Technoscience Journal for Community Development in Africa 4 (2025) 241–254
ISSN 2659-0573

Research Article

A lightweight deep learning model for intrusion detection in Internet of Medical Things (IoMT) devices using weight pruning technique

Yakubu Ibrahim[1*], Ibrahim Yusuf Fagge[2], Audu Musa Mabu[1]

[1]Department of Computer Science, Yobe State University, Damaturu, Nigeria.
[2]Department of Computer Science, Bayero University, Kano, Nigeria.

Abstract: The Internet of Medical Things (IoMT) has transformed modern healthcare by enabling seamless data collection and monitoring through connected medical devices. However, the growing interconnectivity of these devices exposes them to serious cyber security threats, such as unauthorized access, data tampering, and denial-of-service attacks. These risks are heightened by the limited computing and memory resources of IoMT devices, which make traditional intrusion detection systems unsuitable. This study proposes a lightweight deep neural network model optimized for resource-constrained IoMT environments. The model integrates Principal Component Analysis (PCA) for dimensionality reduction and weight pruning techniques to minimize model complexity while maintaining high detection performance. Experimental results demonstrate a significant reduction in model size to 84 KB with 98% detection accuracy, outperforming state-of-the-art methods. This research provides an efficient and deployable security solution that strengthens the resilience of IoMT devices without overwhelming their limited computational capacity.

Keywords: Artificial Neural Network, Deep learning, Internet of Medical Things, Intrusion detection, Machine Learning

1. Introduction

The Internet of Things (IoT) is a connection of internet-connected objects (nodes) that can send and receive or transfer data over a wireless network without human intervention. The nodes in IoT network can be medical sensors, fitness trackers, smart car, smart watches, etc (Ahmad et al., 2022). The IoT has an extensive applicability in several areas, including healthcare. Internet of medical things (IoMT) is a subset of IoT that amalgamate medical devices and applications that can connect to health care information technology systems using networking technologies (Hameed et al., 2021). Wireless Body Area Networks (WBAN) that contains of wearable and implanted medical devices connecting to and monitoring various parts of the body are a major component of Internet of Medical Things. As an alternative to keeping patients in hospitals these devices are able to monitor the patient's health constantly in real-time, and giving them superior physical flexibility and mobility. In another development, medical robots are also capable of accurately executing minor surgeries. Also, they are capable of performing some medical tasks like Cardio-Pulmonary Resuscitation (CPR) (Yaacoub et al., 2020).

IoMT is a domain of connected sensors, wearable tools, medical devices, and clinical frameworks. It allows several applications of human services to reduce costs of social insurance, give proper clinical responses, and target medical treatment. Due to advancements in remote communication, sensor systems, mobile phones, big data analysis, and distributed computing, IoMT is changing the human services industry by providing

* Corresponding author
Email: yakubu.ibrahim@ysu.edu.ng

KWASU PRESS AND PUBLISHING

customized medical treatment and standardizing the communication of clinical information (Rasool et al., 2022). IoMT has great impact with the advent of Industry 4.0, such as the integration of the automobile industry with IoMT. The digital pillars of Industry 4.0 augmented reality, virtual reality, artificial intelligence, machine learning, and sensors are shaping the future of the medical world. The embedded sensors in the IoMT paradigm enable the continuous monitoring of critical parameters such as ECG, heart rate, blood sugar level, temperature, and respiration. The integration of medical embedded devices, equipment, and the internet allows for the implementation of smart intensive care units (ICUs), smart thermal scanners, medical asset administration, and smart pill dispensers (Yaacoub et al., 2020).

Despite all the benefits and projected evolution in the IoMT, severe security concerns patients' confidential information stealing, Distributed Denial of Service (DDoS) need attention to protect networks and devices from vulnerabilities and various known and unknown cyber-attacks. Networks and IoMT devices are vulnerable to various security threats. Without a comprehensive solution to protect against known and zero-day attacks, critical issues such as data breaches, false diagnostics, fatal accidents due to equipment failure, and Distributed Denial of Service (DDoS) attacks can happen. The most dangerous in IoMT is DDoS, which shut down the critical services or make them unresponsive which can cause great harm to the patients (Ahmad et al., 2022).

Unlike traditional networks, IoMT network nodes have limited resources and low capacity and sometime have control manually. Thus, these ubiquitous devices usually become vulnerable to the attackers. Since many unique and variants of cyber-attacks being generated daily, there is a growing concern for the security of these devices (Shareena et al., 2021). For years several security solutions are developed of which some of them have shown promising result to prevent certain types of attacks. Likewise, quick and effective attack detection are needed as the IoMT devices generates huge amount of data. The common occurring attacks in IoT networks are botnets, DoS, man-in-the-middle, infiltration, identity and data theft, ransom ware, etc. Among them, botnet attacks are very common and preventing them entirely would not be possible as their behavior changes over the period (Nguyen et al., 2022).

Although IoMT is involved in large-scale service supply in the medical paradigm, their resource constrained nature exposes them to significant security and privacy challenges. These flaws are not only devastating for IoMT, but also imperil the entire healthcare ecosystem, putting human lives in jeopardy. Due to their varied character, substantial normal behavior, and the increase in vulnerabilities owing to the exponential proliferation of IoT devices, traditional intrusion detection systems (IDS) are inadequate in the IoMT context (Ahmad & Alsmadi, 2021). Several researchers (Allouzi & Khan, 2021; Dina & Manivannan, 2021; Xin et al., 2018; Zhang et al., 2008) have contributed significantly to the development of intrusion detection systems in IoMTs. Deep learning algorithms have recently shown positive performance in safeguarding IoMT networks and devices. Hence, deep learning models require a lot of computational power, which may be too much for the low-capacity nodes in IoMT networks (Rahman et al., 2020). In-order to detect the botnet activity on IoMT devices, this study proposed a Deep Learning (DL) strategy for detecting intrusion attacks in IoMT devices. Deep learning, a subset of Machine Learning (ML) (Ge et al., 2019), is well-known for its superior problem-solving capabilities and efficiency when dealing with large amounts of data. According to the literature, DL can be employed effectively in a wide range of cyber security applications (Jagannath et al., 2019). To detect botnet activity on IoMT devices, we propose a deep neural network (DL) model and compressed it using neural network weight pruning technique (Zhu & Gupta, 2017). The DL algorithms are effective when there is a vast amount of data or a pattern to be recognized (Taye, 2023).

2. Related Works

Several researchers have proposed different machine learning approaches to monitor and detect attacks on IoT networks. McDermott et al. (2018) proposed a deep learning strategy that combines a Bidirectional Long Short-Term Memory Recurrent Neural Network (BLSTM-RNN) with word embedding to transform string data extracted from captured packets into a representation suitable for BLSTM-RNN-based botnet identification. The data for training and evaluation was obtained from GitHub. The model reached 99% accuracy, although LSTM adds overhead and increases processing time. As a result, the model is incompatible with low-resource IoT devices.

To detect an intruder attempting to inject extraneous data In an IoT network, Jan et al. (2019) proposed a lightweight intrusion detection model based on a support vector machine (SVM). The authors reduced system processing time by relying on a single feature packet arrival rate rather than multiple network attributes. The SVM-based classifier was then used to determine if the

input sample was normal or intruded. The model was tested using the CICIDS2017 dataset, and it achieved 98.03% accuracy. In order to identify infiltration in IoT devices, a single feature attribute is insufficient.

To distinguish network traffic as benign or malicious, Ge et al. (2019) proposed a binary classification model based on a feed-forward neural network. The model was evaluated using the Bot-IoT dataset. Although their model attained a high accuracy of 92% for binary classification, it suffers from a poor accuracy of 82% for multiclass classification because their model is confused by subcategories of assaults within the same category. Almogren (2020) also proposed an edge-based intrusion detection strategy using a deep belief network. The system includes three components: data gathering, feature extraction, and detection. The UNSW-NB15 dataset was used in the evaluation. The model's accuracy is 85%, making it ineffective for intrusion detection. As a result, accuracy should be enhanced.

Li et al. (2020) proposed a multi–convolutional neural network (multi-CNN) fusion approach for intrusion detection. The one-dimensional feature data is transformed into a grayscale graph before being input into the CNN for classification. The NSL-KDD dataset was utilized to train and validate the CNN model, yielding accuracy of 86.95% for binary classification and 76.67% for multiclass classification on the KDDTest and KDD Test2, respectively. As a result, their approach needs future improvement. Rahman et al. (2020) proposed two parallel techniques in which parallel models were developed to correspond to partitioned attack datasets, thereby distributing the computational workload. The parallel models are used initially for side-by-side feature selections, followed by a single multi-layer perceptron classification operating on the fog side. In the distributed case, the parallel models conduct both feature selection and multi-layer perceptron classification independently, after which the outputs are integrated by a coordinating edge or fog to make the final judgment. With 97.80% accuracy, the Aegean Wi-Fi Intrusion Dataset (AWID) was used for evaluation. The dataset utilized is out of date and does not relate to IoT.

In the fog node, a network intrusion detection system based on the Exact Greedy Boosting ensemble approach was proposed by Reddy et al. (2021). The algorithm analyzes nodes and networks to detect intrusions and alerts users when threats are identified. The accuracy for binary, multiclass, and subcategory was 99.6%, 96.7%, and 84%, respectively. The system was evaluated using the IoTID20 simulated environment dataset. Because their work focused on data from the simulated environment, it might lead to a variety of challenges based on real-time data.

Nimbalkar and Kshirsagar (2021) proposed a feature selection approach for intrusion detection in IoT that utilizes Information Gain (IG) and Gain Ratio (GR), selecting the top 50% of features for DDoS attack detection. The system generates feature subsets by performing insertion and union operations on subsets generated by ranking 50% IG and GR features. The approach was examined and validated using a JRipclassifier with 99.99% accuracy and 16.5s detection time on the IoT-BoT and KDD Cup 1999 datasets, respectively. As a result, the model has a high time complexity for detecting attack in IoMT. To identify intrusion in the fog layer of an IoT network, Kan et al. (2021) developed a technique combining adaptive particle swarm optimization (APSO) with a convolutional neural network (CNN). The APSO adaptively optimizes the structure parameters of a one-dimensional CNN as well as the loss function value of training the CNN as the PSO fitness value. During evaluation, 96% accuracy was reached. Despite the fact that an optimization approach was applied at the data level, the model has a high time complexity. As a result, optimization at the algorithm level is required.

Shareenaet al. (2021) presented a complex deep neural network to predict whether network traffic is malicious. The Bot-IoT dataset was used for training and evaluation, achieving 94% accuracy. However, there is no evidence that their model is lightweight, making it unsuitable for low-resource devices. Rambabu and Venkatram (2021) proposed a technique to reduce the computation time of an intrusion detection system by employing K-nearest Neighbor (KNN) for dimensionality reduction and ensemble classification using traffic flow metrics (ECTFM). The NLS-KDD dataset was used for training, achieving 96% accuracy. Nonetheless, data-level optimization is not practical for IoT device intrusion detection.

Su, He, and Wu (2022) trained decision trees, random forests, and gradient-boosting machines (GBM) for IoT threat detection and compared their performance using the IoT 2020 dataset. The decision tree has a higher accuracy of 98%, however the random forest has a higher AUC of

99%. Because a minor change in the data might create a huge change in the structure of the decision tree, generating instability, the decision tree model is not a better solution for intrusion detection. Hameed et al. (2021) presented a hybrid lightweight intrusion detection system for IoMT fog attack detection. The system employs six incremental classifiers: Weighted Hoeffding Tree Ensemble, K-Nearest Neighbors, Naive Bayes, Hoeffding Tree Majority Class, Hoeffding Tree Naive Bayes, and Hoeffding Tree Naive Bayes Adaptive. They gathered and retrieved data from seven disparate sensors, including an IoT-fridge, a garage door, a GPS tracker, Modbus, a sensor light, a thermostat, and a weather sensor. By using a sliding window to configure the ML classifiers. They report accuracy of 92.9%, a model space complexity of 6 MiB, and an execution time of 21 seconds. Because the system employs incremental learning, increasing the number of instances results in an increase in execution time. Performance varied across sample frequencies, indicating that their methodologies were not robust and were influenced by concept drift in the data.

Most closely related to our work is Chaganti et al. (2022), who proposed using Particle Swarm Optimization (PSO) for feature extraction and a deep neural network (DNN) for detecting attacks in IoMT networks. They trained and evaluated their model with imbalanced IoMT network traffic and patient biometric datasets. They thoroughly analyzed different machine learning and deep learning algorithm including Logistic Regression (LR), K-Nearest Neighbor (KNN), Decision Tree, AdaBoost, Random Forest (RF), Support Vector Machine (SVM), Deep Neural Networks (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), They reported that DNN perform better and achieved accuracy of 96%, precision of 95% and recall of 95%. Although, the model achieved high accuracy but lack generalization and the performance can still be enhanced. Medical devices have limited processing power and storage capacity, which can make it difficult to run resource-intensive DNNs.

To build upon the work of Chaganti et al. (2022), we propose using Principal Component Analysis (PCA) as a dimensionality reduction technique to decrease the complexity of the data. By applying PCA, we aimed to capture the most important features of the dataset while reducing its dimensionality. To address the issue of imbalanced data, we employed the Synthetic Minority Over-sampling Technique (SMOTE). Considering the resource constraints of Internet of Medical Things (IoMT) devices, we proposed a compression technique that involved pruning the less significant weights of the model. By eliminating the less significant weights, we aimed to reduce the model's size and computational requirements, making it more suitable for deployment on resource-constrained IoMT devices. To ensure that the compressed model maintained its accuracy, we performed fine-tuning. This process involved retraining the model using the pruned weights as a starting point and further optimizing it to achieve a balance between model size and performance.

3. Materials and methods

This section presented the methodology used in the research. The overall framework and steps employed in the research are depicted in Figure 3.1. The framework served as a guide for the execution of the research, while the steps outlined the systematic approach that was followed throughout the study. By adhering to this methodology, the research aimed to ensure rigor, reliability, and validity in the investigation process.

3.1. Research framework

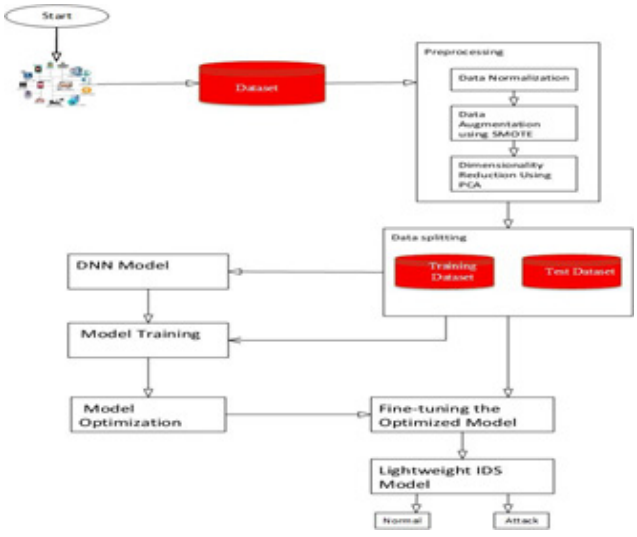

Figure 1: Proposed approach

3.2. Dataset used

Hady et al. (2020) used a real-time health monitoring testbed to create the dataset. The testbed consists of sensor devices attached to the patient's body, a network gateway, and a Software Defined Network (SDN) controller for visualizing network traffic. The network traffic and sensor data generated in the testbed are used to detect anomalies in the data and identify attacks. The attack dataset was created by simulating three attacks in the environment. These are man-in-the-middle attacks, data injection attacks, and spoofing attacks. The ARGUS tool (Argus, 2020) was used to generate a combined dataset of network traffic and patient biometric data. Temperature, peripheral oxygen saturation, pulse rate, systolic blood pressure, diastolic blood pressure, heart rate, respiration rate, and ECG ST Segment data are among the biometric data. To obtain the overall network traffic features, the network traffic flow records and their metrics are captured. The dataset contains 12,000 records out of which 11,000 is the normal traffics and 1000 as attack traffics, and 44 features in total, 35 of which are network traffic features, while 9 are biometric features. The output of the dataset is labeled as an attack or normal traffic. Attack traffic is labeled "0," while normal traffic is labeled "1'.

3.3. Feature scaling

Despite the fact that the features were clean and trainable, the numerical discrepancies in the records were significant, which impaired the model's convergence time and training effect. As a result, the dataset needed to be normalized so that the data in the sample fell between [0, 1]. It was essential to avoid the detrimental influence of the sample mean and variance because the datasets contained both benign and malicious traffic. A basic linear normalization procedure, as demonstrated in Equation (1), was applied to handle the general numerical characteristics:

$$X' = \frac{X - Xmin}{Xmax - Xmin}$$

Here, Xmax represented the maximum value and Xmin represented the minimum value for each feature across all the data. By applying this normalization technique, the data values were scaled to a standardized range, allowing the model to effectively process and learn from the features without being affected by the varying scales or magnitudes of the numerical attributes.

We used the MinMaxScaler function from the scikit-learn library in Python to perform MinMax normalization on the data. This function allows us to transform the data to a specified range, in our case, between 0 and 1.

3.4. Data augmentation

The dataset was imbalanced, indicating an unequal distribution of the target classes. Working with imbalanced datasets presents the challenge that the model may neglect the minority class and perform poorly in predicting it, despite the importance of accurately classifying the minority class (Blagus & Lusa, 2013). To address this issue, we employed the SMOTE (Synthetic Minority Oversampling Technique) approach in our work, which involved oversampling the minority class. The SMOTE method entailed duplicating examples from the minority class, even though these instances did not provide any new information to the model. Instead, new instances were created by combining existing ones (Blagus & Lusa, 2013). The SMOTE algorithm operated mathematically through the following steps:

Step 1: The minority class set, denoted as A, was established. For each x A, the k-nearest neighbors of x were determined by calculating the Euclidean Distance between x and every sample in set A.

Step 2: The sampling rate N was determined based on the imbalanced proportion. For each x A, N examples x1, x2, x3, ..., xn) were randomly selected from its k-nearest neighbors, forming the set A1.

Step 3: For each example xk A1 (k = 1, 2, 3, ..., N), a new example was generated using the formula shown in Equation (2):

$$X' = x + rand(0,1) * |x - xk|$$

Here, rand (0, 1) represented a random number between 0 and 1.

We used SMOTE function from the imbalanced-learn library in Python. This technique allowed us to oversample the minority class and achieve a more balanced dataset.

3.5. Feature extraction

Dimensionality reduction is a method of feature extraction that reduces the number of features in a dataset while retaining as much of the original information as possible (Zebari et al., 2020). Since our dataset had many features, we adopted principal component analysis (PCA) for dimensionality reduction. PCA worked by identifying the directions (principal components) in

which the data varied the most and projecting the data onto a lower-dimensional subspace defined by these directions (Kherif & Latypova, 2020). We used PCA function from the scikit-learn library in Python using 12 number of components.

3.6. Deep neural network

An artificial neural network (ANN) is a computing model inspired by the structure and function of the human brain, consisting of interconnected neurons organized into layers. Deep neural networks (DNNs) are a type of ANN that have more than three hidden layers, allowing them to model more complex patterns in the data (Allouzi & Khan, 2021). In this work, the proposed neural network with multiple input vectors and several hidden layers. The activation function used in the hidden layers is Relu, and the output layer uses a sigmoid activation function for binary classification. The learning rate, epochs, and batch size are hyperparameters that are tuned during the training phase to optimize the performance of the model. The learning rate determines the step size at each cycle as the model progresses towards the minimum of the loss function, and the epochs and batch size control the number of training iterations and the size of the training samples used in each iteration, respectively. Algorithm1 described the pseudo-code of our propose model.

Algorithm 1: Algorithmic Description of the proposed model.
Input: Training data: x and training label: y;
Randomly initialize parameters W and b;
1 load the dataset D(xi, yi)
2 For every xi in D:
3 X'i = (x-xmin)/(xmax - xmin) // Normalization
4 end for
5 For each x'ϵ Minority class:
*6 x'' = x'+ rand(0,1) * |x'-x'k| // SMOTE*
7 end for
8 While training do:
9 Randomly select data point x''i and its label yi ;
10 Compute the model prediction f(x'') for x''i;
11 If yi == f(x''i) then
12 Continue;
13 Else
14 Update the parameters W and b;
15 wi+1 = wi + (yi – f(x'')) x''i;
16 bi+1 = bi + (yi – f(x''i));
17 end if
18 end while
19 Output: Trained DNN intrusion detection model.

3.7. Weights pruning techniques

To compress the model, we proposed a method of removing weights of neural networks by setting unwanted Individual parameters to zero, and make the network sparse. This would reduce the number of parameters in the model while maintaining the same architecture. The pruning process follows the sparsity function *si* that indicates the proportion of pruned weights to all prunable weights of ANN, which given by (Zhu & Gupta, 2017).

$$s_i = \begin{cases} 0 & i < i_{start} \\ s_{end} + (s_{start} - s_{end})\left(1 - \frac{i - i_{start}}{i_{end} - i_{start}}\right)^n & i \in \{i_{start}, i_{start} + \Delta i, \dots, i_{start} + (m-1)\Delta i, i_{end}\} \\ s_{end} & i > i_{start} \end{cases}$$

Where *si* is the sparsity after the itth epoch of training, Sstart is the initial sparsity, and *send* is the target sparsity. m is the number of pruning processes, which is related to, Sstart, Sstart and Δ (*iend*= i_{start}+Δend=istart +mΔi). n is the order of the sparsity function, which controls the pruning schedule (Zhao & Chi, 2020). Algorithm 2 provide the pseudo code of the proposed pruning algorithm.

Algorithm 2: Algorithmic Description of proposed pruning technique.
Input: training data x, initial sparsity: sstart, Final sparsity send, algorithm starting point: istart , algorithm end point: iend , pruning frequency: Δi, prunable parameters: W
1 m = 0;
2 for i = istart + mΔido
3 update si;
4 Sort the weights in W based on their absolute value;
5 Set the smallest si weights as 0;
6 m = m+1;
7 end for
8 for i < iend +1 do
9 Train the prunable parameters of the DNN model based on x;
10 i = i+1
11 end for
12 Output: Pruned DNN model with parameter W

3.8. Evaluation metrics

Machine learning provides numerous metrics for evaluating classifier performance. It is critical to choose the appropriate performance measures based on the practical requirements of a specific domain. Certain classifiers may perform well when examined using one metric but poorly when evaluated using another [(Liu et al., 2014). The performance of our proposed model will be measured using accuracy, precision, recall, sensitivity, and F1-score metrics. Where TP represent True Positive, TN represent True Negative, FP represent False Positive, and FN represent False Negative.

3.8.1. Accuracy

It measures how well the model detects intrusions or attacks. It calculates the proportion of accurately classified intrusion attempts to total inputs (Ahmed et al., 2022). It's calculated as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN} \quad (3)$$

3.8.2. Precision

It is the proportion of actions identified by the algorithm as attack that are truly attack (Shareena et al., 2021). It is calculated as follows:

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

3.8.3. Recall

It is the proportion of actual intrusions that were predicted as intrusions by the model (Zhang et al., 2008). It is calculated as follows:

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

3.8.4. **F1**-*Score*

It is the harmonic mean (reciprocal of the arithmetic mean of the reciprocals) of Precision and Recall (Shareena et al., 2021). It is calculated as follows:

$$\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{Precision + Recall} \quad (6)$$

Where True Positive (TP) is an actual intrusion that classified as an intrusion by the IDS, True Negative (TN) is an actual non-intrusive event that classified as non-intrusive by the IDS, False Positive (FP) is an intrusive event that classified as non-intrusive by the IDS, False Negative (FN) is non-intrusive event classified as intrusive action (Taye, 2023).

3.8.5. Model Size

Model file size is typically measured in bytes or kilobytes (KB), and it represents the size of the serialized model file stored on disk. It includes all the weights, biases, and other model-related parameters that need to be saved for model persistence or deployment [34]. To calculate the model file size, we considered the following equation:

$$Model\ size = \sum (i = 1\ to\ L)(s[i]) \quad (7)$$

Where S[i] represents the size (in bytes) of the parameters in layer i. For example, if a layer has weights and biases stored as 32-bit floating-point numbers, you would calculate the size as:

$$S[i] = (N[i] + 1) * 4 \quad (8)$$

Where, N[i] represents the total number of weights and biases in layer i, and the factor of 4 represents the size of a 32-bit floating-point number in bytes.

3.9. Experimental environment setup

The experiments were conducted in the Google Colab environment, which provided a convenient platform for running Python code in a Jupyter notebook-like environment. The deep learning model was developed using the TensorFlow framework, which offers a comprehensive set of tools and functionalities for building and training neural networks.

3. Results

We defined a Multi-layer Perceptron (MLP) model architecture using the Keras library in Python. The architecture consisted of an input layer with 12 input dimensions and 100 neurons, which utilized the Rectified Linear Unit (ReLU) activation function. Four hidden layers were defined, each with 512 neurons and ReLU activation. Finally, the output layer consisted of 1 neuron with a sigmoid activation function. To train the model we used the following hyperparameters in Table 1.

Table 1: Hyperparameters used for model training

Hyperparameter	**Value**
Learning Rate	0.001
Batch Size	10
Number of Epochs	100
Optimizer	Adam
Loss Function	Binary Crossentropy
Validation Split	0.2

These hyperparameters were chosen to optimize the training process and achieve the best performance for the model. The learning rate determines the step size during gradient descent optimization, the batch size determines the number of samples processed before updating the model, and the number of epochs defines the number of complete passes through the entire dataset during training. The Adam optimizer combines the advantages of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms. Binary crossentropy was used as the loss function since the task involved binary classification. A validation split of 0.2 was used to allocate 20% of the training data for

validation during model training. After the training we got the following result:

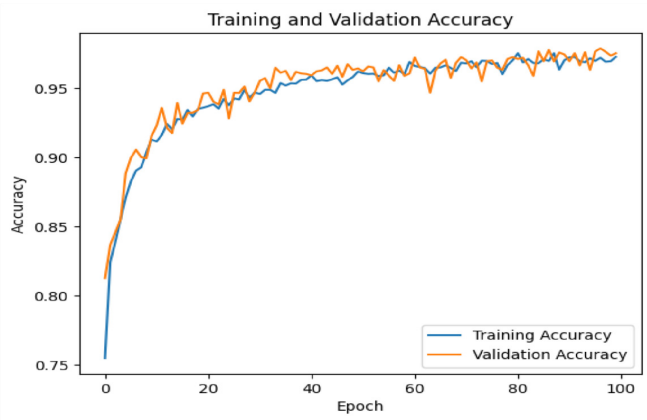

Figure 2: Training and validation accuracy

Figure 2 illustrates the consistent and high performance of the model on both the training and validation datasets. With training and validation accuracies reaching 0.98, the model demonstrates its ability to accurately predict the target variable. The similarity in accuracy between the training and validation sets suggests that the model is not overfitting the training data. This indicates that the model has successfully captured the underlying patterns in the data, allowing it to generalize well to unseen examples. Thus, the results indicate a strong level of generalization and reliable predictions by the model.

Figure 3: Training and validation loss

Figure 3 clearly illustrates the low training and validation loss values, further supporting the model's strong generalization capabilities. The convergence of both training and validation losses on the graph indicates that the model effectively minimizes errors during the training process. This suggests that the model has learned to capture the essential patterns and features of the data without overfitting. The consistency between the low training and validation losses, as depicted in the graph (Fig. 3), reinforces the reliability of the model's predictions and its ability to generalize well to unseen data. Overall, these findings provide additional evidence of the model's robustness and its suitability for accurate predictions.

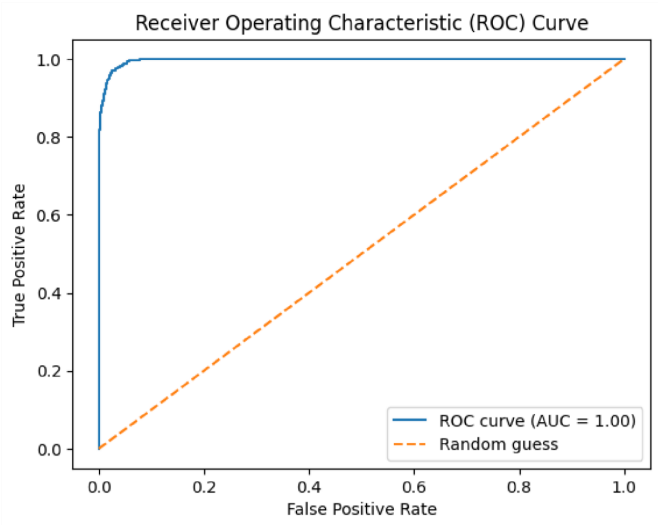

Figure 4: Model AUC

An AUC of 1.0 indicates that the model has achieved a perfect discrimination between the positive and negative classes. It suggests that the model correctly ranks all positive instances above negative instances, resulting in a flawless separation. This exceptional AUC score further validates the model's ability to distinguish between the target classes accurately. Thus, we can confidently conclude that the model exhibits outstanding predictive performance with a flawless discriminatory power, as evidenced by the achieved AUC of 1.0.

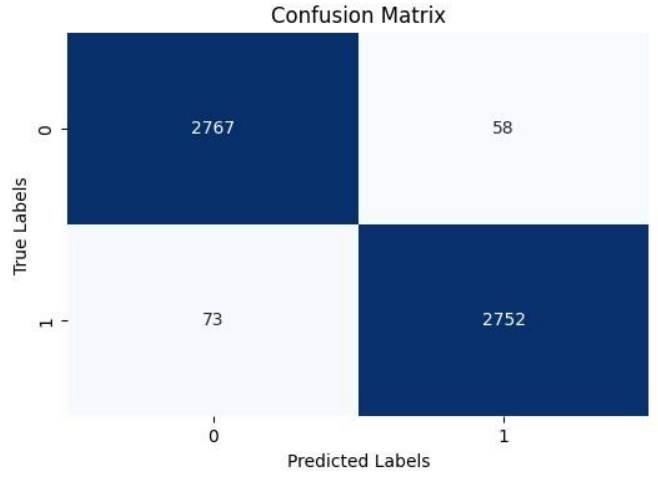

Figure 5: Confusion matrix

The confusion matrix indicates that model demonstrates a high level of accuracy in predicting the target variable, with a significant number of True Positives (TP) and True Negatives (TN). TP = 2752 indicates that the model correctly identified 2752 positive instances, while TN = 2767 indicates that it accurately classified 2767 instances as negative. Although the model's performance is generally strong, it is worth noting that it did have a few misclassifications. The number of False Positives (FP) is 73, indicating instances that were wrongly predicted as positive when they were actually negative. Similarly, there are 58 instances classified as False Negatives (FN), indicating instances that were wrongly predicted as negative when they were actually positive. While these misclassifications exist, they represent a relatively small proportion of the overall dataset. The high number of TP and TN, along with the low values of FP and FN, suggest that the model has a good ability to discriminate between positive and negative instances.

Table 2: Result Summary

Accuracy	Precision	Recall	F1 score	size
0.98	0.98	0.98	0.98	*1MB*

Table 2 presents a summary of the results obtained after training the model. The performance metrics provide insights into the model's effectiveness in predicting the target variable. The accuracy of the model is 0.98, indicating that it correctly predicts the target variable for 98% of the instances. A high accuracy score suggests that the model performs well in classifying both positive and negative instances. The precision of the model is 0.98, which signifies the proportion of correctly predicted positive instances out of all instances predicted as positive. This indicates that the model has a high precision in identifying positive instances. The recall of the model is also 0.98, representing the proportion of correctly predicted positive instances out of all actual positive instances. A high recall score suggests that the model effectively captures the positive instances in the dataset. The F1 score is a harmonic mean of precision and recall, providing a balanced measure of the model's performance. With an F1 score of 0.98, the model demonstrates a strong balance between precision and recall.

4.1. Pruning the weights of the trained model.

We used tensorflow_model_optimization.sparsity.keras module and pruned our trained model using the following hyperparameters listed in Table 3 .

Table 3: Hyperparameters used in model pruning

Hyperparameters	Values
Initial Sparsity	0.50
Final Sparsity	0.90
Begin Step	0
End Step	100

The initial sparsity defines the starting point of the pruning process. It represents the percentage of weights that are initially pruned (set to zero). The initial sparsity of 0.50, means that 50% of the weights in the model are initially pruned. The final sparsity represents the target sparsity level to be achieved at the end of the pruning process. It defines the percentage of weights that will eventually be pruned. The final sparsity of 0.90, indicating that the pruning process aims to prune 90% of the weights in the model. The begin step determines when the pruning process starts. It specifies the training step or epoch at which the pruning begins. The begin step of 0, indicating that pruning starts from the beginning of the training process. The end step indicates the training step or epoch at which the pruning process ends. It defines the point in the training process when the desired final sparsity should be reached. The end step is set to 100, indicating that the pruning process should be completed after 100 training steps or epochs.

4.2. Fine tuning the pruned model

We retrained the pruned model with the same hyperparameters as the baseline model but reduced the number of epochs to 10. We also updated the pruned weights using tfmot.sparsity.keras.UpdatePruningStep. Figure 6 shows the results obtained from evaluating the pruned model.

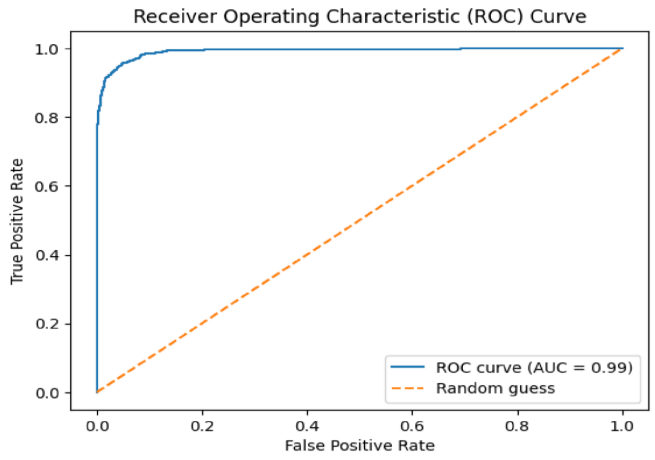

Figure 6: AUC of the pruned model

After applying pruning techniques to reduce the complexity of the model, we observed a slight decrease in the AUC value from 1.0 to 0.99. However, it is important to note that an AUC of 0.99 still indicates excellent performance and a high level of discrimination between positive and negative classes.

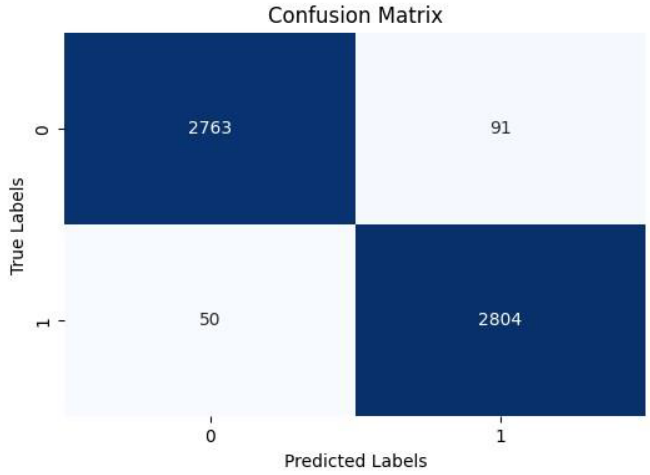

Figure 7: Confusion matrix of pruned model

Table 4: Pruned Model Results Summary

Accuracy	Precision	Recall	F1-score	size
0.98	0.97	0.98	0.97	84KB

The results in Table 4 indicate that the pruned model maintains a high level of performance even after the pruning process, suggesting that unnecessary or less significant weights were successfully removed without significantly impacting the model's predictive capabilities. It demonstrates the effectiveness of the pruning technique in reducing model complexity while preserving performance. Pruning the model allowed us to achieve a more efficient and streamlined model without significantly sacrificing its accuracy. This demonstrates the effectiveness of the pruning technique in optimizing the model for resource-constrained environments, such as IoMT nodes with limited processing power and memory capacity. Therefore, the pruned model retains a high level of performance and offers improved efficiency for real-world deployment. These results highlight the potential of our approach in developing lightweight and efficient deep learning models for IoMT applications.

4.3. Results comparison

We present a detailed comparison of our model with state-of-the-art models in the field. The comparison aims to evaluate the performance and effectiveness of our proposed model in addressing the research problem. By comparing our model to existing approaches, we can gain insights into its strengths, weaknesses, and potential contributions to the field. The evaluation metrics used for comparison include accuracy, precision, recall, F1-score, and model size. Table 5 provides a comprehensive overview of the performance metrics and model sizes of our model and selected state-of-the-art models. This analysis will serve as a foundation for discussing the superiority and advancements of our approach.

Table 5: Comparison with the existing models

Author	Method	Accuracy	Size
(Hady et al., 2020)	KNN	90%	-
(Gupta et al., 2022)	Tree based classifier	93%	-
(Chaganti etal., 2022)	PSO-DNN	96%	1MB
Proposed model	PCA-DNN	98%	1MB
Pruned model	PCA-DNN	97%	84KB

A comparative analysis between the proposed PCA-DNN model and existing intrusion detection approaches for IoMT environments clearly demonstrate that the proposed model outperforms prior studies in terms of detection accuracy while also addressing the critical constraint of model size.

Compared with the KNN-based approach by Hady et al. (2020), which achieved an accuracy of 90%, and the tree-based classifier proposed by Gupta et al. (2022) with 93% accuracy, the proposed PCA-DNN model achieved a substantially higher accuracy of 98%. This improvement highlights the advantage of deep learning models when combined with effective feature reduction techniques in capturing complex attack patterns within IoMT traffic.

Furthermore, when compared to the PSO-DNN model introduced by Chaganti et al. (2022), which reported a 96% accuracy with a model size of approximately 1 MB, the proposed model not only improves detection accuracy but also maintains comparable storage requirements prior to pruning. This demonstrates that replacing feature optimization via particle swarm optimization with Principal Component Analysis can yield better generalization while reducing feature redundancy.

A key contribution of this study is the pruning of the PCA-DNN model, which reduced the model size from 1 MB to 84 KB, while retaining a high accuracy of 97%. This result is particularly significant for resource-constrained IoMT devices, where memory and computational power are limited. Although pruning led to a slight reduction in accuracy (from 0.98 to 0.97), a paired t-test confirmed that the difference was not statistically significant, indicating that pruning effectively reduces computational and storage overhead without meaningful performance degradation.

The implications of these findings are substantial for real-time healthcare systems. High recall values imply a lower false-negative rate, which is crucial in IoMT environments where undetected attacks may compromise patient safety. Overall, the proposed PCA-DNN and its pruned variant provide a superior balance between accuracy, efficiency, and lightweight deployment, outperforming existing IoMT intrusion detection models and making them well-suited for real-world healthcare applications.

5. Conclusion and recommendation

This study presented a lightweight deep learning model for intrusion detection in Internet of Medical Things (IoMT) devices using Principal Component Analysis (PCA) for dimensionality reduction and weight pruning for model compression. The proposed PCA-DNN achieved 98% accuracy with a model size of 84 KB, outperforming prior state-of-the-art techniques. A major contribution of this work is the introduction of a pruning algorithm that makes the neural network sparse by setting the less significant weights to zero. This approach effectively reduces computational and memory requirements while preserving high detection performance. The pruning process significantly optimized the model's complexity, enabling deployment on resource-constrained IoMT devices without substantial loss in accuracy. The results confirm that the proposed method provides an efficient and practical solution for securing IoMT systems. Future work will focus on evaluating the model's performance on larger and more diverse datasets and investigating its robustness against adversarial attacks to further enhance reliability in real-world medical IoT environments. Also, k-fold cross-validation and evaluation on external datasets can be performed to validate the model's generalizability across different IoMT environments.

References

Ahmad, R., & Alsmadi, I. (2021) Machine learning approaches to IoT security: A systematic literature review. *Internet of Things, 14*, 100365.

Ahmad, R., Alsmadi, I., Alhamdani, W., & Tawalbeh, L. A. (2022) A comprehensive deep learning benchmark for IoT IDS. *Computers & Security, 114*, 102588.

Allouzi, M. A., & Khan, J. I. (2021) Identifying and modeling security threats for IoMT edge network using Markov chain and common vulnerability scoring system (CVSS) *arXiv preprint*, arXiv:2104.11580.

Almogren, A. S. (2020) Intrusion detection in Edge-of-Things computing. *Journal of Parallel and Distributed Computing, 137*, 259–265.

Argus. (2020) *Audit record generation and utilization system*. https://openargus.org

Blagus, R., & Lusa, L. (2013) SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics, 14*, 1–16.

Chaganti, R., Mourade, A., Ravi, V., Vemprala, N., Dua, A., & Bhushan, B. (2022) A particle swarm optimization and deep learning approach for intrusion detection system in Internet of Medical Things. *Sustainability, 14*(19), 12828.

Dina, A. S., & Manivannan, D. (2021) Intrusion detection based on machine learning techniques in computer networks. *Internet of Things, 16*, 100462.

Ge, M., Fu, X., Syed, N., Baig, Z., Teo, G., & Robles-Kelly, A. (2019) Deep learning-based intrusion detection for IoT networks. In *Proceedings of the IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 256–260)

Gupta, K., Sharma, D. K., Gupta, K. D., & Kumar, A. (2022) A tree classifier-based network intrusion detection model for Internet of Medical Things. *Computers and Electrical Engineering, 102*, 108158.

Hady, A. A., Ghubaish, A., Salman, T., Unal, D., & Jain, R. (2020) Intrusion detection system for healthcare systems using medical and network data: A comparison study. *IEEE Access, 8*, 106576–106584.

Hameed, S. S., Selamat, A., Abdul Latiff, L., Razak, S. A., Krejcar, O., Fujita, H., & Omatu, S. (2021) A hybrid lightweight system for early attack detection in the IoMT fog. *Sensors, 21*(24), 8289.

Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., & Melodia, T. (2019) Machine learning for wireless communications in the Internet of Things: A comprehensive survey. *Ad Hoc Networks, 93*, 101913.

Jan, S. U., Ahmed, S., Shakhov, V., & Koo, I. (2019) Toward a lightweight intrusion detection system for the Internet of Things. *IEEE Access, 7*, 42450–42471.

Kan, X., Fan, Y., Fang, Z., Li, L., & Wang, S. (2021) A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network. *Information Sciences, 568*, 147–162.

Kherif, F., & Latypova, A. (2020) Principal component analysis. In *Machine learning* (pp. 209–225) Academic Press.

Li, Y., Wang, C., Li, J., & Wang, J. (2020) Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. *Measurement, 154*, 107450.

Liu, Y., Zhou, Y., Wen, S., & Tang, C. (2014) A strategy on selecting performance metrics for classifier

evaluation. *International Journal of Mobile Computing and Multimedia Communications, 6*(4), 20–35.

McDermott, C. D., Majdani, F., & Petrovski, A. V. (2018) Botnet detection in the Internet of Things using deep learning approaches. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8)

Nguyen, X. H., Nguyen, X. D., Huynh, H. H., & Le, K. H. (2022) RealGuard: A lightweight network intrusion detection system for IoT gateways. *Sensors, 22*(2), 432.

Nimbalkar, P., & Kshirsagar, D. (2021) Feature selection for intrusion detection system in Internet-of-Things (IoT) *ICT Express, 7*(2), 177–181.

Rahman, M. A., Asyhari, A. T., Leong, L. S., Satrya, G. B., Tao, M. H., & Zolkipli, M. F. (2020) Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustainable Cities and Society, 61*, 102324.

Rambabu, K., & Venkatram, N. (2021) Ensemble classification using traffic flow metrics to predict distributed denial of service scope in the Internet of Things (IoT) networks. *Computers & Electrical Engineering, 96*, 107444.

Rasool, R. U., Ahmad, H. F., Rafique, W., Qayyum, A., & Qadir, J. (2022) Security and privacy of Internet of Medical Things: A contemporary review in the age of surveillance, botnets, and adversarial ML. *Journal of Network and Computer Applications*, 103332.

Reddy, D. K. K., Behera, H. S., Nayak, J., Naik, B., Ghosh, U., & Sharma, P. K. (2021) Exact greedy algorithm-based split finding approach for intrusion detection in fog-enabled IoT environment. *Journal of Information Security and Applications, 60*, 102866.

Schizas, N., Karras, A., Karras, C., & Sioutas, S. (2022) TinyML for ultra-low power AI and large-scale IoT deployments: A systematic review. *Future Internet, 14*(12), 363.

Shareena, J., Ramdas, A., & Haripriya, A.P. (2021) Intrusion detection system for IoT botnet attacks using deep learning. *SN Computer Science, 2*(3), 1–8.

Su, J., He, S., & Wu, Y. (2022) Feature selection and prediction for IoT attacks. *High-Confidence Computing, 2*(2), 100047.

Taye, M. M. (2023) Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. *Computers, 12*(5), 91.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Wang, M., & Qiu, L. (2018) Machine learning and deep learning methods for cybersecurity. *IEEE Access, 6*, 35365–35381.

Yaacoub, J. P. A., Noura, M., Noura, H. N., Salman, O., Yaacoub, E., Couturier, R., & Chehab, A. (2020) Securing Internet of Medical Things systems: Limitations, issues and recommendations. *Future Generation Computer Systems, 105*, 581–606.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., & Saeed, J. (2020) A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends, 1*(2), 56–70.

Zhang, J., Zulkernine, M., & Haque, A. (2008) Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(5), 649–659.

Zhao, Y., & Chi, N. (2020) Partial pruning strategy for a dual-branch multilayer perceptron-based post-equalizer in underwater visible light communication systems. *Optics Express, 28*(10), 15562–15572.

Zhu, M., & Gupta, S. (2017) To prune, or not to prune: Exploring the efficacy of pruning for model compression. *arXiv preprint*, arXiv:1710.01878.