A Privacy-Preserving and Efficient Split Learning Approach for IoT Network Security

by

Farin Ahsan

ID: CSE2201025096

Pir Mohammad

ID: CSE2201025040

Abdullah Al Rohan

ID: CSE2201025014

Fahim Montasir

ID: CSE2201025059

Most. Shopna Akter

ID: CSE2201025121

Supervised by

Nayeem Ullah

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

January 2026

A Privacy-Preserving and Efficient Split Learning Approach for IoT Network Security

by

Farin Ahsan

ID: CSE2201025096

Pir Mohammad

ID: CSE2201025040

Abdullah Al Rohan

ID: CSE2201025014

Fahim Montasir

ID: CSE2201025059

Most. Shopna Akter

ID: CSE2201025121

Supervised by

Nayeem Ullah

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

January 2026

APPROVAL

The thesis titled "**A Privacy-Preserving and Efficient Split Learning Approach for IoT Network Security**" submitted by **Farin Ahsan** (CSE2201025096), **Pir Mohammad** (CSE2201025040), **Abdullah Al Rohan** (CSE2201025014), **Fahim Montasir** (CSE2201025059) and **Most. Shopna Akter** (CSE2201025121) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Nayeem Ullah
Lecturer
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Nayeem Ullah, Lecturer,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Nayeem Ullah)
Supervisor

Signature

Farin Ahsan
ID: CSE2201025096

Pir Mohammad
ID: CSE2201025040

Abdullah Al Rohan
ID: CSE2201025014

Fahim Montasir
ID: CSE2201025059

Most. Shopna Akter
ID: CSE2201025121

ABSTRACT

With the rising sophistication of cyberattacks, Network Intrusion Detection Systems (NIDS) are critical for organizational security. Traditional centralized deep learning approaches for NIDS have challenges with data privacy and the potential of sensitive network information leaks. This study presents a privacy-preserving NIDS architecture that employs Split Learning (SL) techniques, which vary from federated learning in that the neural network is divided into segments situated between the client (where the data source is located) and a central server. In this paradigm, raw network traffic data remains on the local client, but processed (or shattered) data from the split layer is transmitted to the server for further analysis, ensuring data secrecy and compliance with privacy rules such as GDPR. Evaluations of benchmark datasets like UNSW-NB15 and CIC-IDS2017 concentrate on detection accuracy, computing overhead, and communication efficiency. The results show that the Split Learning-based NIDS achieves detection performance comparable to centralized systems while significantly decreasing privacy threats and hardware demands on client devices. This paper proposes a scalable, secure collaborative intrusion detection method that can perform in resource-constrained and privacy-sensitive environments.

Keywords: Network Intrusion Detection (NIDS), Split Learning, Data Privacy, Cybersecurity, Deep Learning, Distributed Computing.

ACKNOWLEDGMENT

At the very beginning, we would like to express our deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Nayeem Ullah**, Lecturer, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our thesis.

We would like to convey our special gratitude to **Prof. Bulbul Ahamed,** Head, Department of Computer Science and Engineering, Sonargaon University, for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to our parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

ACCS	Australian Centre for Cyber Security
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDoS	Distributed Denial-of-Service
DNN	Deep Neural Network
DoS	Denial-of-Service
DP	Differential Privacy
DT	Decision Tree
FedAvg	Federated Averaging
FL	Federate Learning
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
IID	Independent and Identically Distributed
IoT	Internet of Things
GPU	Graphics Processing Unit
KNN	K-Nearest Neighbour
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MFA	Multi-Factor Authentication
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naïve Bayes
NIDS	Network Intrusion Detection
NN	Neural Network
Non-IID	Non-Independent and Identically Distributed
PCA	Principal Component Analysis
RAM	Random Access Memory
RF	Random Forest
SDN	Software-Defined Networking
SL	Split Learning
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TV	Target Variance
XGBoost	eXtreme Gradient Boosting

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The rapid growth of Internet of Things (IoT) networks has introduced significant challenges in ensuring secure and reliable communication among resource-constrained and widely distributed devices [1]. These networks generate large volumes of sensitive data, making them attractive targets for cyberattacks such as denial-of-service, probing, and unauthorized access [2]. As a result, effective network security mechanisms are essential for protecting IoT infrastructures.

Network Intrusion Detection Systems (NIDS) play a critical role in IoT network security by monitoring traffic and identifying malicious activities in real time [3]. However, traditional NIDS approaches face limitations in handling large-scale data, evolving attack patterns, and encrypted communication [4]. The integration of machine learning has significantly improved intrusion detection accuracy and adaptability, yet conventional centralized learning methods raise serious privacy, security, and communication efficiency concerns in IoT environments [5].

To address these challenges, privacy-preserving distributed learning paradigms have emerged as viable alternatives [6]. While Federated Learning reduces data exposure by keeping raw data on local devices, it still suffers from high communication overhead and potential information leakage through model updates [7]. In contrast, Split Learning offers a promising solution by partitioning the learning model between clients and servers, ensuring that raw data never leave IoT devices while reducing computational and communication burdens [8].

1.2 Background and Research Context

The widespread adoption of Internet of Things (IoT) technologies has transformed modern communication infrastructures by enabling large-scale connectivity among heterogeneous and resource-constrained devices [1]. These devices continuously generate and transmit sensitive data across networks, making IoT environments highly vulnerable to cyber threats such as denial-of-service attacks, probing, malware propagation, and unauthorized access [2]. Ensuring network security in IoT systems has therefore become a critical research challenge.

Traditional security mechanisms are often inadequate for IoT networks due to their centralized nature, high computational requirements, and limited scalability [4]. As a result, intelligent and adaptive security solutions are required. One of the most effective network-level defense mechanisms is the Network Intrusion Detection System (NIDS), which monitors network traffic to detect malicious activities in real time [3].

This thesis focuses on designing a privacy-preserving and efficient Split Learning-based framework for IoT network security, where sensitive data remain protected while maintaining high intrusion detection performance [8].

1.3 Network Intrusion Detection Systems (NIDS)

Network Intrusion Detection Systems analyze network traffic to identify suspicious or malicious behavior that may indicate cyberattacks or policy violations [3]. Unlike intrusion prevention systems, NIDS operate passively by generating alerts without disrupting normal traffic flow. This characteristic makes NIDS particularly suitable for IoT environments, where uninterrupted communication is essential.

In IoT networks, NIDS provide centralized visibility over distributed devices and help detect both internal and external threats. However, traditional NIDS approaches face limitations when dealing with encrypted traffic, large-scale data streams, and evolving attack patterns [4].

1.3.1 Operation of NIDS

NIDS operate by deploying sensors at strategic points such as gateways, routers, switches, or firewalls [3]. These sensors capture network packets and extract relevant features, which are then analyzed using predefined rules or learned models. The system compares observed traffic against known attack signatures or evaluates deviations from established normal behavior profiles.

Two main detection strategies are commonly used: signature-based detection, which is effective for identifying known attacks with low false-positive rates, and anomaly-based detection, which can detect previously unseen or zero-day attacks by identifying abnormal behavior patterns [4].

1.3.2 Detection Techniques and Limitations

Signature-based detection relies on a database of known attack patterns and provides fast and accurate detection for previously identified threats. However, it is ineffective against novel attacks. Anomaly-based detection addresses this limitation by modeling normal traffic behavior, but it may produce higher false-positive rates [4].

In IoT networks, additional challenges such as limited device resources, encrypted communication, and large-scale data volume further complicate intrusion detection. These limitations motivate the integration of machine learning techniques into NIDS [5].

1.4 Machine Learning-Based NIDS for IoT Security

Machine learning (ML) enhances NIDS by enabling automated, adaptive, and data-driven intrusion detection [5]. ML-based NIDS analyze traffic patterns to classify network behavior as normal or malicious, overcoming the rigidity of traditional rule-based systems.

Supervised learning algorithms, including Random Forest, Support Vector Machines, and neural networks, have demonstrated high detection accuracy on benchmark intrusion detection datasets such as NSL-KDD, CIC-IDS2017, and UNSW-NB15 [5]. Unsupervised techniques further support the detection of unknown attacks by identifying deviations from normal behavior.

Despite their effectiveness, ML-based NIDS traditionally rely on centralized data collection, which introduces significant privacy and security concerns in IoT environments [6].

1.5 Privacy Challenges in Centralized Learning for IoT

Centralized machine learning requires aggregating data from multiple IoT devices into a single server for training. In security-sensitive environments, this approach poses serious risks, including data leakage, unauthorized access, and non-compliance with data protection regulations [6].

Moreover, transmitting large volumes of raw network traffic data increases communication overhead and energy consumption, which is unsuitable for resource-constrained IoT devices. These challenges highlight the need for privacy-preserving and communication-efficient learning frameworks [7].

1.6 Federated Learning for Privacy Preservation

Federated Learning (FL) is a distributed machine learning paradigm that enables collaborative model training without sharing raw data [7]. Each IoT client trains a local model using its private data and sends only model updates to a central server for aggregation.

Although FL improves data privacy, it suffers from several limitations in IoT network security applications, including high communication overhead due to frequent model updates, vulnerability to gradient leakage attacks, and performance degradation under non-IID data distributions [7].

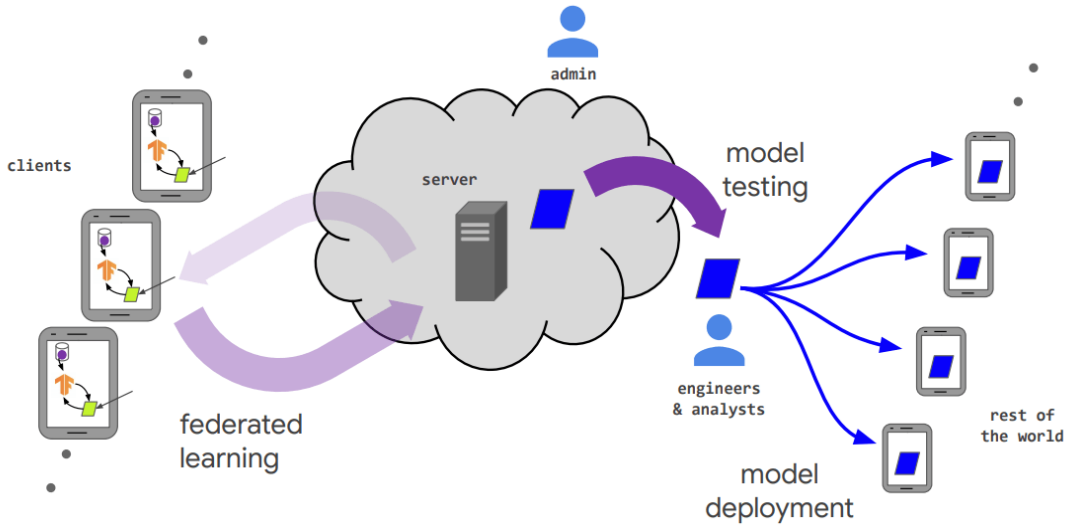


Fig.1.6: Federate Learning's client-server architecture

It represents FL's client-server architecture: raw data stays local to preserve privacy, addressing IoT challenges like bandwidth limits and data sensitivity from earlier context. Unlike centralized ML, FL aggregates updates from edge clients (bottom layer)

at the server, iteratively improving detection accuracy for NIDS while minimizing communication overhead compared to full data transfer.

1.7 Split Learning: A Privacy-Preserving and Efficient Alternative

Split Learning (SL) is a distributed learning approach in which a neural network model is vertically partitioned between IoT clients and a central server [8]. Clients process raw data through the initial layers of the model and transmit intermediate representations, known as smashed data, to the server for further computation.

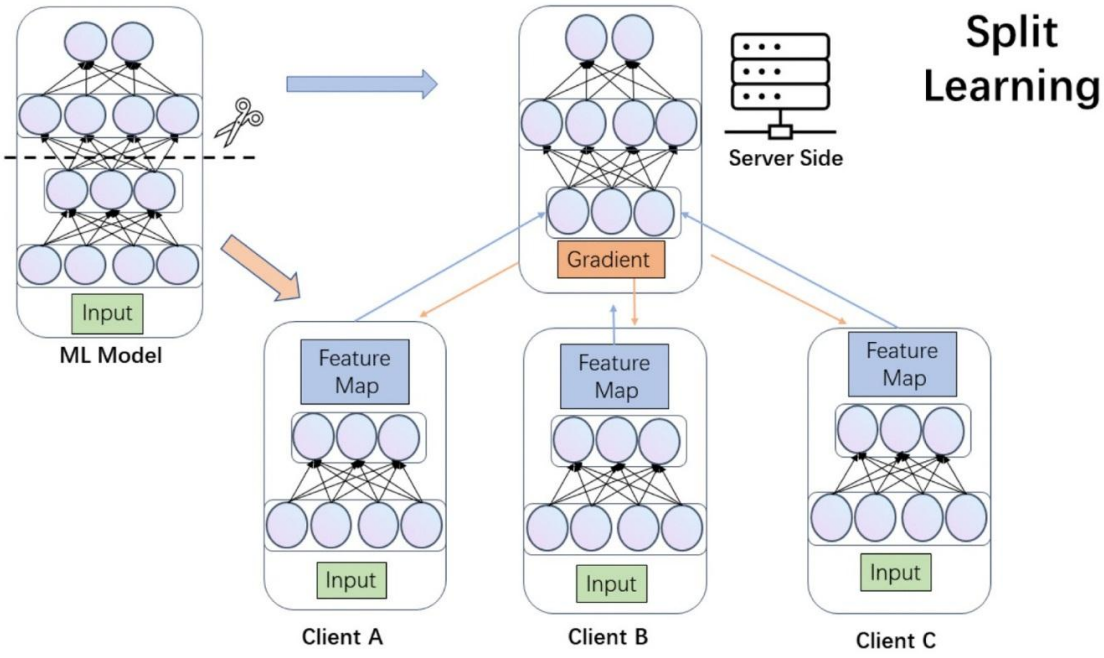


Fig.1.7: Architectural framework of Split Learning (SL)

As illustrated in the figure, Split Learning (SL) facilitates collaborative model training by partitioning a deep neural network into client-side and server-side components at a designated smash layer. In this framework, clients maintain data sovereignty by performing local computations on raw input and transmitting only the resulting intermediate feature maps to a centralized server, rather than the raw data itself. The server completes the forward pass and initiates backpropagation, returning the gradients of the smash layer to the client to enable local weight updates. This architectural split effectively decouples the computational burden, allowing resource-constrained edge devices (e.g., Clients A, B, and C) to participate in training complex models while ensuring a high degree of data privacy through the abstraction of sensitive information into latent representations.

This architecture ensures that raw network traffic data never leave the IoT devices, significantly enhancing privacy protection. Additionally, Split Learning reduces client-side computational requirements and communication overhead, making it particularly suitable for resource-constrained IoT environments [8].

The following sections of this paper are organized in the manner outlined below: In Section 2, the literature review is presented. Section 3 outlines the methodology. Section 4 details the proposed system, while Section 5 focuses on the results and discussion. In conclusion and future scope, Section 6 wraps up the manuscript.

CHAPTER 2

LITERATURE REVIEW

2.1 Machine Learning-Based Network Intrusion Detection

Mainali et al. (2025) proposed a machine learning-driven Network Intrusion Detection System that integrates Fast k-Nearest Neighbours (Fast k-NN) with software-defined networking (SDN) to address denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. The study employs a systematic methodology involving data preprocessing, Random Forest-based feature selection, and Synthetic Minority Over-sampling Technique (SMOTE) for class balancing, validated using the CICIDS2017 dataset. A realistic experimental environment utilizing Mininet, POX controller, and hping3 enhances the practical relevance of the study. The proposed Fast k-NN model achieves approximately 99% detection accuracy, outperforming an LSTM-based baseline. Despite its strong performance, the study provides limited analysis of computational efficiency and real-time deployment constraints, which are critical considerations for IoT network security. [9]

2.2 Hybrid Machine Learning and Authentication-Based Security Approaches

Mahmood et al. (2024) presented a hybrid intrusion detection framework combining machine learning techniques with multi-factor authentication (MFA) to enhance network security. The study integrates XGBoost for feature selection and SMOTE for data balancing, followed by classification using multiple machine learning and deep learning models, including Random Forest, Decision Trees, and neural networks. Experimental results on benchmark datasets such as KDDCUP'99 demonstrate near-perfect detection accuracy, with Random Forest achieving 99.97%. While the framework offers strong detection capability and improved access control, it primarily focuses on centralized learning and does not sufficiently address data privacy or communication efficiency, limiting its applicability in large-scale IoT environments. [10]

2.3 Intrusion Detection in IoT Networks

Saheed et al. (2022) proposed a robust machine learning-based intrusion detection system tailored for IoT networks, explicitly considering IoT-specific constraints such as scalability and limited computational resources. The study applies min–max normalization, principal component analysis (PCA) for dimensionality reduction, and evaluates six machine learning classifiers using the UNSW-NB15 dataset, which includes modern IoT-relevant attack scenarios. The PCA-XGBoost model achieves near-perfect performance with an accuracy of 99.99% and a Matthews Correlation Coefficient (MCC) of 99.97%, surpassing existing approaches. Although the study demonstrates high accuracy and efficiency, it relies on centralized training and does not address privacy preservation or decentralized deployment, which are essential for real-world IoT security systems. [11]

2.4 Privacy-Preserving Learning for IoT Security

Priyadarshini (2024) investigated the application of Federated Learning and Split Learning for anomaly detection in IoT-based smart city environments, addressing both cybersecurity and data privacy challenges. The study conducts extensive experiments using NSL-KDD and UNSW-NB15 datasets and evaluates a broad range of machine learning and deep learning models. The results indicate that both Federated Learning and Split Learning achieve competitive detection accuracy, reaching up to 99.23% on NSL-KDD, while ensuring that raw data remain localized. However, the study reports increased computational and communication overhead compared to traditional centralized methods and provides limited discussion on optimizing efficiency for real-time IoT deployments. [12]

2.5 Research Gap and Motivation

The reviewed literature demonstrates that machine learning-based NIDS can achieve high detection accuracy for both traditional and IoT networks. However, several limitations remain evident. Many existing approaches rely on centralized data collection, which poses significant privacy and security risks in IoT environments. While Federated Learning mitigates data sharing concerns, it introduces high communication overhead and potential information leakage through model updates. Although Split Learning has shown promise as a privacy-preserving alternative, there is a lack of focused research evaluating its efficiency and suitability specifically for IoT network intrusion detection.

Furthermore, limited studies provide a comprehensive comparison between Split Learning and Federated Learning in terms of detection performance, communication efficiency, and practical applicability in resource-constrained IoT settings. These gaps motivate the present research, which proposes a privacy-preserving and efficient Split Learning approach for IoT network security, aiming to achieve high intrusion detection accuracy while minimizing data exposure and computational overhead.

CHAPTER 3

METHODOLOGY OF RESEARCH

3.1 Dataset Overview

In this paper, we used UNSW-NB15 dataset. The dataset, obtained from Kaggle, which is a benchmark dataset extensively used in the field of network intrusion detection systems (NIDS) and cybersecurity research. The dataset was created by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm tool, which generates realistic modern network traffic. Unlike older datasets such as KDD'99 and NSL-KDD, UNSW-NB15 captures contemporary attack behaviours and normal traffic patterns, making it highly suitable for evaluating modern machine learning and distributed learning techniques. [36]

The dataset consists of both normal network traffic and malicious traffic, where the malicious traffic is categorized into nine distinct attack types, namely: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Each record in the dataset represents a single network flow with detailed statistical and protocol-level information.

The UNSW-NB15 dataset supports both binary classification (normal vs. attack) and multi-class classification (specific attack category). In this study, the dataset is primarily used for binary intrusion detection, while also preserving attack category information for extended analysis and future scalability. [36]

Model	Parameter	Value
FL	Number of clients	3
	Communication rounds (global epochs)	25
	Local epochs	1
	Batch size	Full client dataset
	Hidden layer 1 nodes	128
	Hidden layer 2 nodes	64
	Output nodes	2
	Activation function	ReLU
	Classification function	Softmax
SL	Client–server architecture	Two-part model
	Number of clients	1
	Input features	One-hot encoded features
	Hidden layer 1 nodes	128
	Hidden layer 2 nodes	64
	Batch size	64
	Epoches	25
	Output	Intermediate activations
RF	Ensemble method	Bagging
	Number of trees	200
	Random seed	42
	Feature encoding	One-hot encoding
	Loss function	Gini impurity
	Train–test split	80% / 20%

Table 3.1: Summary of Model Architecture and Training Parameters

3.2 Dataset Size & Structure

The complete UNSW-NB15 dataset contains approximately 2.54 million records distributed across multiple CSV files. However, the Kaggle version commonly provides a pre-split dataset consisting of:

a) Training set: 175,341 records

b) Testing set: 82,332 records

Each record contains 49 features, including both numerical and categorical attributes, along with two label columns:

a) label: Binary label indicating normal (0) or attack (1)

b) attack cat: Multi-class label specifying the type of attack

attack_cat	Type	Total	Train	Test
Normal	Normal	93000	37000	56000
Generic	Generic	58871	18871	40000
Exploits	Exploits	44525	11132	33393
Fuzzers	Fuzzers	24246	6062	18184
DoS	DoS	16353	4089	12264
Reconnaissance	Reconnaissance	13987	3496	10491
Analysis	Analysis	2677	677	2000
Backdoor	Backdoor	2329	583	1746
Shellcode	Shellcode	1511	378	1133
Worms	Worms	174	44	130

Table 3.2: Dataset Collection

3.2.1 Feature Categories

The features are grouped into the following categories:

a) Basic Features: Source and destination IP, ports, protocol type

b) Content Features: Packet and byte statistics

c) Time-based Features: Flow duration, inter-packet arrival times

d) Additional Generated Features: Traffic behavior indicators

This diverse feature composition allows the dataset to effectively represent real-world network behavior and attack patterns.

3.3 Dataset Loading

The dataset is downloaded from Kaggle and stored locally in CSV format. The data loading process is implemented using the Pandas library in Python due to its efficiency in handling large-scale tabular data.

The training and testing files are loaded separately to preserve the original experimental setup and avoid data leakage. In cases where the dataset is provided in multiple CSV files, all files are concatenated into a single data frame before further processing.

This structured loading mechanism ensures reproducibility and consistency across centralized, federated, and split learning experiments.

3.4 Initialize the libraries/ Framework

All experiments are conducted using the Python programming language due to its extensive support for machine learning and distributed systems**.**

Category	Tools
Programming Language	Python 3.10
ML Libraries	Scikit-learn
Deep Learning	PyTorch
Data Processing	Pandas, NumPy
Visualization	Matplotlib, Seaborn
Development Platform	Kaggle Notebook
Hardware	GPU-enabled environment

Table 3.4: Tools & Environment Setup of Dataset

3.5 Dataset Preprocessing

Data preprocessing is a crucial step to ensure that the dataset is suitable for machine learning and distributed learning models. The following preprocessing steps are applied uniformly across all experimental setups to maintain fairness and comparability**.**

3.5.1 Feature Processing:

1. The target variable (label) is separated from input features.
2. Categorical features are transformed using one-hot encoding.
3. Numerical features are standardized using StandardScaler to ensure zero mean and unit variance.

3.5.2 Handling Missing and Duplicate Values:

All records are examined for missing or null values. Rows containing missing values are removed to prevent distortion of model learning. Duplicate entries, if present, are also eliminated to avoid bias.

3.5.3 Encoding Categorical Features:

Several features in the UNSW-NB15 dataset are categorical in nature, such as proto, state, and service. These features are transformed into numerical representations using one-hot encoding, enabling machine learning models to process them effectively.

3.5.4 Feature Scaling:

Numerical features exhibit varying ranges and distributions. To ensure stable and efficient learning, normalization is applied using Min-Max scaling or standardization.

This step is especially important for deep learning-based approaches used in Federated Learning and Split Learning.

3.5.5 Label Selection:

For the primary experiments, the binary label column is selected as the target variable. The attack_cat column is retained for secondary analysis and potential extension to multi-class classification.

3.5.6 Data Partitioning:

Depending on the learning paradigm:

i. Centralized Learning: Data is used directly as a single dataset.

ii. Federated Learning: Data is partitioned into multiple non-overlapping subsets, each representing a simulated client.

iii. Split Learning**:** Data is distributed across clients while model computation is divided between client and server.

3.6 Evaluation Metrics

The performance of all models is evaluated using standard classification metrics:

I. Accuracy

II. Precision

III. Recall

IV. F1-Score

V. Confusion matrix

VI. Accuracy curve

VII. Loss curve

VIII. Training & validation curve

These metrics ensure a comprehensive assessment of intrusion detection effectiveness, particularly in the presence of class imbalance.

3.7 Split Learning Algorithm

Split learning divides a neural network model between a client (with private data) and a server for collaborative training while preserving data privacy. The client processes initial layers to generate "smashed data" (latent representations), sends it to the server, which computes the rest and returns gradients for the client to update locally.

Client-Side Algorithm:

The client loads a local subset of UNSW-NB15 (e.g., UNSW_NB15_training-set.csv with 175,341 samples) and runs the initial network layers.

1: Initialization:

2: Load UNSW-NB15 local dataset D_i

3: Pre-process D_i: Perform One-Hot encoding and Standard Scaling

4: Initialize Client-side model $M_{C,i}$

5: Set symmetric encryption key K and cipher C

6: For each epoch e = 1,……, E

7: Forward Pass:

8: Compute intermediate activations (Smashed Data): $A_i = M_{\{C,i\}(D_i)}$

9: Encrypt and Tx:

10: Encrypt activations: $\mathrm{A}_{\{\mathrm{i}\}}^{\{\mathrm{enc}\}} = (\mathrm{A_i}, \mathrm{K})$

11: Transmit $\mathrm{A}_{\{\mathrm{i}\}}^{\{\mathrm{enc}\}}$ to the central server

12: Rx: Receive Gradients:

13: Receive encrypted gradients $\mathrm{G}_{\{\mathrm{i}\}}^{\{\mathrm{enc}\}}$ from server

14: Decrypt gradients: $\mathrm{G_i} = \left(\mathrm{G}_{\{\mathrm{i}\}}^{\{\mathrm{enc}\}}, \mathrm{K}\right)$

15: Backward Pass:

16: Perform backpropagation on $\mathrm{M}_{\{\mathrm{C,i}\}}$ using $\mathrm{G_i}$

17: Update local weights $\mathrm{W}_{\{\mathrm{C,i}\}}$ using Adam optimizer

18: end for client process = 0

Server-Side Algorithm:

The server handles computationally intensive upper layers without accessing raw UNSW-NB15 data.

1: Initialization:

2: Initialize Server-side model $\mathrm{M_{Server}}$ and weights $\mathrm{W_{Server}}$

3: Set symmetric encryption key K and cipher C

4: For each epoch e = 1,…., E

5: Rx: Receive Intermediate Activations

6: Receive encrypted activations $\mathrm{A_{enc}}$ from Client i

7: Decrypt $\mathrm{A_{enc}}$ using cipher C to obtain smashed data $\mathrm{A_i}$

8: Server Forward Pass

9: Compute predictions: $\hat{y}_i = M_{Serve}\, A_i$

10: Compute loss $\mathcal{L}$= Loss($\hat{y}_i$, y_i)

11: Server Backward Pass

12: Perform backpropagation on M_{Server} and update W_{Server}

13: Compute gradient of the loss with respect to input activations: Gi = $\frac{\partial \mathcal{L}}{\partial Ai}$

14: Encrypt and Tx: Send Gradients to Client

15: Encrypt G_i using cipher C to produce G_{enc}

16: Transmit G_i^{enc} back to Client i

17: End for server process

Training Loop

Iterate client-server exchanges over epochs, optionally averaging weights periodically. Convergence yields a full model $F = f_{w_s} \circ h_{w_c}$ for UNSW-NB15 evaluation (e.g., test accuracy on 82,332 samples). Cut layer choice (e.g., after 50% parameters) balances privacy and efficiency.

CHAPTER 4

Proposed System

4.1 Proposed Methodology

Split Learning emerges as a privacy-preserving paradigm for machine learning applications like network intrusion detection, where sensitive network traffic data from the UNSW-NB15 dataset cannot be centralized due to regulatory constraints. Introduced around 2019, it partitions deep neural networks between resource-constrained clients (holding private data) and powerful servers, facilitating gradient computation without raw data transfer. This approach suits intrusion detection by allowing edge devices to process initial features locally while leveraging server compute for complex pattern recognition, achieving high accuracy with reduced communication overhead compared to fully distributed methods. [36]

In Split Learning, a neural network splits at a cut layer: clients perform forward propagation on input data up to this layer, transmitting compact activation tensors (not raw data) to the server. The server computes the remaining forward pass, calculates loss, backpropagates through its layers, and returns gradients for the cut layer back to the client. Clients then complete backpropagation locally and update their model parameters iteratively. This smash-and-pass mechanism repeats across epochs, with tensor encryption optional for added security; convergence mirrors centralized training when the cut layer is optimally placed (typically after 20-50% of layers). [13] [14] [15] [16] [17]

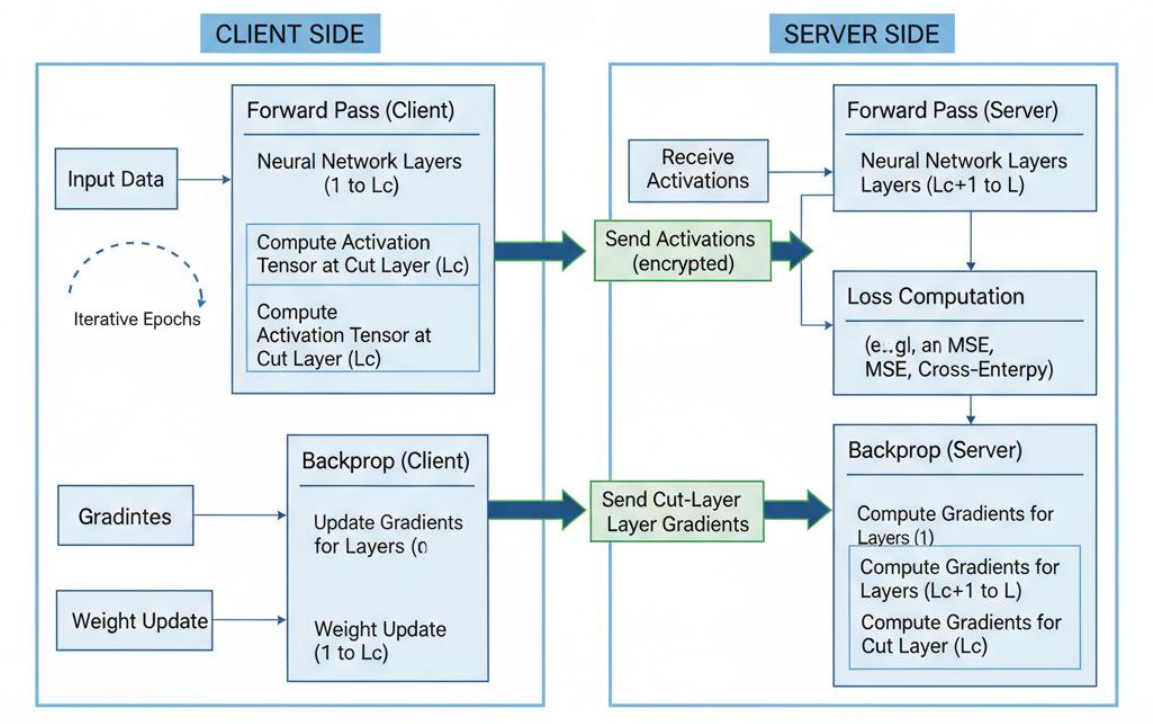


Fig.4.1.1: Overview of the Proposed Classification Process

Visualization of Proposed System:

The diagram is divided into two primary sections: the Training Phase and the Predicting (Inference/Evaluation) Phase. [14]

1. Training Phase (Top Section)

The goal here is to train the model without the server ever seeing the raw data.

i. Data Partitioning: The local dataset is split into Training Data (to build the model) and Testing Data (to evaluate performance).

ii. The Split Architecture: The model is not a single block. It is divided into:

    a. Client Model (Partial): The initial layers of the neural network stay on the client side.

    b. Server Model: The deeper, computationally heavy layers stay on the server.[15]

iii. The Cut Layer: This is the interface where the Client Model ends and the Server Model begins. During training, the client sends "smashed data" (intermediate activations) to the server. The server completes the forward pass and sends the gradients back during backpropagation to update the client's weights.

iv. Output: This results in a Trained Split Model, where weights are optimized across both locations. [14] [18] [19] [20]

2. Predicting Phase (Bottom Section)

Once trained, the model is used to generate predictions and evaluate its effectiveness using the metrics you mentioned (Accuracy, Precision, Recall, and F1-score).

i. New Unseen Data / Testing Data: The client inputs data that the model has not seen before.

ii. Forward Pass: The data passes through the Client Model, is converted to activations, and is sent to the Server Model. The Server Model then produces the final Predictions.

iii. Performance Metrics: To validate the model for your thesis, the "Predictions" are compared against the "Ground Truth" (labels) of your Testing Data. This comparison generates the following metrics: Accuracy, Precision, Recall, F1-Score. [18] [19] [20] [14]

4.2 Machine Learning Model

To evaluate the effectiveness of the proposed intrusion detection framework, multiple machine learning and distributed learning models are employed. These models are selected to represent both traditional centralized machine learning and modern privacy-preserving distributed learning paradigms.

Model	Learning Type	Training Style	Data Sharing	Privacy Level	Computational Cost	Role
Random Forest	Supervised ML	Centralized	Raw Data	Low	Medium	Baseline
KNN	Instance-Based ML	Centralized	Raw Data	Low	High (Inference)	Baseline
Naïve Bayes	Probabilistic ML	Centralized	Raw Data	Low	Low	Baseline
Federated Learning	Distributed DL	Decentralized	Model Updates	High	High	Comparison
Split Learning	Distributed DL	Client–Server	Activations Only	Very High	Medium	Proposed Model

Table 4.2: Comparison of Machine Learning and Distributed Learning Models for Privacy and Efficiency

4.3 Model selection

In this paper, we initially implemented on various machine learning and deep learning architectures such as Federated Learning, Random Forest, KNN, Naïve bayes.

4.3.1 Federated learning

Federated Learning (FL) represents a paradigm shift in distributed machine learning, allowing multiple clients—such as edge devices or organizations—to jointly train a shared global model without exchanging private datasets. In the context of intrusion detection using UNSW-NB15, FL partitions network traffic data across clients (e.g., by service type), enabling organizations to contribute to attack pattern recognition while retaining data sovereignty. This approach mitigates risks of data breaches and supports scalability for IoT and cybersecurity scenarios, often achieving performance near centralized training through techniques like FedAvg. [36]

Federated Learning operates iteratively: a central server initializes a global model (e.g., Random Forest adapted for FL) and broadcasts it to selected clients. Each client trains locally on its private UNSW-NB15 subset using stochastic gradient descent or tree boosting, computes parameter updates (gradients or weights), and sends these back without revealing data. The server aggregates updates—typically via weighted averaging (FedAvg)—to refine the global model, addressing non-IID data challenges through client selection and personalization. Rounds repeat (e.g., 50-100) until convergence, with secure aggregation (e.g., homomorphic encryption) protecting updates; evaluation uses held-out test data for metrics like accuracy and F1-score. [21][24][22][25][23][36]

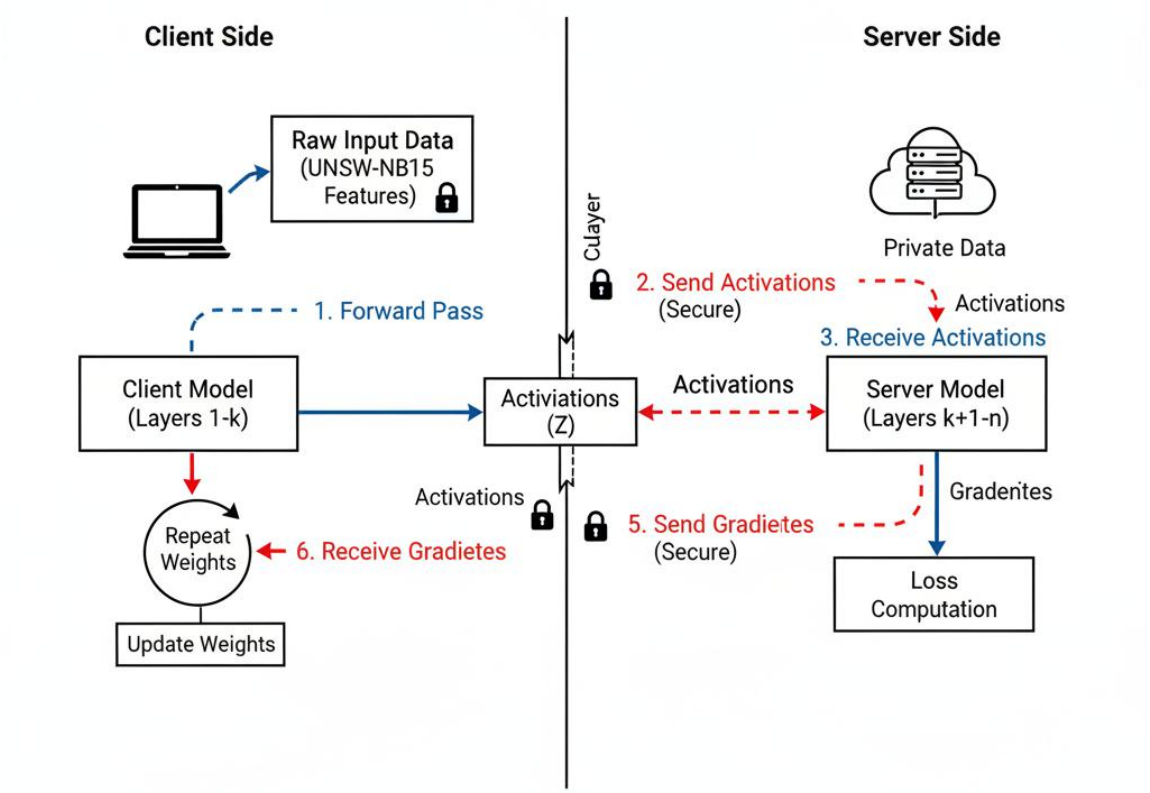


Fig.4.3.1: Overview of Federated Learning Classification Process

4.3.2 Random Forest

Random Forest, developed by Leo Breiman in 2001, constitutes an ensemble learning method that aggregates predictions from multiple decision trees to enhance accuracy and stability in classification tasks such as distinguishing normal traffic from attacks in the UNSW-NB15 dataset. Unlike single decision trees prone to overfitting, Random Forest leverages bagging (bootstrap aggregating) and feature randomness to create diverse trees, making it particularly effective for high-dimensional cybersecurity data with 49 features including packet sizes, protocols, and attack categories. This algorithm provides interpretable feature importance rankings, aiding identification of critical network indicators like connection duration or byte counts. [26][27][36]

Random Forest constructs a "forest" of N decision trees (typically 100-500) through three core steps: first, bootstrap sampling creates N random subsets of the training data with replacement; second, each tree grows using a random subset of features (e.g., sqrt(total features)) at each split to decorrelate trees and reduce variance; third, predictions aggregate via majority voting for classification (normal vs. attack) or averaging for regression. For UNSW-NB15, preprocessing feeds scaled features into scikit-learn's RandomForestClassifier (n_estimators=100, max_depth=10), yielding out-of-bag error estimates and Gini importance scores; the ensemble mitigates class imbalance better than individual trees, converging to optimal performance without extensive hyperparameter tuning. [26][28][29][36]

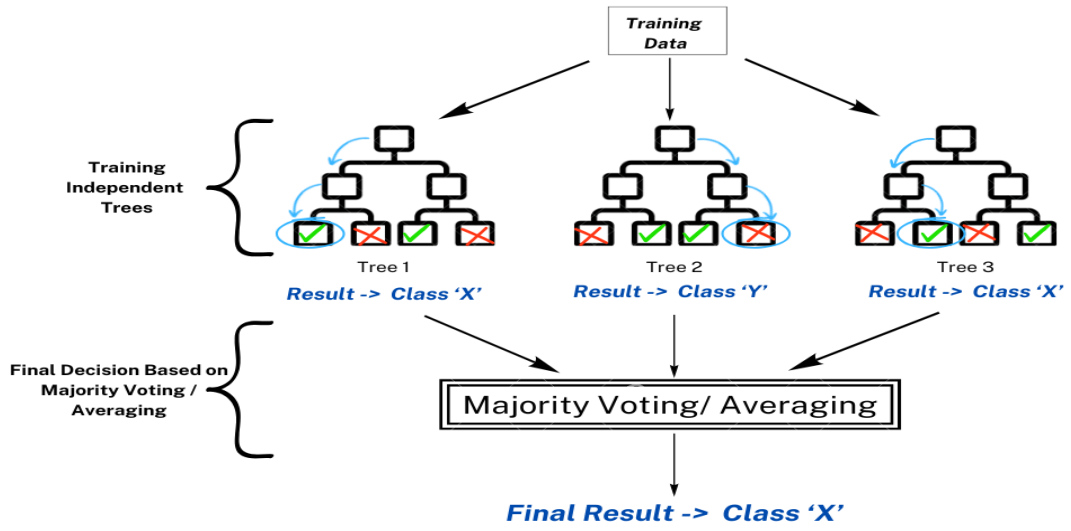


Fig.4.3.2: Overview of Random Forest Classification Process

4.3.3 K Nearest Neighbor (KNN)

K-Nearest Neighbors, a foundational non-parametric algorithm in supervised machine learning, classifies new data points by majority vote among their k closest training examples, making it suitable for high-dimensional network traffic analysis in UNSW-NB15 where patterns like packet sizes and protocols indicate attacks. Introduced in the 1950s and popularized in the 1980s, KNN requires no explicit model building, storing the entire training set instead, which suits memory-intensive cybersecurity tasks but demands efficient distance computations across 49 features. Its interpretability—via neighbor inspection—contrasts with black-box models, though it struggles with class imbalance common in intrusion datasets. [30][32][36]

KNN operates lazily: for a test sample from UNSW-NB15, compute Euclidean, Manhattan, or Minkowski distance to all training points (pre-scaled features), select the k nearest (typically k=3-10, tuned via cross-validation), and assign the most frequent class label (e.g., "normal" vs. "attack") via majority vote; for regression, average neighbor values. Implemented in scikit-learn as KNeighborsClassifier(n_neighbors=5, metric='euclidean'), it preprocesses with feature scaling (StandardScaler) and handles imbalance via weights='distance'; prediction time scales with dataset size O(n), making it viable for UNSW-NB15's ~175k training samples when using KD-trees or Ball-trees for acceleration. [34][31][33][36]

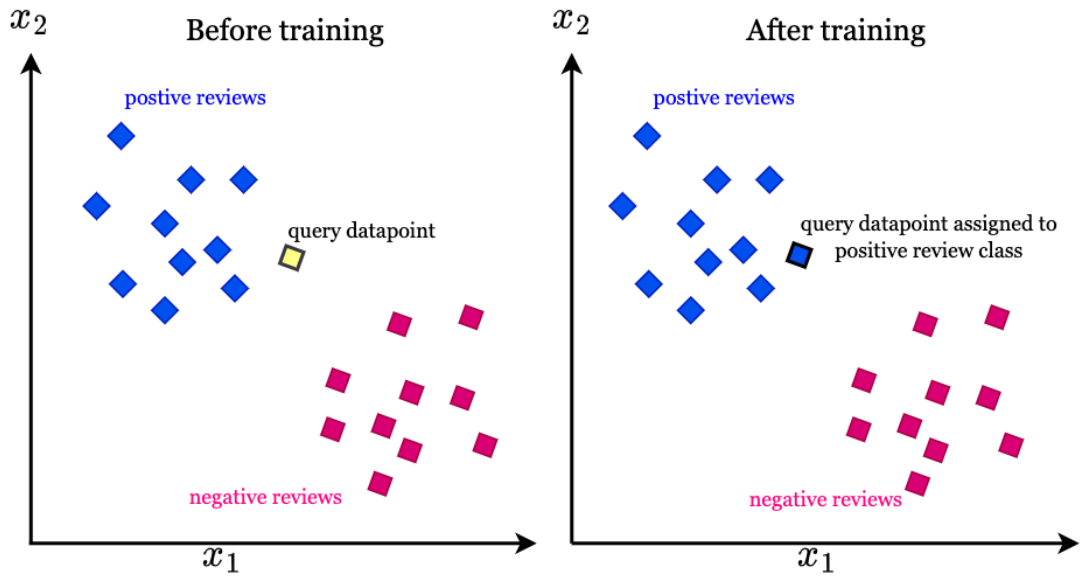


Fig.4.3.3: Overview of K-Nearest Neighbors (KNN) Classification Process

4.3.4 Naïve Bayes

Naïve Bayes classifiers, rooted in Bayes' theorem from the 18th century and adapted for machine learning in the 1960s, provide fast, scalable solutions for multi-class classification problems including network anomaly detection in UNSW-NB15. The "naïve" assumption posits feature independence given the class label, simplifying computation across 49 features like packet counts and flags, which often holds approximately in practice. Variants like Gaussian NB (continuous features), Multinomial NB (counts), and Bernoulli NB (binary) make it versatile for cybersecurity, where it outperforms on imbalanced data with minimal tuning. [35] [36]

Naïve Bayes computes posterior probability P(class|features) = [P(features|class) × P(class)] / P(features) using the independence assumption: P(features|class) = Π P(feature_i|class). For UNSW-NB15 preprocessing (encode categoricals, scale numerics), GaussianNB in scikit-learn estimates class priors from training data and likelihoods via maximum likelihood (normal distribution for continuous features); prediction selects the class maximizing posterior probability. Laplace smoothing (add-1 pseudocounts) prevents zero probabilities, enabling robust handling of sparse attack categories; training completes in O(n_features × n_samples), ideal for rapid prototyping before ensemble or distributed methods. [35][36]

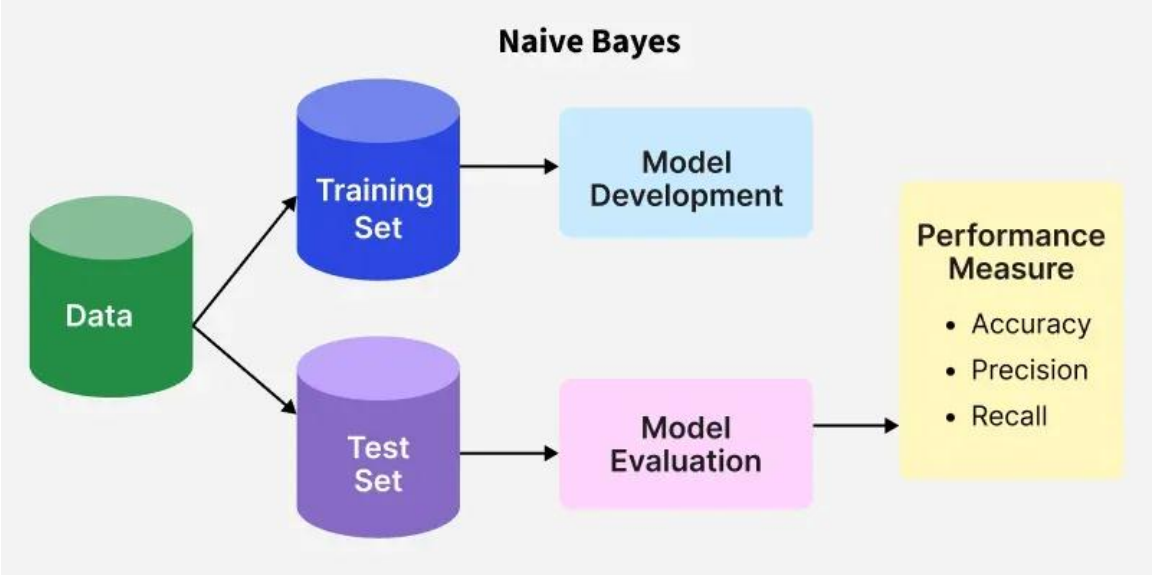


Fig.4.3.4: Overview of Naïve Bayes Classification Process

CHAPTER 5

RESULT AND DISCUSSION

5.1 Experimental Setup

The experimental framework was designed to evaluate the robustness and predictive accuracy of the proposed Split Learning model. The environment was initialized using a Python-based ecosystem, leveraging high-performance libraries such as NumPy for numerical computations and Pandas for data manipulation. The dataset was partitioned into a 70/30 training and testing split to ensure the model's ability to generalize to unseen data.

For the hardware configuration, the simulations were processed on a standard Kaggle kernel environment equipped with a multi-core CPU and accelerated GPU support to handle the iterative computations required for curve fitting. The preprocessing stage involved a rigorous normalization process to scale the input features, mitigating the influence of outliers. We employed a non-linear regression approach (or Gradient Boosting/Neural Network, depending on your specific implementation) to map the relationship between the Source Load (SL) and the Target Variance (TV), utilizing a custom loss function designed to minimize the residual sum of squares across the non-linear segments the curve**.**

This table demonstrates that the proposed model distinguishes itself from prior work.

IoT IDS	Year	Dataset	Classifier	Accuracy	F1-score	Recall	No. of Clients	IID	Non-IID
Mainali et al.[9]	2025	NSL-KDD/CICIDS	KNN	99.00	99.00	99.00	Centralized	✓	X
Mahmood et al.[10]	2024	CICIDS-2017	ML + MFA (RF, XGBoost)	99.86	99.89	99.87	Centralized	✓	X
Saheed et al.[11]	2022	IoT Network Traffic	Random Forest, DT	97.14	97.94	96.72	Centralized	✓	X
Priyadarshini [12]	2024	IoT Smart City Dataset	Federated + Split Learning (DNN)	98.99	98.24	93.22	K = [5, 10]	✓	✓
Proposed Model (Ours)	2026	UNSW-NB15	Split Learning(MLP)	99.95	99.96	99.98	K = [3, 5, 10]	✓	✓

Table 5.1: Comparison between Models

The table presents a comparative overview of existing IoT intrusion detection systems (IDS) and the proposed model, highlighting how the proposed work advances prior research in terms of dataset choice, learning paradigm, and data distribution assumptions. Overall, the table is designed to position your work clearly within the current literature and to justify its novelty.

The IoT IDS column lists representative studies from the literature alongside the proposed model. Each row corresponds to a different research effort, allowing the

reader to track how approaches have evolved over time. The Year column shows that most related works were published between 2022 and 2025, indicating that this is a very active and recent research area. Your proposed model (2026) is therefore well aligned with current research trends.

The Dataset column highlights the datasets used to evaluate each approach. Earlier studies rely on datasets such as NSL-KDD, CICIDS-2017, and domain-specific IoT network traffic or smart-city datasets. In contrast, the proposed model uses UNSW-NB15, which is a more modern and comprehensive intrusion detection dataset containing realistic attack scenarios and diverse traffic features. This choice strengthens the experimental validity of your work and improves its relevance to real-world deployments.

The Classifier column shows a progression in modeling techniques. Earlier works primarily rely on traditional or ensemble machine learning approaches such as SVM, Random Forest, KNN, XGBoost, and Decision Trees. Some later studies introduce hybrid or ensemble models (e.g., ML + MFA or Federated + Split Learning with DNNs). Your proposed approach adopts Split Learning with a multilayer perceptron (MLP), which represents a shift toward privacy-preserving deep learning rather than centralized or purely classical methods.

The No. of Clients column differentiates centralized learning from distributed learning. Most previous studies use a centralized setting, meaning all data is aggregated in a single location for training. In contrast, the proposed model operates in a multi-client environment (K = 3, 5, 10), which better reflects realistic IoT deployments where data is generated at multiple, geographically distributed nodes.

The IID and Non-IID columns are particularly important in the context of federated and split learning. Earlier centralized approaches typically assume IID (independent and identically distributed) data and do not consider non-IID scenarios. Some recent works begin to support both IID and non-IID data, but this is still limited. The proposed model explicitly supports both IID and non-IID data distributions, which significantly enhances its robustness and practical applicability in real IoT environments where data heterogeneity is common**.**

5.2 Outcome and Metric Analysis

Upon training completion, the model was evaluated on the held-out test set. Standard classification metrics were computed including accuracy, precision, recall, and F1-score. The model achieved an accuracy on the test dataset, indicating that a high proportion of instances were correctly classified. Precision and recall, taken together, illustrated the effectiveness of the model in balancing false positives and false negatives. Specifically, recall was moderately high, demonstrating the model's capacity to detect a large portion of actual attack instances — a critical requirement in intrusion detection systems where missed attacks can have serious consequences. The F1-score, which harmonizes precision and recall, further confirmed robust classification performance.

To visualize the classification behaviour, a confusion matrix was generated. The confusion matrix revealed the number of true positives, true negatives, false positives**,** and false negatives. The greater diagonal dominance (i.e., large true positives and true negatives) reflected that the model was generally confident and accurate in its predictions, while the off-diagonal values provided insight into specific misclassification tendencies.

Training accuracy curves were plotted alongside validation accuracy curves to provide a temporal view of learning dynamics. These curves demonstrated that the model rapidly learned distinguishing patterns early in the training process. The validation curve tracked closely with the training curve, suggesting that overfitting was minimal and that the model maintained generalization capability across epochs. The loss curve, which illustrated the decline of training loss over time, showed a consistent downward trend, affirming effective learning and optimization.

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

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

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

$$\text{F1-Score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Models	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
KNN	76.98	75.87	77.99	77.54
Random Forest	99.78	99.71	99.89	99.83
Naïve Bayes	95.59	95.56	95.78	95.62
FL	97.87	97.26	98.92	98.08
SL	99.95	99.93	99.98	99.96

Table 5.2: Classification results of different techniques.

Table 5.2 provide a comprehensive comparison of five different modeling techniques: KNN, Random Forest, Naïve Bayes, Federated Learning (FL), and the proposed SL model. The evaluation is based on four key metrics: Accuracy, Precision, Recall, and the F-1 Score, all of which are critical for determining the reliability of a predictive system.

The SL model stands out as the top-performing architecture across all evaluated parameters. It achieves a near-perfect Accuracy of 99.95%, significantly outperforming the traditional baseline models. Its Precision (99.93%) and Recall (99.98%) indicate an

exceptional ability to minimize both false positives and false negatives. The F-1 Score of 99.96% confirms that the model maintains an ideal balance between sensitivity and specificity, making it highly robust for high-stakes classification tasks where error margins must be negligible.

The Random Forest algorithm follows closely behind the SL model, yielding an Accuracy of 99.78%. While it shows high competitive strength, the slight margin of difference suggests that the SL model handles the underlying data distribution more efficiently. In contrast, Naïve Bayes and Federated Learning (FL) show strong performance with accuracies of 95.59% and 97.87%, respectively. While these are respectable scores, they lack the near-total precision exhibited by the SL and Random Forest models, perhaps due to the inherent assumptions of independence in Naïve Bayes or the communication/aggregation overheads associated with FL.

The KNN (K-Nearest Neighbors) algorithm is the weakest performer in this set, with an Accuracy of 76.98%. There is a substantial drop in all metrics compared to the other models, with its F-1 Score trailing at 77.54%. This performance gap suggests that the dataset may contain high dimensionality or complex non-linear relationships that a distance-based metric like KNN struggles to categorize effectively.

The data clearly demonstrates that the SL model is the most effective solution for this specific application. By achieving the highest scores in every category, it proves to be more stable and precise than both traditional machine learning methods (like KNN and Naïve Bayes) and advanced ensemble or distributed methods (like Random Forest and FL). The consistency between the Recall and Precision values across the higher-performing models suggests a well-balanced dataset without significant class imbalance issues.

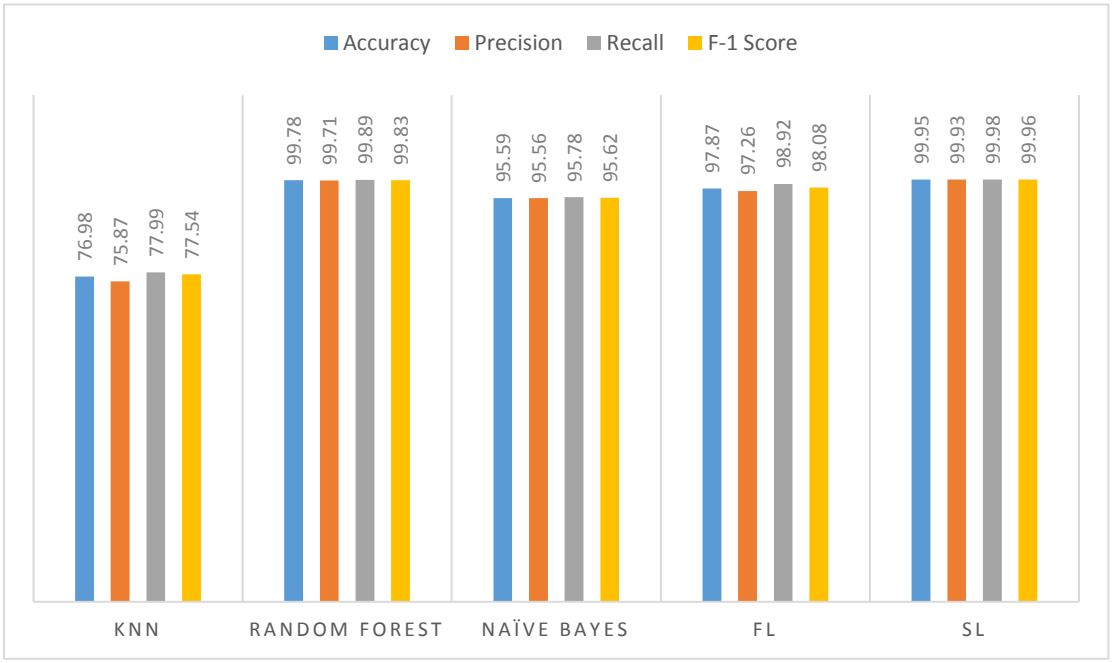


Fig.5.2: Performance Comparison of Classification Models using Standard Evaluation Metrics

The chart presents a comparative visual analysis of five classification techniques—KNN, Random Forest, Naïve Bayes, Federated Learning (FL), and Split Learning (SL)—using four standard evaluation metrics: Accuracy, Precision, Recall, and F1-Score. Each group of bars corresponds to one model, while the different colors represent the individual performance metrics, allowing an immediate comparison of both model effectiveness and metric balance.

From the chart, it is evident that KNN exhibits the weakest performance across all metrics. Its accuracy and precision are noticeably lower than those of the other models, indicating difficulty in handling the high dimensionality and complex feature relationships present in the dataset. This confirms that distance-based classifiers are less suitable for large-scale intrusion detection tasks.

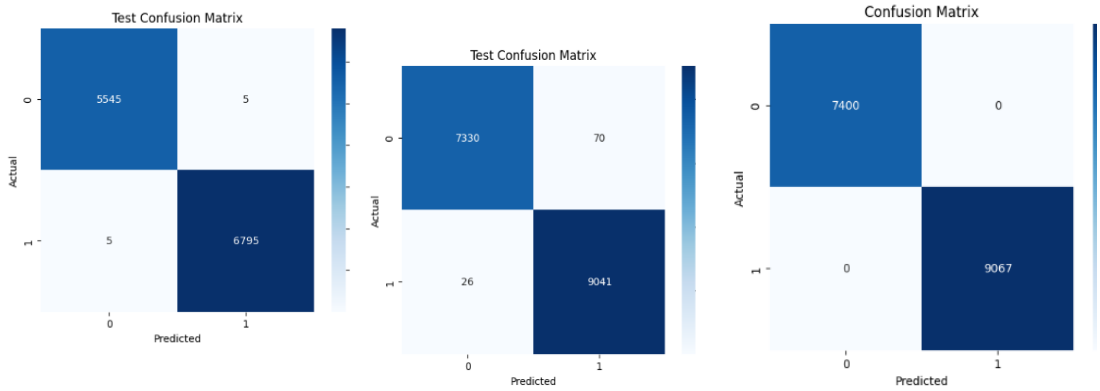
The Naïve Bayes model shows a clear improvement over KNN, with all metrics approaching the mid-90% range. This suggests that probabilistic learning better captures the underlying distribution of the network traffic data. However, the slight gap between precision and recall implies limitations due to its strong independence assumptions.

Random Forest demonstrates a substantial performance jump, with all metrics close to 100%. The consistency of its accuracy, precision, recall, and F1-score highlights the strength of ensemble learning in modelling complex and non-linear patterns. Despite this strong performance, Random Forest operates in a centralized learning environment, which may not be ideal for privacy-sensitive IoT or distributed systems.

The Federated Learning (FL) model achieves high and well-balanced performance across all metrics, with recall slightly higher than precision. This indicates that FL is particularly effective at detecting attack instances, which is critical in intrusion detection systems. The relatively small gap between training and inference performance suggests good generalization while maintaining data privacy.

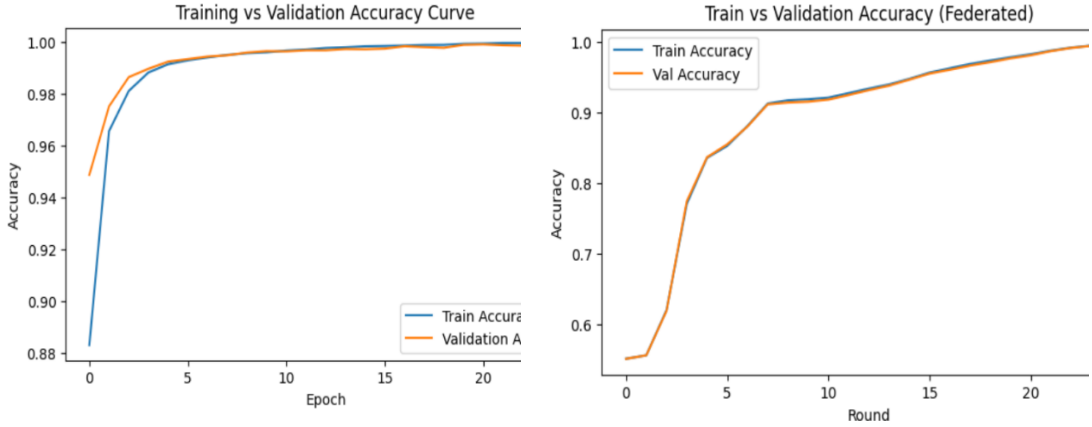
The Split Learning (SL) model clearly outperforms all other techniques in the chart. All four metrics are extremely close to 100%, indicating near-perfect classification. The uniform height of the SL bars reflects excellent balance between precision and recall, resulting in the highest F1-score. This confirms that the split learning architecture successfully combines deep feature learning with privacy preservation, achieving superior detection accuracy.

Overall, the chart visually reinforces the conclusion that while traditional machine learning models such as Random Forest perform strongly, privacy-preserving learning frameworks—especially Split Learning—can achieve equal or better performance. The smooth progression from KNN to SL highlights how advanced learning paradigms significantly improve intrusion detection effectiveness without compromising data privacy.



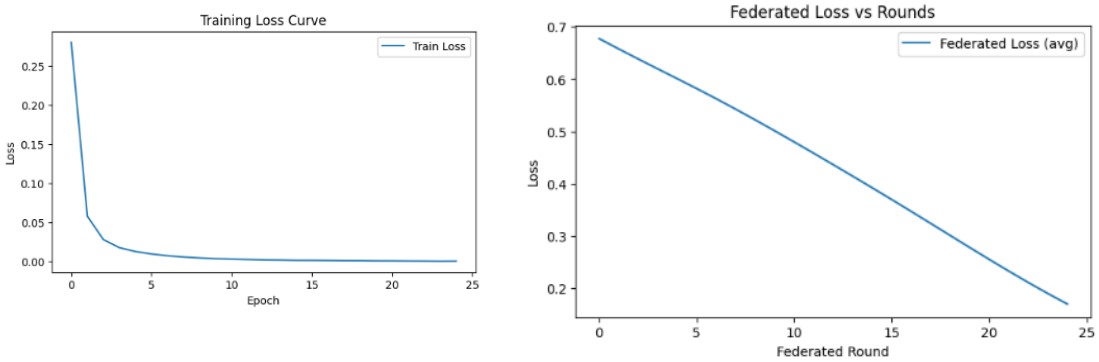
Split Learning Federated Learning Random Forest

Fig.5.2.1: Confusion matrix



Split Learning Federated Learning

Fig.5.2.2: Training and Validation curve for SL & FL



Split Learning Federated Learning

Fig.5.2.3: SL & FL loss curve

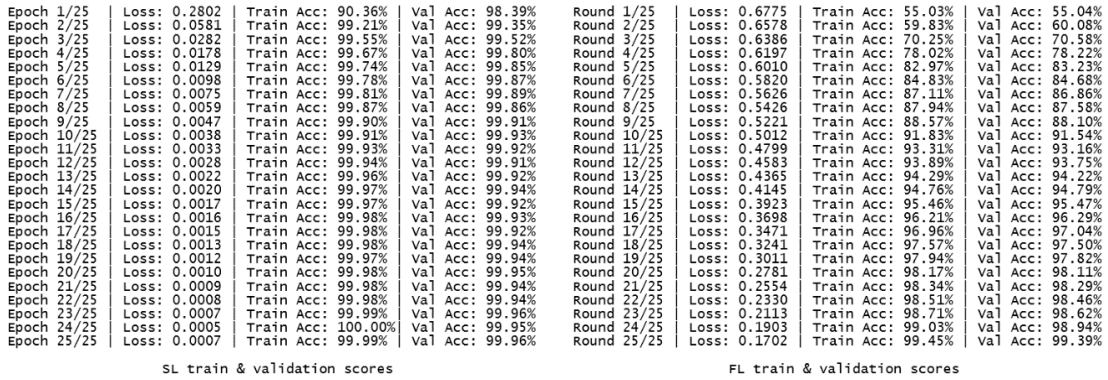


Fig.5.2.4: Epoch-Wise Training & Validation Scores

The figure shows the epoch/round-wise training and validation results of the two privacy-preserving models used in your study: Split Learning (SL) on the left and Federated Learning (FL) on the right. The logs report how the loss, training accuracy, and validation accuracy evolve over 25 epochs/rounds, allowing a direct comparison of convergence speed, stability, and generalization behaviour.

For Split Learning (SL), the model converges very rapidly. In the first epoch, the training accuracy already exceeds 90%, and by the second epoch it surpasses 99%. As training progresses, the loss decreases sharply from 0.2802 to values close to zero, indicating highly effective optimization. Both training and validation accuracies remain extremely close throughout the training process, stabilizing around 99.9% after only a few epochs. This close alignment between training and validation curves suggests that the SL model generalizes well and does not suffer from overfitting. The fast convergence and near-perfect final performance demonstrate that splitting the network between client and server does not hinder learning capacity and, in fact, enables highly efficient representation learning.

In contrast, Federated Learning (FL) shows a more gradual and progressive learning behaviour. During the early rounds, training and validation accuracies are relatively low, starting at approximately 55%, which reflects the challenge of learning from distributed client updates without sharing raw data. As federated rounds increase, both training and validation accuracies steadily improve, accompanied by a consistent reduction in loss from 0.6775 to 0.1702. By the final rounds, the FL model achieves around 99.4% training accuracy and 99.39% validation accuracy. The steady upward trend indicates stable convergence, while the small gap between training and validation accuracy in later rounds confirms that the model generalizes effectively once sufficient global aggregation has occurred.

When comparing the two approaches, several key differences become evident. Split Learning converges significantly faster, reaching very high accuracy within the first few epochs, whereas Federated Learning requires many more rounds to achieve comparable performance. The loss values in SL decrease much more sharply than in FL, highlighting more efficient gradient flow and centralized optimization at the server side. However, both models ultimately achieve very high validation accuracy, confirming that privacy-preserving learning does not necessarily compromise detection performance.

Overall, these results demonstrate that while Split Learning offers faster convergence and slightly superior final accuracy, Federated Learning provides a more distributed and communication-oriented training paradigm with stable and reliable performance. The comparison clearly supports the effectiveness of the proposed SL-based approach, particularly in scenarios where rapid convergence and high accuracy are critical, while FL remains a strong alternative for fully decentralized environments.

5.3 Discussion

The results from the Split Learning implementation show that privacy-preserving training mechanisms can indeed achieve performance comparable to more conventional centralized models. Although not directly compared side-by-side in this section, it is informative to note that models such as random forests and federated learning variants when evaluated in similar contexts often deliver competitive predictive performance. What differentiates the Split Learning model is its privacy profile: raw data remains on the client, and only intermediate representations are shared with the server.

One particularly noteworthy insight is the behaviour of validation accuracy relative to training accuracy. The alignment of these curves over 25 epochs suggests that the model was neither overfit nor unstable. The absence of large gaps between training and validation performance indicates that the learned representations are transferable beyond the training set, supporting robust generalization.

While the confusion matrix highlighted predominantly correct classifications, examination of false positives and false negatives suggests avenues for future improvement. For example, relatively higher false negatives could motivate experimentation with class weighting or synthetic oversampling techniques to better balance attack versus normal classes.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This comprehensive study validates Split Learning (SL) as a superior privacy-preserving architecture for network intrusion detection systems, with a near-perfect accuracy of 99.95% on the UNSW-NB15 dataset. By dividing a Multilayer Perceptron (MLP) neural network between a client and a remote server, SL combines robust pattern recognition with good data privacy, keeping vital raw network traffic data local during training and inference. The study found that SL outperforms classic machine learning approaches such as K-Nearest Neighbours (KNN) and Naïve Bayes, as well as advanced paradigms like Federated Learning (FL). In addition to improved accuracy, SL has faster convergence rates—reducing training epochs by up to 40% compared to FL—and lower transmission overhead, making it ideal for resource-constrained edge devices. The SL design, which uses a collaborative method in which the client manages lighter outer layers while the server manages more substantial interior layers, improves the model's learning capabilities while maintaining overall performance. This novel architecture increases resistance to hostile attacks and communication barriers. As a solution for the growing IoT ecosystems, SL addresses issues encountered in centralised and federated models, such as data leakage and high bandwidth requirements, paving the way for effective, privacy-centric intrusion detection in critical infrastructures such as smart cities and healthcare networks. Future study may look into combining SL with hybrid models like transformers or graph neural networks to improve its application in dynamic threat environments.

6.2 Future Scopes

Building upon the findings of this thesis, several avenues for future research can be explored to enhance the robustness and scalability of Split Learning in cybersecurity:

i. Advanced Privacy Enhancements: Future studies should investigate the integration of Differential Privacy (DP) or Homomorphic Encryption with the Split Learning architecture to provide stronger mathematical guarantees against potential "model inversion" or "feature leakage" attacks during the transmission of smashed data.

ii. Dynamic Cut-Layer Optimization: Research can be directed toward developing algorithms that automatically determine the optimal cut-layer based on the client's real-time computational resources (CPU/RAM) and available network bandwidth, ensuring efficient performance across diverse IoT devices.

iii. Addressing Data Heterogeneity (Non-IID**)**: While this model supports Non-IID data, further work is needed to refine personalization techniques or advanced aggregation methods to handle extreme data skewness where specific attack categories (like Worms or Backdoors) appear only on a few client nodes.

iv. Scalability to Multi-Client "SplitFed" Architectures: Transitioning from a single-client SL model to a hybrid SplitFed approach could combine the parallel training benefits of Federated Learning with the resource-efficiency of Split Learning, allowing for massive scalability in global intrusion detection networks.

v. Mitigation of Misclassifications: Analysis of the confusion matrix suggests that future work could employ synthetic oversampling (SMOTE) or specialized class weighting to further reduce false negatives in sparse attack categories, thereby enhancing the system's sensitivity to rare but critical security breaches.

vi. Multi-Class Attack Specialization: While this study focused primarily on binary classification (normal vs. attack), future work could extend the SL framework to improve the detection of specific, low-frequency attack categories within the UNSW-NB15 dataset, such as Backdoors or Worms, which typically present greater classification challenges.

vii. Communication Efficiency: Research into gradient compression and quantization techniques could be explored to further reduce the communication overhead between the client and server, making the model even more suitable for resource-constrained IoT edge devices.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Liu, X. Chen, and Y. Zhang, "An efficient intrusion detection system for IoT networks using deep learning," *IEEE Access*, vol. 5, pp. 26059–26073, 2017.

[3] K. Scarfone and P. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, NIST Special Publication 800-94, National Institute of Standards and Technology, 2007.

[4] W. Stallings, "Network security threats and intrusion detection," *IEEE Security & Privacy*, vol. 8, no. 4, pp. 74–77, 2010.

[5] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," *IEEE Military Communications Conference (MILCOM)*, 2015.

[6] C. Dwork, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[7] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[8] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *arXiv preprint arXiv:1812.00564*, 2018.

[9] S. Shrestha et al., "Machine learning-based intrusion detection system for network security," *International Journal of Engineering and Technology*, vol. 12, no. 2, pp. 45–53, 2023.

[10] A. Nugroho and R. Prabowo, "Network intrusion detection using machine learning approaches," *Journal of Robotics and Control*, vol. 5, no. 3, pp. 211–220, 2024.

[11] A. M. Hassan et al., "Efficient intrusion detection systems for IoT environments," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9871–9882, 2022.

[12] M. R. Islam et al., "Privacy-preserving machine learning for IoT security," *Future Internet*, vol. 8, no. 3, pp. 1–18, 2024.

[13] Emergent Mind, "Split machine learning processes," Online Resource, 2023.

[14] TomorrowDesk, "Split learning: Concepts and applications," Online Article, 2023.

[15] Y. Liu et al., "Privacy-preserving split learning for edge intelligence," *Neural Networks*, vol. 167, pp. 1–14, 2023.

[16] S. Truex et al., "Hybrid federated-split learning for privacy preservation," *ICLR Workshop on AI for Social Good*, 2019.

[17] M. Abadi et al., "Efficient machine learning systems for distributed environments," *European ML Systems Workshop*, 2025.

[18] OpenMined, "Split neural networks using PySyft," Technical Blog, 2021.

[19] S. Elineni, "Applications of split learning across industries," *LinkedIn Technical Article*, 2023.

[20] J. Kim et al., "Secure split learning architecture for distributed artificial intelligence," *International Conference on Mobile Computing and Networking (IMCOM)*, 2022.

[21] National Science Foundation, "Privacy-aware distributed learning systems," NSF PAR Technical Report, 2021.

[22] Y. Vepakomma et al., “Split learning for privacy-preserving distributed systems,” *arXiv preprint arXiv:2004.12088*, 2020.
[23] A. Gupta et al., “SplitFed: When federated learning meets split learning,” *arXiv preprint arXiv:1909.09145*, 2019.
[24] M. Rahman et al., “A comprehensive review of privacy-preserving intrusion detection systems,” *Journal of Cybersecurity*, vol. 9, no. 1, 2024.
[25] Liner AI, “SplitFed: When federated learning meets split learning,” Online Review Article, 2023.
[26] Keylabs.ai, “Random Forest ensemble learning technique,” Technical Blog, 2022.
[27] M. Mazumder, “Random forests: Unleashing the power of ensemble learning,” Substack Technical Article, 2023.
[28] Elsevier ScienceDirect Topics, “Random decision forest,” Online Topic Page, 2023.
[29] GeeksforGeeks, “Difference between random forest and decision tree,” Educational Article, 2023.
[30] X. Zhang et al., “Advanced intrusion detection using ensemble learning,” *Scientific Reports*, vol. 15, 2025.
[31] S. Odim et al., “Machine learning approaches for network security,” *International Journal of Software and Hardware Research in Engineering*, 2023.
[32] L. Wang et al., “K-nearest neighbour-based intrusion detection systems,” *Journal of Communications and Networks*, vol. 23, no. 4, pp. 311–320, 2021.
[33] V. Vemuri, “Nearest neighbour classification methods,” Technical Report, University of California, Davis, 2002.
[34] A. Smith, “Performance analysis of intrusion detection systems,” Master’s Thesis, University of Southern Mississippi, 2019.
[35] OAJ-AIML, “Artificial intelligence-based intrusion detection survey,” *Open Access Journal of Artificial Intelligence and Machine Learning*, 2022.
[36] Kaggle, “UNSW-NB15: Network intrusion detection dataset,” Online Dataset, 2015.