Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Tabnet And Optuna-Driven Optimization For Iot Network Intrusion Detection

Nitu Dash[1], Sujata Chakravarty[2], Amiya Kumar Rath[3], Sunil Kumar Mohapatra[4], Pritam Pattanaik[5]

[1]Department of Computer Science & Engineering, BPUT, Rourkela, Odisha, India. nitudash@gmail.com
[2 4 5]Department of Computer Science & Engineering, CUTM, Odisha, India. chakravartys69@gmail.com, sunil.mbj@gamil.com, pritampattanaik378@gmail.com
[3]Department of Computer Science & Engineering, Faculty of Engineering, BPUT, Rourkela, Odisha, India. amiyaamiya@rediffmail.com

The widespread adoption of the Internet of Things (IoT) due to its ease of use and versatility has simultaneously raised significant security issues concerning interconnected devices and networks worldwide. This highlights the pressing need for robust Intrusion Detection Systems (IDS) to counter cyber threats. Deep learning (DL) methodologies present a highly effective solution, enabling the detection of anomalies in network traffic and significantly bolstering IoT network security while reducing the risk of cyberattacks. This paper investigates Long Short-Term Memory (LSTM) and Attentive Interpretable Tabular Learning (TabNet) DL methods for enhancing IDS classification in IoT network. Additionally, the Optuna framework is utilized to optimize classifier performance by fine-tuning hyperparameters and determining the ideal model configuration. These models are evaluated using publicly available NSL KDD and BoTNeTIoT-L01 datasets. Experimental results indicate that optimization process significantly improves model performance by achieving high accuracy. The TabNet model exhibits exceptional feature interpretation capabilities, providing insights into the most critical factors for detecting intrusion in IoT networks. Comparative analysis reveals that the Optuna-optimized TabNet model surpasses the standard LSTM, TabNet and Optuna based LSTM model in terms of accuracy, precision, and other computational efficiency. These findings highlights the efficacy of the model in significantly enhancing intrusion detection in IoT networks by providing timely actions to security breaches.

Keywords: Intrusion Detection System (IDS); Long Short-Term Memory (LSTM); Attentive Interpretable Tabular Learning (TabNet), Optuna.

1. Introduction

The Internet of Things (IoT) plays a crucial role in revolutionizing industries and transforming daily life by creating a connected ecosystem of devices and systems. IoT enables the seamless exchange of data between physical objects, sensors, and cloud-based platforms, fostering unprecedented levels of automation, efficiency, and operational insights. With IoT technology, organizations can optimize processes, develop innovative business models, and enhance decision-making through real-time data analytics [1]. IoT applications has the ability to

enhance convenience, safety, and sustainability across various domains ranging from smart homes to industrial automation and smart cities. By harnessing the power of interconnected devices, IoT drives technological advancements, improves resource management, and paves the way for a more efficient, interconnected future. The International Data Corporation (IDC) forecasted that by 2025, there will be 41.6 billion networked IoT devices producing a whopping 79.4 zettabytes (ZB) of data [2]. Despite managing and generating significant amounts of data, IoT devices are typically inexpensive and possess limited CPU, storage, and memory resources. Without adequate security measures, these devices can be vulnerable to cyber-attacks, potentially leading to unforeseen disruptions within private networks. Such security breaches can jeopardize service availability, compromise data confidentiality, and infringe upon user privacy [3]. Consequently, the IoT ecosystem presents an attractive target for cybercriminals, necessitating innovative solutions to address cybersecurity threats and safeguard data protection [4].

Intrusion Detection Systems (IDS) serve as vital security measures for networks, identifying unauthorized access and potential attacks by analyzing network activity and internal behaviors [5]. The shift towards intelligent, data-driven solutions is essential, moving beyond traditional knowledge-based IDS approaches. Recently, researchers have veered towards innovative methodologies such as anomaly-based IDS leveraging deep learning methods in analyzing vast datasets across various contexts. These methods facilitate event correlation, pattern recognition, and the detection of abnormal activities that might otherwise go unnoticed [6]. However, the efficiency of deep learning models heavily reliant on the selection of optimal hyperparameters. This task can be quite difficult and time consuming because it necessitates in-depth domain expertise and experimentation.

In this study, Optuna, an open-source framework for efficient hyperparameter optimization is utilized to enhance the accuracy and productivity of DL approaches for IoT IDS. Optuna automates the hyperparameter tuning process, employing various optimization algorithms and pruning techniques to efficiently explore the hyperparameter search space [7]. By integrating seamlessly with deep learning libraries, Optuna enables the building of optimized models that can accurately detect intrusions while minimizing false positives, ultimately fortifying the security of IoT networks. Moreover, Optuna's support for distributed and parallel optimization, as well as its efficient resource utilization, make it particularly well-suited for the resource-constrained environment of IoT network devices.

The primary contributions of this paper are listed below:

- Designing deep learning architectures for IoT intrusion detection.
- It enhances interpretability by leveraging TabNet's sparse feature selection and decision rule learning, allowing for automatic feature explanations, helping users understand model predictions and improving feature identification.
- To address IoT resource constraints, the models are optimized using OPTUNA, ensuring computational efficiency and suitability for real-time, resource-limited environments.
- Conducting extensive evaluations of the optimized IDS models on diverse datasets, benchmarking against state-of-the-art solutions.

The structure of the paper is as follows: Section 2 presents a literature survey of various machine learning and deep learning method in the context of IDS classification. Section 3 outlines the Proposed framework, and the methodologies employed for constructing the IDS model. This includes a description of the LSTM and TabNet model, as well as insights into the Optuna technique. Section 4 details the NSL KDD and BoTNeTIoT-L01 dataset followed by data pre-processing steps. Section 5 elaborates the performance metrics and delineates the investigational outcomes along with their analysis. Finally, Section 6 presents the conclusion.

2. Related Work

Various studies have been done to improve IoT security and protect these systems from possible attacks by using machine learning and DL methods. These cutting-edge approach have shown remarkably adept at precisely and quickly identifying security vulnerabilities in IoT environments, allowing for preventative actions to reduce risks before they have a significant negative impact.

Researchers investigated detection against adversarial attacks applying self-normalizing neural network (SNN) and feedforward neural network (FNN) [8]. They employed Bot-IoT dataset and achieved 95.1% accuracy with FNN. SNN demonstrated higher resilience (9%) against adversarial attacks. Despite feature normalization improving SNN's resilience, it reduced its accuracy to below 50%, rendering it inappropriate for real-world defenses. In [9], FNN is applied for traffic analysis in a network setting, with a comparative study against the support vector classifier (SVC). FNN achieved outstanding performance using Bot-IoT dataset, with 99.414% accuracy for DDoS/DoS attacks and 99% across all metrics. IoT IDS framework using CNN [10] using Bot-IoT dataset achieved 91.27% as highest accuracy with 128 batch size, but lower accuracies were observed with smaller batch sizes. A deep blockchain framework (DBF), combining bidirectional LSTM (BiLSTM) and blockchain to enhance confidentiality and identify malicious behavior is introduced in [11]. Examination of the Bot-IoT and UNSW-NB15 datasets yielded accuracies of 98.91% and 99.41%, respectively. However, the solution's effectiveness diminishes under heavy network traffic, particularly in detecting complex attacks. A cloud-based detection system is depicted in[12] using Distributed CNN for IoT devices and LSTM for cloud hosts to improve detection of various malicious attacks. Analysis using the N_BaIoT dataset revealed CNN achieving accuracy of 94.30% and LSTM achieving high accuracy i.e. 97.84%. However, the solution's inability to identify emerging attacks exposes IoT devices to threats. A robust framework for IoT attack detection leverages distributed techniques and deep learning [13]. It deploys detectors on fog nodes for efficiency and evaluates six DL models i.e. DNN, LSTM, BiLSTM, GRU, CNN, CNN-LSTM, with LSTM performing best. The framework achieves exceptional detection rates and accuracy, surpassing 99% in binary and multi-class classifications. Another paper examines several ML algorithms like KNN, Decision Trees (DT), Naive Bayes (NB), Random Forest (RF), SVM, ANN, and Logistic Regression for IoT IDS networks [14]. The Bot-IoT dataset shows that RF achieves 99% accuracy in binary classification of HTTP DDoS attacks, while KNN excels in multi-class classification with 99% accuracy. An ensemble IDS model combining logistic regression, NB, and DT with a voting classifier is presented in [15]. Evaluated on CICIDS2017 dataset, it achieved 88.96% accuracy for multi-class classification and reached 88.92% overall accuracy for binary classification. The research [16] focused on

choosing the right hidden layers and neurons to prevent overfitting for ANN and compared with Random Forest using the N-BaIoT dataset for detecting Mirai malware in IoT devices. Results showed 92.8% accuracy, 0.3% False Negative rate, and an F-1 score of 0.99. Major findings contribute to cost-effective solutions for detecting Mirai malware strains. A Collaborative IDS that combines network and host data utilizing TabNet, named CIDS-Net for improved performance is proposed in [17]. SCVIC-CIDS-2021 derived from CIC-IDS-2018 is used for experiments. CIDS-Net improves IDS performance by achieving 99.98% F-Score but the paper doesn't describe accuracy or any other performance metrics. A lightweight NIDS specifically designed for IoT gateways is depicted in [18] utilizing TabNet, a model developed by Google for handling tabular data. Evaluation results from BoT-IoT and UNSW-NB15 datasets demonstrate the effectiveness of the proposed system, achieving high accuracy rates of 98.53% and 97.95% in intrusion detection tasks. A hybrid deep learning model [19] combining LSTM and CNN was built to detect zero-day attacks in IoT networks. Real-time zero-day attack data was curated for the training process after identifying the top 12 features via Explainable AI. Time series GAN (TGAN) was employed to generate additional attack samples. By merging this data with the AWID dataset, the hybrid model achieved 93.53% accuracy, surpassing the 84.29% accuracy on the AWID dataset alone. Another study employs XGBoost-based feature selection algorithm with LSTM, GRU, Simple RNN to enhance detection performance on NSL-KDD and UNSW-NB15 datasets [20]. Results show significant improvements in test accuracy, with XGBoost-LSTM achieving 88.13% for binary classification on NSL-KDD and XGBoost-Simple-RNN achieved 87.07% for UNSW-NB15. For multiclass classification, XGBoost-LSTM outperforms others on NSL-KDD i.e. 86.93%, while XGBoost-GRU excels on UNSW-NB15 which is 78.40%. Overall, the proposed IDS framework performs better than existing methods. Researchers work have attempted to achieve a sophisticated IDS by implementing machine learning in large-scale IoT environments [21]. Different feature extraction techniques were investigated, involving image filters and transfer learning models like VGG-16 and DenseNet, complemented by different ML models such as RF, K-nearest neighbors, SVC, and stacked models for classification. The research was performed using the IEEE Dataport dataset and a top accuracy rate of 98.3 percent was achieved using the combination of VGG-16 with Stack. The novel model called TabNet-IDS [22] utilizes attentive mechanisms to automatically select important features from datasets. This model aims to enhance the explainability of the results while effectively training the IDS . It is implemented using the TabNet algorithm within the PyTorch DL framework. The outcomes demonstrate high accuracy on various tabular datasets related to IoT security, with reported accuracies of 97%, 95%, 98% for CIC-IDS2017, CSE-CICIDS2018 and CIC-DDoS2019 dataset. LSTM-based IDS with Dynamic Access Control (DAC) is introduced in [23]. It achieves 97.16% validation accuracy in detecting 14 different threats. It demonstrated 98% detection rate and fast 1.2 second response time, making it a highly accurate and efficient IDS for enhancing IoT security. A study proposes an intelligent framework to enhance cybersecurity in Industry 4.0 WSNs using MLP, DT and Autoencoder models for intrusion detection and threat prioritization [24]. The framework achieves high accuracy of 99.5% for MLP and DT, 91% for Autoencoder and outperforms benchmark models, offering a proactive and effective approach to prevent and mitigate cybersecurity threats by prioritizing and addressing high-risk intrusions. The study [25] evaluates feature selection and extraction

techniques for ML-based intrusion detection in IoT networks using the diverse ToN-IoT dataset. Feature extraction yielded better performance with smaller feature sets, whereas feature selection excelled in reducing training and inference times.

Ekundayo [26] leveraged Optuna to fine-tune the CNN-LSTM for predicting various hyperparameters related to consumption of electric energy, resulting in reduced mean square error than traditional methods. The paper [27] compares four Python hyperparameter optimization libraries—Optuna, HyperOpt, Optunity, and SMAC—across two benchmarks: a CASH problem and the NeurIPS MLP optimization challenge. Optuna excelled in the CASH task, while HyperOpt performed best for MLP selection. Srinivas and Katarya [28] applied Optuna to optimize the hyperparameters of XGBoost for predicting cardiovascular diseases, achieving 94.7% accuracy on the Cleveland dataset. Li et al. [29] employed Optuna to tune the hyperparameters influencing the accuracy of LSTM model accuracy in their research on drilling pressure prediction systems. Consequently, it demonstrated a reduction in Mean Squared Error (MSE) on both the training and test datasets. Researchers [30] presented an advanced solution using BILSTM with Optuna for hyperparameter optimization, leading to a significant 7% accuracy improvement over traditional models. This advancement enhances ship navigation safety and efficiency, with implications for autonomous collision avoidance systems and maritime traffic management. ASXAML, an anti-money laundering framework using XGBoost, which aims to reduce false positives proposed in [31]. It combines feature selection with Optuna for hyperparameter tuning, achieving 86% F-beta score with just 11% of money laundering cases misclassified out of 1926 in the test data.

Table 1. Related word for IDS using ML & DL techniques

Authors Name	**Year**	**Methodology Used**	**Dataset Used**	**Findings**	**Accuracy (%)**	**Limitations**
Ibitoye et. al. [8]	2019	FNN and SNN	Bot-IoT	Average accuracy with precision, recall, and F1-score all reaching 95%, showcasing its impressive performance.	95.10	Feature normalization of Bot-Io indicates that if applied, accuracy could plummet to below 50%.
Ge et. al. [9]	2019	FNN	Bot-IoT	DDoS/DoS attacks attained an accuracy of 99.414%. Across all evaluation metrics the model consistently achieved a remarkable 99% performance.	99.41	Binary classification efficiency decreased in defending against keylogging attacks and mitigating information theft.
Susilo et. al. [10]	2020	CNN	Bot-IoT	Highest accuracy is 91.27% using 128 batch size and the lowest accuracy 88.30% using 32 batch size.	91.27	Accuracy drops to 88.30% using 32 batch size=32 and 90.64% using batch size=64.

Alkadi et. al. [11]	2020	Smart contracts with BiLSTM for classification	UNSW-NB15 and BoT-IoT	DBF effectively detects insider and outsider attacks in the cloud and IoT systems.	98.91 & 99.41	performance deteriorates under high network traffic and struggles to effectively detect complex attacks.
Parra et. al. [12]	2020	Distributed CNN for IoT devices and LSTM for cloud hosts	N_BaIoT	LSTM implemented on client devices maps CNN sections, enabling detection and defense at the point of attack origin.	97.84	lacks the capability to detect emerging attacks
Samy et. al. 13]	2020	DNN, LSTM, BiLSTM, GRU, CNN, CNNLSTM	N_BaIoT-2018, CICIDS-2017, NSL-KDD	It deploys detectors on fog nodes for efficiency and evaluates six DL models	99	The suggested model requires large datasets and takes more time to train.
Churcher et. al. [14]	2021	KNN, SVM, DT, NB, RF, ANN, LR	Bot-IoT	For binary classification of attacks, RF performed best, achieving 99% accuracy on HTTP DDoS attacks and for multi-class classification KNN outperformed other methods with 99% accuracy.	99	Lack of diverse data sources containing a mix of different types of attacks
Abbas et. al. [15]	2021	LR, NB, and DT with voting classifier	CICIDS 2017	The ensemble model improves multi-class accuracy to 88.96% by combining DT with NB and LR and 88.92% accuracy for binary classification	88.96	the need for data from real-time networks
Palla et. al. [16]	2021	ANN	N-BaIoT	ANN out performs with 92.8% accuracy, low False Negative rate (0.3%), high F-1 score (0.99), and cost-effective solutions for detecting Mirai malware strains.	92.8	Comparative analysis limited to ANN and Random Forest models.

Liu et. al. [17]	2022	TabNet	SCVISC-CIDS-2021	CIDS-Net deep learning model for network intrusion classification with host features.	-	Accuracy not given but TabNet achieves F-Score of 99.89%
Nguyen et. al. [18]	2022	TabNet	BOT-IoT and UNSW-NB1	A lightweight NIDS using TabNet model for IoT gateways.	98.53 & 97.95	Limited Testing on Diverse Platforms
Asaduzzaman et. al. [19]	2022	CNN-LSTM	AWID-GAN	LSTM-CNN deep learning model detects zero-day attacks in IoT networks with 93.53% accuracy, surpassing baseline results on the AWID dataset.	93.53	limited number of real-time attacks were conducted
S. M. Kasongo [20]	2023	XGBoost for feature selection and LSTM, GRU, Simple RNN	NSL-KDD, UNSW-NB15	Achieves better accuracy for binary classification than multi class classification	88.13 87.07	Training time is more and performance decreases on minority classes
Musleh et. al. [21]	2023	Feature extractors: VGG-16 and DenseNet. RF, KNN, SVM, and different stacked models	IEEE Dataport	This study examined various feature extractors and machine learning algorithms on IEEE Dataport dataset. Combining VGG-16 with stacking achieved the highest accuracy	98.3	Difficulty in distinguishing threats
Zegarra et. al. [22]	2023	TabNet	CIC-IDS2017, 2018, 2019	TabNet-IDS, utilizes attentive mechanisms for automatic feature selection in IoT intrusion detection.	97, 95.58, 98.51	Challenge of model explainability
Alazab et. al. [23]	2024	LSTM with Dynamic Access Control algorithm	CICIDS 2017 & BoT-IoT	LSTM-based IDS with Dynamic Access Control that achieves 97.16% validation accuracy in detecting 14 different threats	98.73	high implementation cost using Raspberry Pi, lack of testing against adversarial machine learning attacks
Al-Quayed et. al. [24]	2024	Decision Tree, MLP, and Autoencoder, LR	WSN-DS	The framework achieves high accuracy of 99.5% for Decision Tree and MLP, 91% for Autoencoder	99.52	specific domain of WSNs in Industry 4.0

Li et. al. [25]	2024	DT, RF, KNN, NB, MLP with feature selection & extraction techniques	ToN-IoT	Feature extraction outperformed feature selection in detection performance	88.22 is highest accuracy	accuracy, F1-score, and runtime need to be enhanced

3. Materials and Methods

3.1 The Proposed Framework

This article presents a methodology for IDS that uses DL approaches to discover and categorize potential vulnerabilities and threats in Internet of Things (IoT) networks. First phase involves gathering the essential data required for constructing the models. In this study NSL KDD and BoTNeTIoT-L01 datasets are selected for experimentation. The collected data then goes through preprocessing, which includes data cleaning, selection, and normalization. This refined data is divided into training and testing sets to feed into the learning model. The system employs LSTM and TabNet DL models which are trained on the processed data to identify patterns that differentiate normal network behavior from potential attacks. To optimize the performance of these models, the pipeline incorporates OPTUNA, a hyperparameter optimization framework. This step fine-tunes the model parameters to achieve the best possible performance. Once trained and optimized, the model produces outcomes that are then classified into two categories: normal traffic or attack. The performance of this classification is calculated through various metrics. This evaluation feeds back into the optimization process, allowing for iterative improvement of the model. The different phases of the proposed framework is represented in Figure 1.

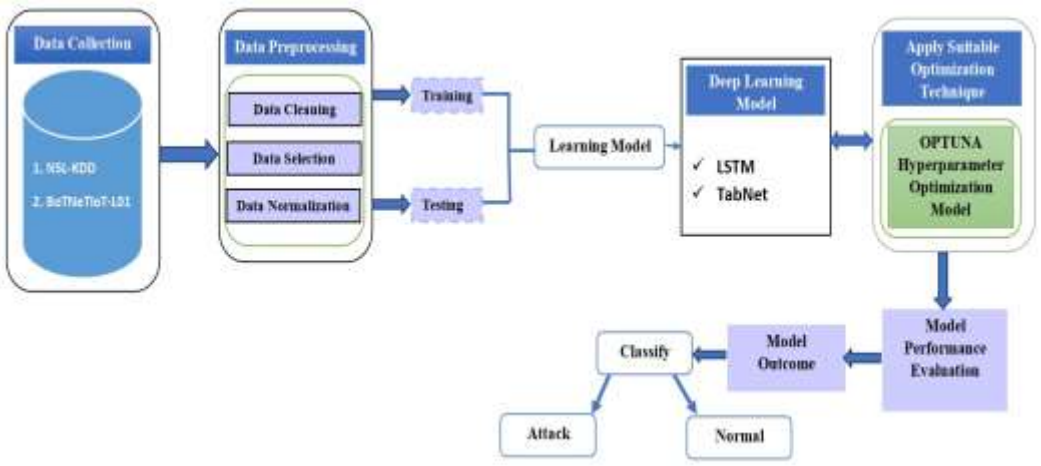


Figure 1. The Proposed Framework

3.2 Long Short-Term Memory (LSTM)

LSTM addresses the vanishing gradient issue commonly encountered in training traditional RNNs. It introduces a memory cell with key components, including input, forget, and output gates, along with a cell state [32]. The cell state acts as the network's memory, enabling information to be retained and transmitted across long sequences. Each gate (input, forget, and output) is implemented using sigmoid layers, which control the flow of information into and out of the memory cell.

Forget Gate: It examines the input data and the information from the previously hidden layer. LSTM then determines which information to forget from the cell state using a sigmoid function. It is calculated below in Equation 1.

$$f_a = \sigma(W_{fg} \,.\, [h_{t-1,} x_t] + b_{fg}) \quad (1)$$

Input/Update Gate: The LSTM decides which information to retain in the cell state. Initially, the input gate layer determines which information to update using a sigmoid function. Subsequently, a Tanh layer suggests a new vector to incorporate into the cell state. The LSTM then updates the cell state by discarding the information marked as forget and incorporating the new vector values. It is calculated below in Equation 2 and 3.

$$i_t = \sigma(W_{ig} [h_{t-1}, x_t] + b_{ig}) \quad (2)$$

$$c_t = \tanh(W_{cs}[h_{t-1,} x_t] + b_{cs}) \quad (3)$$

Output Gate: Determines the output by employing a sigmoid function to specify the segment of the LSTM cell to output. The outcome undergoes a Tanh layer transformation (with values ranging between -1 and 1) to transmit solely the chosen information to the succeeding neuron. It is calculated below in Equation 4 and 5.

$$o_t = \sigma(W_{og} [h_{t-1,} x_t] + b_{og}) \quad (4)$$

$$h_t = o_{tl} * \tanh(c_{tl}) \quad (5)$$

3.3 Attentive Interpretable Tabular Learning (TabNet)

TabNet is an advanced DL architecture explicitly designed for tabular data, making it highly suitable for tasks like network intrusion detection. It combines the strength of neural networks with the interpretability of decision trees, offering a unique approach to handling structured data [33]

At its core, TabNet employs a sequential attention mechanism that processes data through multiple decision steps. Each step focuses on different features, allowing to capture intricate relationships within the data. The key innovation lies in its feature selection process, governed by the equation given in Equation 6.

$$M[a] = \text{sparsemax}(P[a] \odot M[a\text{-}1]) \quad (6)$$

Here, M[a] signifies the feature mask at step i, P[a] denotes the feature selection parameters, and ⊙ signifies element-wise multiplication. The sparsemax function ensures sparse feature selection, promoting interpretability and efficiency.

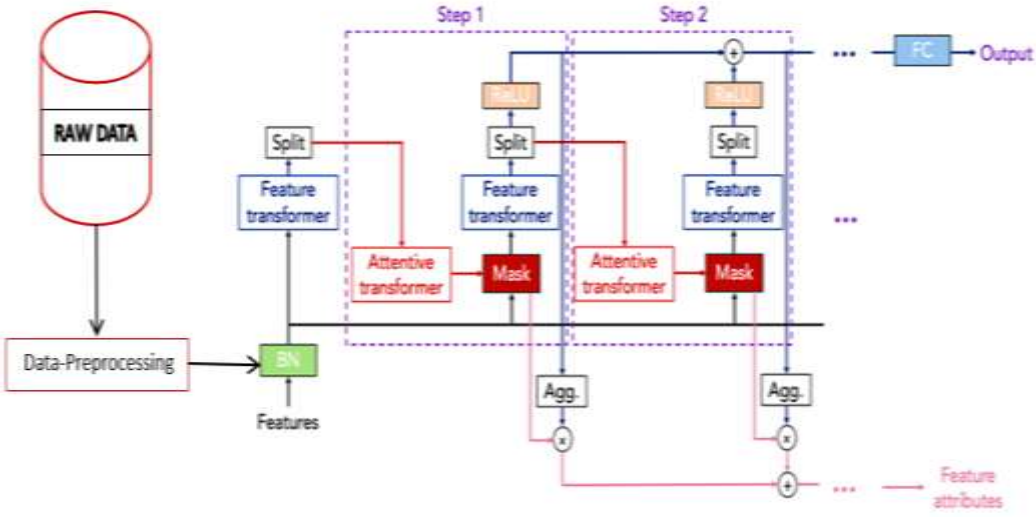


Figure 2. Architecture of TabNet

TabNet's architecture as shown in Figure 2 [34] consists of several components:

- Feature Transformer: Applies learnable transformations to input features.
- Attentive Transformer: Uses the mask M[a] to select relevant features.
- Feature Selection Network: Determines which features to focus on in each step.

The model iterates through these components for a predefined number of steps, refining its feature selection and decision-making process. For classification tasks, the final output is typically processed through a softmax layer as given in Equation 7.

$$y\_pred = softmax(W * h + b) \quad (7)$$

where W is the weight matrix, h is the final hidden representation and b is the bias.

TabNet excels in classification tasks for several reasons:

- Adaptive Feature Selection: It can identify and emphasize on the relevant features for each specific instance, enhancing its ability to capture nuanced patterns.
- Interpretability: The feature selection process provides insights into which features are most important for each decision, making the model's reasoning more transparent.
- Handling Mixed Data Types: TabNet efficiently processes both numerical and categorical data without extensive preprocessing.
- Balancing Complexity and Performance: Its architecture allows for deep learning capabilities while maintaining computational efficiency.

- Regularization: TabNet incorporates built-in regularization techniques, reducing overfitting and improving generalization.

TabNet can effectively learn complex patterns in network traffic data. It can distinguish between normal and malicious activities with high accuracy while providing valuable insights into which network characteristics are most indicative of potential threats. This makes TabNet particularly powerful for creating robust, explainable intrusion detection systems that can adapt to evolving cyber threats. Its interpretability feature also aids security analysts in understanding and validating the model's decisions, which is essential in critical security applications.

3.4 OPTUNA

Optuna is a powerful open-source framework for hyperparameter optimization specially designed for DL models [35]. Optuna offers several key advantages: firstly, a user-friendly define-by-run style API; secondly, an effective pruning and sampling mechanism; and thirdly, simple setup procedures. Optuna's define-by-run API permits users to dynamically build hyperparameter search spaces through an objective function during runtime, thus offering flexible, real-time optimization without pre-defining all elements. The cost-effectiveness of hyperparameter optimization relies on efficient searching and performance estimation strategies to select and evaluate parameters effectively. Optuna offers both independent sampling and relational sampling methods for hyperparameter optimization [36]. While independent sampling, like tree-structured Parzen estimator (TPE), can perform well without using parameter correlations, the effectiveness of each method varies based on the environment and task. Optuna can handle various independent sampling methods and relational sampling methods like covariance matrix adaptation evolution strategy (CMA-ES). Relational sampling in define-by-run frameworks implementation can be challenging, but Optuna can identify trial results indicative of parameter correlations, enabling the use of relational sampling algorithms like CMA-ES and GP-BO as optimization progresses. Specifically, Optuna permits customized sampling procedures.

Pruning involves monitoring intermediate objective values and terminating trials that do not meet criteria. Optuna uses the Asynchronous Successive Halving (ASHA) algorithm, which allows for independent early stopping in a distributed environment, leading to efficient parallel processing and linear scalability. The pruning process is governed by the 'report API' for monitoring and 'should_prune API' for premature termination, ensuring optimal resource utilization. Optuna's last design feature is a simple setup process that handles tasks from heavy experiments to lightweight computations. It offers flexible storage options, excels in lightweight tasks, supports in-depth analysis, and includes a web dashboard for real-time visualization. Optuna simplifies storage deployment, integrates seamlessly with container-orchestration systems, and is easy to install. It's open-source and scales linearly for distributed computations. Optuna workflow for hyperparameter optimization is given below.

- Design the objective function: Create a function that takes hyperparameters as input and returns a performance metric for your model.Specify the hyperparameter search space: Determine the types and ranges of hyperparameters you want to optimize.

- Initialize an Optuna study: Set up a new study object for your specific optimization task.
- Execute the optimization process: Use the study's optimization method, providing your objective function and the desired number of trials.
- Retrieve optimization results: Extract the best-performing hyperparameter set and its corresponding performance metric.
- Finalize model training: Use the optimal hyperparameters to train your model for deployment or further evaluation.

4. Dataset Description

In this study, NSL-KDD and BoTNeTIoT-L01 datasets are taken into consideration.

4.1 NSL KDD Dataset

The NSL-KDD dataset [37] is an refined version of the original KDD Cup 1999 dataset, which is extensively used for NIDS research. It addresses the redundancy and imbalance issues found in the KDD dataset by eliminating duplicate records and providing a balanced representation of normal and attack types. Each instance in the dataset represents a network connection, described by 41 features that include basic attributes like protocol type, service, and flags, as well as derived attributes such as counts of packets and connections. The dataset is identified as either normal or one of four main attack types: Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R) and Probe. This dataset is often used for benchmarking IDS because of its cleaner and more representative structure. The attack traffic distribution is outlined in Table 2.

Table 2. Attack types of NSL KDD

Category	**Attack Type**
DoS	Apache2, Teardrop, Land, Back, Pod, Mailbomb Neptune, Edstrom, Process table, Smurf, Worm
R2L	Snmpgetattack, Multihop Guess_password, Phf, Named, imap, Xlock, Sendmail, Ftp_write, Httptunnel, Snmpguess, Waremaster, Xsnoop.
U2R	Rootkit, Buffer_overflow, Perl, Ps, Xterm, Loadmodule, Sqiattack,
Probe	Nmap, Satan, IPsweep, Saint, Portsweep, Mscan

4.2 BoTNeTIoT-L01 dataset

The BoTNeTIoT-L01 dataset is a recent and comprehensive collection of IoT network traffic designed for intrusion detection research, particularly focusing on botnet attacks. Captured in a smart home testbed at the University of New South Wales Cyber Range Lab, the dataset

includes network traffic from nine IoT devices such as cameras, door locks, bulbs, switches, and remotes, all connected via a central switch [38]. The traffic was recorded using Wireshark and contains two well-known botnet attacks i.e. Mirai and Gafgyt, along with their subdivisions as described in Table 3. A key feature of BoTNeTIoT-L01 is its inclusion of 23 statistically engineered features, derived from .pcap files. Seven statistical measurements were used to produce these features: the mean, variance, covariance, count, magnitude, radius, and correlation coefficient. The calculations were performed accross a 10-second time window, employing a decay factor of 0.1 (referred to as L0.1 in related research). The dataset contains 2,426,574 instances with a binary target label indicating normal traffic (0) or attack traffic (1). The processed data is divided into 80% training 80% and 20% testing sets for subsequent analysis and model development.

Table 3. Attack types of BoTNeTIoT-L01

Category	**Attack Types**
Mirai	UDP flooding, Ack flooding, UDP plain, Scan, Syn flooding
Gafgyt	Junk, Como, TCP, UDP flooding

4.3 Data Preprocessing

Data preprocessing is a key procedure that converts raw data into a refined, uniform, and significant format suitable for analysis. This process is crucial for ensuring data reliability and suitability for classification models, especially in IoT IDS security [39].

Step-1 Data Cleaning: The dataset contains both duplicate records, which have been eliminated, and missing values, which have been addressed through either imputation or the removal of incomplete records. This process ensures data integrity and improves the quality of the dataset for analysis purposes.

Step-2 Data Selection: The dataset contains a mix of character and numeric attributes. To ensure consistency, label encoding is used for multi-class labels while one-hot encoding is applied to categorical columns. Additionally, LabelBinarizer is utilized for one-hot encoding of categorical labels, converting them into binary arrays suitable for deep learning algorithms.

Step-3 Data Normalization: It is employed to prevent bias towards features with larger values by maintaining their scale. In this study normalization is opted, specifically min-max scaling, to standardize features. The dataset is normalized between 0 and 1 using Equation 8

$$f_{normalized} = \frac{f-min}{max-min} \quad (8)$$

where f = feature value, max = maximum value, min = minimum value of the feature.

5. Experiments and Result Analysis

The constructed models are thoroughly examined and juxtaposed using the below key evaluation criteria as outlined in Equations 9 through 12 correspondingly.

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (9)$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP+FP})} \quad (10)$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP+FN})} \quad (11)$$

$$\text{F-score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (12)$$

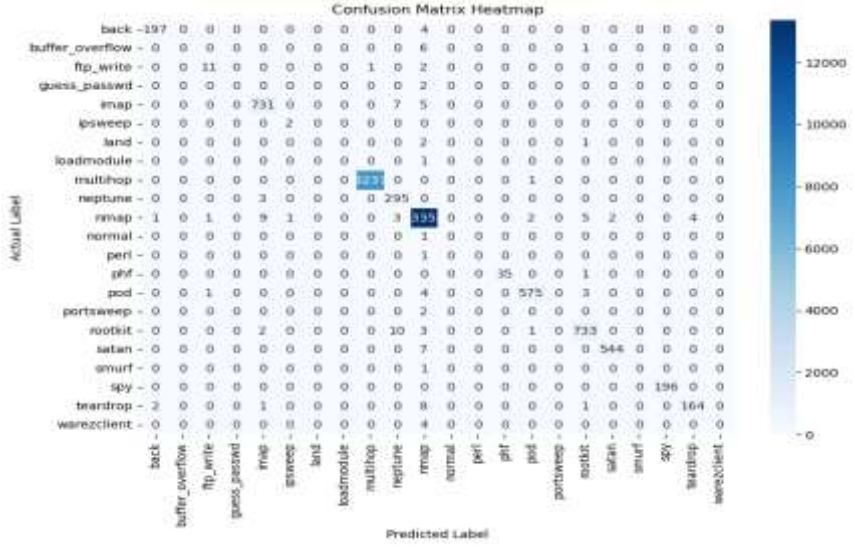
Here the experiments are divided into two fundamental design models. The LSTMIDS model and Optuna based LSTMIDS model are built and analyzed in phase 1. Further TabNetIDS and Optuna based TabNetIDS models are developed and analyzed in phase 2.

Experiments are carried out using Jupyter Notebook, an interactive development environment for Python, which is part of the Anaconda distribution. This platform is chosen for its effectiveness in implementing and assessing the proposed methodology. For the deep learning aspects of the study, TensorFlow 2 and Keras 3.3 are employed [40]. These Python-based frameworks are instrumental in building and training the models. The computational work is performed on a high-performance laptop with the following specifications: Intel Core i9-10900K Processor, 128 GB of RAM, NVIDIA RTX 3060 Ti GPU (with cuDNN support) Graphics and 64-bit Windows 11 Operating System. The programming language used throughout the study is Python 3.11. For comprehensive data analysis, pandas-profiling version 3.6.6 is utilized, an open-source Python module known for its robust data exploration capabilities.

5.1 Phase 1: Result Analysis using LSTM

5.1.1 LSTM Model for NIDS

This section presents an experimental analysis of LSTM model for NIDS, applying the benchmark and widely used NSL-KDD dataset.



.Figure 3. Confusion Matrix Heatmap for NSL-KDD Dataset

The above Figure 3 displays a confusion matrix heatmap for the NSL-KDD dataset. It envisages the performance of a classification model by comparing predicted labels against actual labels. The matrix includes several network attack types. The diagonal elements of the matrix have higher values, indicating that the model performs well in correctly classifying many instances. For example: 'normal' class has 9711 correct predictions, 'neptune' has 4657 correct predictions, 'satan' has 3633 correct predictions. There are some off-diagonal elements showing misclassifications. For instance:Some 'back' attacks (197) are misclassified as 'neptune', a few 'buffer_overflow' instances were misclassified as 'normal'. Some attack types appear to be very rare or not well-predicted, with mostly zeros in their rows and columns (e.g., spy, perl, multihop). The 'normal' class seems to be the most frequent, followed by 'neptune' and 'satan', while many other attack types have much lower frequencies. The heatmap uses a blue color gradient, with darker blue indicating higher numbers of instances.

Table 4. Hyperparameter of LSTM approach.

S.L. No.	**LSTM Model Hyperparameters**
1	LSTM Unit = 50
2	Dropout_rate = 0.2
3	Return_sequence = True
4	Dense layer = 1
5	Activation function =softmax
6	Optimizer = adam

The hyperparameters employed in the LSTM model is outlined in Table 4. The model utilizes a single LSTM layer with 50 hidden units, which are responsible for capturing complex temporal patterns in the network data. To thwart overfitting, a dropout rate of 0.2 is implemented, randomly deactivating 20% of neurons while training. The 'Return_sequence = True' setting allows the model to analyze the entire sequence of hidden states at each time step, crucial for understanding long-term dependencies in network traffic. A single dense layer including activation function as softmax is applied to map the learned features from the LSTM layer to the final output, which is a probability distribution over the different attack categories. Model tuning is done by Adam optimizer, an adaptive learning rate algorithm that helps accelerate training and improve model performance.

Table 5. Performance of LSTM model on NSL-KDD

Loss	**Accuracy**	**Precision**	**Recall**
0.049453739076	0.984210908412	0.985450983047	0.982796907424

The efficacy of the trained LSTM model for NSL-KDD dataset is presented below in Table 5. The model achieves a low loss value of 0.0494, indicating that it has effectively minimized errors during training. It demonstrates high accuracy (0.9842), correctly classifying 98.42% of the instances, emphasizing the model's outstanding efficiency in classifying between normal and malicious traffic. The precision of 0.9854 suggests a low false-positive rate, meaning it

rarely misclassifies normal traffic as an attack. Similarly, the recall of 0.9828 indicates a low false-negative rate, effectively identifying most actual attacks.

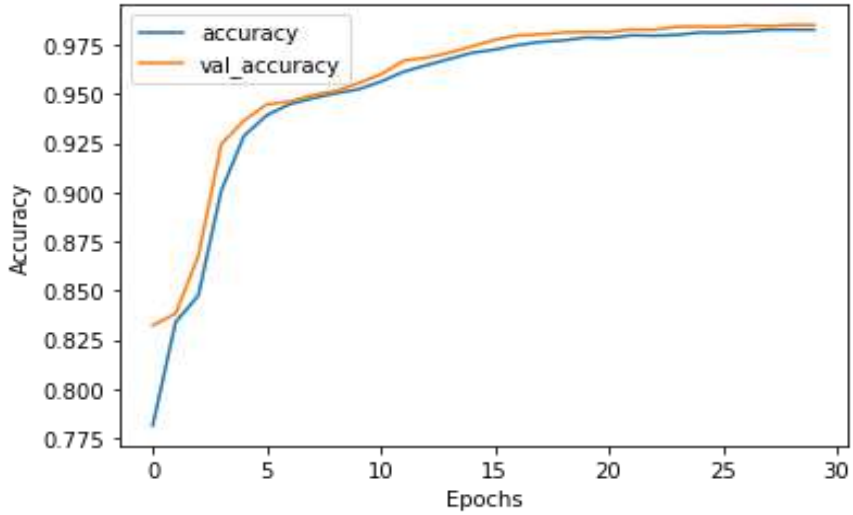
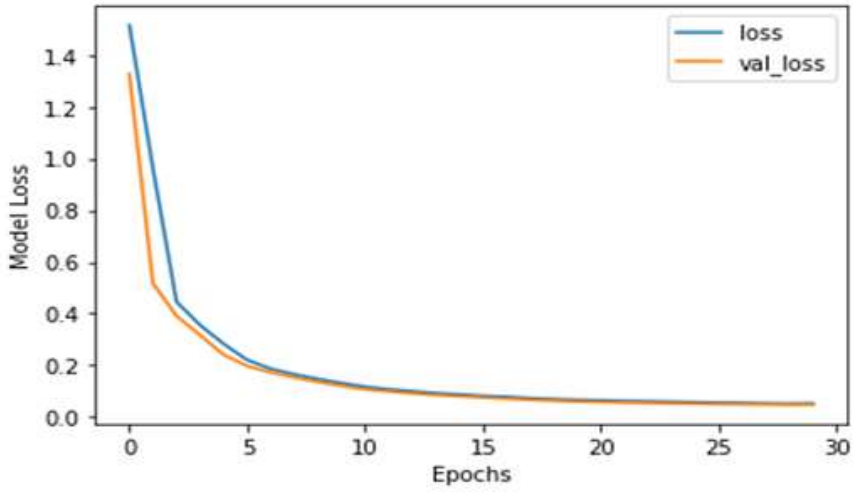


Figure. 4: Plot of accuracy vs epoch for LSTM train and test dataset

The accuracy of LSTM model over successive epochs for training and testing datasets is given in Figure 4. Initially, the training accuracy (depicted by the blue line) rises steeply, indicating that the quick learning patterns of the model are from the training data. Parallel to this, the validation accuracy (shown by the orange line) also rises sharply, closely following the training accuracy. This close alignment signifies that the model's outcomes on unknown data is improving in tandem with its training performance, which is a positive indication of generalization.

As epochs progress from 0 to 30, both lines begin to plateau, reaching a point where increases in accuracy are marginal. Overall, the alignment and convergence of these two

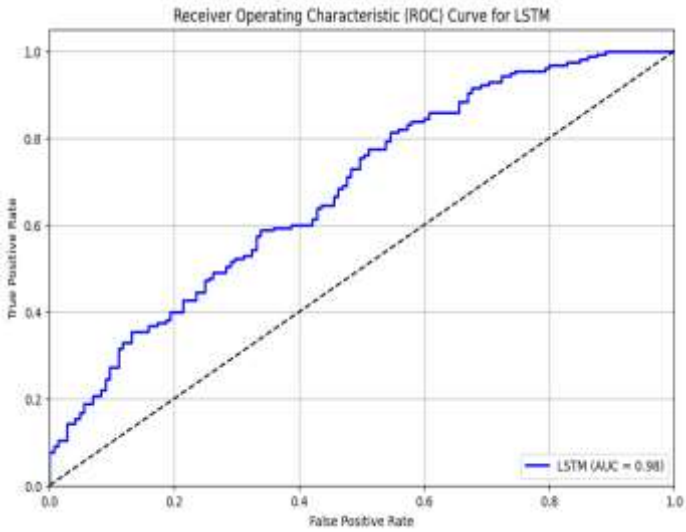


metrics over the epochs demonstrate that the LSTM model is successfully capturing the data's underlying patterns achieving a balance between bias and variance.

Figure 5. Plot of loss vs epoch for LSTM training and test dataset

Figure 5 depicts the loss of an LSTM model over successive epochs for both training and validation datasets. Initially, the training loss (blue line) and validation loss (orange line) decreased sharply. This rapid decline suggests that the model quickly learns to reduce the discrepancy among its predictions and the actual outcomes, fitting the training data effectively. As the epochs progress, the rate of reduction in both training and validation losses slows down, forming an asymptotic curve that gradually levels off. This slowdown indicates that the model is approaching its optimal performance, having captured the essential patterns in the data while reaching its learning capacity. The close alignment between the training and validation loss implies that the model is generalizing well to unseen data.

Figure 6**.** ROC curve for classification accuracy of LSTM model



The ROC (Receiver Operating Characteristic) curve illustrates the trade-off between the true positive rate (sensitivity) on the y-axis and the false positive rate on the x-axis at various threshold settings as given in Figure 6. The model's AUC of 0.98, with a curve close to the top left corner, indicates excellent classification performance with a high true positive rate and low false positive rate.

5.1.2 Optuna Hyperparameters Tuning Method for LSTM

Optuna is a framework for automated hyperparameter optimization. It significantly simplifies the process of finding optimal hyperparameters for LSTM models by automatically exploring a wide range of values. This approach reduces the number of trials needed, lowers computational costs, and improves model performance, allowing faster development with minimal manual effort.

Algorithm:

Step 1. Define the Objective Function for Optuna

Step 1.1: Create an objective function that Optuna will use to optimize the hyperparameters.

Step 2. Suggest Hyperparameters

Step 2.1:`units`: Number of LSTM units, suggested as an integer between 50 and 200.

Step 2.2: `dropout_rate`: Dropout rate, suggested as a uniform float between 0.1 and 0.5.

Step 2.3: `learning_rate`: Learning rate, suggested as a logarithmic uniform float between 1e-5 and 1e-1.

Step 3. Build the LSTM Model

Step 3.1: Add an LSTM layer with the suggested number of units and `return_sequences=True`.

Step 3.2: Add a dropout layer with the suggested dropout rate.

Step 3.3: Add another LSTM layer with the same number of units.

Step 3.4: Add another dropout layer.

Step 3.5: Add a dense layer using sigmoid activation function.

Step 4. Compile the Model

Step 4.1: Use the Adam optimizer with the suggested learning rate.

Step 4.2: Use `binary_crossentropy` as loss function and `accuracy` as metric.

Step 5. Train the Model

Step 5.1: Train the model with a batch size of 64 for 10 epochs on the training data.

Step 5.2: Utilize 20% of the training data for validation.

Step 6. Evaluate the Model

Step 6.1: Evaluate the model on the test data and return the accuracy.

Step 7. Create and Optimize the Optuna Study

Step 7.1: Create an Optuna study to maximize the objective function.

Step 7.2: Run the optimization process for 50 trials.

Step 8. Output the Best Hyperparameters and Accuracy

Step 8.1 Print the best hyperparameters learned by the Optuna.

Step 8.2 Print the best accuracy achieved by the model during the study.

Step 9. Exit

Optuna Model Hyper Parameters:

```
Number of finished trials: 201
Best trial:
  Value: 0.595
  Params:
    max_depth: 7
    learning_rate: 0.9980673832300351
    n_estimators: 284
    min_child_weight: 8
    gamma: 0.149040762182943
    subsample: 0.09102154634335183
    colsample_bytree: 0.013470551011872159
    reg_alpha: 0.1543583837166755
    reg_lambda: 0.055126817319760955
```

Best hyperparameters**:** {'units': 145, 'dropout_rate': 0.19983772432277946, 'learning_rate': 0.014379399241973167}

Here Optuna has explored 201 combinations of hyperparameter values i.e. trials to discover the excellent set for the LSTM model. This large number of trials indicates a thorough search for the optimal configuration. For this LSTM implementation, the best hyperparameters identified are 145 units, a dropout rate of approximately 0.20, and a learning rate of 0.014. The "units" parameter refers to the neuron numbers in the LSTM layers, which influence the model's ability to capture temporal dependencies in sequential data. The dropout rate prevents overfitting by randomly setting fraction of input units to zero during training, enhancing the model's ability to generalize to new data. 0.014 of learning rate signifies a modest step size during optimization, which balances the speed and stability of convergence to an optimal solution. Achieving a best accuracy of 0.98 indicates that the model accurately classifies instances in the dataset, reflecting the success of the hyperparameter tuning process in enhancing model robustness and performance. This optimized setup underscores the importance of tuning for leveraging the full potential of the LSTM model, showcasing Optuna's role in streamlining and improving the development process.

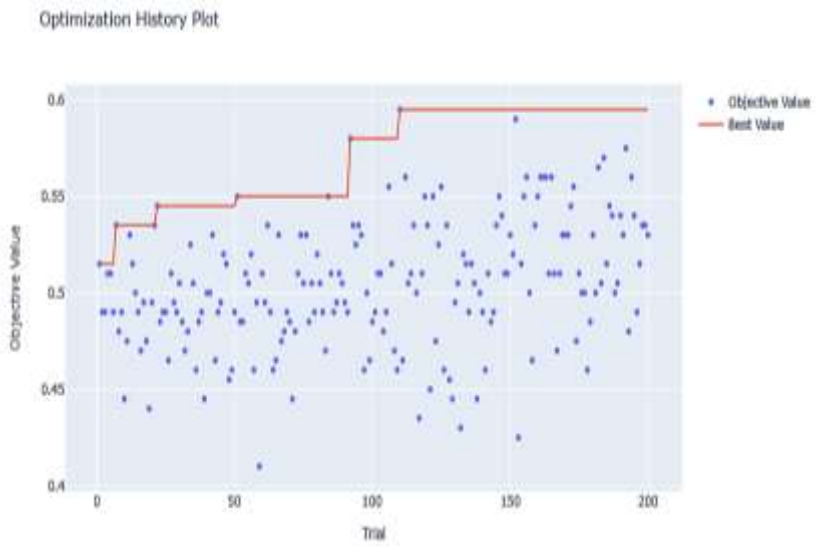


Figure 7. Optuna History plot hyperparameters for LSTM using NSL KDD

The Optuna Optimization History Plot visualizes the progression of the hyperparameter optimization process across multiple trials. The x-axis represents the trial number, indicating each step in the tuning process, while the y-axis displays the objective value being optimized, such as validation accuracy or loss as illustrated in Figure 7.

Blue dots depict the objective values for each trial, showcasing the diverse performance outcomes from different hyperparameter settings. The scattered nature signifies Optuna's exploration of the search space. A red line traces the best value achieved over time, indicating performance improvements as Optuna identifies superior hyperparameter combinations. The plot highlights how Optuna effectively navigates and exploits the search space, steadily enhancing model performance through 201 trails.

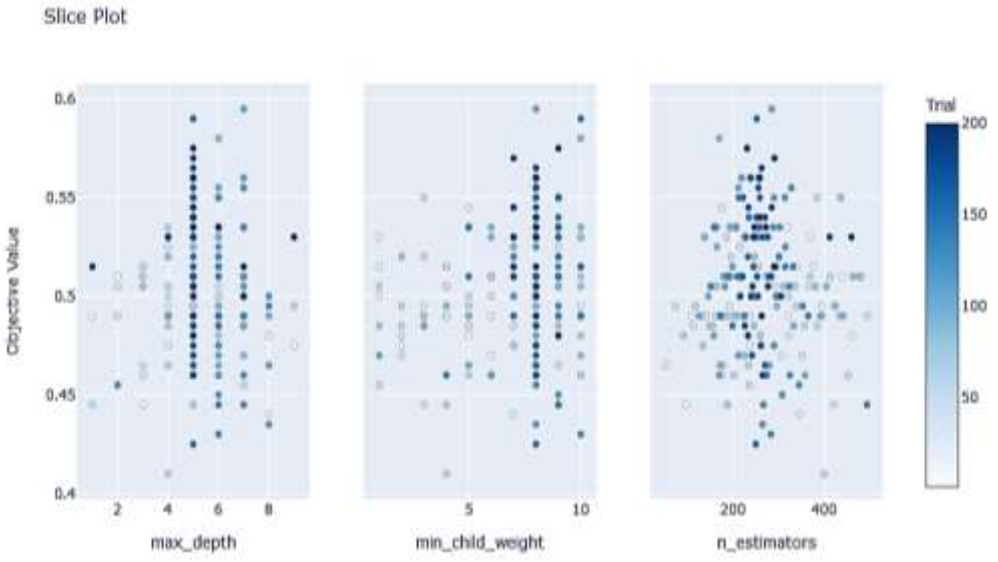


Figure 8. Objective value of hyper parameter during optimization for LSTM using optuna

The Slice Plot as depicted in Figure 8 visualizes the impact of three hyperparameters: max depth, min child weight, and number of estimators on the objective value across trials. Each subplot illustrates how changes in these parameters affect performance, with dots representing individual trials colored by trial number. This plot helps identify trends and optimal ranges for each hyperparameter, revealing which settings may achieve better results during optimization.

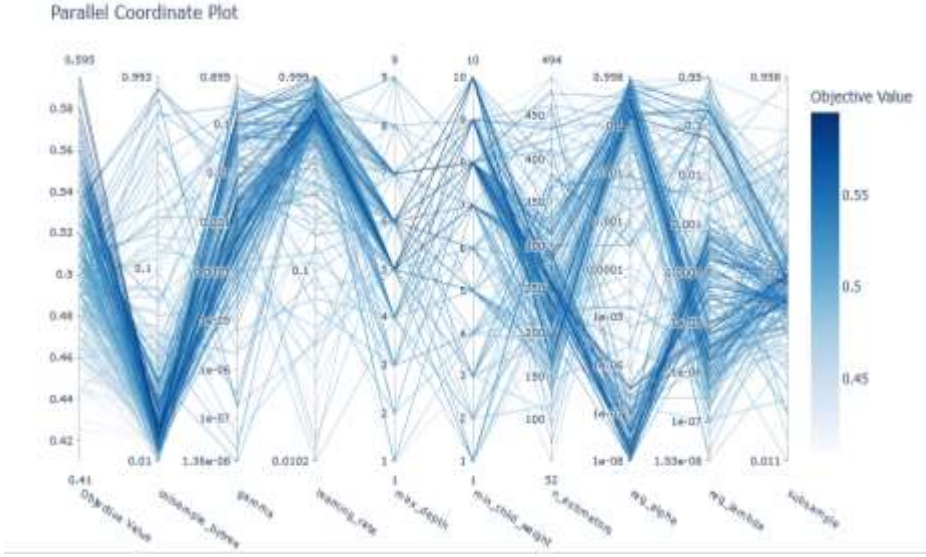


Figure 9. Optuna Parallel Coordinate plot hyperparameters for LSTM using NSL KDD

The Parallel Coordinate Plot illustrates the interplay between various hyperparameters and their collective impact on the objective value as shown in Figure 9. Each line in the plot correspond to a single trial, connecting specific parameter settings across multiple axes, which include colsample_bytree, subsample, gamma, and others. The color intensity of these lines, ranging from light to dark blue, reflects the associated objective value, with darker lines indicating better performance. This visualization enables a clear comparison of how different combinations of hyperparameters influence model efficacy, revealing complex interdependencies that can guide the selection of optimal settings for improved overall performance. It effectively highlights which parameter configurations tend to yield the highest objective values, offering insights into potential paths for further optimization.

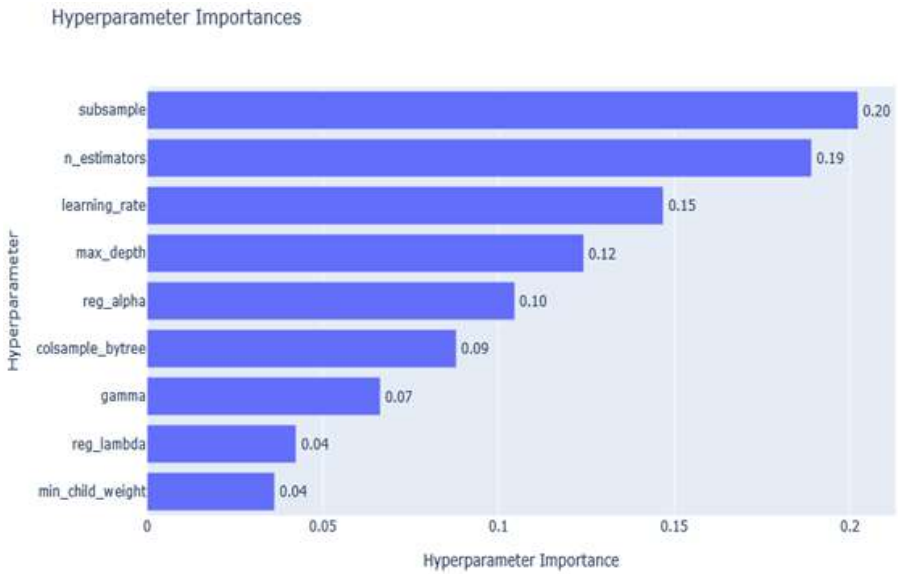


Figure 10. Hyperparameter importance for NSL-KDD dataset using Optuna-LSTM model

Figure 10 shows a hyperparameter importance plot generated by the Optuna optimization framework. This plot displays the hyperparameters on the y-axis and their corresponding importance scores on the x-axis. The horizontal bars are organized in descending order of importance. "subsample" is the most important hyperparameter with a score of 0.20", "n_estimators" follows closely with 0.19, "learning_rate" is the third most important at 0.15, "max_depth" and "reg_alpha" are in the middle range of importance, the least important parameters are "gamma", "reg_lambda", and "min_child_weight".

This visualization helps to focus on tuning the most influential hyperparameters, potentially streamlining the optimization process. It suggests that adjusting the subsample rate, number of estimators, and learning rate could yield the most significant improvements in model performance.

Table 6. Performance of Optuna-LSTM Model for NSL-KDD dataset

Loss	Accuracy	Precision	Recall	F-score
0.029413789076	0.986489000021	0.981475541021	0.980135610811	0.981137613911

Table 6 summarizes the Optuna-LSTM model's performance on the NSL-KDD dataset. The model recorded a loss of 0.0294, indicating minimal error in its predictions. Its accuracy is 98.65%, showing that the model correctly classified a large majority of the instances. The precision, recall, F-Score are 98.15%, 98.01%, and 98.11% reflecting a balanced performance between precision and recall.

Table 7. Accuracy comparison of LSTM and OPTUNA based LSTM for NSL-KDD Data

DL Models	Accuracy using NSL-KDD Dataset
LSTM	0.9842109084129333
OPTUNA with LSTM	0.9864890000214576

Table 7 compares the accuracy of two models, an LSTM and an OPTUNA-optimized LSTM, using the NSL-KDD dataset. The standard LSTM achieves an accuracy of approximately 98.42%. In contrast, when the LSTM model's hyperparameters are fine-tuned using OPTUNA, the accuracy improves to 99.00%. This indicates that hyperparameter optimization with OPTUNA improves the LSTM model's efficiency on the dataset, achieving a notable increase in accuracy of 0.58 percentage points.

Table 8. Error metrics analysis of LSTM and OPTUNA with LSTM for NSL-KDD data

Error Metrics	LSTM Error Value	OPUNA with LSTM Error value
Mean Absolute Error (MAE)	0.5	0.2951642467785
Mean Square Logarithmic Error (MSLE)	0.07242365465	0.0572574259124
Mean Squared Error (MSE)	0.375	0.1077083011482
Root Mean Squared Error (RMSE)	0.61237243569	0.3281894287576
Neural Network Square Error (NNSE)	1.1898	1.4079162954399
R-squared (R2 Score)	0.95860813704	0.9779162954399

An analysis of error metrics for two models based on LSTM: a standard LSTM and an LSTM optimized with OPTUNA is presented in Table 8. For the MAE, the OPTUNA with LSTM model shows a lower error value of approximately 0.295 compared to 0.5 for the standard LSTM, indicating improved average predictive accuracy. Similarly, the MSLE is lower in the OPTUNA-optimized model at about 0.057 compared to 0.072 for the LSTM. The MSE and RMSE are also reduced to 0.108 and 0.328 values in the OPTUNA model, compared

to 0.375 and 0.612 for the standard LSTM, reflecting better predictive performance and error minimization. The NNSE is slightly higher for the OPTUNA model, at 1.408 compared to 1.1898, suggesting some trade-offs in model characteristics. Lastly, the R2 Score is slightly lower at approximately 0.958 for the OPTUNA LSTM compared to 0.9779 for the standard LSTM, indicating a minor decrease in the proportion of variance. Overall, the error metrics suggest that OPTUNA's hyperparameter optimization generally enhances the LSTM's prediction accuracy and error handling, despite a slight variation in NNSE and R2 Score.

5.2 Phase 2: Result Analysis using TabNet

This section presents an experimental analysis of TabNet model [41] for IoT NIDS using the NSL-KDD and BoTNeTIoT-L01 dataset.

Table 9. Performance of the TabNet on BoTNeTIoT-L01 and NSL-KDD datasets

Dataset	**Accuracy**	**Precision**	**Recall**	**F-Score**
BoTNeTIoT-L01	0.9891	0.9740	0.9821	0.9949
NSL-KDD	0.9853	0.9734	0.9821	0.9789

This Table 9 compares the TabNet model performance on BoTNeTIoT-L01 and NSL-KDD datasets. For the BOTNETIOT-L01 dataset, TabNet achieved an accuracy of 0.9891, precision of 0.9720, recall of 0.9821 and F-Score of 0.9949. On the NSL-KDD dataset, the model performed similarly well, with an accuracy of 0.9853, precision of 0.9734, recall of 0.9821 and F-Score of 0.9789.

Algorithm for optuna TabNet model :

Input: Trained data, Set no. of trails = 100, No. of hyperparameters for tuned

Output: Optimal best hyperparameters

Step 1: Define the Objective Function

Step 2: Configure Optuna Study

Step 3: Define Hyperparameter Search Space

Step 4 : Train the TabNet-IDS Model

Step 5: Execute the Optimization Process

Step 6: Retrieve the Best Hyperparameters

Step 7: Evaluate the Model with the Best Hyperparameters

Step 8: Exit

The feature importance of NSL-KDD for Optuna based TabNet model is displayed in Figure 11. The chart shows a wide range of features on the vertical axis, with their corresponding importance scores represented by horizontal bars extending to the right. The importance scores range from 0 to approximately 0.25 on the horizontal axis.

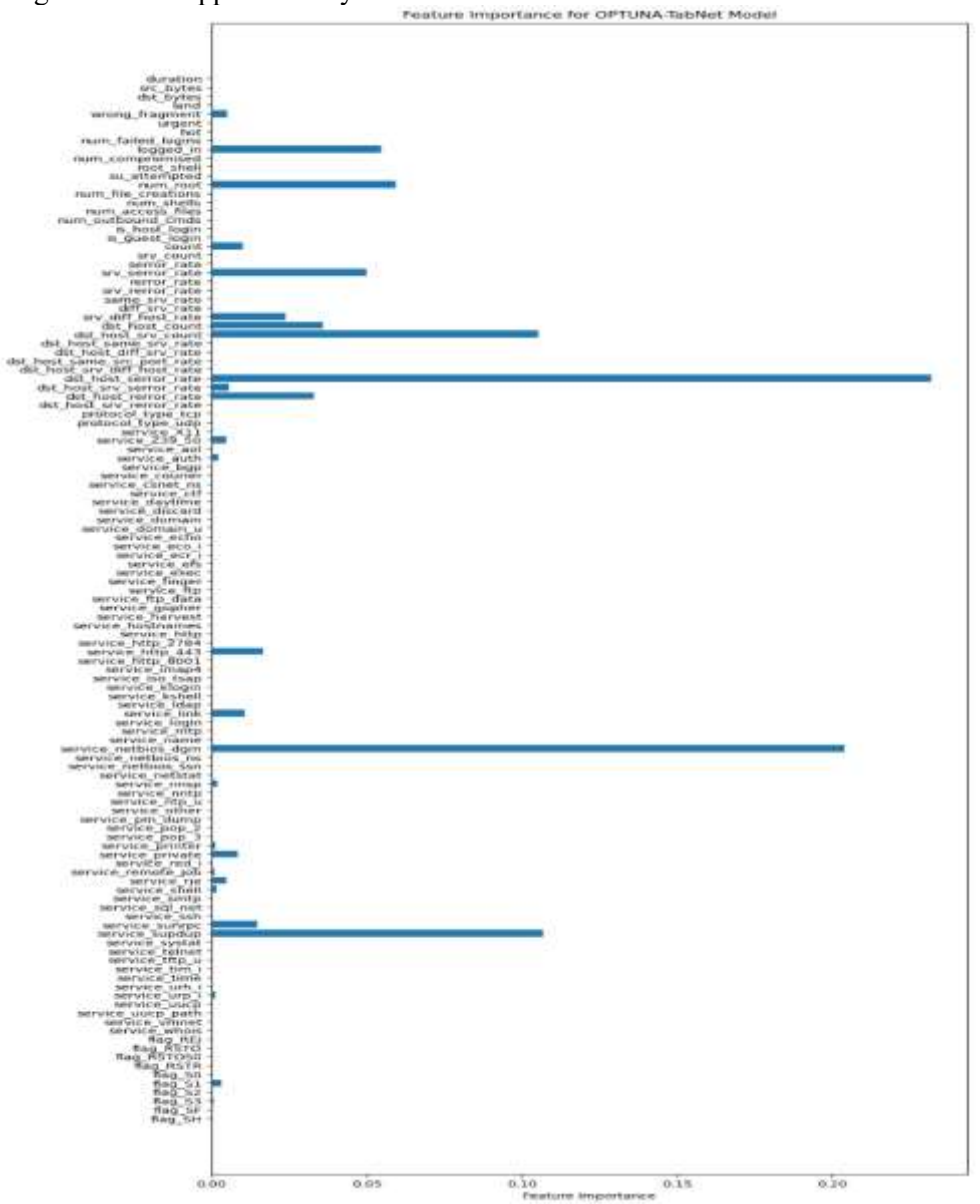


Figure 11. Feature importance of NSL-KDD for Optuna based TabNet model

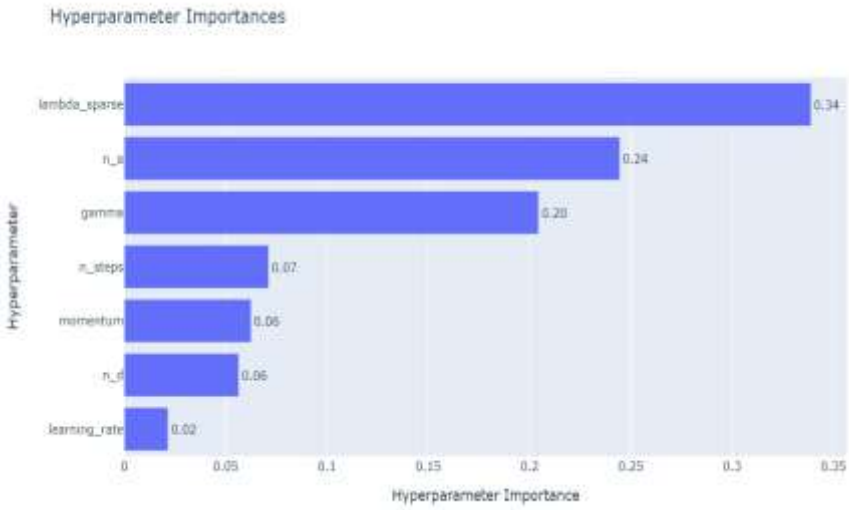
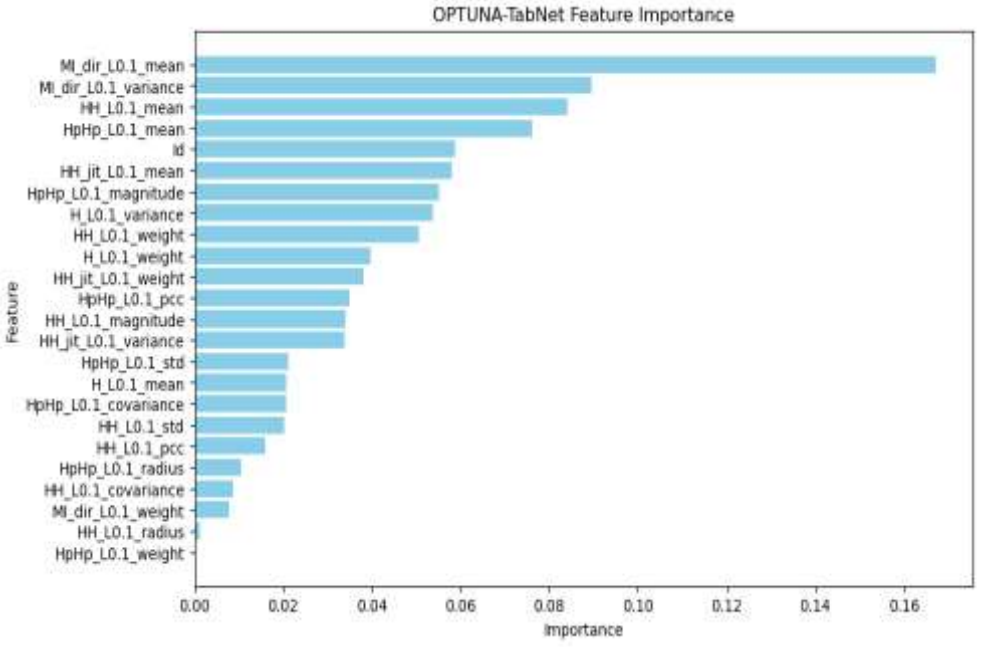


Figure 12. Hyperparameter importance for NSL-KDD dataset using Optuna-Tabnet model

The hyperparameter importance for the NSL-KDD dataset using an Optuna-optimized TabNet model is presented in Figure 12. The chart displays various hyperparameters on the vertical axis and their respective importance scores on the horizontal axis, ranging from 0 to 0.35. The hyperparameters are ranked in descending order of importance. The top three parameters (lambda_sparse, n_a, and gamma) are crucial for fine-tuning the model, while parameters like learning_rate has less influence on the overall performance. This visualization



helps in understanding which hyperparameters has significant influence on the model's performance when trained on the NSL-KDD dataset.
Figure 13. Feature importance of BoTNeTIoT-L01 dataset for Optuna-Tabnet model

The feature importance ranking for various metrics in the OPTUNA-TabNet model is shown in Figure 13 using BoTNeTIoT-L01 dataset. The features are listed on the y-axis, while their respective importance scores are denoted by horizontal bars on the x-axis. The most important feature is "MI_dir_L0.1_mean", followed by "MI_dir_L0.1_variance" and "HH_L0.1_mean". The importance scores range from about 0 to 0.16.

Table 10. TabNet hyperparameters description with Optuna-based optimized search result value

TabNet Hyper Parameters	**Parameter Description**	**Search Space**	**Optimized Search Result Value**
n_d	Width of the decision prediction layer	[8,64]	62
n_a	Embedding size for attention mask	[8,64]	62
n_steps	Number of sequential steps in the architecture	[3, 10]	3
n_independent	Number of independent Gated Linear Unit layers in each step	[1, 5]	2
gamma	Feature re-use coefficient, where values near 1 reduce correlation between layers	[1.0, 2.0]	1.5
n_shared	Number of shared Gated Linear Unit layers at each step	[1, 5]	2
epsilon	A constant value, not tuned	1e-15	-
momentum	Momentum applied in batch normalization	[0.01, 0.4]	0.02
lambda_sparse	Sparsity penalty coefficient; higher values encourage more sparse feature selection	[1e-3, 1e-5]	1e-3
optimizer_fn	PyTorch optimizer function to decrease complexity and achieve minima, default = torch. optim.Adam.	-	-
scheduler_fn	Function to adjust learning rate during training	-	-
scheduler_params	Parameters applied to the learning rate scheduler	{"gamma": 0.95, "step_size": 10}	{"gamma": 0.95, "step_size": 10}

mask_type	Masking function for feature selection	["sparsemax", "entmax"]	"sparsemax"
patience	Number of epochs to wait before early stopping if no improvement	[15, 30]	25

The hyperparameters of the TabNet model and their optimized values using Optuna are described in Table 10. It lists 13 hyperparameters, their descriptions, search spaces, and the optimized values found. Key optimized results include n_d and n_a both set to 62 and lambda_sparse at 1e-3. This table provides insights into the model's architecture and training process, showing how Optuna helped fine-tune these parameters for optimal performance.

Table 11. Performance of the Optuna-TabNet on BoTNeTIoT-L01 and NSL-KDD datasets

Dataset	**Accuracy**	**Precision**	**Recall**	**F-Score**
BoTNeTIoT-L01	0.9981	0.9972	0.9945	0.9939
NSL-KDD	0.9941	0.9952	0.9919	0.9928

The performance metrics of the TabNet model optimized using Optuna on BoTNeTIoT-L01 and NSL-KDD dataset are given in Table 11. For the BoTNeTIoT-L01 dataset, the model accomplished remarkable results with an accuracy of 0.9981, precision of 0.9972, recall of 0.9945 and F-Score of 0.9939. Similarly, for the NSL-KDD dataset, the model performed exceptionally well, with an accuracy of 0.9941, precision of 0.9952, recall of 0.9919 and F-Score of 0.9928. The results indicate excellent performance across all metrics for both datasets.

Comparing this to the previous table 9, it is observed that the Optuna-optimized TabNet model surpassed the original TabNet model. For the BoTNeTIoT-L01 dataset, all metrics improved, with notable increases in Precision (from 0.9720 to 0.9972) and Accuracy (from 0.9891 to 0.9981). The improvements were even more pronounced for the NSL-KDD dataset, where Precision increased from 0.9734 to 0.9952, and Accuracy from 0.9853 to 0.9941. The F-Score and Recall also saw improvements across both datasets. This comparison suggests that the use of Optuna for hyperparameter optimization significantly heightened the performance of the TabNet model, resulting in increased accurate and reliable predictions for both cybersecurity datasets.

Table 12. Proposed model performance comparison with other models

References	**Method used**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-Score (%)**	**Hyper Parameter Optimization**
Parra et. al. [12]	LSTM	97.84	97.81	95	96.2	No

Liu et al [17]	TabNet	-	-	-	99.89	No
Nguyen et al [18]	TabNet (BoT-IoT)	98.53	98.65	98.53	98.57	No
	TabNet (UNSW-NB15)	97.95	97.84	97.95	97.67	No
Asaduzzaman et. al. [19]	CNN-LSTM	93.53	57.45	31.87	41	No
Zegarra et. al. [22]	TabNet (CIC-IDS2017)	97.03	97.03	97.02	96.97	No
	TabNet (CSE-CICIDS2018)	95.58	95.69	95.59	95.55	No
	TabNet (CIC-DDoS2019)	98.51	98.50	98.40	98.44	No
Alazab et. al. [23]	LSTM	98.73	97.07	97.11	97.05	No
Proposed Model -A (NSL-KDD)	LSTM	98.42	98.54	98.27	98.16	No
	OPTUNA - LSTM	98.64	98.14	98.01	98.11	Yes
Proposed Model-B (NSL-KDD)	TabNet	98.53	97.34	97.89	98.21	**No**
	TabNet - Optuna	99.41	99.52	99.19	99.28	Yes
Proposed Model- D (BoTNeTIoT-L01)	TabNet	98.91	97.20	99.49	98.21	No
	TabNet - Optuna	**99.81**	**99.72**	**99.45**	**99.39**	**Yes**

6. Conclusion

The extensive investigation into advanced deep learning methodologies for IoT network intrusion detection has yielded compelling results, underscoring the efficacy of the Optuna-optimized TabNet NIDS model in enhancing cybersecurity measures. This study's findings demonstrate a marked improvement in intrusion detection capabilities when leveraging the synergy between TabNet's interpretable learning approach and Optuna's hyperparameter optimization framework. The superior performance of the optimized TabNet NIDS model, evidenced by its heightened accuracy, precision, and computational efficiency, surpasses that of traditional LSTM and non-optimized TabNet implementations. Notably, the model's robust feature interpretation capabilities provide invaluable insights into the critical factors governing intrusion detection in IoT networks, thereby facilitating more informed security strategies. The model's capability to generate rapid responses to potential security breaches, coupled with its interpretability makes it a powerful tool in combating cyber threats within the growing IoT ecosystem. These outcomes not only validate the effectiveness of our proposed approach but also open doors for future innovations in adaptive, intelligent IDS. As IoT networks continue to proliferate, the integration of such sophisticated, optimized deep learning models becomes increasingly crucial in fortifying digital infrastructures against evolving cybersecurity challenges.

Acknowledgments

The authors gratefully acknowledge the Management of BPUT, Rourkela, Odisha, India for providing the facilities to carry out the research work.

Author contributions

In this manuscript, all authors have equally contributed.

Funding

This research received no funding.

Conflict of interest statement

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Additional information

Correspondence and requests for materials should be addressed to Prof Sujata Chakravarty

Data availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Lee, J. (2019). The internet of things for enterprises: An ecosystem, architecture, and IoT service business model. Internet of Things, 7, Article 100078.

2. Sahu, A. K., Dwivedi, Y. K., Tripathi, A. K., & Lal, B. (2021). Internet of Things attack detection using hybrid Deep Learning Model. Computer Communications, 176, 146-154.
3. Hromada, D., Szalai, I., Szűcs, V., & Holczinger, T. (2023). Security aspects of the internet of things. In Research anthology on convergence of blockchain, internet of things, and security (pp. 67-87). IGI Global.
4. Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in IoT intrusion detection. Journal of Network and Systems Management, 30(1), Article 8.
5. DLTIDS: A dual-layer trust-based intrusion detection system for Blackhole attacks in wireless sensor networks. (2024). Nanotechnology Perceptions, 20(S6). https://doi.org/10.62441/nano-ntp.v20is6.13
6. Dayal, I., Singh, J., Saraswat, N., Padmanabhan, J., Kaur, A., & Chandani, P. (2024). Deep Learning to Improve Cyber Security: Swarm-Intelligent Fully Convolutional Neural Network Model for Efficient Intrusion Detection in Big Data Environment. Nanotechnology Perceptions, 584-593. https://doi.org/10.62441/nano-ntp.v20iS4.44
7. Lai, J. P., Lin, Y. L., Lin, H. C., Shih, C. Y., Wang, Y. P., & Pai, P. F. (2023). Tree-based machine learning models with optuna in predicting impedance values for circuit analysis. Micromachines, 14(2), Article 265.
8. Ibitoye, O., Shafiq, O., & Matrawy, A. (2019). Analyzing Adversarial Attacks against Deep Learning for Intrusion Detection in IoT Networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
9. 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 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC) (pp. 256-25609). IEEE.
10. Susilo, B., & Sari, R. F. (2020). Intrusion Detection in IoT Networks Using Deep Learning Algorithm. Information, 11(5), Article 279.
11. Alkadi, O., Moustafa, N., Turnbull, B., & Choo, K. K. R. (2020). A Deep Blockchain Framework-enabled Collaborative Intrusion Detection for Protecting IoT and Cloud Networks. IEEE Internet of Things Journal, 8(1), 1-1.
12. Parra, G. D. L. T., Rad, P., Choo, K. K. R., & Beebe, N. (2020). Detecting Internet of Things attacks using distributed deep learning. Journal of Network and Computer Applications, 163, Article 102662.
13. Samy, A., Yu, H., & Zhang, H. (2020). Fog-Based Attack Detection Framework for Internet of Things Using Deep Learning. IEEE Access, 8, 74571-74585.
14. Churcher, A., Ullah, R., Ahmad, J., Masud, F., Gogate, M., Alqahtani, F., Nour, B., & Buchanan, W. J. (2021). An experimental analysis of attack classification using machine learning in IoT networks. Sensors, 21(2), Article 446.
15. Adeel, A., Khan, M. A., Khan, M. A., Latif, S., Ajaz, M., Shah, A. A., & Ahmad, J. (2021). A New Ensemble-Based Intrusion Detection System for Internet of Things. Arabian Journal for Science and Engineering. https://doi.org/10.1007/S13369-021-06086-5
16. Palla, T. G., & Tayeb, S. (2021). Intelligent Mirai malware detection for IoT nodes. Electronics, 10(11), Article 1241.
17. Liu, J., Simsek, M., Kantarci, B., Bagheri, M., & Djukic, P. (2022). Collaborative feature maps of networks and hosts for ai-driven intrusion detection. In GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 2662-2667). IEEE.
18. Nguyen, T. N., Dang, K. M., Tran, A. D., & Le, K. H. (2022). Towards an attention-based threat detection system for iot networks. In International Conference on Future Data and Security Engineering (pp. 301-315). Springer.

19. Asaduzzaman, M., & Rahman, M. M. (2022). An Adversarial Approach for Intrusion Detection Using Hybrid Deep Learning Model. In 2022 International Conference on Information Technology Research and Innovation (ICITRI) (pp. 18-23). IEEE.
20. Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications, 199, 113-125.
21. Dhiaa, M., Alhaidari, F., Atta-ur-Rahman, & Mustafa, R. A. M. (2023). Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT. Journal of Sensor and Actuator Networks, 12(2), Article 29.
22. Zegarra Rodríguez, D., Daniel Okey, O., Maidin, S. S., Umoren Udo, E., & Kleinschmidt, J. H. (2023). Attentive transformer deep learning algorithm for intrusion detection on IoT systems using automatic Xplainable feature selection. PLOS ONE, 18(10), Article e0286652.
23. Alazab, M., Alazab, M., Shalaginov, A., Mesleh, A., & Awaysheh, F. (2024). A Novel IDS with a Dynamic Access Control Algorithm to Detect and Defend Intrusion at IoT Nodes. Sensors, 24(7), Article 2188.
24. Al-Quayed, F., Ahmad, Z., & Humayun, M. (2024). A situation based predictive approach for cybersecurity intrusion detection and prevention using machine learning and deep learning algorithms in wireless sensor networks of industry 4.0. IEEE Access.
25. Li, J., Othman, M. S., Chen, H., Ding, Y., Faisal, M., & Huang, J. (2024). Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning. Journal of Big Data, 11, Article 36.
26. Ekundayo, I. (2020). Optuna Optimization Based Cnn-lstm Model for Predicting Electric Power Consumption [Master's thesis]. National College of Ireland.
27. Shekhar, S., Bansode, A., & Salim, A. (2021). A comparative study of hyper-parameter optimization tools. In Proceedings of the IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) (pp. 1-6). IEEE.
28. Srinivas, P., & Katarya, R. (2022). hyOPTXg: OPTUNA hyper-parameter optimization framework for predicting cardiovascular disease using XGBoost. Biomedical Signal Processing and Control, 73, Article 103456.
29. Li, Z. H., Xu, J., Wang, W. Q., Zhang, X. Y., & Wang, Y. J. (2023). Research on mine pressure prediction method based on Optuna-LSTM. Mining Research and Development, 43(03), 98-102.
30. Zhou, Y., Dong, Z., & Bao, X. (2024). A Ship Trajectory Prediction Method Based on an Optuna--BILSTM Model. Applied Sciences, 14(9), Article 3719.
31. Bakry, A. N., Alsharkawy, A. S., Farag, M. S., & Raslan, K. R. (2024). Automatic suppression of false positive alerts in anti-money laundering systems using machine learning. The Journal of Supercomputing, 80(5), 6264-6284.
32. Altunay, H. C., & Albayrak, Z. (2023). A hybrid CNN+ LSTM-based intrusion detection system for industrial IoT networks. Engineering Science and Technology, an International Journal, 38, Article 101322.
33. Zegarra Rodríguez, D., Daniel Okey, O., Maidin, S. S., Umoren Udo, E., & Kleinschmidt, J. H. (2023). Attentive transformer deep learning algorithm for intrusion detection on IoT systems using automatic Xplainable feature selection. PLOS ONE, 18(10), Article e0286652.
34. https://medium.com/@okpo65/paper-review-tabnet-deep-neural-net-for-tabular-a97c43290969
35. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2623-2631). ACM.
36. Zhou, Y., Dong, Z., & Bao, X. (2024). A Ship Trajectory Prediction Method Based on an Optuna--BILSTM Model. Applied Sciences, 14(9), Article 3719.

37. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1-6). IEEE. https://doi.org/10.1109/cisda.2009.5356528
38. Alshowaideh, A. (n.d.). IoT dataset for intrusion detection systems (IDS) [Data set]. Kaggle. https://www.kaggle.com/datasets/azalhowaide/iot-dataset-for-intrusion-detection-systems-ids
39. Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2007). Data preprocessing for supervised learning. International Journal of Computer Science, 1(12), 6.Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2019). A detailed analysis of the CICIDS2017 data set. In Information Systems Security and Privacy (pp. 172-188). Springer. https://doi.org/10.1007/978-3-030-25109-3_9
40. Salih, A. A., & Abdulazeez, A. M. (2021). Evaluation of classification algorithms for intrusion detection system: A review. Journal of Soft Computing and Data Mining, 2(1), 31-40. https://doi.org/10.30880/jscdm.2021.02.01.004
41. https://dreamquark-ai.github.io/tabnet/generated_docs/README.html#