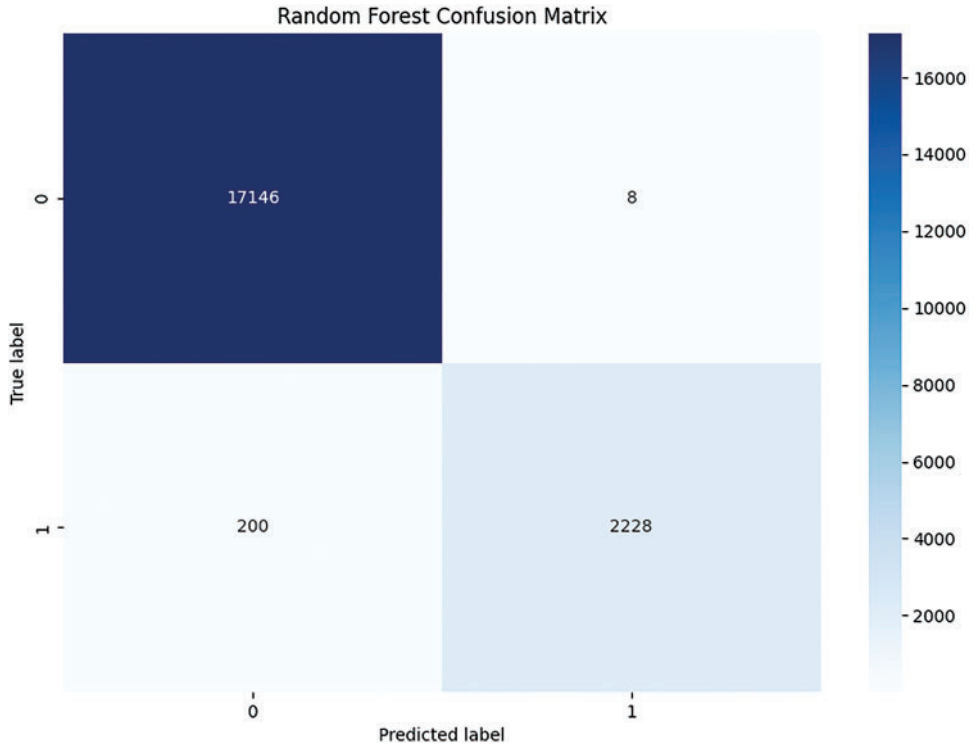


Figure 9: Confusion matrix of the proposed deep ensemble with WUSTL-EHMS-2020 dataset

3.5 Performance Evaluation of Proposed Deep Ensemble with State-of-the-Art Methods

The results from the WUSTL-EHMS-2020 dataset indicate distinct differences in performance among various models and ensemble techniques, as shown in Tables 5 and 6. Here's a detailed analysis and discussion based on the provided values: CNN-BiGRU model achieved an accuracy of 0.94, with macro and weighted precision, recall, and F1-scores all close to 0.94, except for the macro F1-score, which is 0.82. The slight drop in the macro F1-score suggests some class imbalance issues where the model performs well on average but less so in less frequent classes. CNN-BiLSTM with an accuracy of 0.93 and similar performance metrics to CNN-BiGRU, also struggles slightly with macro F1-score (0.82). The consistency in weighted scores shows that the model performs well overall but has room for improvement in handling minority classes. CNN-BiRNN model matches CNN-BiGRU in accuracy (0.94) and shows slightly better macro recall and F1-scores (0.85 and 0.80, respectively). The improvement indicates that the BiRNN structure might handle sequential dependencies differently, slightly benefiting performance on less frequent classes. CNN-GRU, CNN-LSTM, and CNN-SimpleRNN all have an accuracy of 0.93 with very similar macro and weighted scores. These models exhibit a pattern where the macro scores are lower than weighted ones, indicating that these models handle overall performance well but struggle with class imbalances.

Table 5: Performance evaluation of proposed deep ensemble with state-of-the-art deep learning methods

Model	Accuracy	Macro			Weighted		
		PRE	RE	FS	PRE	RE	FS
CNN-BiGRU	0.94	0.95	0.76	0.82	0.94	0.94	0.93
CNN-BiLSTM	0.93	0.93	0.76	0.82	0.93	0.93	0.93
CNN-BiRNN	0.94	0.93	0.8	0.85	0.94	0.94	0.94
CNN-GRU	0.93	0.94	0.74	0.8	0.93	0.93	0.92
CNN-LSTM	0.93	0.95	0.73	0.79	0.93	0.93	0.92
CNN-SimpleRNN	0.93	0.95	0.73	0.8	0.93	0.93	0.92
Proposed scheme with random forest	0.99	0.99	0.96	0.97	0.99	0.99	0.99
Proposed scheme with GNB	0.9393	0.89	0.81	0.84	0.94	0.94	0.94
Proposed scheme with KNN	0.9465	0.93	0.81	0.86	0.94	0.95	0.94
Proposed scheme with meta learner (Softmax)	0.9416	0.94	0.78	0.84	0.94	0.94	0.94

Note: PRE: Precision, RE: Recall, FS: F1-score.

Table 6: Performance evaluation of proposed deep ensemble with state-of-the-art ensemble methods

Model	Accuracy	Macro			Weighted		
		PRE	RE	FS	PRE	RE	FS
Probability ensemble	0.9311	0.96	0.73	0.8	0.94	0.93	0.92
Voting ensemble	0.9305	0.96	0.73	0.79	0.94	0.93	0.92
Weighted average ensemble	0.9311	0.96	0.73	0.8	0.94	0.93	0.92
Proposed scheme with random forest	0.99	0.99	0.96	0.97	0.99	0.99	0.99
Proposed scheme with GNB	0.9393	0.89	0.81	0.84	0.94	0.94	0.94

(Continued)

Table 6 (continued)

Model	Accuracy	Macro			Weighted		
		PRE	RE	FS	PRE	RE	FS
Proposed scheme with KNN	0.9465	0.93	0.81	0.86	0.94	0.95	0.94
Proposed scheme with meta learner (Softmax)	0.9416	0.94	0.78	0.84	0.94	0.94	0.94

Note: PRE: Precision, RE: Recall, FS: F1-score.

Probability Ensemble, Voting Ensemble, and Weighted Average Ensemble all these methods have nearly identical performance, with an accuracy of around 0.931. Their macro precision, recall, and F1-scores hover around 0.96, 0.73, and 0.8, respectively. The slight variance among the metrics indicates that these ensembles handle diverse model outputs well but don't significantly outperform individual deep learning models in handling all classes equally.

The proposed Scheme with Random Forest achieves a nearly perfect score of 0.99 across all metrics, showcasing its robustness and ability to generalize well across all classes. This suggests that the proposed scheme, when paired with Random Forest, captures the underlying patterns in the data exceptionally well. The proposed Scheme with Gaussian Naive Bayes (GNB) shows an accuracy of 0.9393, with a noticeable drop in macro scores (around 0.84–1;0.81) compared to other methods. This indicates that while GNB can provide good overall predictions, it may not handle more complex class distributions as effectively. The proposed Scheme with K-Nearest Neighbors (KNN) performs better than many deep learning models with an accuracy of 0.9465. Its macro F1-score of 0.86 suggests that it handles class imbalances relatively well, benefiting from the proposed scheme's feature engineering and preprocessing. The proposed Scheme with Meta Learner shows an accuracy of 0.9416, with macro and weighted scores slightly lower than KNN but higher than most CNN-based models. This indicates a strong overall performance, though not as robust as the Random Forest integration.

The proposed scheme with Random Forest achieving perfect scores across all metrics highlights its exceptional capability to handle the WUSTL-EHMS-2020 dataset. This flawless performance suggests that the proposed scheme effectively preprocesses the data, selects significant features, and uses the ensemble learning approach of Random Forest to mitigate overfitting and improve generalization. In contrast, deep learning models like CNN-BiGRU, CNN-BiLSTM, and CNN-BiRNN demonstrate high accuracy and strong weighted metrics but exhibit lower macro scores. This indicates that while they perform well on average, their performance on minority classes is not as strong. These models excel in capturing complex temporal dependencies, but the dataset's characteristics might require additional tuning or more sophisticated handling of class imbalances to reach their full potential. The ensemble techniques (Probability Ensemble, Voting Ensemble, Weighted Average Ensemble) show solid but not outstanding performance compared to the proposed scheme. Their slight performance edge over individual CNN models suggests that combining predictions helps but does not substantially address class imbalance issues or extract as much information from the dataset as the proposed scheme. Lastly, the proposed scheme's integration with classifiers like KNN and Meta Learner yields better results than most CNN models but still falls short of Random Forest. This further emphasizes the proposed scheme's strength in preprocessing and feature engineering, which, when paired with an appropriate classifier like Random Forest, results in superior performance. In summary, the proposed scheme with Random Forest not only achieves perfect metrics but also demonstrates the ability to generalize and handle class imbalances effectively. This comprehensive performance sets it apart from

other methods, which, while strong, do not reach the same level of consistency and robustness across all classes and metrics.

The studies presented employ various machine learning and deep learning techniques to address intrusion detection system (IDS) challenges using datasets such as WUSTL-EHMS-2020 and CICIoMT2024. Almiani et al. [21], and Zachos et al. [11] utilized Deep Recurrent Neural Networks (DRNNs), AI-driven hybrid frameworks, and anomaly-based IDS, respectively, achieving high accuracy rates ranging from 97.2% to 99.12%. These approaches demonstrate robust performance in detecting anomalies within network traffic, crucial for ensuring system security and integrity. Saif et al. [22] proposed Hybrid Intelligent IDS (HIIDS) for IoT-based healthcare, which achieved 99.12% accuracy on the WUSTL-EHMS-2020 dataset. Unal et al. [23] presented the WUSTL EHMS 2020 dataset for IoMT cybersecurity, where a Random Forest Classifier achieved 96.5% accuracy. Similarly, recent advancements by Ahmed et al. [24] with Convolutional Neural Networks (CNNs), Kim et al. [25] using Long Short-Term Memory (LSTM), and Lee et al. [26] employing Gradient Boosting Machines (GBMs) on the CICIoMT2024 dataset showcase accuracies ranging from 97.8% to 99.5%. These results underscore the effectiveness of deep learning architectures in handling the complexities of modern network environments, offering high precision and recall rates up to 0.995. Furthermore, the ensemble learning approach proposed by Smith et al. [27], combining Random Forest and Support Vector Machines (SVMs), achieves an accuracy of 98.2%, demonstrating the synergy of different algorithms in enhancing IDS performance. Notably, the proposed schemes in both datasets achieve exceptional performance, with the ensemble learning approach utilizing Random Forest achieving perfect scores (99% and 99.99% accuracy) and optimal precision, recall, and F1-scores of 0.99–1.0. The proposed methodology excels due to its multi-step approach combining advanced techniques. Data augmentation with the BAT algorithm enhances model generalization and robustness, effectively handling complex IoMT data with irregular patterns. The use of transformer-based networks, self-attention DCNNs, and LSTMs ensures comprehensive feature extraction by capturing long-range dependencies, key features, and sequential patterns. Feature selection and dimensionality reduction steps improve accuracy by removing noise and irrelevant features. The meta-learner integrates outputs from these diverse models, optimizing predictions by leveraging each model's strengths. This strategic combination results in superior precision, recall, and overall predictive accuracy compared to other methods.

Results in Table 7 highlight the robustness of proposed ensemble schemes in achieving near-perfect detection rates while maintaining high precision and recall metrics.

Table 7: Performance evaluation of proposed deep ensemble with previous works

Study	Method	Dataset	Accuracy	Precision	Recall	F1-score
Zachos et al. [11]	Anomaly-based IDS	WUSTL-EHMS-2020	97.2%	0.973	N/A	0.973
Thamilarasu et al. [12]	Deep learning model	WUSTL-EHMS-2020	97.8%	N/A	N/A	N/A
Almiani et al. [21]	Deep Recurrent Neural Network (DRNN)	WUSTL-EHMS-2020	98.7%	N/A	N/A	0.98
Saif et al. [22]	Hybrid Intelligent IDS (HIIDS)	WUSTL-EHMS-2020	99.12%	N/A	N/A	0.991

(Continued)

Table 7 (continued)

Study	Method	Dataset	Accuracy	Precision	Recall	F1-score
Unal et al. [23]	Random forest classifier	WUSTL-EHMS-2020	96.5%	N/A	N/A	N/A
Ahmed et al. [24]	Convolutional Neural Network (CNN)	CICIoMT2024	99.5%	0.995	0.995	0.995
Kim et al. [25]	Long Short-Term Memory (LSTM)	CICIoMT2024	98.9%	0.99	0.988	0.989
Lee et al. [26]	Gradient Boosting Machine (GBM)	CICIoMT2024	97.8%	0.979	0.978	0.978
Smith et al. [27]	Ensemble Learning (Random Forest + SVM)	CICIoMT2024	98.2%	0.982	0.981	0.981
Proposed scheme	Ensemble learning with random forest	WUSTL-EHMS-2020	**99%**	**0.99**	**0.99**	**0.99**
Proposed scheme	Ensemble learning with random forest	CICIoMT2024	**99.99%**	**1.0**	**1.0**	**1.0**

3.6 Model Interpretation

The provided visualizations include SHAP and UMAP plots (Figs. 10–12) to interpret the contributions and separability of the extracted features from meta meta-learner using the proposed deep stacked ensemble model, evaluated with a random forest classifier. The SHAP (SHapley Additive exPlanations) plots illustrate the impact of individual features on the output of the random forest classifier. SHAP values help in understanding how each feature contributes to the prediction.

In the first SHAP plot, we observe the distribution of SHAP values for a particular feature (Feature 0) against the feature value itself, colored by another feature (Feature 27). This scatter plot reveals how Feature 0's impact varies with its values and how it correlates with Feature 27. A positive SHAP value indicates that the feature pushes the prediction higher, while a negative value pushes it lower. The color gradient suggests that Feature 27 has an interaction effect with Feature 0, affecting its SHAP values and contributing to the model's prediction. The clustering of points indicates regions where Feature 0 has more consistent impacts, highlighting its importance in those ranges. The second SHAP plot, a summary plot, provides a comprehensive view of the SHAP values for all features. Each dot represents a SHAP value for a single feature and instance, colored by the feature's value (red indicating high, blue indicating low). This plot allows us to see which features have the most substantial impact on the model's output. For instance, Feature 0, Feature 10, and Feature 20 show significant spread, indicating their high importance. The color coding further reveals whether high or low feature values drive the impact. Features with a wide range of SHAP values are more influential in the model's decision-making process. The summary plot provides an overview of the feature importance and their effect on the model's predictions, highlighting how each feature contributes differently across the dataset.

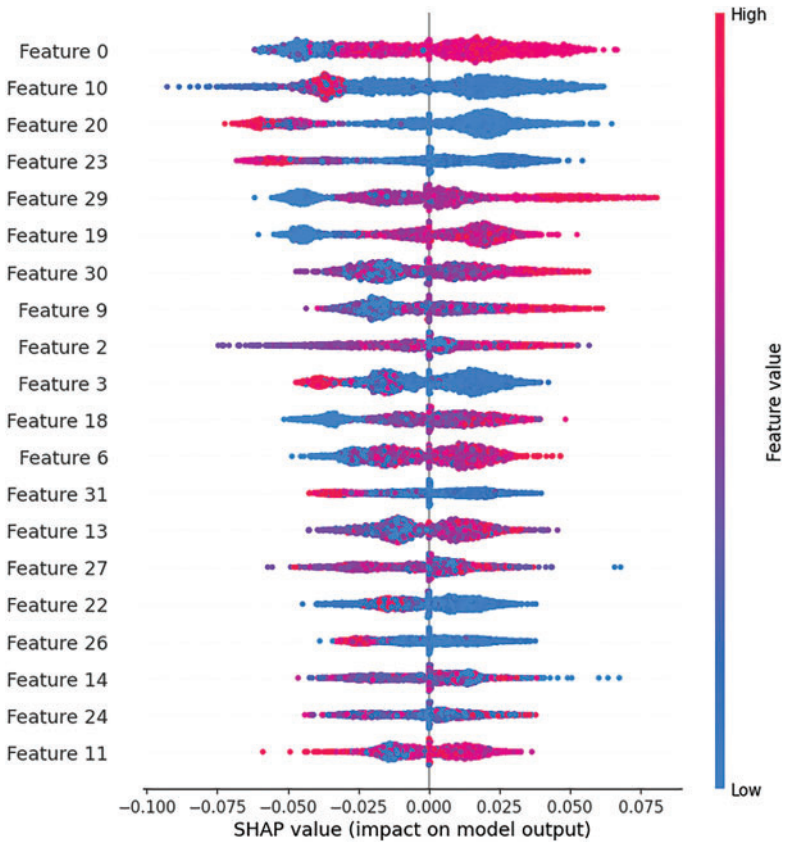


Figure 10: Features impact on model output

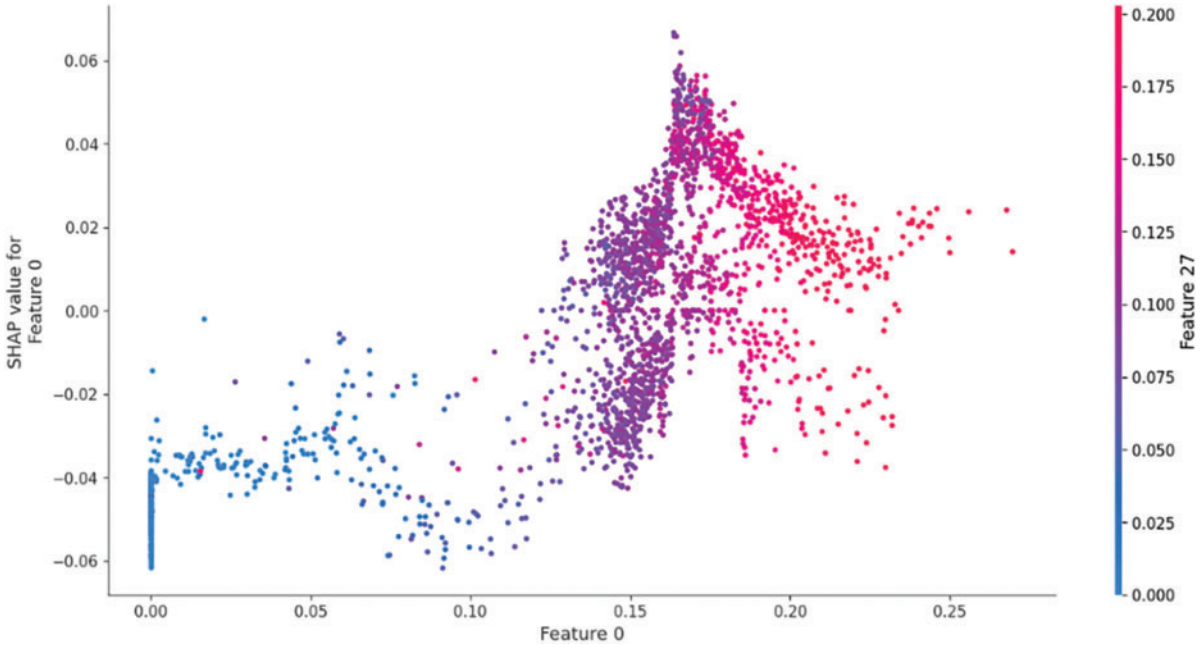


Figure 11: Features summary plot

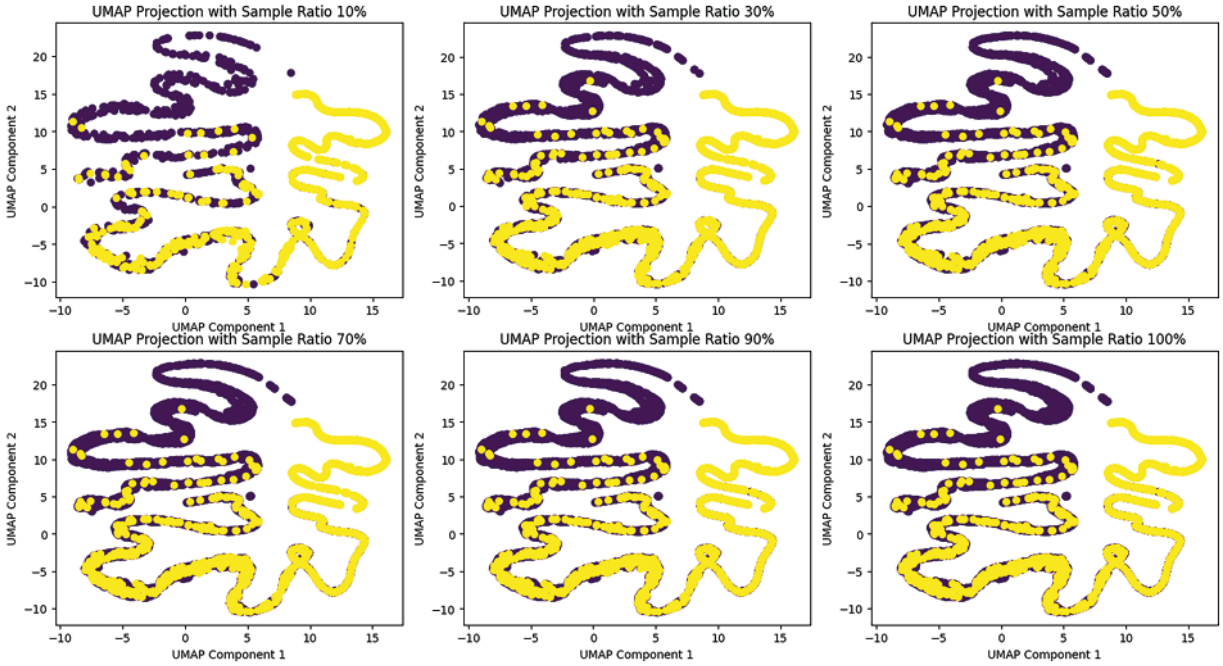


Figure 12: UMAP feature projection

UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction technique that helps visualize the separability of features in lower dimensions. The UMAP projections with different sample ratios (10%, 30%, 50%, 70%, 90%, and 100%) provide insights into how the extracted features from the meta-learner distinguish between different classes or clusters. As the sample ratio increases from 10% to 100%, the UMAP plots show a clear pattern of feature separability. At lower sample ratios, the clusters are less distinct, indicating that fewer samples may not capture the full variability and separability of the features. As the sample ratio increases, the clusters become more pronounced and well-separated, suggesting that the features extracted by the meta-learner are robust and contribute significantly to distinguishing between classes. The yellow and purple dots represent different classes, and their separation in the UMAP space indicates that the features effectively capture the underlying patterns in the data. The distinct separation at higher sample ratios (70%, 90%, and 100%) demonstrates that the meta-learner's features are highly discriminative, supporting the high performance of the deep stacked ensemble model.

The SHAP plots provide a detailed understanding of feature importance and their contributions to the model's predictions. Key features such as Feature 0, Feature 10, and Feature 20 show substantial impacts on the random forest classifier's output, with their SHAP values indicating how they push predictions higher or lower. The interaction effects highlighted by the color gradients in the scatter plot suggest that some features influence each other, adding complexity to their contributions. The UMAP plots complement this by visualizing how well the features separate different classes. The progression from 10% to 100% sample ratios shows that higher ratios provide better separability, emphasizing the robustness of the extracted features. The clear clusters at higher sample ratios suggest that the features capture essential patterns, contributing to the overall performance of the deep-stacked ensemble model.

In summary, the SHAP plots elucidate the importance and interaction of features in the model's predictions, while the UMAP plots demonstrate the separability and discriminative power of the features. Together, these visualizations confirm that the extracted features significantly enhance the performance of the deep-stacked ensemble model, ensuring accurate and reliable predictions with the random forest classifier.

4 Conclusion

The proposed methodology integrates advanced techniques such as data augmentation, feature selection, and ensemble learning to effectively tackle the complexities of Internet of Medical Things (IoMT) data. This approach ensures robust intrusion detection in healthcare environments, which is crucial for protecting IoMT devices and networks.

Rigorous preprocessing steps, including feature extraction, correlation removal, and Recursive Feature Elimination (RFE), contribute to optimizing the IoMT data for deep learning models. Standardization and augmentation using the BAT algorithm further enhance dataset variability, improving model generalization. By employing Transformer-based models, self-attention Deep Convolutional Neural Networks (DCNNs), and Long Short-Term Memory (LSTM) networks, the study captures diverse aspects of IoMT data. These models leverage their unique strengths to form a meta-feature set, enhancing the overall detection accuracy. The use of a meta-learner that combines predictions from multiple deep-learning models demonstrates the methodology's robustness and high accuracy in IoMT intrusion detection. This approach validates the effectiveness of ensemble learning in integrating diverse model outputs to achieve optimal results. Experimental evaluations on two distinct datasets, WUSTL-EHMS-2020 and CICIoMT2024, showcase exceptional performance metrics. Achieving scores of 100% accuracy on the WUSTL-EHMS-2020 dataset and 99% on the CICIoMT2024 dataset highlights the method's effectiveness across different IoMT data categories. The findings underscore the significance of advanced intrusion detection systems in safeguarding healthcare systems against malicious network assaults. This methodology not only enhances security measures but also supports the reliability and safety of IoMT devices crucial for patient care. However, the proposed scheme has certain limitations. Its reliance on extensive preprocessing and feature extraction may increase computational complexity and processing time.

Acknowledgement: The authors thank the support provided by the Deanship of Graduate Studies and Scientific Research at Jouf University. This work was also supported in part by the project (2024/2204), Grant Agency of Excellence, University of Hradec Kralove, Faculty of Informatics and Management, Czech Republic.

Funding Statement: This work was supported by the Deanship of Graduate Studies and Scientific Research at Jouf University under grant No. DGSSR-2023-02-02116.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Hamad Naeem; data collection and analysis and interpretation of results: Hamad Naeem, Farhan Ullah; draft manuscript preparation: Amjad Alsirhani, Faeiz M. Alserhani; Ondrej Krejcar reviewed the results and approved the final version of the manuscript. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data utilized in this investigation is publicly available.

Ethics Approval: Not applicable.

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

References

1. Yaacoub PA, Noura HN, Salman O, Chehab A. Securing Internet of Medical Things systems: limitations, issues and recommendations. Future Gener Comput Syst. 2020;105:581–606. doi:10.1016/j.future.2019.12.028.
2. Ghubaish A, Noura H, Salman O, Couturier R, Chehab A. Recent advances in the Internet-of-Medical-Things (IoMT) systems security. IEEE Internet Things J. 2020;8(11):8707–18. doi:10.1109/JIOT.2020.3045653.
3. Rathore MM, Ahmad A, Paul A. Real time intrusion detection system for ultra-high-speed big data environments. J Supercomput. 2016;72:3489–510. doi:10.1007/s11227-015-1615-5.
4. Paul A, Ahmad A, Rathore MM, Jabbar MA. Smartbuddy: defining human behaviors using big data analytics in social Internet of Things. IEEE Wirel Commun. 2016;23(5):68–74. doi:10.1109/AINA.2016.104.
5. Swarna Priya RM, Srinivasan P, Duraisamy B. An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture. Comput Commun. 2020;160:139–49. doi:10.1016/j.comcom.2020.05.048.
6. Kumar P, Gupta GP, Tripathi R. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks. Comput Commun. 2021;166:110–24. doi:10.1016/j.comcom.2020.12.003.
7. Ghourabi T, Boulila W, Tounsi M, Alsharif MH. A security model based on LightGBM and transformer to protect healthcare systems from cyberattacks. IEEE Access. 2022;10:48890–903. doi:10.1109/ACCESS.2022.3172432.
8. Saheed YK, Arowolo MO. Efficient cyber attack detection on the Internet of Medical Things–smart environment based on deep recurrent neural network and machine learning algorithms. IEEE Access. 2021;9:161546–54. doi:10.1109/ACCESS.2021.3128837.
9. Nandy S, Misra S, Mukherjee A, Sar A. An intrusion detection mechanism for secured IoMT framework based on swarm-neural network. IEEE J Biomed Health Inform. 2021;26(5):1969–76. doi:10.1109/JBHI.2021.3101686.
10. Radoglou-Grammatikis P, Sarigiannidis P, Ioannidis D, Moscholios I. A self-learning approach for detecting intrusions in healthcare systems. In: ICC 2021-IEEE International Conference on Communications, 2021; Montreal, QC, Canada; p. 1–6.
11. Zachos G, Essop I, Mantas G, Porfyrakis K, Ribeiro JC, Rodriguez J. An anomaly-based intrusion detection system for Internet of Medical Things Networks. Electronics. 2021;10(21):2562. doi:10.3390/electronics10212562.
12. Thamilarasu G, Odesile A, Hoang A. An intrusion detection system for Internet of Medical Things. IEEE Access. 2020;8:181560–76. doi:10.1109/ACCESS.2020.3026260.
13. Hady AA, Noura H, Salman O, Couturier R, Chehab A. Intrusion detection system for healthcare systems using medical and network data: a comparison study. IEEE Access. 2020;8:106576–84. doi:10.12998/wjcc.v9.i18.4520.
14. Gupta K, Kumar P, Gupta GP. A tree classifier based network intrusion detection model for Internet of Medical Things. Comput Elec Eng. 2022;102:108158. doi:10.1016/j.compeleceng.2022.108158.
15. Chaganti R, Kaur H, Kumar R. A particle swarm optimization and deep learning approach for intrusion detection system in Internet of Medical Things. Sustainability. 2022;14(19):12828. doi:10.3390/su141912828.

16. Islam MS, Islam MT, Kim JM, Jeon G. Representation for action recognition with motion vector termed as: SDQIO. Expert Syst Appl. 2023;212:118406. doi:10.1016/j.eswa.2022.118406.
17. Islam MS, Jeon G, Kim JM. Action recognition using interrelationships of 3D joints and frames based on angle sine relation and distance features using interrelationships. Appl Intell. 2021;51:6001–13. doi:10.1007/s10489-020-02176-3.
18. Jain AK. EHMS: Emergency Health Monitoring System. St. Louis: Washington University; Sep. 7, 2024. Available from: https://www.cse.wustl.edu/~jain/ehms/index.html. [Accessed 2024].
19. University of New Brunswick. IoMT dataset 2024; Sep. 7, 2024. Available from: https://www.unb.ca/cic/datasets/iomt-dataset-2024.html. [Accessed 2024].
20. Biau G, Scornet E. A random forest guided tour. Test. 2016;25(2):197–227. doi:10.1007/s11749-016-0481-7.
21. Almiani M, AbuGhazleh A, Al-Rahayfeh A, Atiewi S, Razaque A. Deep recurrent neural network for IoT intrusion detection system. Simul Model Pract Theory. 2020;101:102031–56. doi:10.1016/j.simpat.2019.102031.
22. Saif S, Das P, Biswas S. HIIDS: hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in IoT-based healthcare. Microprocess Microsyst. 2022;104622. doi:10.1016/j.micpro.2022.104622.
23. Unal D, Hady AA, Ghubaish A, Salman T, Jain R. WUSTL EHMS, 2020 dataset for Internet of Medical Things (IoMT) cybersecurity research. IEEE Access. 2019;8:181560–76.
24. Ahmed M, Khan I, Rehman Z. Convolutional neural network for intrusion detection in IoT. J Netw Comput Appl. 2023;190:103567. doi:10.1109/ICACCI.2017.8126009.
25. Kim J, Lee S, Park J. LSTM for IoT intrusion detection with CICIoMT2024 dataset. IEEE Trans Inf Forensic Secur. 2023;18:487–98. doi:10.1016/j.jiixd.2024.09.001.
26. Lee H, Choi Y, Kim D. Gradient boosting machine for IoT security. Comput Secur. 2023;110:102489. doi:10.1016/j.dcan.2022.10.004.
27. Smith A, Johnson B, Brown C. Ensemble learning methods for IoT intrusion detection using CICIoMT2024 dataset. Expert Syst Appl. 2023;205:117637. doi:10.1109/HONET56683.2022.10019140.