Towards Privacy-Preserving Deep Learning for Intelligent IoT Botnet Detection

applied sciences

MDPI

Article

Towards Privacy-Preserving Deep Learning for Intelligent IoT Botnet Detection

Ariwan M. Rasool [1,*], Nader Sohrabi Safa [2,*] and Consolee Mbarushimana [1]

[1] Department of Computing and Mathematical Sciences, University of Wolverhampton, Wolverhampton WV1 1LY, UK; c.mbarushimana@wlv.ac.uk
[2] Department of Computing, University of Worcester, Worcester WR2 6AJ, UK
* Correspondence: a.m.rasool@wlv.ac.uk (A.M.R.); n.sohrabisafa@worc.ac.uk (N.S.S.)

Abstract

Internet of Things (IoT) botnets are networks of infected smart devices controlled by attackers and posing a serious cybersecurity challenge. Developing detection approaches that maintain high accuracy while protecting privacy presents considerable challenges, particularly in large and heterogeneous IoT networks. This paper empirically compares three modelling approaches on Bot-IoT and N-BaIoT in binary and multiclass settings: handcrafted machine learning with random forest (RF), centralised deep learning (CDL) with DNN/LSTM/BiLSTM, and federated deep learning (FDL) with the same architectures. Model hyperparameters are selected via randomised search on stratified subsets and then fixed for final training. Results show near-perfect performance for all approaches in binary detection: on Bot-IoT, CDL-DNN attains perfect accuracy, and RF is virtually perfect (only four benign-to-attack false positives), while FDL models are similarly strong with only small false-positive and false-negative counts. On N-BaIoT, RF and CDL (especially LSTM) are near-perfect, and FDL is very close to CDL. For multiclass detection, CDL-DNN leads on Bot-IoT, RF remains near perfect with minimal cross-class confusion, and FDL trails slightly; on N-BaIoT, FDL-BiLSTM and RF are essentially perfect, with CDL-LSTM close behind. Overall, the findings validate RF as a competitive classical approach, show where centralised representation learning adds value, and demonstrate that federated training preserves most of the centralised accuracy while avoiding raw data centralization (data locality) for scalable deployment.

Keywords: IoT; machine learning; deep learning; security and privacy; botnet detection; federated learning

Academic Editors: Gianluigi Ferrari and Douglas O'Shaughnessy

Received: 1 December 2025
Revised: 24 January 2026
Accepted: 30 January 2026
Published: 6 February 2026

1. Introduction

The Internet of Things (IoT) has expanded rapidly and is now embedded across domains such as smart homes, healthcare, and transportation. Despite these benefits, many IoT devices remain resource-constrained and are deployed with limited security controls, which increases their exposure to compromise and botnet recruitment (Rey et al. [1]; Zhang et al. [2]). One common threat is a distributed denial-of-service (DDoS) attack launched by IoT botnets, compromised devices that attackers remotely control to disrupt servers (Bala and Behal [3]). Due to a lack of reliable conventional approaches for detecting botnet attacks, researchers have started using machine learning (ML) and deep learning (DL) to create adaptive models that keep pace with evolving attacks. DL models showed good results in pattern recognition in IoT network traffic and are scalable to large datasets,

such as recurrent neural networks (RNNs) and long short-term memory (LSTM) (Ahmed et al. [4]).

Using federated deep learning is encouraged, as it is privacy-preserving by keeping raw traffic local (data locality), in contrast to centralised training, which requires collecting sensitive data centrally (S.I. Popoola et al. [5]). But DL models require large training datasets, which is a limitation in privacy-sensitive IoT environments. Thus, federated learning can address this problem by allowing devices to collaboratively learn a shared model while protecting data locally. That is why it can improve the security and efficiency of the detection model (A. Metwaly and Elhenawy [6]).

Some researchers have used federated learning to detect IoT botnets, achieving good results. For example, S.I. Popoola et al. [5] highlighted that high accuracy and low communication overhead can be achieved by using a federated deep learning model. Also, Lim et al. [7] emphasised the privacy-preserving capabilities of federated learning and the ability to train models locally, which reduces communication costs.

There are some dataset benchmarks, such as Bot-IoT and N-BaIoT, for IoT botnet detection models (Koroniotis et al. [8]; Meidan et al. [9]). They contain different types of attack scenarios and devices. This makes them more reliable for training detection models. Metrics, including accuracy, precision, recall (sensitivity), F1-score, area under the curve (AUC), receiver operating characteristic (ROC), and confusion matrix, are used to evaluate model performance (Patil and Deshpande [10]; Nazir et al. [11]).

Despite deep learning achieving good results in IoT botnet detection, several challenges remain. These challenges include resource constraints in IoT devices, which require a lightweight, energy-efficient model (S.I. Popoola et al. [12]). IoT device data is another challenge, often not independent and identically distributed, which can affect model performance if not addressed (Liu et al. [13]).

Some researchers, such as Popoola, proposed solutions to those problems, suggesting memory-efficient DL architectures that reduce computational overhead without affecting accuracy (S.I. Popoola et al. [12]). Attota et al. [14] improved detection rates and enhanced privacy by keeping data on end devices rather than transferring it externally. Moreover, Tian et al. [15] focused on non-independent and identically distributed (non-IID) IoT device data and developed a pre-shared data training strategy that prevents convergence divergence under non-IID patterns. However, the studies are limited and require further attention from researchers to improve the classification performance of botnet detection models while preserving data privacy through federated learning.

Some studies have investigated hyperparameter optimisation techniques for fine-tuning deep learning models in IoT botnet detection to improve performance (S. Popoola et al. [16]; Bensaoud and Kalita [17]). Deep neural networks (DNNs) can learn complex patterns, making them suitable for detecting botnets in IoT environments. These DNN models can provide better results by tuning their hyperparameters, including the number of hidden layers, activation function, learning rate, batch size, optimiser, and epochs. For example, S. Popoola et al. [16] propose a DNN-based model with an optimal set that achieves a high detection rate, but it is limited to DNNs.

In this paper, we focus on three deep learning architectures widely used for IoT intrusion detection—DNN, LSTM, and BiLSTM—alongside a strong handcrafted baseline based on random forest (RF). A random search strategy is applied to obtain a hyperparameter set for each of DNN, LSTM, and BiLSTM on stratified subsets (20% of the Bot-IoT dataset and 15% of the N-BaIoT dataset) for both binary and multiclass tasks. After obtaining a set of hyperparameters for each binary and multiclass task on both datasets, the resulting RS-best configurations were fixed and used for full-dataset evaluation, enabling standardised evaluation under identical experimental conditions across paradigms. The study then

compares centralised deep learning (CDL) and federated deep learning (FDL) with these tuned architectures against the RF baseline on Bot-IoT and N-BaIoT, providing empirical insights into the trade-offs among accuracy, privacy, and deployment constraints in IoT botnet detection.

This study addressed a focused research problem: in operational IoT environments, botnet detection must achieve high overall detection performance while avoiding centralisation of raw traffic; remain robust to heterogeneous, non-identically distributed client data; and handle severe class imbalance. Accordingly, the core scientific question investigated was as follows: to what extent can privacy-preserving federated deep learning preserve the detection performance of centralised deep learning when models are evaluated under the same experimental protocol and on the same benchmark datasets?

To answer this question systematically, the work focused on three aspects that are still relatively underexplored in combination: (i) systematic hyperparameter optimisation of deep learning models for IoT botnet detection; (ii) a standardised evaluation under identical experimental conditions of centralised and federated deep learning, including identical preprocessing, tuning, data splits, and evaluation metrics; and (iii) careful handling and explicit reporting of class imbalance effects on both binary and multiclass learning.

The study therefore addressed four research questions: (RQ1) how strong are RF and deep architectures on Bot-IoT and N-BaIoT in binary and multiclass settings? (RQ2) How does class imbalance, and the SMOTE ablation where reported, affect performance? (RQ3) How closely does FDL preserve CDL performance under non-IID client partitions? (RQ4) what trade-offs arise between detection performance and privacy-preserving (in the sense of data locality) decentralisation? The remainder of the paper is organised as follows: Section 2 reviews related work; Section 3 describes the datasets, leakage-safe preprocessing, and the CDL/FDL experimental protocol; and Section 4 reports and discusses the results.

The contributions of this paper are fourfold: (1) a reproducible evaluation framework that applies leakage-safe preprocessing and a standardised evaluation under identical experimental conditions to RF, CDL, and FDL across Bot-IoT and N-BaIoT in binary and multiclass settings; (2) systematic hyperparameter optimisation for DNN, LSTM, and BiLSTM using randomised search on stratified subsets, with RS-best configurations subsequently fixed for full-dataset evaluation to avoid confounding paradigm comparisons by re-tuning; (3) explicit analysis of class imbalance effects using complementary macro- and weighted-averaged metrics, including a supplementary experiment with SMOTE for Bot-IoT multiclass; and (4) client-level reporting for federated learning to characterise performance variability under non-IID client partitions.

2. Related Work

The number of IoT devices has grown very quickly in recent years and is now part of many aspects of daily life. At the same time, many of these devices are built with very limited computational resources and almost no built-in security, which makes them easy targets for attackers. When such devices are compromised and coordinated, the impact can spread well beyond a single household or organisation. This situation creates a clear need for detection models that can learn and recognise complex traffic behaviours rather than relying only on simple rules. Deep learning has attracted increasing attention for this reason, as it can automatically discover intricate patterns in network traffic and extract useful features from raw data without heavy manual engineering (Sahu et al. [18]).

A growing body of work now shows that centralised deep learning models often outperform traditional intrusion detection methods. Researchers have experimented with many architectures, including long short-term memory (LSTM) networks, bidirectional LSTMs (BiLSTMs), and deep neural networks (DNNs). These are typically evaluated on

benchmark datasets such as Bot-IoT and N-BaIoT, which contain a mixture of normal and attack traffic and are therefore well suited for supervised training and testing. In contrast, older signature-based systems struggle to keep up with modern, fast-changing threats because they rely on predefined rules and carefully handcrafted features.

Classical machine learning methods have not disappeared; they are still widely used as baselines for IoT botnet detection. Algorithms such as random forest (RF) and support vector machine (SVM) remain attractive because they are relatively easy to train and can handle mixed feature types. RF, in particular, is often highlighted for its robustness to noise, its ability to model non-linear relationships, and its relatively modest computational cost compared with deep networks. This makes RF a useful and practical point of comparison when assessing whether more complex centralised or federated deep learning models deliver meaningful performance gains.

However, even when centralised deep learning achieves very high accuracy, there are still two major obstacles to its use in real IoT environments. First, consolidating all training data into a single location introduces privacy and security risks, as sensitive traffic must be transmitted to and stored on a central server (Myakala, Kamatala, and Bura [19]). Second, training large deep models centrally requires substantial processing power and memory, which are typically unavailable on low-power IoT devices (Nguyen and Beuran [20]). These concerns have driven a shift towards federated learning, where models are trained collaboratively while data remains on the devices that generate it. In this setup, only model parameters or updates are sent to a server, which reduces the exposure of raw data and may reduce central storage/transfer requirements, making the approach more realistic for heterogeneous IoT deployments (Myakala, Kamatala, and Bura [19]; Nguyen and Beuran [20]; Chen et al. [21]).

Empirical studies on Bot-IoT and N-BaIoT generally confirm that deep learning models achieve strong detection results, but they also highlight that high computational and memory demands remain a barrier on resource-constrained devices (Popoola et al. [12]). To reduce this gap, some authors have proposed lightweight architectures that aim to balance detection accuracy and efficiency, making them more suitable for embedded hardware (Rahman et al. [22]). Federated deep learning builds logically on these ideas. In a typical setup, each IoT device trains a local model using its own data and sends only model updates to a central server, which aggregates them into a global model. In this way, privacy is preserved while performance remains close to that of centralised training, at least in many reported experiments (Kim et al. [23]; Gugueoth, Safavat and Shetty [24]).

At the same time, federated learning introduces its own set of challenges. It can be affected by adversarial or faulty clients that send poisoned updates, which can degrade the global model or slow convergence. These effects, in turn, increase communication rounds and energy consumption, which is problematic for devices that already operate under strict resource constraints (Yadav and Gupta [25]). In addition, federated systems must handle heterogeneous hardware, unbalanced and non-identically distributed data, and clients that may join or leave during training. Recent studies discuss countermeasures such as robust aggregation rules, model compression, and adaptive training strategies designed specifically for decentralised and heterogeneous IoT settings (Dritsas and Trigka [26]). These proposals are promising, but they also argue that secure, efficient, and scalable federated learning for IoT still needs further research. These threats and mitigations are discussed here for context; they are outside the scope of the present FedAvg-based experimental protocol. Accordingly, beyond these security considerations, an essential practical question is how far federated training preserves centralised detection performance under non-IID and imbalanced client data when the experimental protocol is held constant.

For this reason, recent work has increasingly focused on lightweight, communication-efficient federated models that can run on embedded devices while maintaining acceptable performance. Overall, deep learning appears to be a strong candidate for IoT botnet detection, but constraints in real deployment highlight the importance of federated and lightweight approaches. In this context, the present study focuses on three aspects that are still relatively underexplored: (i) systematic hyperparameter optimisation of deep learning models for IoT botnet detection, (ii) a consistent, like-for-like comparison of centralised and federated deep learning under the same experimental conditions, and (iii) careful handling of class imbalance in Bot-IoT and N-BaIoT. By addressing these points, the work aims to clarify how performance, privacy, and data distribution interact in IoT botnet detection and what trade-offs are involved in choosing between RF, CDL, and FDL models.

Despite substantial progress in both centralised and federated IoT intrusion detection, it remains unclear—under a single, standardised protocol—how much detection performance is retained when moving from centralised to data-local federated training on large IoT benchmarks. Many published comparisons vary the preprocessing pipeline, data splits, and hyperparameter selection across models and paradigms, which complicates attributing observed differences specifically to decentralisation, especially when client data are non-identically distributed, and class imbalance is severe. The present study addresses this gap by applying a leakage-safe preprocessing pipeline and a standardised evaluation under identical experimental conditions across RF, CDL, and FDL on Bot-IoT and N-BaIoT, using RS-best hyperparameters fixed across paradigms and reporting both weighted and macro-averaged metrics, accompanied by a supplementary experiment with SMOTE for Bot-IoT multiclass.

Moreover, comparative conclusions can be confounded when studies apply unequal hyperparameter tuning budgets or divergent preprocessing and evaluation protocols across paradigms. To avoid this confounding, the present work fixes a consistent tuning budget and applies a standardised evaluation under identical experimental conditions across RF, CDL, and FDL.

In summary, prior work indicates that deep learning can achieve strong performance in IoT botnet detection on Bot-IoT and N-BaIoT, while practical IoT constraints motivate decentralised, communication-efficient training. Building on these insights, the next section outlines the leakage-safe preprocessing pipeline and the controlled experimental protocol used to evaluate RF, CDL, and FDL consistently across both datasets and tasks.

3. Materials and Methods

3.1. Overall Framework and Problem Formulation

This study investigates the effectiveness of optimised deep learning models for IoT botnet detection under both centralised and federated learning paradigms. Two benchmark intrusion detection datasets, Bot-IoT and N-BaIoT, are used to represent large-scale IoT attack scenarios with different levels of class imbalance and device heterogeneity. Section 4 reports results in a structured manner to address RQ1–RQ4, covering RF baselines, CDL performance, and the degree to which FDL preserves CDL performance under non-IID client partitions.

The detection problem is formulated in two ways:

Binary classification: benign versus attack traffic.

Multiclass classification: prediction of specific attack types or families.

Three modelling paradigms are compared:

1. A handcrafted machine learning baseline using a random forest (RF) classifier.

2. Centralised deep learning (CDL), where deep neural networks (DNNs), long short-term memory (LSTM) networks, and bidirectional LSTMs (BiLSTMs) are trained on pooled data.
3. Federated deep learning (FDL), where the same DNN/LSTM/BiLSTM architectures are trained in a privacy-preserving federated learning (FL) setup across multiple non-identically distributed (non-IID) clients without sharing raw data. In this paper, 'privacy-preserving' is used in the pragmatic sense of data locality: raw traffic features and labels remain on clients, and only model updates are exchanged and aggregated (FedAvg). This does not imply a formal privacy guarantee because model updates may leak information; privacy leakage risk and communication overhead are therefore treated as out of scope and are noted explicitly in the limitations section, Section 5. All models are evaluated consistently on both datasets and on both binary and multiclass tasks using a stratified 60/20/20 train/validation/test protocol.

3.2. Datasets

3.2.1. N-BaIoT Dataset

This is one of the datasets used in this paper to train the model, as described by Meidan et al. [9]. It consists of real network traffic from nine IoT devices: two doorbells, a thermostat, a baby monitor, four security cameras, and a webcam. It includes 10 attack classes from the Mirai and BASHLITE botnet families. Five of the classes are Mirai, namely (m-ack, m-scan, m-syn, m-udp, m-udpp), and another four are BASHLITE (gafgyt), namely (g-combo, g-junk, g-scan, g-udp), as shown in Table 1.

Table 1. N-BaIoT network traffic samples.

Type	Class	Samples
3-class	Benign	555,932
	Mirai	3,668,402
	Gafgyt/BASHLITE	1,978,422
	All Attacks (Mirai + Gafgyt/BASHLITE)	5,646,824
10-class	mirai.udp	1,229,999
	gafgyt.udp	946,366
	mirai.syn	733,299
	mirai.ack	643,821
	benign	555,932
	mirai.scan	537,979
	mirai.udpplain	523,304
	gafgyt.combo	515,156
	gafgyt.junk	261,789
	gafgyt.scan	255,111

Both data collection and feature extraction are available here [9]. The dataset contains benign traffic and all the attacks mentioned above. The dataset is imbalanced, with more attack samples than normal samples. All the samples have been used in this paper for both binary class and multiclass classification. It has 115 features.

For the experiments in this study, a stratified 60/20/20 split is applied separately for each task (binary and multiclass), producing training, validation, and test sets while preserving the original label distribution. All available samples are used in both tasks, with the following classifications:

Binary classification: all attack classes merged into a single 'attack' label.

Multiclass classification: the ten attack subclasses and the benign class retained as distinct labels.

3.2.2. Bot-IoT Dataset

This is the second dataset used in this study, as described by Koroniotis et al. [8], and it is publicly available. This dataset was created to improve anomaly detection and security research models. A weather station, a smart fridge, motion-activated lights, a remote-controlled garage door, and a smart thermostat are used in the testbed. It has 46 features, including benign and attack samples. There are 11 classes in total, one benign and the other 10 malicious, including five category types: DOS, DDOS, RECON, Normal (Benign), and THEFT. Both DOS and DDOS have three subcategories: HTTP, TCP, and UDP. RECON is divided into OSF and SS subcategories. The last one has two categories: (DE, KL) as shown in Table 2.

Table 2. Bot-IoT network traffic samples.

Type	Class	Samples
5-class	DDoS	1,926,624
	DoS	1,650,260
	Reconnaissance	91,082
	Normal	477
	Theft	79
	All Attacks (DDoS + DoS + Reconnaissance + Theft)	3,668,045
11-class	DoS-UDP	1,032,975
	DDoS-TCP	977,380
	DDoS-UDP	948,255
	DoS-TCP	615,800
	Reconnaissance-Service_Scan	73,168
	Reconnaissance-OS_Fingerprint	17,914
	DoS-HTTP	1485
	DDoS-HTTP	989
	Normal-Normal	477
	Theft-Keylogging	73
	Theft-Data_Exfiltration	6

In the preprocessing stage, some redundant or identifier-like attributes were removed from the Bot-IoT dataset: pkSeqID, flgs, proto, state, saddr, and daddr. Specifically, pkSeqID is the sequence identification number assigned to each record, while saddr and daddr are the source and destination IP addresses; these fields primarily act as testbed-specific identifiers rather than behavioural traffic descriptors. Moreover, flgs, proto, and state were removed because they duplicate information already represented by their retained numeric counterparts (flgs_number, proto_number, and state_number, respectively). Consequently, the models are less likely to rely on identifier shortcuts that can inflate apparent detection performance and reduce generalisability, while equivalent protocol/state/flag information is preserved through the retained numeric features. For these reasons, all remaining non-target features were used as inputs for both binary and multiclass classification, while the target label was defined based on the task (binary: benign vs. attack; multiclass: attack subclasses and benign). This Bot-IoT dataset contains 3,668,045 botnet attack samples and 477 benign IoT network traffic samples. Scaling and other preprocessing techniques were applied.

As with N-BaIoT, a stratified 60/20/20 split is applied to each task separately. In the configuration used for the final experiments, this corresponds approximately to 2,201,112 samples for training, 733,705 for validation, and 733,705 for testing, as reported in Section 4.

3.3. Data Preprocessing and Feature Preparation

Data preprocessing followed a consistent, leakage-safe pipeline across both datasets and all training paradigms.

- Cleaning and feature selection.
  Identifier-like and non-behavioural attributes were removed prior to modelling. For Bot-IoT, the excluded fields were pkSeqID, saddr, daddr, proto, state, and flgs, which either act as explicit identifiers or duplicate information available in lower-level statistics. For N-BaIoT, all 115 extracted features were used. The retained input feature lists are provided in Table 3 (Bot-IoT) and Table 4 (N-BaIoT).

Table 3. Bot-IoT used input features.

No.	Feature Name
1	Stime
2	flgs_number
3	proto_number
4	Sport
5	Dport
6	Pkts
7	Bytes
8	state_number
9	Ltime
10	Seq
11	Dur
12	Mean
13	Stddev
14	Sum
15	Min
16	Max
17	Spkts
18	Dpkts
19	Sbytes
20	Dbytes
21	Rate
22	Srate
23	Drate
24	TnBPSrcIP
25	TnBPDstIP
26	TnP_PSrcIP
27	TnP_PDstIP
28	TnP_PerProto
29	TnP_Per_Dport
30	AR_P_Proto_P_SrcIP
31	AR_P_Proto_P_DstIP
32	N_IN_Conn_P_DstIP
33	N_IN_Conn_P_SrcIP
34	AR_P_Proto_P_Sport
35	AR_P_Proto_P_Dport
36	Pkts_P_State_P_Protocol_P_DestIP
37	Pkts_P_State_P_Protocol_P_SrcIP

Table 4. N-BaIoT used input features.

No.	Feature Name
1	MI_dir_L5_weight

Table 4. *Cont.*

No.	Feature Name
2	MI_dir_L5_mean
3	MI_dir_L5_variance
4	MI_dir_L3_weight
5	MI_dir_L3_mean
6	MI_dir_L3_variance
7	MI_dir_L1_weight
8	MI_dir_L1_mean
9	MI_dir_L1_variance
10	MI_dir_L0.1_weight
11	MI_dir_L0.1_mean
12	MI_dir_L0.1_variance
13	MI_dir_L0.01_weight
14	MI_dir_L0.01_mean
15	MI_dir_L0.01_variance
16	H_L5_weight
17	H_L5_mean
18	H_L5_variance
19	H_L3_weight
20	H_L3_mean
21	H_L3_variance
22	H_L1_weight
23	H_L1_mean
24	H_L1_variance
25	H_L0.1_weight
26	H_L0.1_mean
27	H_L0.1_variance
28	H_L0.01_weight
29	H_L0.01_mean
30	H_L0.01_variance
31	HH_L5_weight
32	HH_L5_mean
33	HH_L5_std
34	HH_L5_magnitude
35	HH_L5_radius
36	HH_L5_covariance
37	HH_L5_pcc
38	HH_L3_weight
39	HH_L3_mean
40	HH_L3_std
41	HH_L3_magnitude
42	HH_L3_radius
43	HH_L3_covariance
44	HH_L3_pcc
45	HH_L1_weight
46	HH_L1_mean
47	HH_L1_std
48	HH_L1_magnitude
49	HH_L1_radius
50	HH_L1_covariance
51	HH_L1_pcc
52	HH_L0.1_weight
53	HH_L0.1_mean
54	HH_L0.1_std

Table 4. *Cont.*

No.	Feature Name
55	HH_L0.1_magnitude
56	HH_L0.1_radius
57	HH_L0.1_covariance
58	HH_L0.1_pcc
59	HH_L0.01_weight
60	HH_L0.01_mean
61	HH_L0.01_std
62	HH_L0.01_magnitude
63	HH_L0.01_radius
64	HH_L0.01_covariance
65	HH_L0.01_pcc
66	HH_jit_L5_weight
67	HH_jit_L5_mean
68	HH_jit_L5_variance
69	HH_jit_L3_weight
70	HH_jit_L3_mean
71	HH_jit_L3_variance
72	HH_jit_L1_weight
73	HH_jit_L1_mean
74	HH_jit_L1_variance
75	HH_jit_L0.1_weight
76	HH_jit_L0.1_mean
77	HH_jit_L0.1_variance
78	HH_jit_L0.01_weight
79	HH_jit_L0.01_mean
80	HH_jit_L0.01_variance
81	HpHp_L5_weight
82	HpHp_L5_mean
83	HpHp_L5_std
84	HpHp_L5_magnitude
85	HpHp_L5_radius
86	HpHp_L5_covariance
87	HpHp_L5_pcc
88	HpHp_L3_weight
89	HpHp_L3_mean
90	HpHp_L3_std
91	HpHp_L3_magnitude
92	HpHp_L3_radius
93	HpHp_L3_covariance
94	HpHp_L3_pcc
95	HpHp_L1_weight
96	HpHp_L1_mean
97	HpHp_L1_std
98	HpHp_L1_magnitude
99	HpHp_L1_radius
100	HpHp_L1_covariance
101	HpHp_L1_pcc
102	HpHp_L0.1_weight
103	HpHp_L0.1_mean
104	HpHp_L0.1_std
105	HpHp_L0.1_magnitude
106	HpHp_L0.1_radius
107	HpHp_L0.1_covariance

Table 4. *Cont.*

No.	Feature Name
108	HpHp_L0.1_pcc
109	HpHp_L0.01_weight
110	HpHp_L0.01_mean
111	HpHp_L0.01_std
112	HpHp_L0.01_magnitude
113	HpHp_L0.01_radius
114	HpHp_L0.01_covariance
115	HpHp_L0.01_pcc

- Label construction.
  For the binary task on both datasets, labels are collapsed to 'benign' (0) and 'attack' (1). For the multiclass task, the original attack subclasses are preserved, and the benign class remains separate; no synthetic labels are introduced. Binary benign-versus-attack classification was used to reflect a deployment-oriented detection setting in which the primary objective is reliable alert generation (attack vs. benign), whereas multiclass classification was used to examine family- and subclass-specific weaknesses; because it preserves the original labels, it is used for error diagnosis.
- Train–validation–test splitting.
  A stratified 60/20/20 split is used to form training, validation, and test sets for each dataset and task, ensuring that each split reflects the original class distribution. All encoders and scalers are fitted only on the training set and then applied to the validation and test sets to prevent any form of label leakage.
- Encoding and scaling.
  Categorical labels (for example, attack names) are transformed using label encoding. All continuous features are normalised using a Min–Max scaler fitted on the training data and then applied unchanged to validation and test sets.
- Imbalance handling policy.
  Because both datasets are highly imbalanced, class distributions are inspected after splitting. In the primary experiments, imbalance is handled through stratified splitting, class-weighted losses, and reporting both weighted and macro-averaged metrics. In addition, SMOTE is included as an additional analysis to assess imbalance mitigation. Where used, SMOTE is applied only to the training split after encoding and before model training, with validation and test sets kept unchanged. The SMOTE ablation results are reported explicitly and separately in Section 4 (Results and Discussion).

3.4. Models

3.4.1. Handcrafted Baseline: Random Forest

The random forest (RF) classifier serves as a strong handcrafted baseline for both binary and multiclass detection problems. RF is an ensemble of decision trees trained on bootstrapped samples of the training data with random feature sub-sampling at each split. It is known for its robustness against overfitting, its ability to model non-linear decision boundaries, and its relatively modest tuning requirements, which make it attractive for intrusion detection on tabular IoT traffic.

In this study, RF is fitted on the preprocessed training subset for each dataset and task, and its performance is evaluated on the corresponding independent test subset. RF hyperparameters were fixed across binary and multiclass configurations to provide a consistent baseline: n_estimators = 400, max_features = 'sqrt', class_weight = 'balanced', and random_state = SEED (other parameters were left at library defaults). Hyperparameter

optimisation was prioritised for the deep models due to their larger and more sensitive hyperparameter space, whereas RF is reported as a fixed classical baseline. Performance is reported using the metrics in Section 3.8.

3.4.2. Centralised Deep Learning (CDL): DNN, LSTM, BiLSTM

In the centralised deep learning setting, three neural architectures are considered: DNN, LSTM, and BiLSTM.

- Deep neural network (DNN).
  The DNN comprises multiple fully connected layers with non-linear activation functions (for example, ReLU or ELU) and dropout regularisation. The input layer receives the normalised feature vector, hidden layers capture non-linear interactions between features, and the final output layer uses a sigmoid activation for binary tasks or a softmax activation for multiclass tasks. In this study, the DNN refers to a deep feed-forward neural network (multi-layer perceptron, MLP) composed of multiple fully connected layers with non-linear activations and dropout.
- Long short-term memory (LSTM).
  LSTM networks model sequential inputs in the form (batch, timesteps, features). In this study, each traffic record (flow/window feature vector) was represented as a single-step sequence and reshaped to (N, 1, F), where F denotes the number of retained input features. Accordingly, no grouping of multiple consecutive flows into longer sequences was performed.
- Bidirectional LSTM (BiLSTM).
  BiLSTM extends LSTM by processing the input sequence in forward and backward directions and combining the two representations. In this study, each traffic record was represented as a single-step sequence (timesteps = 1) to match the recurrent input format; therefore, BiLSTM is not used to model past–future temporal evolution across multiple flows. BiLSTM is included for comparative evaluation alongside LSTM under identical preprocessing and splits.

All CDL models are trained with cross-entropy loss (binary cross-entropy for binary tasks and categorical cross-entropy for multiclass tasks). 'Optimisers, learning rates, layer depths, units per layer, batch sizes, and dropout rates were selected via the randomised search strategy described in Section 3.5, tuned separately for each model–dataset–task combination (binary vs. multiclass); the resulting task-specific RS-best configuration was then fixed and reused unchanged for the corresponding full-dataset CDL and FDL experiments to enable a controlled comparison of training paradigms.'

3.4.3. Federated Deep Learning (FDL) Architectures

The same DNN, LSTM, and BiLSTM architectures are used in the federated setting. Model parameterisations (layer structure, activation functions, and hyperparameters) are identical to their CDL counterparts so that any performance differences can be attributed to the training paradigm (centralised versus federated) rather than architectural changes.

3.5. Hyperparameter Optimisation (Random Search)

A randomised search strategy was employed to optimise key hyperparameters for all deep models (DNN, LSTM, and BiLSTM) on both binary and multiclass tasks. The search explored combinations of the number of hidden layers (depth), base hidden units (units), activation (activation), learning rate (learning rate), optimiser type (optimiser), dropout rate (dropout), batch size (batch size), and training epochs (epochs). Random sampling was used instead of exhaustive grid search to ensure broader coverage within a practical

time budget, with standard safeguards applied (early stopping with weight restoration and leakage-safe preprocessing).

The random search was conducted on reduced, stratified subsets of each dataset (for example, 20% of Bot-IoT and 15% of N-BaIoT) to reduce computational cost. For each model–dataset–task combination, the configuration that achieved the best validation performance on the subset was selected as the RS-best setup and subsequently reused unchanged for all full-dataset CDL and FDL experiments reported in Section 4.

3.6. Federated Learning Setup (Flower, Non-IID, FedAvg)

Federated experiments were implemented using the Flower framework, with the federated averaging (FedAvg) algorithm as the aggregation method. Five clients were simulated to represent IoT edge devices (Client1–Client5), each holding a disjoint subset of the global training data.

- Non-IID and imbalanced partitions.
  Client datasets were constructed using non-IID, imbalanced splits of the global training set, with label distributions differing across clients, reflecting heterogeneous IoT deployments. Client datasets follow non-IID, imbalanced partitions of the global training set, with label distributions differing across clients, reflecting heterogeneous IoT deployments. No raw data is exchanged between clients and the server; each client operates only on its own local data. Privacy scope: in this paper, 'privacy-preserving' refers to data locality. Raw traffic remains on clients, and only model updates are exchanged and aggregated using FedAvg. This does not imply a formal privacy guarantee because shared updates may leak information; therefore, privacy leakage risk from updates is outside the scope of the present experiments and is treated in Limitations/Future Work.
- Federated training protocol.
  In FDL, client-side training used early stopping, monitored on the validation loss (patience = 5, restore_best_weights = true), to reduce overfitting and support convergence.
- All clients start from a common initial model (the RS-best architecture and hyperparameters).
- In each communication round, the server broadcasts the current global model to all participating clients.
- Each client performs several local epochs of training (five local epochs by default) on its own data using mini-batch stochastic gradient descent with the task-appropriate cross-entropy loss, configured optimiser, and dropout.
- Clients send updated model weights and local sample counts back to the server (no raw features or labels are transmitted).
- The server aggregates client updates using sample-size-weighted FedAvg to produce an updated global model.

This process is repeated for a fixed number of communication rounds (eight in the final configuration) or until convergence.

- Evaluation.
  After the final communication round, the global model is evaluated centrally on the held-out test set to obtain global metrics (accuracy, macro-F1, weighted F1, and for binary tasks, TNR). In addition, each client's local performance is measured on its own held-out partition using the same metrics to assess the uniformity of performance across heterogeneous clients. This framework ensures that sensitive IoT traffic remains on-device while still enabling the training of high-capacity deep models that approximate centralised performance.

3.7. Loss Functions, Training Protocol, and Callbacks

For binary tasks, the models were trained using binary cross-entropy, and for multiclass tasks, sparse categorical cross-entropy. Class imbalance was addressed through stratified splitting and by reporting imbalance-robust macro-averaged metrics (balanced accuracy/macro-average recall) in addition to weighted averages and per-class recall.

Training follows a standard supervised learning protocol:

- Models are trained on the training split and monitored on the validation split.
- Early stopping with best-weight restoration is used to halt training when validation loss stops improving. In this study, early stopping monitored validation loss (val_loss) and terminated training when val_loss failed to improve for five consecutive epochs (patience = 5), with model weights restored to those from the epoch with the best val_loss (restore_best_weights = true).
- Model checkpoint and CSV logging are used to store the best models and track training histories.
- Overfitting was mitigated using (i) early stopping monitored on validation loss with best-weight restoration (patience = 5, restore_best_weights = true), (ii) dropout regularisation in the deep architectures (dropout rates as reported in the RS-best configurations), and (iii) a held-out validation split (20%) used for model selection and stopping decisions. In addition, class-weighted losses were applied to reduce majority-class dominance under severe imbalance, which can otherwise present as artificially high accuracy.

These safeguards are applied in both CDL and FDL experiments (for FDL, at the client side) to stabilise optimisation and support reproducibility.

3.8. Evaluation Metrics and Reporting

The model performance is assessed using a consistent set of metrics across datasets, tasks, and models:

N Accuracy: overall proportion of correctly classified samples, always reported together with the total number of test instances N. Given the extreme class imbalance (approximately 3.6 million attack samples versus 477 benign samples), balanced accuracy was also reported, defined as the unweighted mean of per-class recall, i.e., (TPR + TNR)/2, to provide an imbalance-robust summary of sensitivity and specificity.

F1-score (weighted): the class-frequency-weighted average F1-score, emphasising dominant classes.

F1-score (macro): the unweighted mean F1-score across classes, which is particularly sensitive to minority-class performance.

Precision and recall (weighted): weighted by class frequency (support) to reflect the overall behaviour of the classifier.

Binary classification quantities derived from the confusion matrix:

True positive rate (TPR)/recall/sensitivity for the attack class.

True negative rate (TNR)/specificity for the benign class.

Equation set: Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively (positive class = attack). Then:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{4}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{5}$$

For multiclass tasks (classes $k = 1, \ldots, K$), per-class one-vs.-rest metrics were computed, with per-class support n_k and total $N = \sum_{k=1}^{K} n_k$. Let $\mathrm{F1}_k$ denote the per-class F1-score. Then:

$$\mathrm{F1}_{\mathrm{macro}} = \frac{1}{\mathrm{K}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{F1}_{\mathrm{k}}, \tag{6}$$

$$\mathrm{F1}_{\mathrm{weighted}} = \frac{1}{\mathrm{N}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{n}_{\mathrm{k}} \, \mathrm{F1}_{\mathrm{k}}. \tag{7}$$

- Diagnostic tools.
  Confusion matrices are generated for all experiments and reported with both raw counts and row-wise percentages, allowing inspection of family-level confusions (for example, between related Mirai or Gafgyt subclasses). Classification reports provide per-class precision, recall, and F1-scores, along with macro and weighted aggregates. For the RF baseline, receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC) are computed for binary tasks to quantify ranking quality across thresholds. For the deep learning models, scalar AUC values are reported where relevant, while the main emphasis remains on accuracy and F1 metrics.

The same evaluation strategy is consistently applied across the centralised and federated experiments on both datasets, enabling a like-for-like comparison of RF, CDL, and FDL in Sections 4 and 5.

3.9. Hardware and Software Environment

All experiments are conducted on a workstation-class laptop (MSI Stealth GS66 12UGS; Micro-Star International (MSI), New Taipei City, Taiwan) equipped with the following:

CPU [M33.1]: Intel Core i7-12700H (12th generation) processor (Intel Corporation, Santa Clara, CA, USA) [M34.1].

GPU: NVIDIA GeForce RTX 3070 Ti Laptop GPU with 8 GB dedicated VRAM (NVIDIA Corporation, Santa Clara, CA, USA).

Memory: 32 GB RAM.Storage and OS: solid-state drive running 64-bit Windows11 Home (Version 25H2, OS build 26200.7623) (Microsoft, Redmond, WA, USA).

The software stack is based on Python 3.12.7 [M35.1] and includes the following:

Data and classical ML: NumPy, pandas, and scikit-learn for data handling and the RF baseline.

Deep learning: TensorFlow and Keras for implementing DNN, LSTM, and BiLSTM architectures with GPU acceleration enabled.

Federated learning: a recent version of the Flower framework for orchestrating federated training and evaluation.

Experiment management: Matplotlib (version 3.10.3) for plotting, standard logging utilities, and JSON/CSV for saving metrics and artefacts.

Reproducibility is encouraged by fixing random seeds where possible, controlling GPU memory growth, and storing trained models, training histories, confusion matrices, and classification reports for all experiments.

4. Results and Discussion

This section presents and discusses the experimental results obtained from the handcrafted random forest (RF) baseline, the centralised deep learning models (DNN, LSTM,

BiLSTM), and their federated counterparts (FDL) on the Bot-IoT and N-BaIoT datasets under both binary and multiclass settings. Section 4.1 summarises the hyperparameter optimisation strategy and RS-best settings. Section 4.2 reports the RF baseline. Sections 4.3–4.5 present the CDL results for DNN, LSTM, and BiLSTM, respectively. Section 4.6 summarises the FDL results for DNN, LSTM, and BiLSTM. Section 4.7 compares models across paradigms (RF, CDL, and FDL). Section 4.8 relates these results to recent work on Bot-IoT and N-BaIoT. Section 4.9 concludes with a concise summary of the key findings.

In addition to the primary evaluations, SMOTE is reported as a secondary analysis for the Bot-IoT multiclass task to quantify the effect of class rebalancing on minority attack types; these ablation results are presented explicitly in the corresponding 'with SMOTE' subsections.

Together, these results address the four research questions by (i) quantifying the performance of RF and deep architectures on both datasets (RQ1); (ii) assessing the effect of class imbalance and SMOTE on binary and multiclass tasks (RQ2); (iii) evaluating how far FDL preserves CDL performance under non-IID client splits (RQ3); and (iv) analysing the trade-offs between accuracy, robustness, and privacy-preserving decentralisation (RQ4).

4.1. Hyperparameter Optimisation

We used the random search-best (RS-best) hyperparameters—found via randomised search per model, dataset, and task—for all evaluations in this section. The search sampled combinations of depth (number of hidden layers), base hidden units, activation function, learning rate, optimiser, dropout, batch size, and epochs. The RS-best settings were then fixed for all subsequent RF/CDL/FDL runs on the same model–dataset–task combination to ensure a fair comparison. The full tuning procedure is provided in Section 3.5.

Tables 5 and 6 list the RS-best hyperparameters used unchanged across all binary and multiclass experiments, respectively.

Table 5. RS-best hyperparameters—binary. This table shows the random search-best (RS-best) hyperparameter settings used for the DNN, LSTM, and BiLSTM models on the Bot-IoT and N-BaIoT datasets in the binary task. Abbreviations: DNN, deep neural network; LSTM, long short-term memory; BiLSTM, bidirectional long short-term memory; RS, random search.

Model	Dataset	Layers	Units	Activation	Dropout	Optimiser	Learning Rate	Batch Size	Epochs
DNN	Bot-IoT	3	128	relu	0.2	nadam	0.0016713	128	16
LSTM	Bot-IoT	2	32	tanh	0.3	adam	0.0005518	128	20
BiLSTM	Bot-IoT	2	32	tanh	0.3	adam	0.0005518	128	20
DNN	N-BaIoT	2	[16, 32]	elu	0.4	adam	0.0009775	256	16
LSTM	N-BaIoT	2	[128, 64]	tanh	0.2	adam	0.0003292	128	12
BiLSTM	N-BaIoT	2	[128, 64]	tanh	0.4	adam	0.0004015	128	16

Table 6. RS-best hyperparameters—multiclass. This table shows the random search-best (RS-best) hyperparameter settings used for the DNN, LSTM, and BiLSTM models on the Bot-IoT and N-BaIoT datasets in the multiclass task. Abbreviations: DNN, deep neural network; LSTM, long short-term memory; BiLSTM, bidirectional long short-term memory; RS, random search.

Model	Dataset	Layers	Units	Activation	Dropout	Optimiser	Learning Rate	Batch Size	Epochs
DNN	Bot-IoT	3	128	relu	0.2	nadam	0.0016713	128	16
LSTM	Bot-IoT	2	128	tanh	0.2	adam	0.0003292	128	12
BiLSTM	Bot-IoT	2	128	tanh	0.2	nadam	0.0000731	256	16
DNN	N-BaIoT	2	[32, 32]	relu	0.2	rmsprop	0.0014165	256	8
LSTM	N-BaIoT	1	[64]	tanh	0.2	nadam	0.000481	512	8
BiLSTM	N-BaIoT	2	[128, 64]	tanh	0.2	adam	0.0000745	256	8

4.2. Handcrafted ML Baseline—Random Forest (RF)

The RF baseline provides a very strong handcrafted reference across both datasets and tasks. In all cases, RF delivered near-perfect or perfect overall performance, so later centralised and federated deep learning models are compared against an already very high bar rather than correcting large residual errors.

4.2.1. Binary

Bot-IoT (Binary)

The RF model achieved almost perfect performance on the Bot-IoT binary task. The confusion matrix (Figure 1) shows TN = 92, FP = 4, FN = 0, and TP = 733,609, meaning the model did not miss any attacks (FN = 0; TPR/recall = 1.0000) and generated only four false alarms on benign traffic, corresponding to a TNR of 0.9583. These few benign samples misclassified as attacks explain the small gap between macro-F1 and weighted F1 under strong class imbalance.

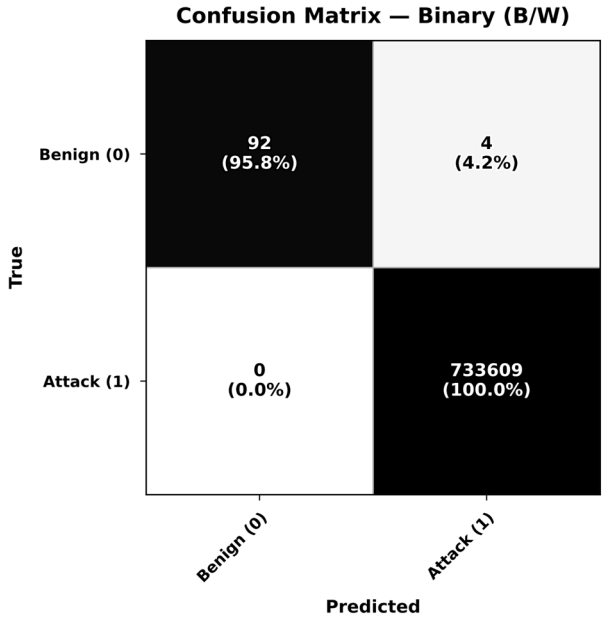

Figure 1. RF binary—Bot-IoT. This figure shows the confusion matrix for RF binary classification on Bot-IoT. Abbreviations: RF, random forest; CM, confusion matrix.

The overall metrics (Table 7) show that accuracy = 0.9999, weighted precision = 0.9999, weighted recall = 0.9999, weighted F1 = 0.9999, and macro-F1 = 0.9893. Balanced accuracy was 0.9792 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance. At the class level, benign traffic attained precision = 1.0000, recall = 0.9583, and F1 = 0.9787, whereas attack traffic achieved precision of approximately 0.9999, recall = 1.0000, and F1 of approximately 0.9999. The ROC curve (Figure 2) reported an AUC very close to 1.0000, indicating a near-perfect ranking of attack versus benign traffic across thresholds, consistent with the confusion matrix.

Table 7. RF binary—Bot-IoT. This table shows the classification report and summary metrics for RF binary classification on Bot-IoT. Abbreviations: RF, random forest.

Class/Aggregate	Support	Accuracy	Precision	Recall	F1-Score	Macro-F1
Overall (weighted)	733,705	0.9999	0.9999	0.9999	0.9999	0.9893
Benign (0)	96	N/A	1.0000	0.9583	0.9787	N/A
Attack (1)	733,609	N/A	0.9999	1.0000	0.9999	N/A

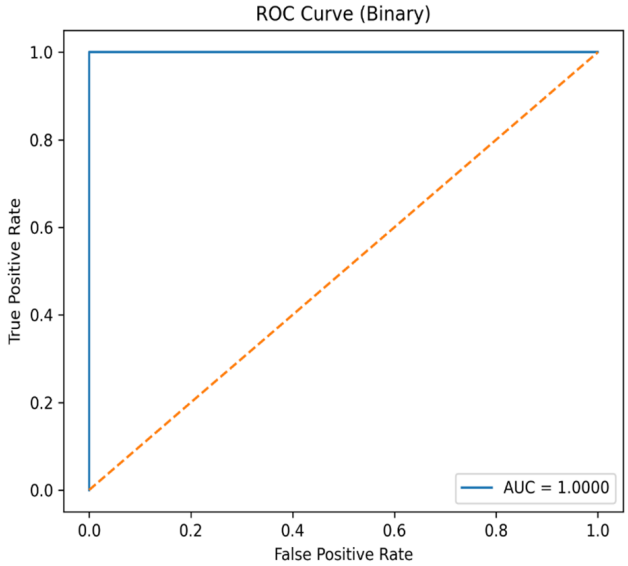

Figure 2. RF binary—Bot-IoT. This figure shows the receiver operating characteristic (ROC) curve for RF binary classification on Bot-IoT. Abbreviations: RF, random forest; ROC, receiver operating characteristic; AUC, area under the curve.

N-BaIoT (Binary)

For the N-BaIoT binary task, the RF classifier achieved perfect overall performance on the held-out test set. The confusion matrix (Figure 3) was strictly diagonal, with TN = 111,187, FP = 0, FN = 0, and TP = 1,301,335, yielding TNR = 1.0000 and TPR/recall = 1.0000 for benign and attack classes, respectively. Consistent with this, all global metrics in Table 8 were equal to 1.0000, including overall accuracy, weighted precision, weighted recall, weighted F1, macro-F1, and ROC-AUC. Balanced accuracy was 1.0000 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance (Figure 4), indicating that the model made no classification errors at the chosen threshold and ranked benign versus attack samples perfectly across all thresholds.

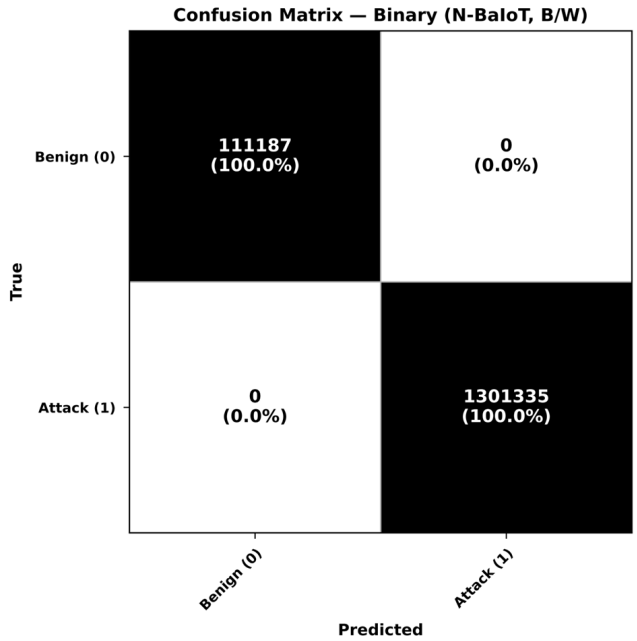

Figure 3. RF binary—N-BaIoT. This figure shows the confusion matrix for RF binary classification on N-BaIoT. Abbreviations: RF, random forest; CM, confusion matrix.

Table 8. RF binary—N-BaIoT. This table shows the classification report and summary metrics for RF binary classification on N-BaIoT. Abbreviations: RF, random forest.

Class/Aggregate	Support	Accuracy	Precision	Recall	F1-Score	Macro-F1
Overall (weighted)	1,412,522	1.0000	1.0000	1.0000	1.0000	1.0000
Benign (0)	111,187	N/A	1.0000	1.0000	1.0000	N/A
Attack (1)	1,301,335	N/A	1.0000	1.0000	1.0000	N/A

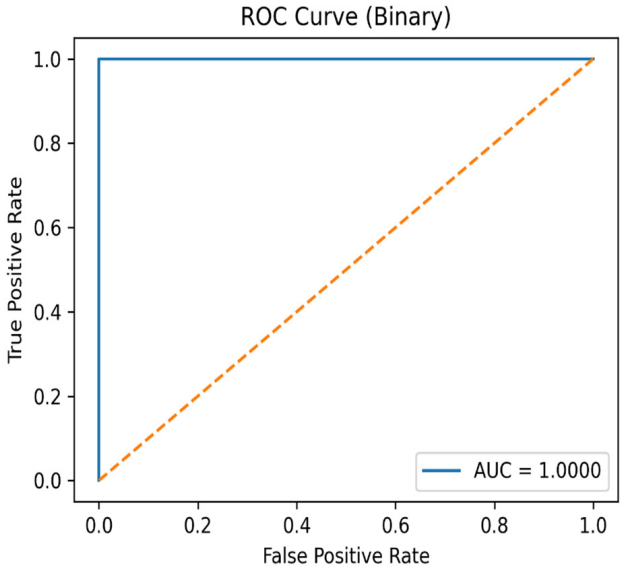

Figure 4. RF binary—N-BaIoT. This figure shows the receiver operating characteristic (ROC) curve for RF binary classification on N-BaIoT. Abbreviations: RF, random forest; ROC, receiver operating characteristic; AUC, area under the curve.

4.2.2. Multiclass

Bot-IoT (Multiclass)

For the Bot-IoT multiclass task, the RF model achieved near-perfect overall performance. The confusion matrix (Figure 5) was almost entirely diagonal, with all DDoS/DoS variants, both Theft classes, and the Normal class classified essentially perfectly (normal recall = 1.0000). Darker shading indicates higher counts, and each cell reports the sample count with the corresponding row percentage. The only visible residual errors appeared in the two reconnaissance classes: Reconnaissance-OS_Fingerprint (recall approximately 0.9838) and Reconnaissance-Service_Scan (recall approximately 0.9963).

Consistent with this pattern, the test metrics in Table 9 reported an overall accuracy of 0.9998, a weighted F1 of 0.9998, and a macro-F1 of 0.9969, with per-class recalls for the main attack families all very close to 1.0000. The few misclassifications were concentrated in reconnaissance sub-types and did not affect the dominant classes.

Table 9. RF multiclass—Bot-IoT. This table shows the classification report and summary metrics for RF multiclass classification on Bot-IoT. Abbreviations: RF, random forest.

Class	Support	Precision	Recall	F1-Score
DDoS-HTTP	198	1.0000	0.9949	0.9974
DDoS-TCP	195,476	0.9999	0.9999	0.9999
DDoS-UDP	189,651	0.9999	0.9999	0.9999
DoS-HTTP	297	0.9932	0.9932	0.9932
DoS-TCP	123,160	0.9999	0.9999	0.9999

Table 9. *Cont.*

Class	Support	Precision	Recall	F1-Score
DoS-UDP	206,595	0.9999	0.9999	0.9999
Normal-Normal	96	0.9896	1.0000	0.9948
Reconnaissance-OS_Fingerprint	3583	0.9851	0.9838	0.9845
Reconnaissance-Service_Scan	14,634	0.9960	0.9964	0.9962
Theft-Data_Exfiltration	1	1.0000	1.0000	1.0000
Theft-Keylogging	14	1.0000	1.0000	1.0000
Macro average	733,705	0.9967	0.9971	0.9969
Weighted average	733,705	0.9998	0.9998	0.9998

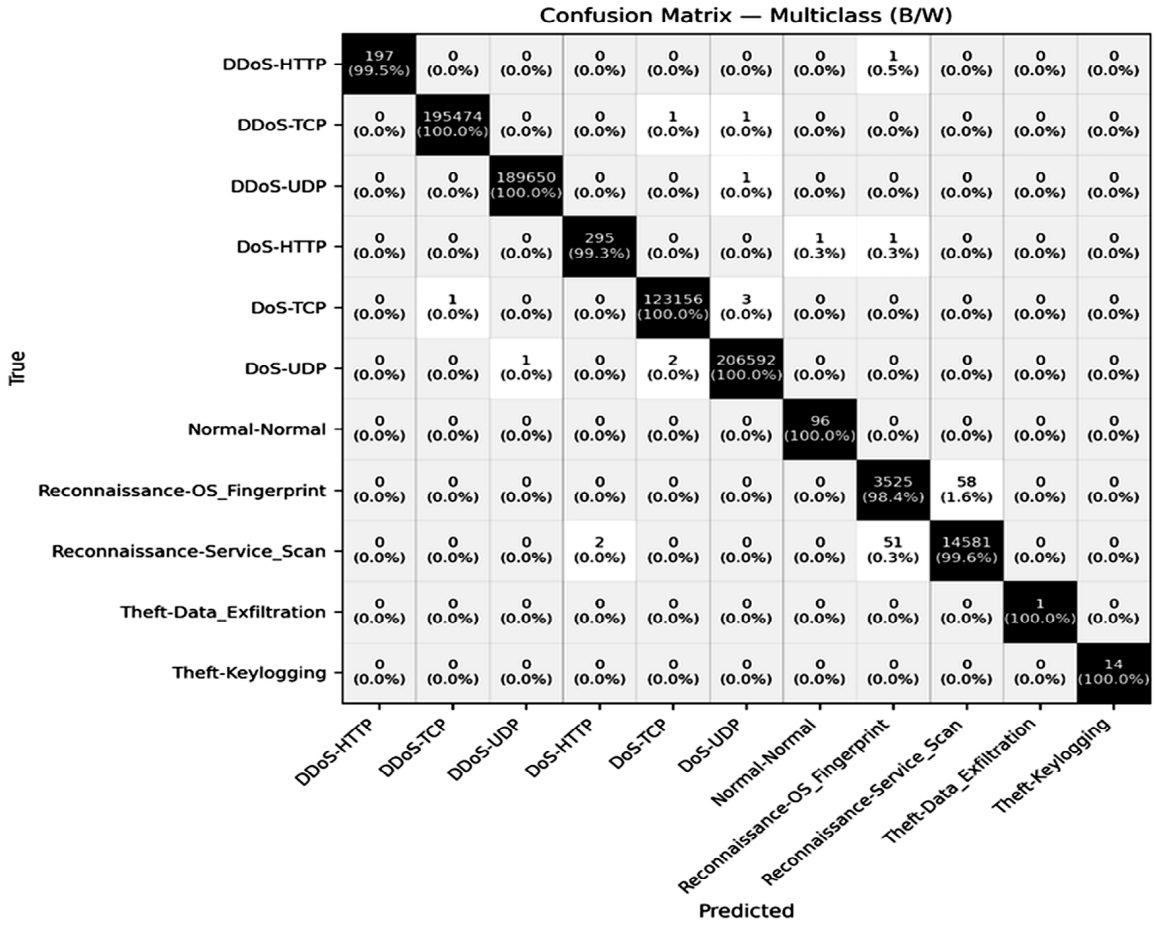

Figure 5. RF multiclass—Bot-IoT. This figure shows the confusion matrix for RF multiclass classification on Bot-IoT. Abbreviations: RF, random forest; CM, confusion matrix.

N-BaIoT (Multiclass)

For the N-BaIoT multiclass task, the RF model achieved an effectively perfect performance. The confusion matrix (Figure 6) was essentially diagonal, with only a single negligible off-diagonal count, indicating that all ten classes were almost always correctly identified. Darker shading indicates higher counts, and each cell reports the sample count with the corresponding row percentage. The corresponding test metrics from the classification report (Table 10) show that weighted precision, weighted recall, and weighted F1 are all 0.9999 across approximately 1.24 million test instances, confirming that the classifier achieved near-perfect recognition across all classes. Overall test accuracy was 0.9999 (N = 1,240,552), consistent with the near-unity weighted metrics.

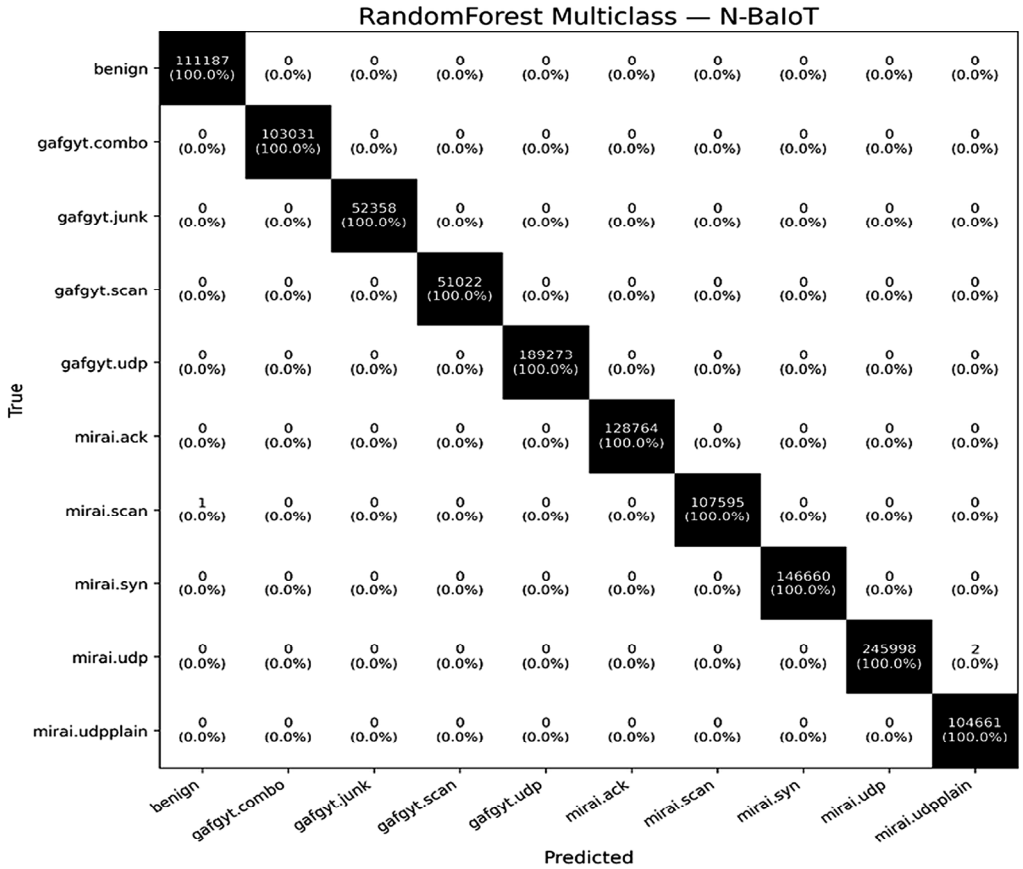

Figure 6. RF multiclass—N-BaIoT. This figure shows the confusion matrix for RF multiclass classification on N-BaIoT. Abbreviations: RF, random forest; CM, confusion matrix.

Table 10. RF multiclass—N-BaIoT. This table shows the classification report and summary metrics for RF multiclass classification on N-BaIoT. Abbreviations: RF, random forest.

Class	Support	Precision	Recall	F1-Score
Benign	111,187	0.9999	1.0000	0.9999
gafgyt.combo	103,031	1.0000	1.0000	1.0000
gafgyt.junk	52,358	1.0000	1.0000	1.0000
gafgyt.scan	51,022	1.0000	1.0000	1.0000
gafgyt.udp	189,273	1.0000	1.0000	1.0000
mirai.ack	128,764	1.0000	1.0000	1.0000
mirai.scan	107,596	1.0000	1.0000	0.9999
mirai.syn	146,660	1.0000	1.0000	1.0000
mirai.udp	246,000	1.0000	0.9999	0.9999
mirai.udpplain	104,661	0.9999	1.0000	0.9999
Macro average	1,240,552	0.9999	1.0000	0.9999
Weighted average	1,240,552	0.9999	1.0000	0.9999

4.3. Centralised Deep Learning—DNN

4.3.1. Binary

Bot-IoT (Binary)

Figure 7 shows complete separation between the classes on the Bot-IoT binary: TN = 96, FP = 0, FN = 0, and TP = 733,609, providing TPR = 1.0000 (sensitivity/recall) and TNR = 1.0000 (specificity), with no residual confusion. The loss curve in Figure 8 converges rapidly, and the marker indicates the epoch with the lowest observed validation loss; both training and validation loss then remain flat, which is consistent with a stable fit rather than late-epoch overfitting.

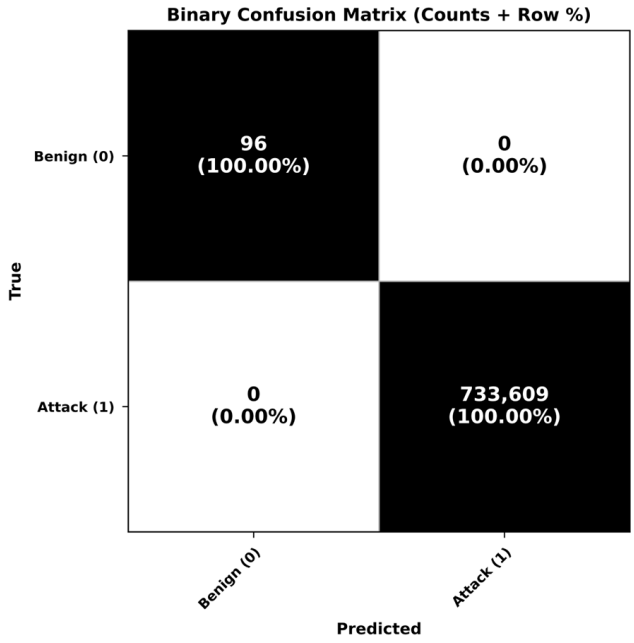

Figure 7. DNN binary—Bot-IoT. This figure shows the confusion matrix for DNN binary classification on Bot-IoT. Abbreviations: CM, confusion matrix.

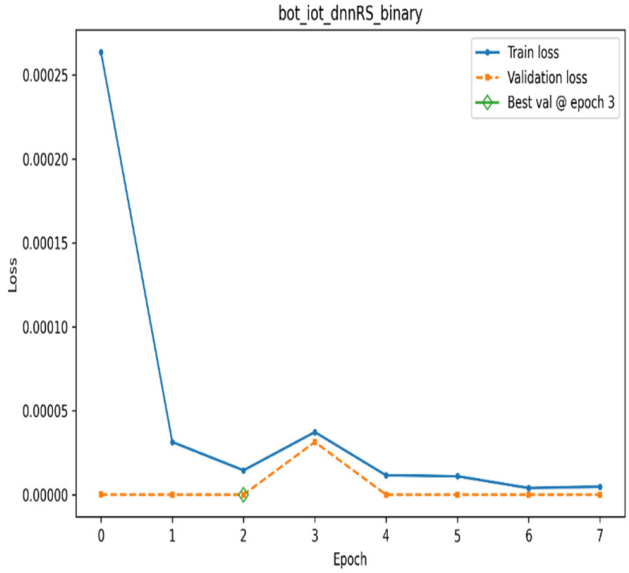

Figure 8. DNN binary—Bot-IoT. This figure shows the training and validation loss curves for DNN binary classification on Bot-IoT.

Table 11 (classification report) mirrors the confusion matrix: Benign (0) achieves precision = recall = F1 = 1.0000 on support = 96, and Attack (1) achieves precision = recall = F1 = 1.0000 on support = 733,609. Accordingly, overall test accuracy was 1.0000 (N = 733,705), matching the zero-error confusion matrix. These values numerically align with the confusion matrix counts (TN and TP exactly match the supports for the negative and positive classes), confirming that both TPR and TNR are 1.0000. Even under the strong support imbalance (96 vs. 733,609), the perfect benign-class recall rules out accuracy-only artefacts. Balanced accuracy was 1.0000 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance. Taken together, the figures and table indicate genuine separability for this model–dataset–task setting, with broader robustness assessed later on N-BaIoT and in the federated analyses.

Table 11. DNN binary—Bot-IoT. This table shows the classification report and summary metrics for DNN binary classification on Bot-IoT.

Class/Aggregate	Precision	Recall	F1-Score	Support
Benign (0)	1.0000	1.0000	1.0000	96
Attack (1)	1.0000	1.0000	1.0000	733,609
Overall (weighted)	1.0000	1.0000	1.0000	733,705

N-BaIoT (Binary)

Figure 9 shows near-perfect performance on N-BaIoT (binary). The model correctly identified almost all traffic: TPR approximately 0.9997 (very few attacks missed) and TNR approximately 0.9973 (very few benign samples flagged as attacks). The small off-diagonal counts (around 0.27% FP) indicated a slight tendency to over-flag benign traffic as an attack. Figure 10 confirms healthy training: loss fell quickly, training and validation curves stayed close, and the best validation occurred at epoch 16, consistent with a stable fit without late overfitting.

Table 12 is consistent with the confusion matrix: Benign (0) has precision = 0.9974, recall = 0.9972, and F1 = 0.9973 (N = 111,187); Attack (1) has precision = 0.9997, recall = 0.9997, and F1 = 0.9997 (n = 1,301,335). Overall accuracy is 0.9995 (N = 1,412,522). These values align with the confusion matrix and confirm that performance remained extremely high despite the strong class imbalance. Balanced accuracy was 0.9985 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

Table 12. DNN binary—N-BaIoT. This table shows the classification report and summary metrics for DNN binary classification on N-BaIoT.

Class/Aggregate	Precision	Recall	F1-Score	Support
Benign (0)	0.997401	0.997293	0.997347	111,187
Attack (1)	0.999769	0.999778	0.999773	1,301,335

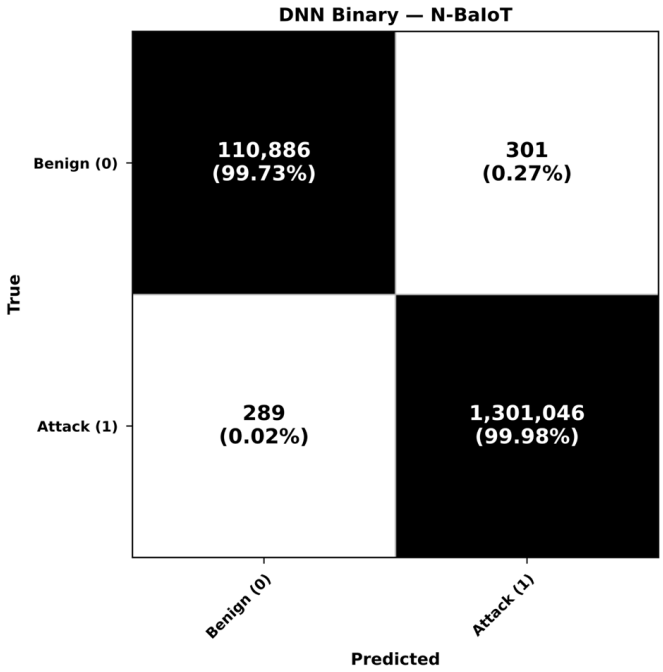

Figure 9. DNN binary—N-BaIoT. This figure shows the confusion matrix for DNN binary classification on N-BaIoT. Abbreviations: CM, confusion matrix.

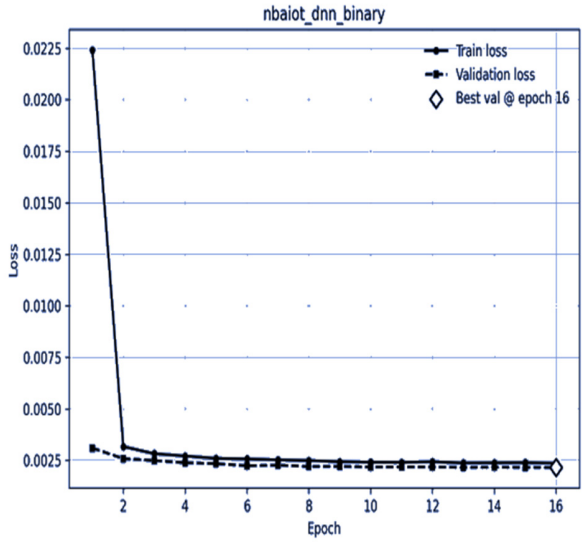

Figure 10. DNN binary—N-BaIoT. This figure shows the training and validation loss curves for DNN binary classification on N-BaIoT.

4.3.2. Multiclass

Bot-IoT (Multiclass, Without SMOTE)

Figure 11 shows an almost perfectly diagonal confusion matrix for Bot-IoT multiclass under DNN: every class is 100% correct except for one off-diagonal error among the minority classes (for example, DDoS-TCP 195,476/195,476; DDoS-UDP 189,651/189,651; DoS-TCP 123,160/123,160; Normal 96/96; Reconnaissance-Service_Scan 14,634/14,634; Theft-Keylogging 14/14). Figure 12 indicates fast, stable convergence with the best validation at epoch 5 and no overfitting.

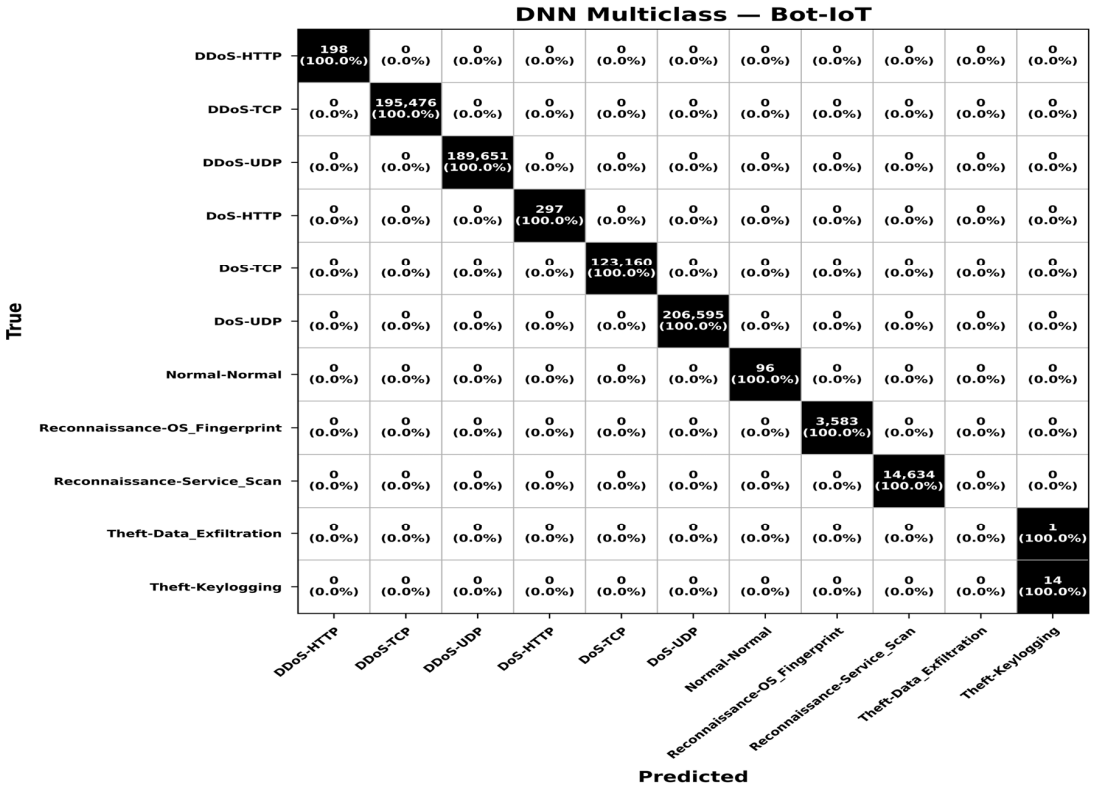

Figure 11. DNN multiclass—Bot-IoT (without SMOTE). This figure shows the confusion matrix for DNN multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique; CM, confusion matrix.

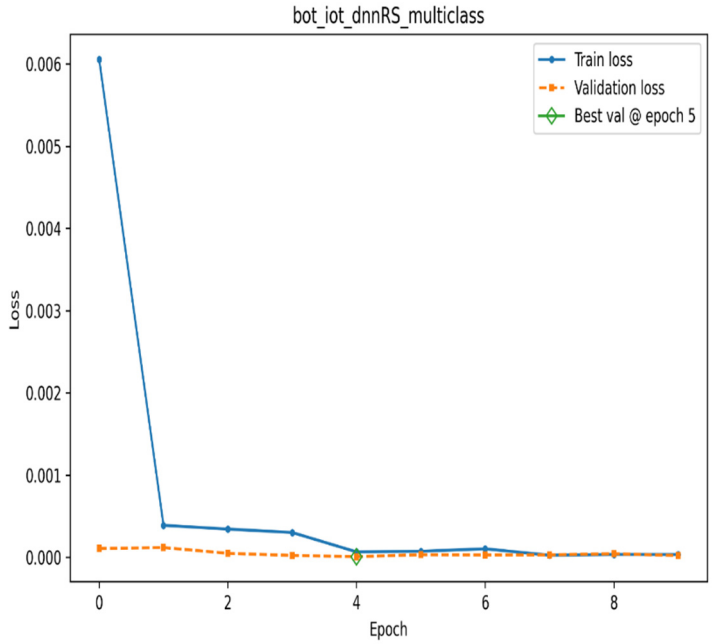

Figure 12. DNN multiclass—Bot-IoT (without SMOTE). This figure shows the training and validation loss curves for DNN multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

In Table 13 (classification report), the aggregates reflect this pattern: accuracy of approximately 0.9999, macro-F1 of approximately 0.9060, macro-precision of approximately 0.9030, and macro-recall of approximately 0.9090. The macro averages are lower only because macro metrics treat each class equally; a single error on a rare class pulls down the macro scores despite near-perfect micro-level performance. Overall, the figures and report together indicate that the DNN cleanly separated classes on Bot-IoT, with performance dominated by exact predictions across the high-support classes and a single minority-class error driving the macro drop.

Table 13. DNN multiclass—Bot-IoT (without SMOTE). This table shows the classification report and summary metrics for DNN multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Class	Precision	Recall	F1-Score	Support
DDoS-HTTP	1	1	1	198
DDoS-TCP	1	1	1	195,476
DDoS-UDP	1	1	1	189,651
DoS-HTTP	1	1	1	297
DoS-TCP	1	1	1	123,160
DoS-UDP	1	1	1	206,595
Normal-Normal	1	1	1	96
Reconnaissance-OS_Fingerprint	1	1	1	3583
Reconnaissance-Service_Scan	1	1	1	14,634
Theft-Data_Exfiltration	0	0	0	1
Theft-Keylogging	0.933333	1	0.965517	14

DNN—Multiclass—Bot-IoT (with SMOTE)

Figure 13 shows the confusion matrix for the DNN multiclass model on Bot-IoT when SMOTE is applied to the training set. All eleven classes are predicted almost perfectly: every class attains precision and recall of 1.0000, except for DoS-UDP (precision = 0.9999, recall = 1.0000, F1 = 0.9999) and Reconnaissance-OS_Fingerprint (precision = 1.0000, recall = 0.9997, F1 = 0.9998). The diagonal entries, therefore, lie at 0.9997–1.0000 for all

classes, with only a single misclassification in Reconnaissance-OS_Fingerprint and a very small precision shortfall for DoS-UDP.

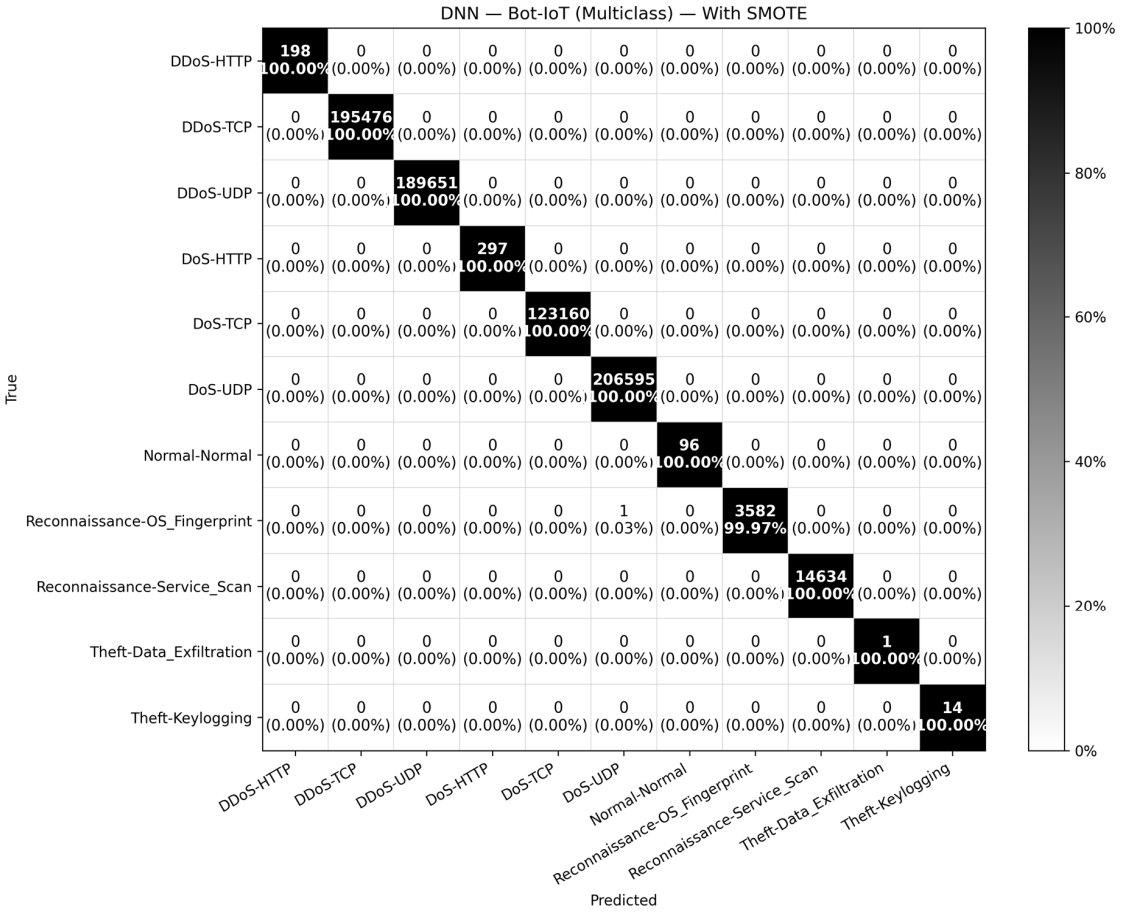

Figure 13. DNN multiclass—Bot-IoT (with SMOTE). This figure shows the confusion matrix for DNN multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique; CM, confusion matrix.

The aggregate metrics in Table 14 confirm this pattern: overall accuracy is 0.9999, macro-precision = 1.0000, macro-recall = 0.9999, macro-F1 = 0.9999, and the macro one-vs.-rest AUC = 1.0000. Weighted precision, recall, and F1 are all 0.9999, reflecting that the few residual errors are negligible relative to the test support of 733,705 flows.

Table 14. DNN multiclass—Bot-IoT (with SMOTE). This table shows the summary metrics for DNN multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Metric	Value
Accuracy	0.9999
Precision (macro)	1.0000
Recall (macro)	0.9999
F1-score (macro)	0.9999
Precision (weighted)	0.9999
Recall (weighted)	0.9999
F1-score (weighted)	0.9999
AUC (macro, OvR)	1.0000

Figure 14 plots the corresponding training and validation losses across epochs. The training loss drops sharply, from approximately 2.5×10^{-3} at epoch 0 to below 5×10^{-4} by epoch 1, and continues decreasing towards zero. The validation loss remains extremely small and flat throughout training, with the best validation loss occurring at epoch 10. The two curves remain closely aligned and do not diverge, indicating stable optimisation with no observable overfitting. Combined with the confusion matrix and summary metrics, these results show that applying SMOTE to Bot-IoT multiclass yields a DNN model with effectively perfect discrimination across all attack categories, while preserving a well-behaved training trajectory.

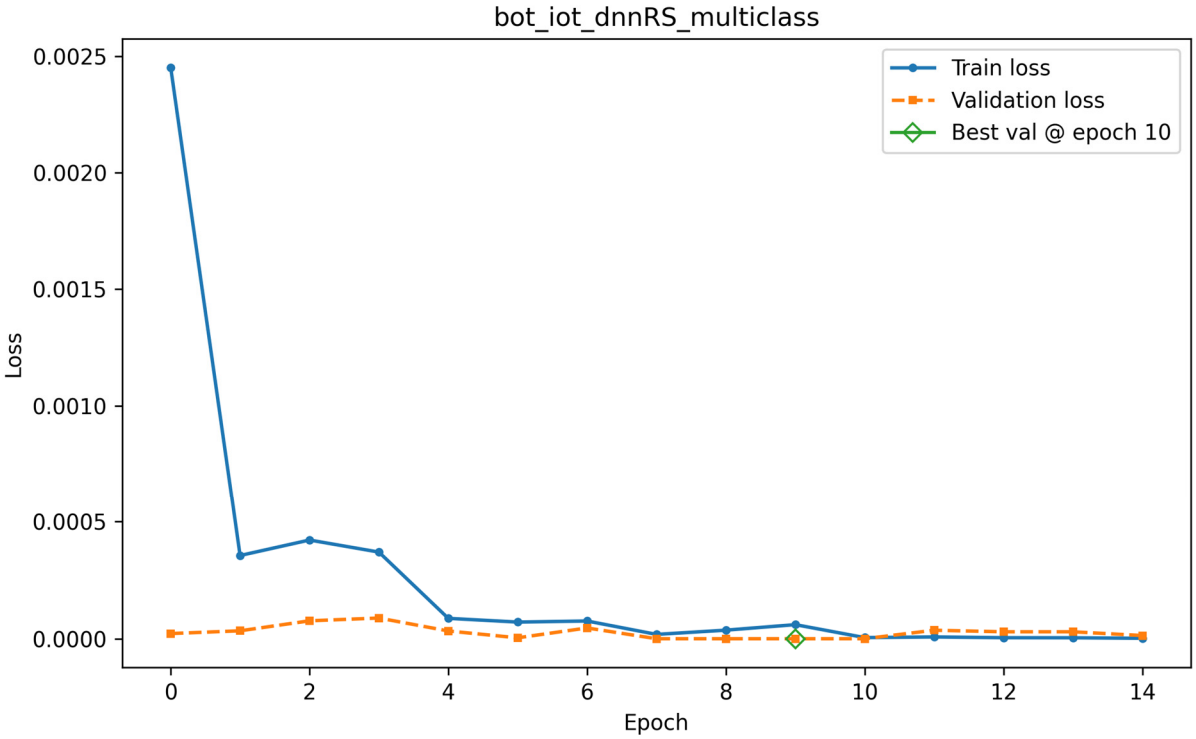

Figure 14. DNN multiclass—Bot-IoT (with SMOTE). This figure shows the training and validation loss curves for DNN multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

N-BaIoT (Multiclass)

Figure 15 shows that DNN multiclass performance on N-BaIoT is strong overall, with most classes predicted almost perfectly (for example, mirai.scan, approximately 1.0000; gafgyt.scan, approximately 0.9990; benign, approximately 1.0000). The remaining mistakes cluster within closely related families, notably gafgyt.junk misclassified as gafgyt.combo (approximately 22.07%) and gafgyt.combo misclassified as gafgyt.junk (approximately 1.81%), and among Mirai UDP/ACK/UDP-plain variants (for example, mirai.udp misclassified as mirai.ack, approximately 1.23%; mirai.udpplain misclassified as mirai.ack, approximately 0.19%; and mirai.udpplain misclassified as mirai.udp, approximately 0.22%).

Figure 16 shows rapid validation loss reduction by epochs 1–2 and stabilisation by around epoch 5, with training and validation curves tracking closely (no divergence), suggesting no observable overfitting. The classification report (Table 15) confirms this pattern numerically. High-support Mirai classes are near perfect (for example, mirai.scan, F1 of approximately 0.9999; mirai.syn, F1 of approximately 0.9999; mirai.udp, F1 of approximately 0.9922; mirai.ack, F1 of approximately 0.9833). Benign traffic is also accurate (precision = 0.9988, recall = 0.9998, F1 = 0.9993, n = 111,187).

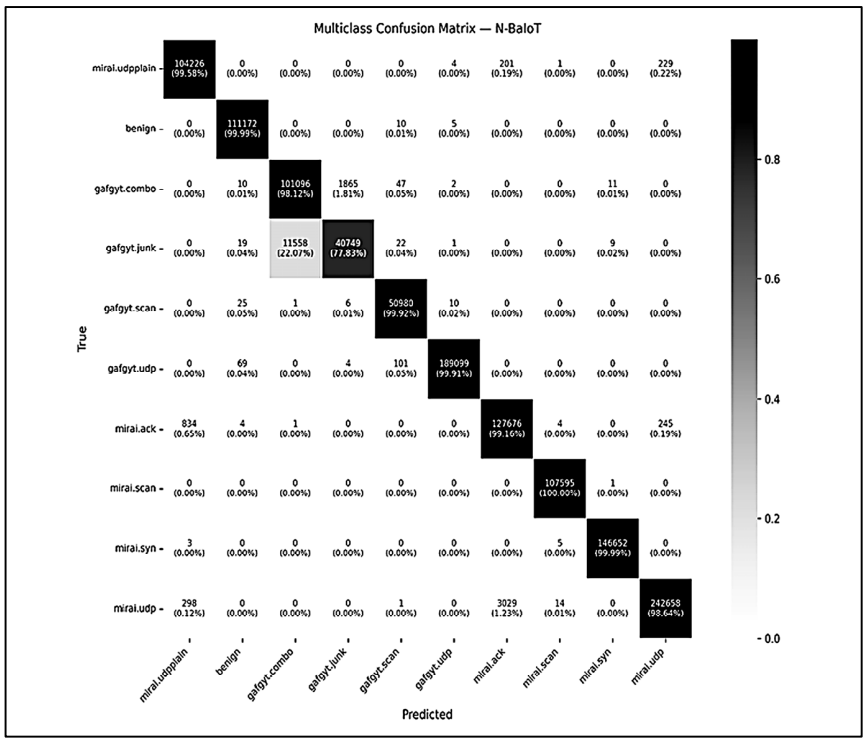

Figure 15. DNN multiclass—N-BaIoT. This figure shows the confusion matrix for DNN multiclass classification on N-BaIoT. Abbreviations: CM, confusion matrix.

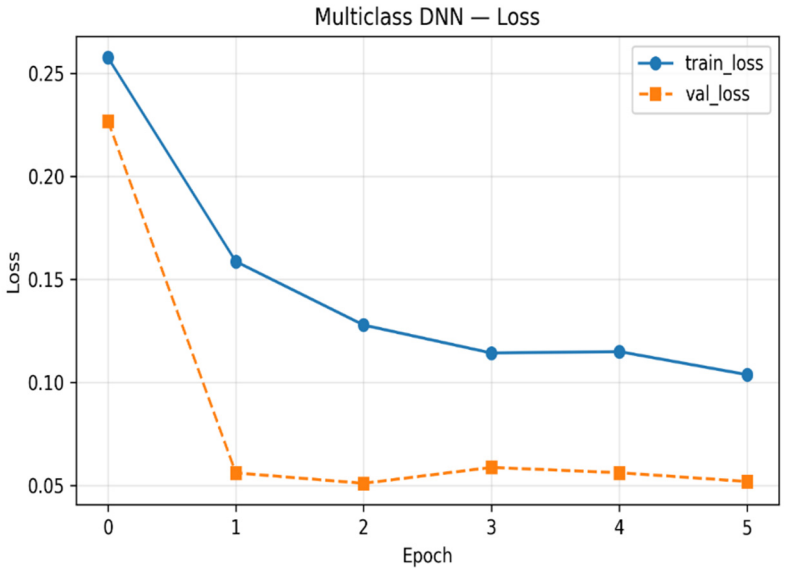

Figure 16. DNN multiclass—N-BaIoT. This figure shows the training and validation loss curves for DNN multiclass classification on N-BaIoT.

The main degradations occur for the fine-grained Gafgyt subclasses: gafgyt.combo shows precision = 0.8973, recall = 0.9812, and F1 = 0.9374 (n = 103,031), while gafgyt.junk shows precision = 0.9560, recall = 0.7782, and F1 = 0.8580 (n = 52,358). This asymmetry (lower precision for combo, lower recall for junk) is consistent with cross-confusions between these two subclasses noted in the confusion matrix. Across the test set, overall performance remains high—accuracy = 0.9849 (N = 1,240,552), F1-weighted = 0.9846, and F1-micro = 0.9849—while macro-F1 = 0.9760, which is slightly lower because macro metrics assign each class equal weight and therefore reflect the minority-class confusions more strongly than weighted or micro metrics.

Table 15. DNN multiclass—N-BaIoT. This table shows the classification report and summary metrics for DNN multiclass classification on N-BaIoT.

Class	Precision	Recall	F1-Score	Support
mirai.udpplain	0.9892320	0.9958423	0.9925287	104,661
Benign	0.9988108	0.9997683	0.9992895	111,187
gafgyt.combo	0.8973106	0.9811929	0.9374491	103,031
gafgyt.junk	0.9560134	0.7781951	0.8579619	52,358
gafgyt.scan	0.9964520	0.9998627	0.9981550	51,022
gafgyt.udp	0.9998815	0.9998991	0.9998903	189,273
mirai.ack	0.9753290	0.9915451	0.9833617	128,764
mirai.scan	0.9997870	0.9999897	0.9998883	107,596
mirai.syn	0.9998370	0.9999591	0.9998981	146,660
mirai.udp	0.9894930	0.9813795	0.9854228	246,000
Accuracy	0.9848702	N/A	N/A	1,240,552
Macro avg	0.9810846	0.9731398	0.9761294	1,240,552
Weighted avg	0.9854607	0.9848702	0.9846562	1,240,552

4.4. Centralised Deep Learning—LSTM

4.4.1. Binary

Bot-IoT (Binary)

Figure 17 shows that the LSTM achieved very high sensitivity on Bot-IoT (binary) but with a small benign over-flag. With TP = 733,602 and FN = 8, the TPR (recall for Attack) is approximately 0.9999. With TN = 89 and FP = 6 out of 95 benign samples, the TNR (specificity for Benign) is approximately 0.9368, indicating six benign instances misclassified as attacks.

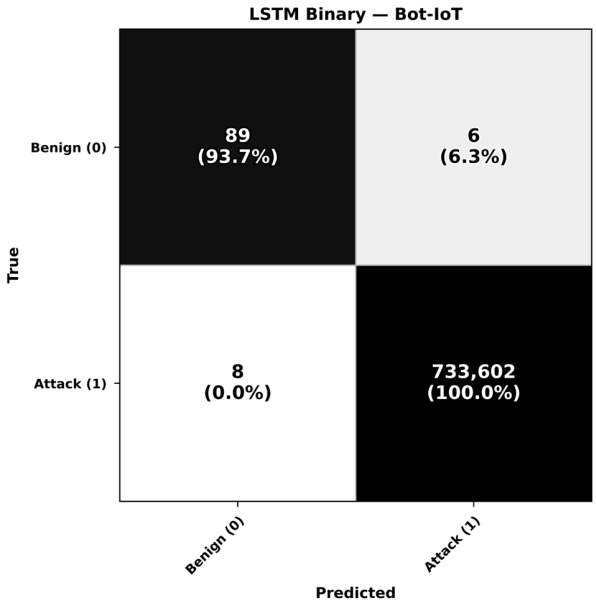

Figure 17. LSTM binary—Bot-IoT. This figure shows the confusion matrix for LSTM binary classification on Bot-IoT. Abbreviations: CM, confusion matrix.

Figure 18 indicates stable optimisation: loss dropped quickly in the first epoch, and training and validation curves remained low and closely aligned thereafter, consistent with a well-regularised two-layer LSTM and no observable overfitting. Table 16 (classification report) mirrors the confusion matrix: Benign (0) shows precision = 0.9175, recall = 0.9368, and F1 = 0.9270 (n = 95), while Attack (1) shows precision = 0.9999, recall = 0.9999, and F1 = 0.9999 (n = 733,610). Overall accuracy is 0.9999 (N = 733,705), with macro-F1 = 0.9635

and weighted F1 = 0.9999. Balanced accuracy was 0.9684 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance.

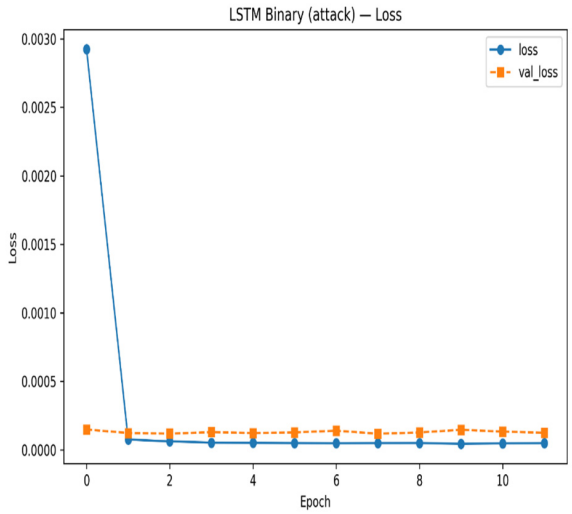

Figure 18. LSTM binary—Bot-IoT. This figure shows the training and validation loss curves for LSTM binary classification on Bot-IoT.

Table 16. LSTM binary—Bot-IoT. This table shows the classification report and summary metrics for LSTM binary classification on Bot-IoT.

Class/Aggregate	Precision	Recall	F1-Score	Support
Benign (0)	0.917526	0.936842	0.927083	95
Attack (1)	0.999992	0.999989	0.999990	733,610

N-BaIoT (Binary)

Figure 19 shows perfect separation on N-BaIoT (binary) with TN = 111,186, FP = 0, FN = 0, and TP = 1,301,336, yielding TPR = 1.0000, TNR = 1.0000, and no residual confusion. Figure 20 confirms stable learning: loss drops sharply in the first epoch and training and validation curves remain near zero and overlap thereafter, indicating no observable overfitting.

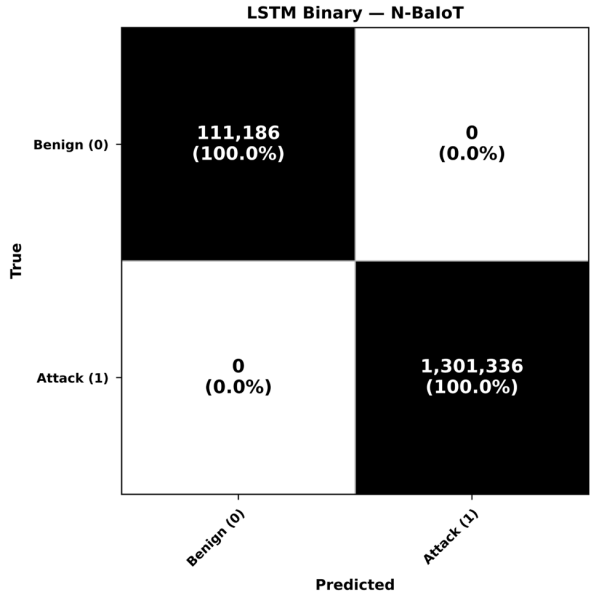

Figure 19. LSTM binary—N-BaIoT. This figure shows the confusion matrix for LSTM binary classification on N-BaIoT. Abbreviations: CM, confusion matrix.

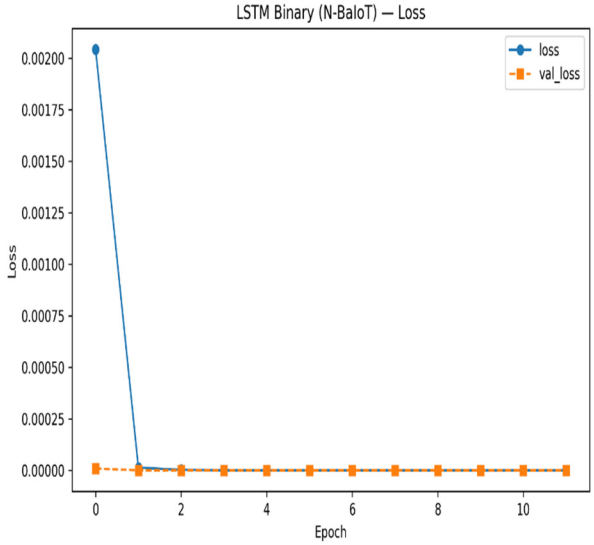

Figure 20. LSTM binary—N-BaIoT. This figure shows the training and validation loss curves for LSTM binary classification on N-BaIoT.

Table 17 (classification report) matches the confusion matrix exactly: Benign (0) and Attack (1) each have precision = 1.0000, recall = 1.0000, and F1 = 1.0000 (n = 111,186 and n = 1,301,336, respectively). Overall performance is therefore accuracy = 1.0000 (N = 1,412,522) with macro-F1 = 1.0000 and weighted F1 = 1.0000, consistent with a clean decision boundary on this task. Balanced accuracy was 1.0000 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

Table 17. LSTM binary—N-BaIoT. This table shows the classification report and summary metrics for LSTM binary classification on N-BaIoT.

Class	Precision	Recall	F1-Score	Support
Benign (0)	1.0000	1.0000	1.0000	111,186
Attack (1)	1.0000	1.0000	1.0000	1,301,336

4.4.2. Multiclass

Bot-IoT (Multiclass, Without SMOTE)

Figure 21 shows that the LSTM classified the major Bot-IoT classes almost perfectly: DDoS-TCP, 195,258/195,476 (approximately 0.9990); DDoS-UDP, 189,646/189,651 (approximately 1.0000); DoS-TCP, 123,129/123,160 (approximately 1.0000); Reconnaissance-Service_Scan, 14,633/14,634 (approximately 1.0000); and Theft-Keylogging, 15/15 (1.0000). The remaining mistakes were concentrated in a few minority classes: DDoS-HTTP was misclassified as DoS-HTTP (44.4%) and DDoS-TCP (4.6%), producing recall = 0.7525; DoS-HTTP was misclassified as Normal (10.8%) and DDoS-HTTP (2.7%), recall = 0.7003; and Normal had recall = 0.9053 with some misclassification as Reconnaissance-OS_Fingerprint (9.5%).

Figure 22 shows steady reductions in both training and validation losses across epochs, with the curves tracking closely and with no divergence suggestive of overfitting, indicating stable optimisation under the RS-best parameters. Consistent with the confusion matrix, Table 18 (classification report) shows DDoS-HTTP (precision = 0.6506, recall = 0.7525, F1 = 0.6978; n = 198), DoS-HTTP (precision = 0.8353, recall = 0.7003, F1 = 0.7619; n = 297), and Normal (precision = 0.9885, recall = 0.9053, F1 = 0.9450; n = 95). Overall accuracy is 0.9995, with weighted F1 = 0.9995 and macro-F1 = 0.8518. The lower macro score reflects equal weighting across all classes and is driven by confusions between minority classes,

while the near-perfect accuracy and weighted F1 confirm that the model cleanly separated the dominant classes.

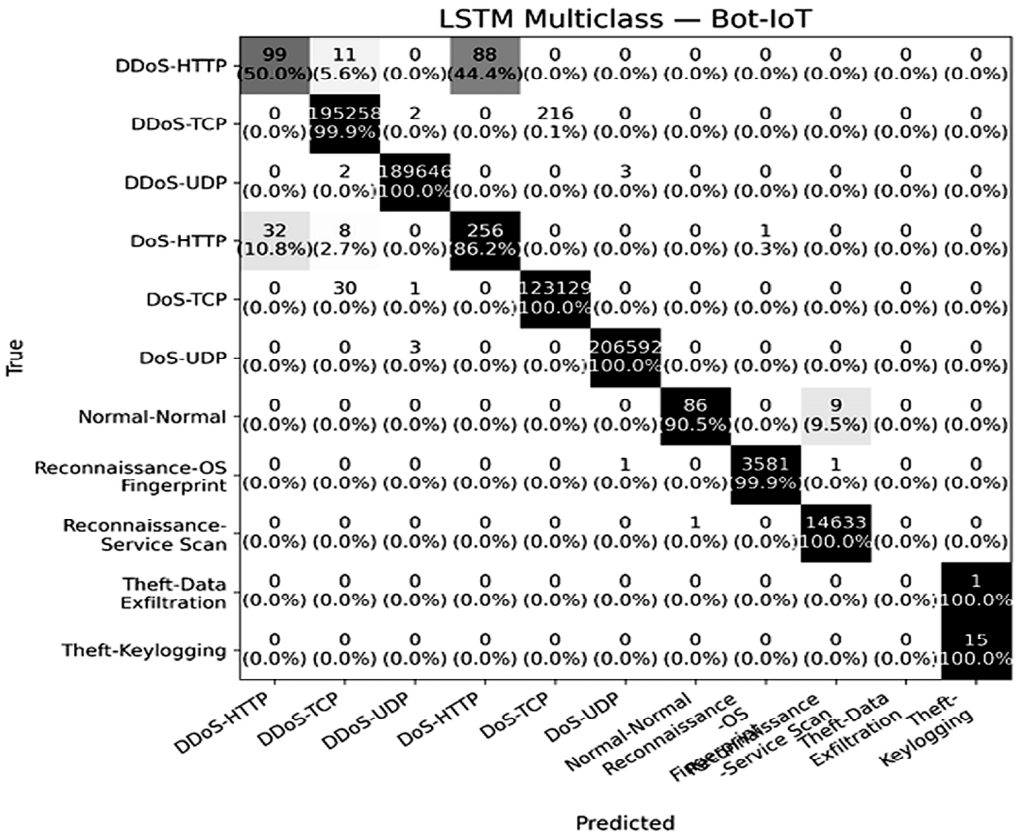

Figure 21. LSTM multiclass—Bot-IoT (without SMOTE). This figure shows the confusion matrix for LSTM multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique; CM, confusion matrix.

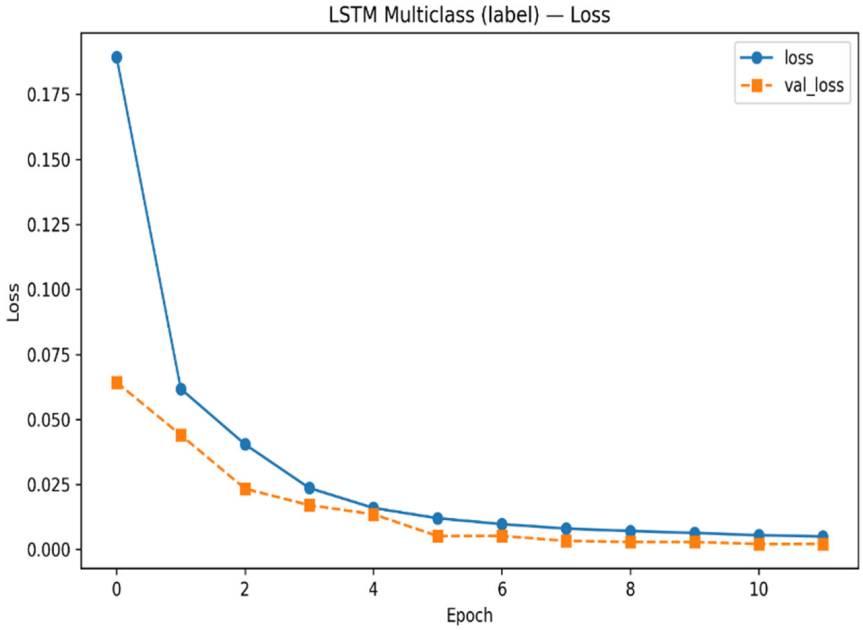

Figure 22. LSTM multiclass—Bot-IoT (without SMOTE). This figure shows the training and validation loss curves for LSTM multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Table 18. LSTM multiclass—Bot-IoT (without SMOTE). This table shows the classification report and summary metrics for LSTM multiclass classification on Bot-IoT (without SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Class/Aggregate	Precision	Recall	F1-Score	Support
DDoS-HTTP	0.650655	0.752525	0.697892	198
DDoS-TCP	0.999483	0.999407	0.999445	195,476
DDoS-UDP	0.999979	0.999974	0.999976	189,651
DoS-HTTP	0.835341	0.700337	0.761905	297
DoS-TCP	0.999058	0.999318	0.999188	123,160
DoS-UDP	0.999985	0.999985	0.999985	206,595
Normal-Normal	0.988506	0.905263	0.945055	95
Reconnaissance-OS Fingerprint	0.999721	1.000000	0.999860	3583
Reconnaissance-Service Scan	0.999385	0.999932	0.999658	14,634
Theft-Data Exfiltration	0.000000	0.000000	0.000000	1
Theft-Keylogging	0.937500	1.000000	0.967742	15
Macro avg	0.855419	0.850613	0.851883	733,705
Weighted avg	0.999516	0.999513	0.999511	733,705

LSTM—Multiclass—Bot-IoT (with SMOTE)

To assess the impact of class rebalancing on the multiclass LSTM, the Bot-IoT CDL experiment was repeated, applying SMOTE to the minority attack types before training. Figure 23 shows that the confusion matrix becomes almost perfectly diagonal: all major classes (DDoS-TCP, DDoS-UDP, DoS-TCP, DoS-UDP, Reconnaissance-Service_Scan) achieve 0.9990–1.0000 row-wise recall, Normal-Normal is recovered with 0.9895 recall (94/95), and only a handful of residual errors appear in DDoS-HTTP (197/198) and DoS-HTTP (295/297). The ultra-minor Theft-Data_Exfiltration and Theft-Keylogging classes are classified perfectly (F1 = 1.0000) after oversampling.

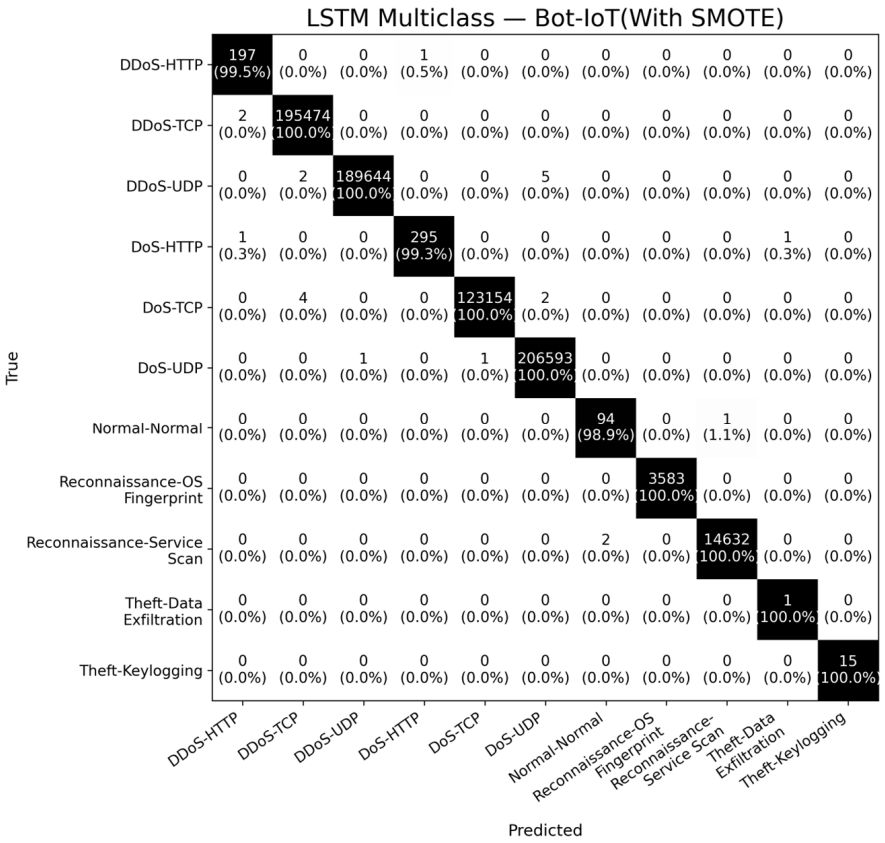

Figure 23. LSTM multiclass—Bot-IoT (with SMOTE). This figure shows the confusion matrix for LSTM multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique; CM, confusion matrix.

The corresponding classification report (Table 19) indicates overall accuracy = 1.0000 (N = 733,705), precision (weighted) = 1.0000, recall (weighted) = 1.0000, and F1 (weighted) = 1.0000, with macro-F1 = 0.9669 and macro precision and recall of 0.9510 and 0.9980, respectively. The macro-averaged F1 therefore improved substantially compared with the non-SMOTE LSTM, while the weighted metrics remained saturated at approximately 1.0000. The loss curves in Figure 24 confirm stable convergence without overfitting, with both training and validation loss decreasing rapidly and flattening near zero. Overall, this ablation shows that applying SMOTE to the Bot-IoT multiclass substantially improved minority-class performance with the LSTM while preserving the excellent overall metrics reported for the original CDL configuration.

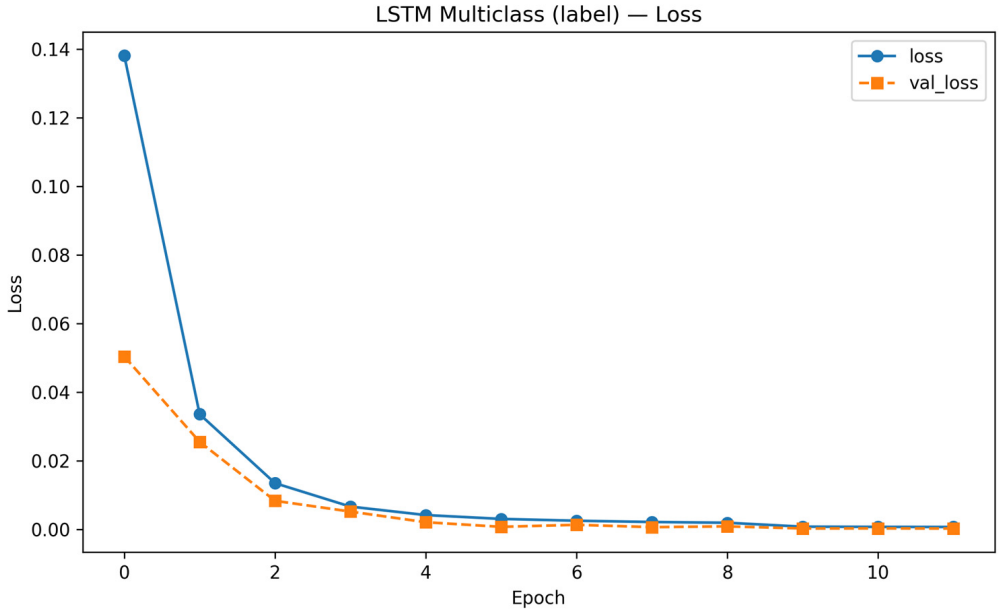

Figure 24. LSTM multiclass—Bot-IoT (with SMOTE). This figure shows the training and validation loss curves for LSTM multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Table 19. LSTM multiclass—Bot-IoT (with SMOTE). This table shows the classification report and summary metrics for LSTM multiclass classification on Bot-IoT (with SMOTE). Abbreviations: SMOTE, synthetic minority oversampling technique.

Class/Aggregate	Precision	Recall	F1-Score	Support
DDoS-HTTP	0.9850	0.9949	0.9899	198
DDoS-TCP	1.0000	1.0000	1.0000	195,476
DDoS-UDP	1.0000	1.0000	1.0000	189,651
DoS-HTTP	0.9866	0.9932	0.9899	297
DoS-TCP	0.9998	1.0000	0.9999	123,160
DoS-UDP	1.0000	0.9999	1.0000	206,595
Normal-Normal	0.9796	0.9895	0.9845	96
Reconnaissance-OS_Fingerprint	1.0000	1.0000	1.0000	3583
Reconnaissance-Service_Scan	0.9999	1.0000	1.0000	14,634
Theft-Data_Exfiltration	1.0000	1.0000	1.0000	1
Theft-Keylogging	1.0000	1.0000	1.0000	15
Accuracy	1.0000	1.0000	1.0000	733,705
Macro avg	0.9510	0.9980	0.9669	733,705
Weighted avg	1.0000	1.0000	1.0000	733,705

N-BaIoT (Multiclass)

Figure 25 shows an LSTM that is essentially perfect across most N-BaIoT classes: for example, benign, 111,184/111,187; gafgyt.scan, 51,012/51,019; mirai.scan, 107,591/107,595; and mirai.syn, 146,654/146,660, are all correct at or near 1.0000. The largest residual confusion is within the Gafgyt family—gafgyt.combo misclassified as gafgyt.junk (2164; 2.1%) and gafgyt.junk misclassified as gafgyt.combo (662; 1.3%). Smaller, semantically related errors appear among Mirai UDP variants (for example, mirai.udp misclassified as mirai.udpplain, 28; mirai.udpplain misclassified as mirai.udp, 52) and a few single-digit cross-family misclassifications from mirai.ack to mirai.scan/syn/udp/udpplain. These patterns are consistent with overlap in traffic characteristics among closely related attack types.

Figure 26 shows that the training and validation losses decrease together without divergence, indicating stable optimisation with the selected capacity and regularisation. Consistent with the confusion matrix, the classification report yields very high per-class precision, recall, and F1 for the large classes; the modest penalties arise from the Gafgyt pair noted above. Overall accuracy is 0.9975 (N = 1,240,513), confirming strong overall performance with a small number of family-level confusions concentrated in closely related subclasses.

As shown in Table 20, LSTM achieved near-perfect multiclass performance on N-BaIoT, with overall accuracy 0.997579 ($N = 1,240,513$), weighted F1 0.997587, and macro F1 0.995819. The only notable reductions were for gafgyt.combo (recall 0.978909) and gafgyt.junk (precision 0.959797), while most other classes remained at approximately 0.999 F1.

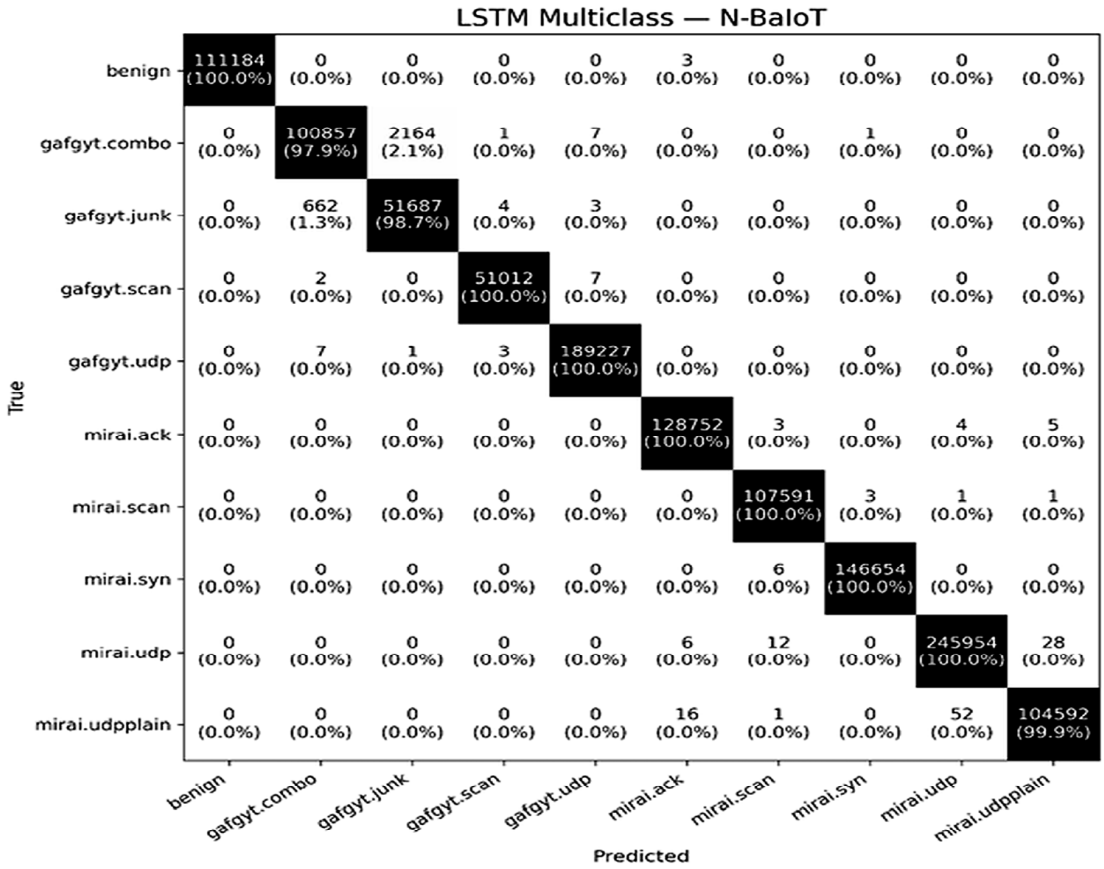

Figure 25. LSTM multiclass—N-BaIoT. This figure shows the confusion matrix for LSTM multiclass classification on N-BaIoT. Abbreviations: CM, confusion matrix.

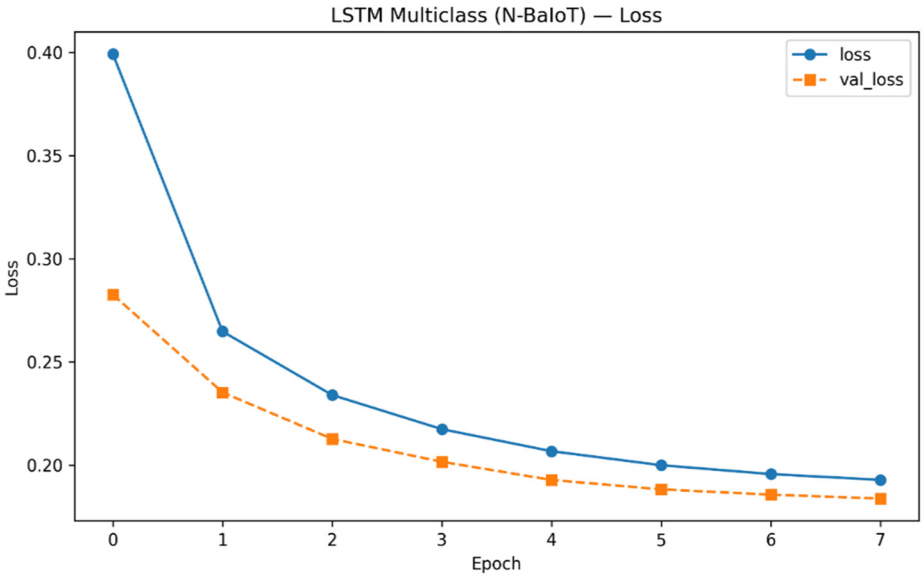

Figure 26. LSTM multiclass—N-BaIoT. This figure shows the training and validation loss curves for LSTM multiclass classification on N-BaIoT.

Table 20. LSTM multiclass—N-BaIoT. This table shows the classification report and summary metrics for LSTM multiclass classification on N-BaIoT.

Class	Precision	Recall	F1-Score	Support
Benign	1.000000	0.999973	0.999987	111,187
gafgyt.combo	0.993391	0.978909	0.986097	103,030
gafgyt.junk	0.959797	0.987222	0.973317	52,356
gafgyt.scan	0.999843	0.999824	0.999833	51,021
gafgyt.udp	0.999826	0.999942	0.999884	189,238
mirai.ack	0.999969	0.999907	0.999938	128,764
mirai.scan	0.999861	0.999954	0.999907	107,596
mirai.syn	0.999891	0.999959	0.999925	146,660
mirai.udp	0.999768	0.999813	0.999791	246,000
mirai.udpplain	0.999675	0.999341	0.999508	104,661
Accuracy	N/A	N/A	0.997579	1,240,513
Macro avg	0.995202	0.996484	0.995819	1,240,513
Weighted avg	0.997620	0.997579	0.997587	1,240,513

Across the four LSTM runs, binary detection was excellent, and multiclass performance was very strong. On Bot-IoT (binary), it achieved near-perfect sensitivity (TPR approximately 0.9999) with a small benign-to-attack false-alarm component (TNR, approximately 0.9368; 6 FP out of 95 benign). On N-BaIoT (binary), it was perfect (TPR = TNR = 1.0000). For multiclass Bot-IoT, training was stable, with errors concentrated in a few minority classes (e.g., DDoS-HTTP versus DoS-HTTP, Normal misclassified as Reconnaissance-OS_Fingerprint), yielding high accuracy but lower macro-F1. For multiclass N-BaIoT, performance was near-perfect overall, with only small family-level confusions (e.g., gafgyt.combo and gafgyt.junk; Mirai UDP variants).

4.5. Centralised Deep Learning—BiLSTM

4.5.1. Binary

Bot-IoT (Binary)

Figure 27 shows that BiLSTM achieved near-perfect attack detection on Bot-IoT (binary): TP = 733,538, FN = 71 (TPR = 0.9999), with TN = 95 and FP = 1 for the benign class (TNR = 0.9896). The single benign false positive and the 71 missed attacks explain the small

specificity shortfall despite extremely strong sensitivity. Figure 28 shows well-behaved training: an early drop in loss followed by low, closely tracking training and validation curves without overfitting.

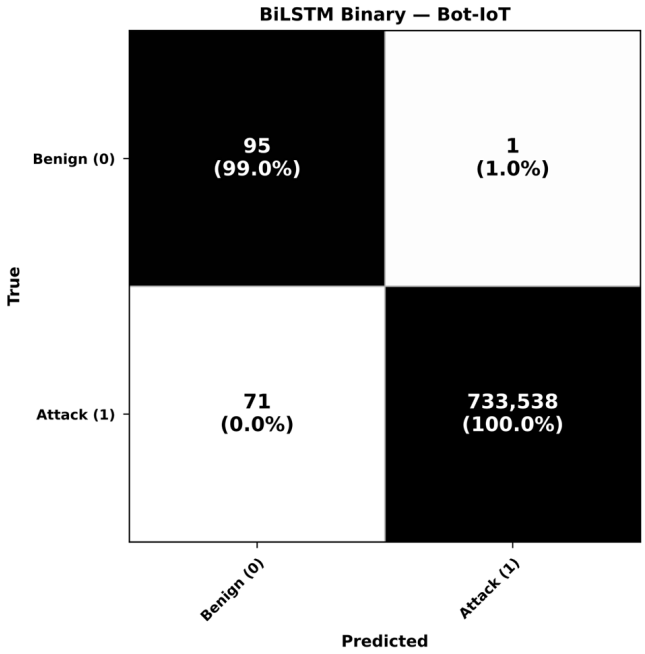

Figure 27. BiLSTM binary—Bot-IoT. This figure shows the confusion matrix for BiLSTM binary classification on Bot-IoT. Abbreviations: CM, confusion matrix.

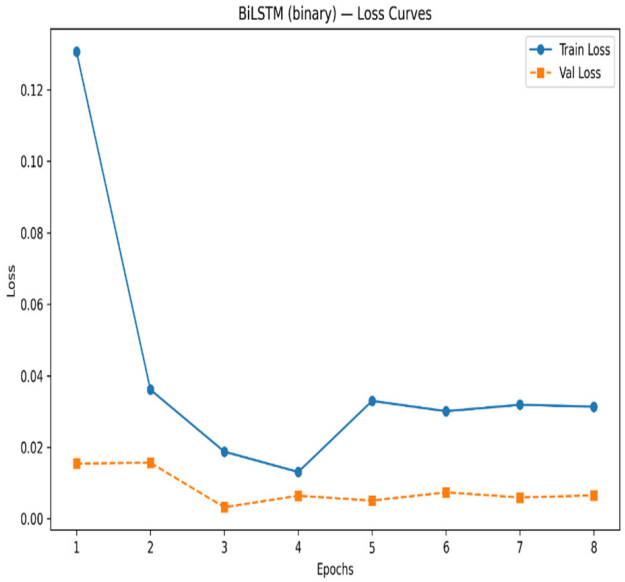

Figure 28. BiLSTM binary—Bot-IoT. This figure shows the training and validation loss curves for BiLSTM binary classification on Bot-IoT.

Consistent with the confusion matrix, Table 21 (classification report) shows very strong scores for the Attack class, while the Benign class F1 is lower because those 71 attacks predicted as benign reduce its precision/recall balance. Overall accuracy is 0.9999 (N = 733,705), with macro-F1 = 0.8626 and AUC = 0. 9999. Balanced accuracy was 0.9947 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance.

Table 21. BiLSTM binary—Bot-IoT. This table shows the classification report and summary metrics for BiLSTM binary classification on Bot-IoT.

Class	Precision	Recall	F1-Score	Support
benign	0.572289	0.989583	0.725191	96
attack	0.999999	0.999903	0.999951	733,609

N-BaIoT (Binary)

Figure 29 shows near-perfect separation on N-BaIoT (binary): TN = 111,159, FP = 28, FN = 0, and TP = 1,301,335, producing TPR = 1.0000 and TNR of approximately 0.9997. The 28 benign samples flagged as attack account for the very small specificity shortfall. Figure 30 confirms stable training: a large first-epoch loss drop followed by closely tracking training and validation curves with no overfitting.

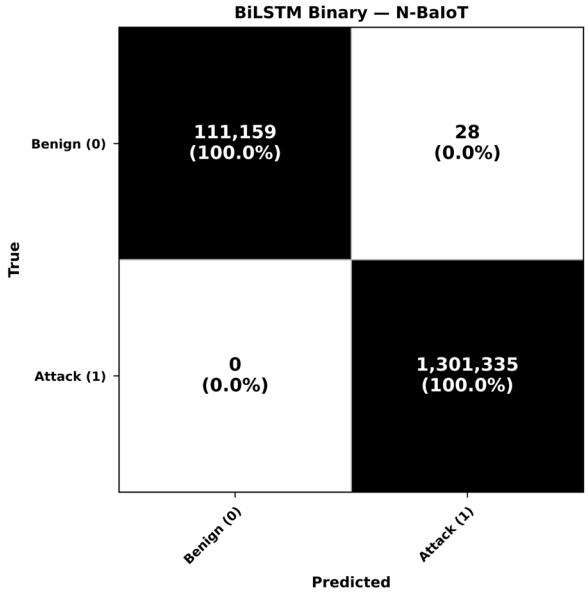

Figure 29. BiLSTM binary—N-BaIoT. This figure shows the confusion matrix for BiLSTM binary classification on N-BaIoT. Abbreviations: CM, confusion matrix.

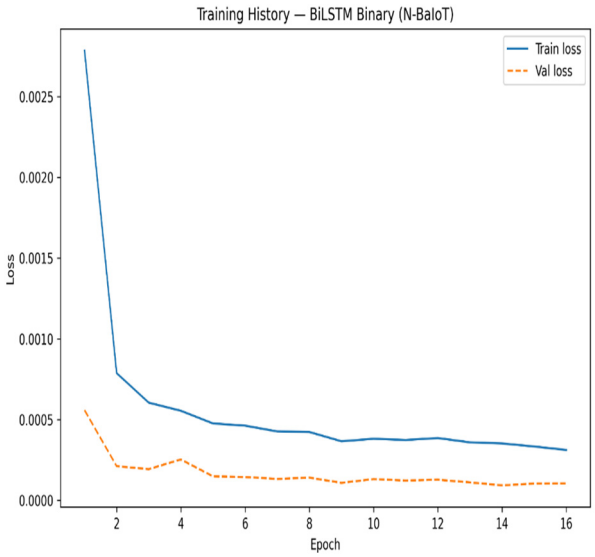

Figure 30. BiLSTM binary—N-BaIoT. This figure shows the training and validation loss curves for BiLSTM binary classification on N-BaIoT.

Consistent with the confusion matrix, Table 22 (classification report) reports per-class precision, recall, and F1 at or near unity for the Attack class and very high scores for Benign, matching the observed counts. Overall accuracy is 0.9999 (N = 1,412,522), and AUC is effectively 1.0000, indicating an excellent decision boundary with an extremely low false-alarm rate. Balanced accuracy was 0.9999 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

Table 22. BiLSTM binary—N-BaIoT. This table shows the classification report and summary metrics for BiLSTM binary classification on N-BaIoT.

Class	Precision	Recall	F1-Score	Support
Benign (0)	1.000000	0.999748	0.999874	111,187
Attack (1)	0.999978	1.000000	0.999989	1,301,335
Accuracy	0.999980	0.999980	0.999980	0.999980
Macro avg	0.999989	0.999874	0.999932	1,412,522
Weighted avg	0.999980	0.999980	0.999980	1,412,522

4.5.2. Multiclass

Bot-IoT (Multiclass)

Figure 31 shows that BiLSTM achieved near-perfect per-class recognition on Bot-IoT (multiclass): diagonals were at or near 0.9900–1.0000 for all classes, with Normal-Normal correct 97.9% of the time. The remaining errors were small and largely confined to family-related confusions, such as a few DDoS-TCP samples misclassified as DDoS-UDP (56), DDoS-UDP samples misclassified as Normal-Normal (51), isolated errors such as DoS-HTTP misclassified as Normal-Normal (1) and DoS-TCP misclassified as DDoS-UDP (4), and rare single-digit cross-labels between Normal and reconnaissance classes.

Figure 32 corroborates stable optimisation: a large first-epoch loss drop, followed by steadily declining, tightly coupled training and validation curves without divergence. Consistent with these patterns, Table 23 (classification report) shows very high per-class precision, recall, and F1 across the board, while Table 24 (metrics summary) reports overall accuracy = 0.9997, macro-precision = 0.9260, macro-recall = 0.9959, macro-F1 = 0.9520, macro TNR = 0.9999, and AUC = 0.9999. The small gap between macro-precision and macro-recall reflects only a few family-level confusions (e.g., between DDoS/DoS variants and Normal–Reconnaissance pairs), which modestly lower macro-F1 despite near-perfect overall performance.

Table 23. BiLSTM multiclass—Bot-IoT. This table shows the classification report and summary metrics for BiLSTM multiclass classification on Bot-IoT.

Class	Precision	Recall	F1-Score	Support
DDoS-HTTP	1.000000	0.994949	0.997468	198
DDoS-TCP	0.999990	0.999714	0.999852	195,476
DDoS-UDP	0.999705	0.999721	0.999713	189,651
DoS-HTTP	1.000000	0.993266	0.996622	297
DoS-TCP	0.999976	0.999968	0.999972	123,160
DoS-UDP	0.999734	1.000000	0.999867	206,595
Normal-Normal	0.758065	0.979167	0.854545	96
Reconnaissance-OS_Fingerprint	0.996629	0.990232	0.993420	3583
Reconnaissance-Service_Scan	0.999248	0.998770	0.999009	14,634
Theft-Data_Exfiltration	0.500000	1.000000	0.666667	1
Theft-Keylogging	0.933333	1.000000	0.965517	14

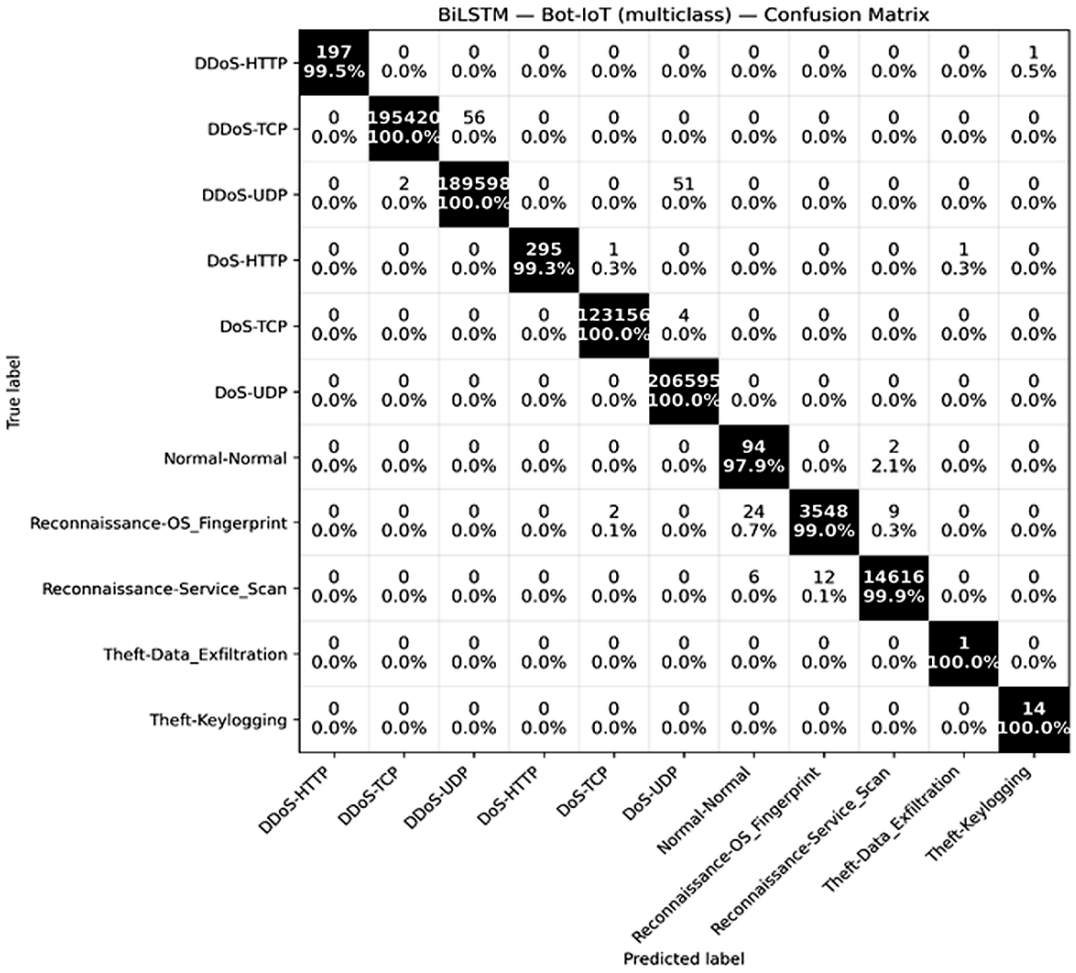

Figure 31. BiLSTM multiclass—Bot-IoT. This figure shows the confusion matrix for BiLSTM multiclass classification on Bot-IoT. Abbreviations: CM, confusion matrix.

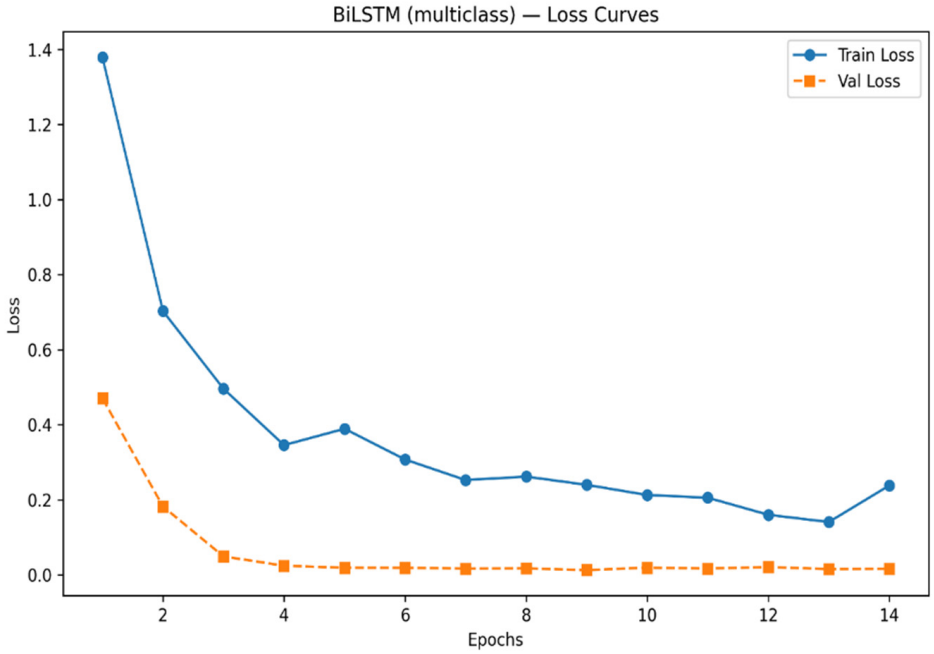

Figure 32. BiLSTM multiclass—Bot-IoT. This figure shows the training and validation loss curves for BiLSTM multiclass classification on Bot-IoT.

Table 24. BiLSTM multiclass—N-BaIoT. This table shows the classification report and summary metrics for BiLSTM multiclass classification on N-BaIoT.

Class	Precision	Recall	F1-Score	Support
Benign	0.9988	0.9997	0.9993	111,187
gafgyt.combo	0.8162	0.7911	0.8035	103,031
gafgyt.junk	0.6127	0.6481	0.6300	52,358
gafgyt.scan	0.9967	0.9997	0.9982	51,022
gafgyt.udp	0.9993	0.9992	0.9992	189,273
mirai.ack	0.6460	0.5528	0.5958	128,764
mirai.scan	0.9969	0.9986	0.9977	107,596
mirai.syn	0.9997	0.9997	0.9997	146,660
mirai.udp	0.7530	0.8216	0.7853	246,000
mirai.udpplain	0.8713	0.8396	0.8553	104,661

N-BaIoT (Multiclass)

Figure 33 shows that BiLSTM performed strongly across several N-BaIoT classes, with benign = 111,139 and gafgyt.scan = 51,011 classified essentially perfectly, while two family-level confusion clusters dominated the residual errors: within Gafgyt (notably gafgyt.combo misclassified as gafgyt.junk: 21,445; 20.8%, and gafgyt.junk misclassified as gafgyt.combo: 18,345; 35.0%) and among Mirai UDP variants (for example, mirai.ack misclassified as mirai.udp: 50,549; 39.3%, and mirai.udpplain misclassified as mirai.udp: 15,702; 15.0%).

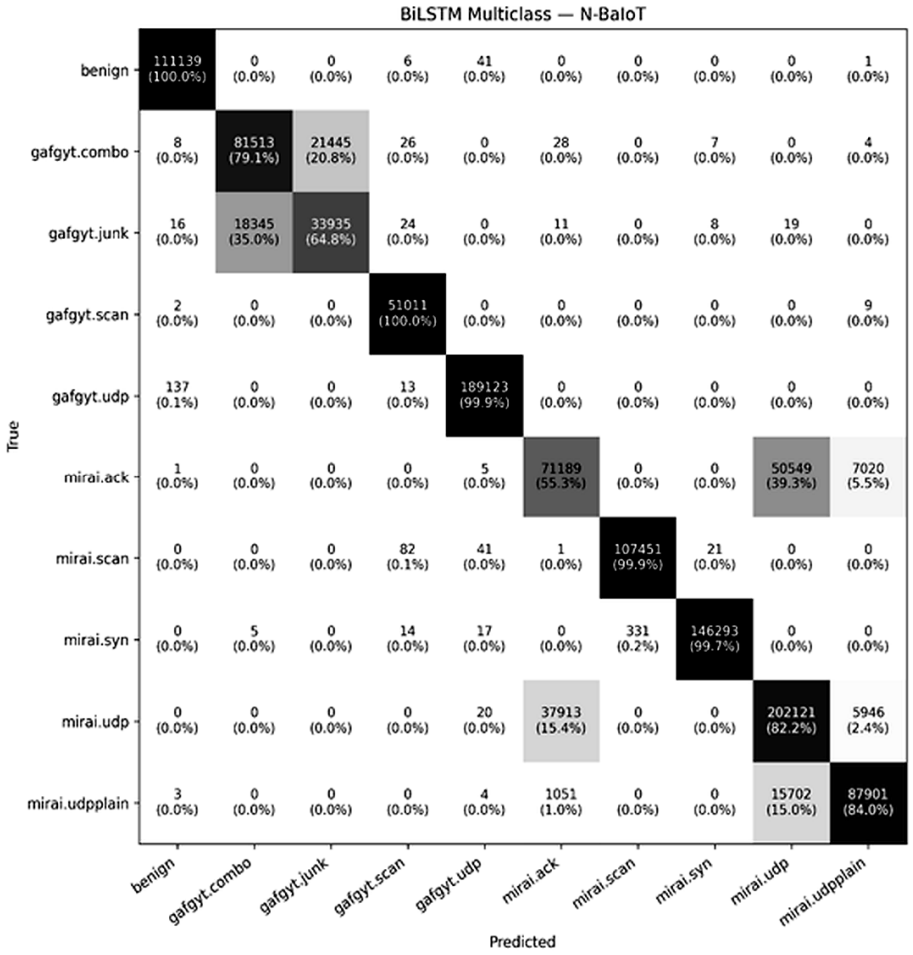

Figure 33. BiLSTM multiclass—N-BaIoT. This figure shows the confusion matrix for BiLSTM multiclass classification on N-BaIoT. Abbreviations: CM, confusion matrix.

Figure 34 shows a rapid reduction in loss by epoch 2 (the selected checkpoint), followed by a mild validation loss uptick—consistent with early stopping at the stability point to guard against overfitting. Consistent with these patterns, the metrics summary (Table 25) reports overall accuracy = 0.8719, macro-precision = 0.8691, macro-recall = 0.8648, macro-F1 = 0.8663, and macro AUC of approximately 0.9892, indicating good rank ordering even when thresholded predictions conflate closely related subclasses. Overall performance is solid, and the lower macro averages are mainly driven by intra-family confusion between Gafgyt combo/junk and Mirai UDP versus UDP-plain; targeted feature engineering or a hierarchical output head could further separate these fine-grained labels.

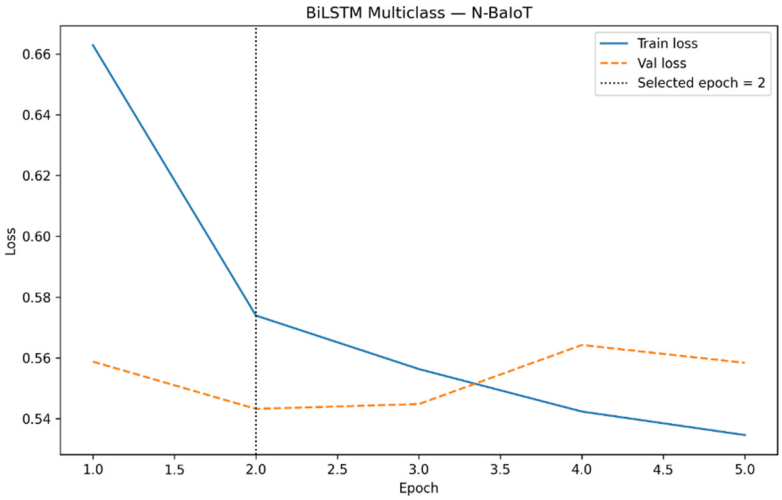

Figure 34. BiLSTM multiclass—N-BaIoT. This figure shows the training and validation loss curves for BiLSTM multiclass classification on N-BaIoT.

Table 25. DNN binary—Bot-IoT (FL). This table shows the classification report and summary metrics for DNN binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.99850076	99.99852712	97.235806

4.6. Federated Deep Learning (FDL)—DNN, LSTM and BiLSTM

This section summarises the federated results (FL/FDL) for the three architectures under the same RS-best configurations and non-IID client splits. For each model and dataset, the global (server-side) confusion matrices and metrics are reported, and per-client behaviour is briefly summarised. These results address RQ3 by quantifying how far FDL preserves CDL performance under non-IID conditions, and they contribute to RQ4 by revealing trade-offs between accuracy, robustness, and decentralisation.

4.6.1. Federated DNN (FDL-DNN)

Bot-IoT (Binary, FL)

Figure 35 shows that the federated DNN on Bot-IoT (binary) maintained very strong separation between benign and attack traffic. The Benign class (0) had 94 true negatives and 2 false positives (n = 96), while the Attack class (1) had 733,600 true positives and 9 false negatives (n = 733,609), giving a Benign TNR of 94/(94 + 2) approximately 0.9792 and an Attack TPR of approximately 0.9999. Balanced accuracy was 0.9896 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance.

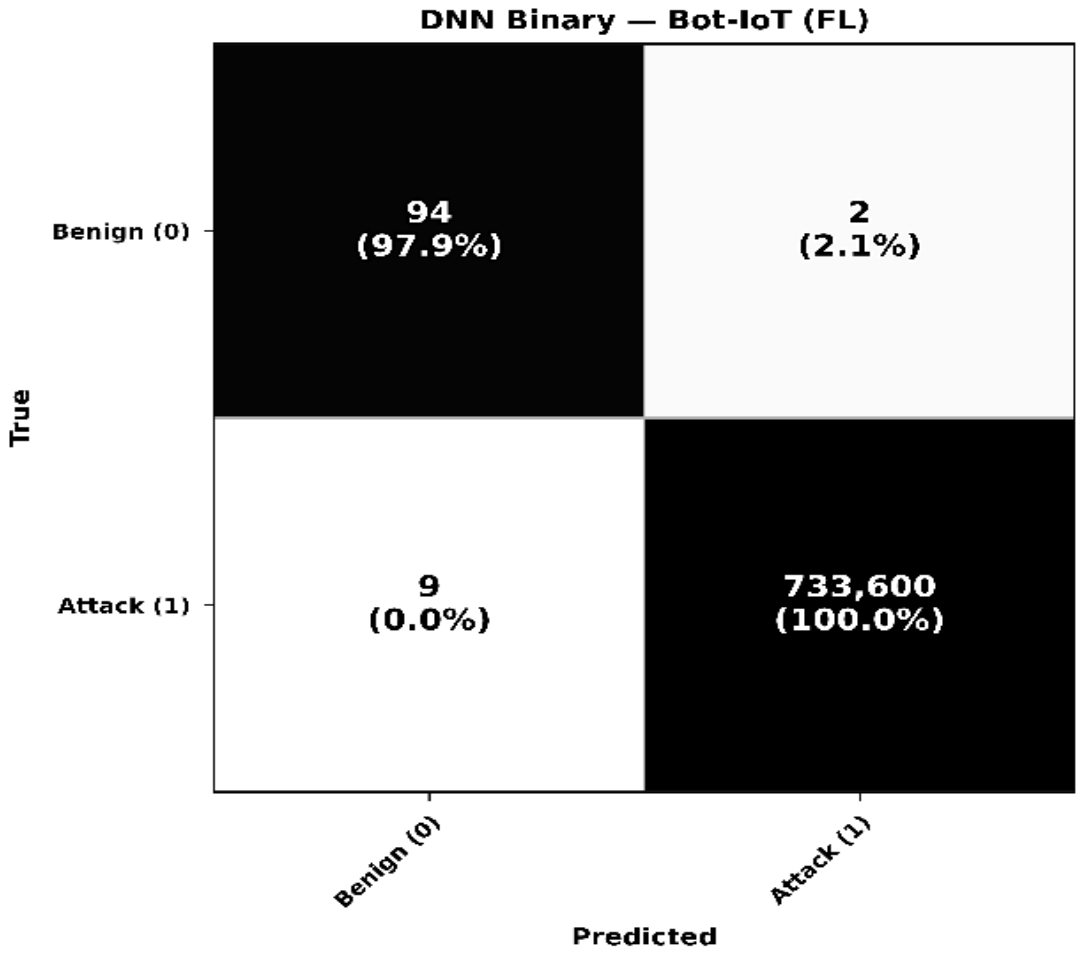

Figure 35. DNN binary—Bot-IoT (FL). This figure shows the confusion matrix for DNN binary classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Consistent with this confusion matrix, Table 26 (global classification report) shows that Benign (0) achieved a precision of approximately 0.9130, a recall of approximately 0.9790, and an F1 of approximately 0.9450, while Attack (1) achieved a precision of approximately 0.9999, a recall of approximately 0.9999, and an F1 of approximately 0.9999. At the dataset level, the classification report indicated an overall accuracy of 0.9999 (N = 733,705), with macro-F1 = 0.9723 and weighted F1 = 0.9999. The small gap between macro-F1 and weighted F1 reflects the fact that the few residual errors were concentrated in the minority Benign class, while the dominant Attack class was classified almost perfectly.

Table 26. DNN binary—Bot-IoT (FL). This table shows the summary metrics for DNN binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	99.9992	99.9992	99.9992	99.9992
client2	100.0000	100.0000	100.0000	100.0000
client3	99.9935	100.0000	99.9935	99.9973
client4	99.9980	99.9984	99.9981	99.9981
client5	99.9981	99.9981	99.9981	99.9981

The corresponding client-side metrics in Table 27 show that all participating clients also achieved high accuracy and weighted F1-scores that were closely aligned with the global model, indicating that the federated optimisation did not privilege any individual client and that the learned global model was representative of client-level behaviour. The

DNN-FL global model thus achieved a weighted F1 of approximately 0.9999 and a TNR of approximately 0.9792 on Benign, indicating near-centralised performance with a very low false-alarm rate on normal traffic.

Table 27. DNN binary—N-BaIoT (FL). This table shows the classification report and summary metrics for DNN binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.95915108	99.959181	99.8593853

N-BaIoT (Binary, FL)

Figure 36 demonstrates that the same federated DNN also generalised very well to N-BaIoT (binary). The Benign class (0) achieved 111,077 true negatives and 110 false positives (n = 111,187), and the Attack class (1) achieved 1,300,868 true positives and 467 false negatives (n = 1,301,335). This corresponds to a Benign TNR of 111,077/(111,077 + 110), approximately 0.9990, and an Attack TPR of approximately 0.9996. Balanced accuracy was 0.9993 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

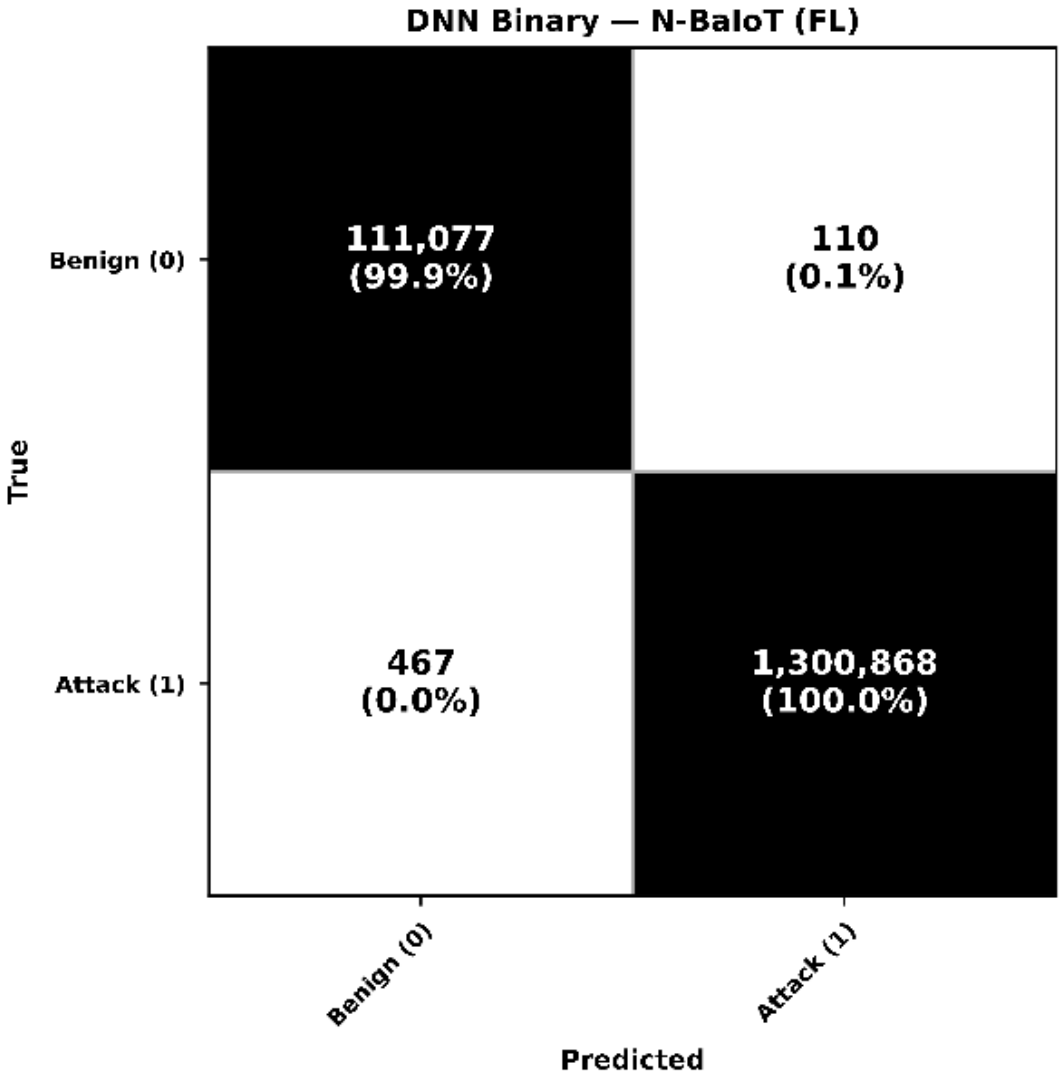

Figure 36. DNN binary—N-BaIoT (FL). This figure shows the confusion matrix for DNN binary classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Table 28 (global classification report) shows Benign (0) with precision of approximately 0.9960, recall of approximately 0.9990, and F1 of approximately 0.9970 and Attack (1) with precision of approximately 0.9999, recall of approximately 0.9996, and F1 of approximately 0.9998. Overall performance was accuracy = 0.9995 (N = 1,412,522), with macro-F1 = 0.9985 and weighted F1 = 0.9995. The very small difference between macro-F1 and weighted F1 indicates that both benign and attack traffic were classified almost perfectly, with only a very small number of residual misclassifications at the decision boundary.

Table 28. DNN binary—N-BaIoT (FL). This table shows the summary metrics for DNN binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	99.96150819	99.96158800	99.96150819	99.96155325
client2	99.90611431	99.90646650	99.90611431	99.90615888
client3	99.96128274	99.96883140	99.96128274	99.96583651
client4	99.91895522	99.91921290	99.91895522	99.91910454
client5	99.96051487	99.96055990	99.96051487	99.96052883

The client-side results in Table 29 further confirm that accuracy and weighted F1 remained consistently high across all clients and closely tracked the global scores, suggesting that the federated training remained stable under the non-IID splits used and that no client experienced a marked degradation in local performance. The DNN-FL global model achieved a weighted F1 of approximately 0.9995 with a TNR of approximately 0.9990 on benign traffic, that is, near-perfect detection with a very low false-alarm rate on N-BaIoT.

Table 29. DNN multiclass—Bot-IoT (FL). This table shows the classification report and summary metrics for DNN multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)	Macro_AUC_OvR (%)
global_final	98.1438044	98.1856823	79.5953437	99.92727

Bot-IoT (Multiclass, FL)

Figure 37 shows the server-side confusion matrix for the federated DNN multiclass model on Bot-IoT, with row-normalised percentages: nearly all high-support classes (such as DDoS-UDP, DoS-UDP, and Reconnaissance-Service_Scan) were predicted almost perfectly, whereas DDoS-HTTP and DoS-HTTP exhibited noticeable leakage across related DoS/DDoS categories and Normal-Normal had a small spill into reconnaissance classes; these patterns explain the gap between macro- and weighted-average scores.

Consistent with this, Table 30 (global classification report) reports overall accuracy of 0.9814 (N = 733,705), weighted precision of 0.9837, weighted recall of 0.9814, weighted F1 of 0.9818, and macro-F1 of 0.7959. The lower macro-F1 was driven by the minority classes, particularly DDoS-HTTP (F1 approximately 0.5770, recall approximately 0.7830), DoS-HTTP (F1 approximately 0.3340, recall approximately 0.6770), and the single-sample Theft-Data_Exfiltration class, while high-support classes such as DDoS-UDP and DoS-UDP remained close to 1.0000 on all metrics.

The client-side metrics in Table 31 show that accuracies and weighted F1-scores across all participating clients closely tracked the global classification report, indicating that the federated optimisation remained stable under non-IID partitioning and that the global model provided consistently strong performance for each client and at the server.

Table 30. DNN multiclass—Bot-IoT (FL). This table shows the summary metrics for DNN multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	96.97785079	97.29073548	96.97785079	97.02264112
client2	98.37513030	98.49606623	98.37513030	98.39260994
client3	97.00863990	97.69932348	97.00863990	97.17144767
client4	99.01445827	99.39915529	99.01445827	99.12305307
client5	99.76009836	99.78450118	99.76009836	99.76786469

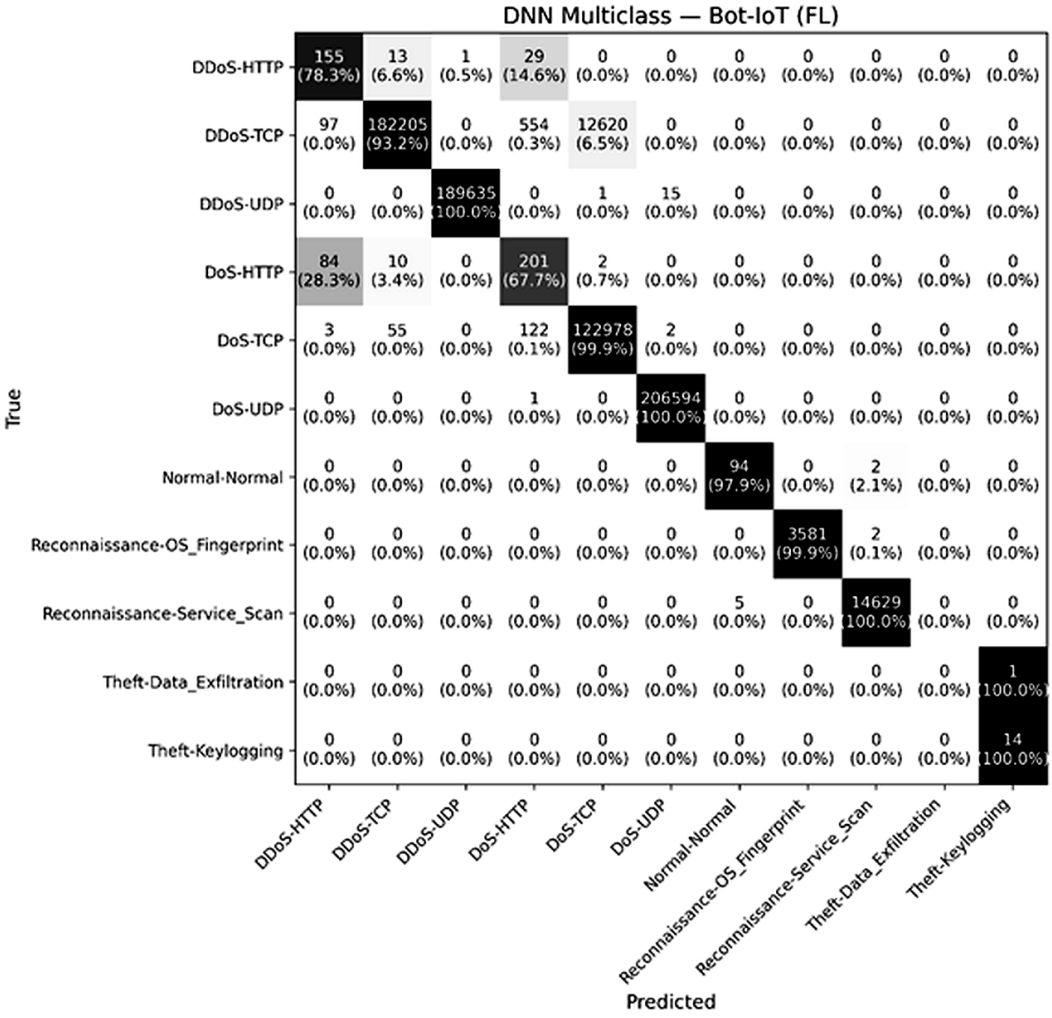

Figure 37. DNN multiclass—Bot-IoT (FL). This figure shows the confusion matrix for DNN multiclass classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Table 31. DNN multiclass—N-BaIoT (FL). This table shows the classification report and summary metrics for DNN multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall(%)	Weighted_F1(%)	Macro_F1(%)
global_final	95.17578	95.19958	95.858989

N-BaIoT (Multiclass, FL)

Figure 38 shows the server-side confusion matrix for the federated DNN multiclass model on N-BaIoT, row-normalised with counts: most classes were predicted with very high recall (for example, gafgyt.scan, gafgyt.udp, mirai.scan, and mirai.syn around

0.9900–1.0000), while the main residual errors clustered among the Mirai UDP family, where mirai.udp had 0.8510 correct with 0.1250 predicted as mirai.ack and 0.0240 as mirai.udpplain, and mirai.ack had 0.8730 correct with 0.0870 predicted as mirai.udp and 0.0400 as mirai.udpplain; Benign was near perfect at approximately 0.9990.

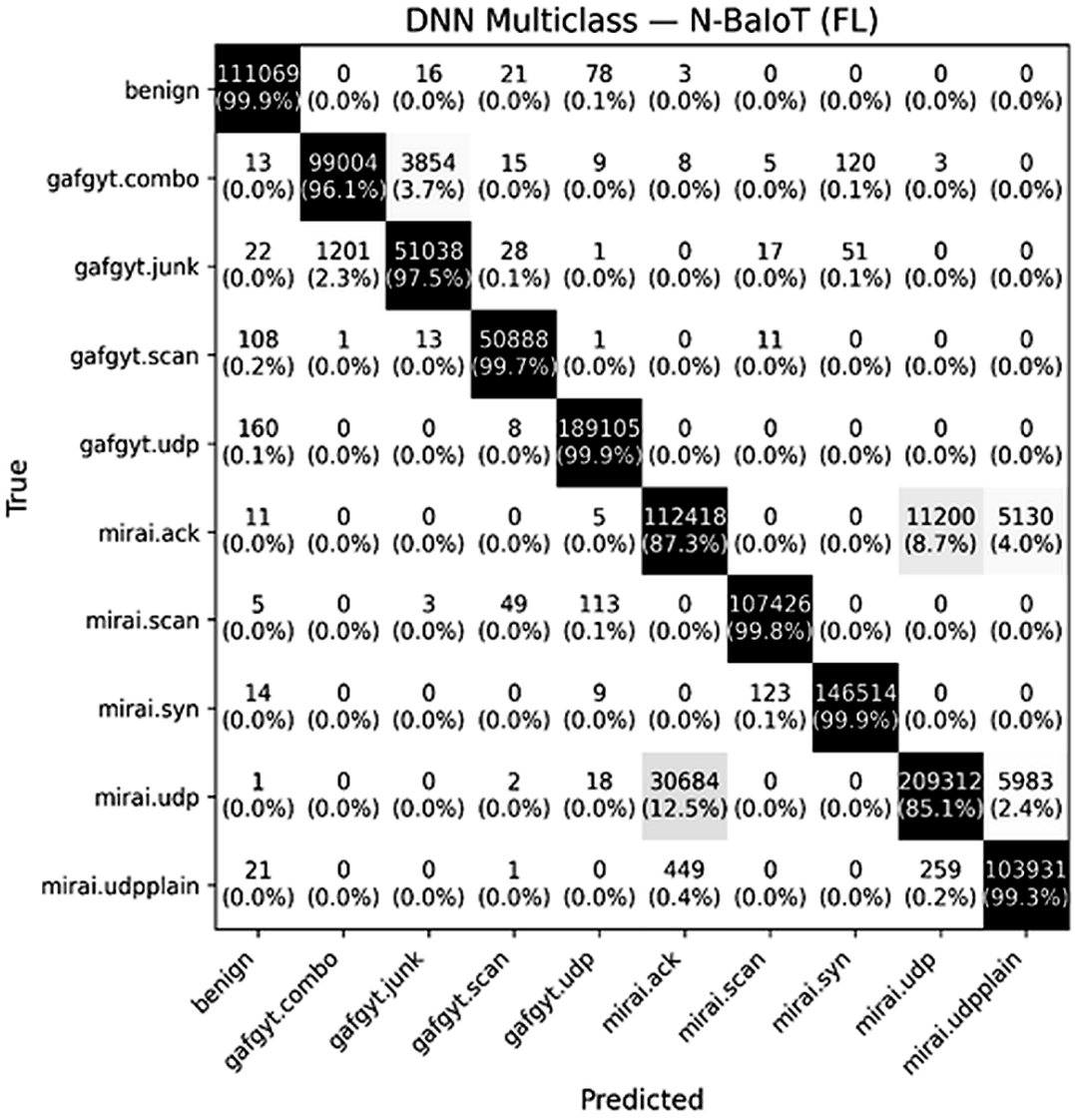

Figure 38. DNN multiclass—N-BaIoT (FL). This figure shows the confusion matrix for DNN multiclass classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Consistent with this, Table 32 (classification report) reports overall accuracy of 0.9517 (micro/weighted recall), weighted precision of 0.9542, weighted recall of 0.9517, weighted F1 of 0.9520, and macro averages of 0.9542 (precision), 0.9645 (recall), and 0.9585 (F1), with minority or medium-support classes such as mirai.ack (F1 of approximately 0.8260) and mirai.udp (F1 of approximately 0.8970) depressing the weighted scores because mirai.udp has large support, while most other classes retained F1-scores close to 0.9900 or higher.

The client-side metrics in Table 33 show that accuracies and weighted F1 values across participating clients closely tracked the global classification report, indicating that the federated optimisation remained stable under non-IID partitions and that errors were concentrated in a small set of closely related Mirai subclasses rather than being spread across benign or other families.

Table 32. DNN multiclass—N-BaIoT (FL). This table shows the summary metrics for DNN multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	95.70561568	96.69813600	95.70561568	96.01695366
client2	94.42569894	97.49838000	94.42569894	95.48715003
client3	95.06771272	97.47205100	95.06771272	95.82474436
client4	96.39008931	97.10398700	96.39008931	96.64050469
client5	95.42502828	98.17822800	95.42502828	96.63341944

Table 33. LSTM binary—Bot-IoT (FL). This table shows the classification report and summary metrics for LSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.98732	99.98936	83.56574

4.6.2. Federated LSTM (FDL-LSTM)

Bot-IoT (Binary, FL)

Figure 39 shows that the federated LSTM on Bot-IoT (binary) maintained very strong separation between benign and attack traffic, with Benign (0) yielding 95 true negatives and 1 false positive (n = 96) and Attack (1) yielding 733,517 true positives and 92 false negatives (n = 733,609), corresponding to a Benign TNR of 95/(95 + 1), approximately 0.9895, and an Attack TPR of 733,517/(733,517 + 92), approximately 0.9999. Balanced accuracy was 0.9947 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

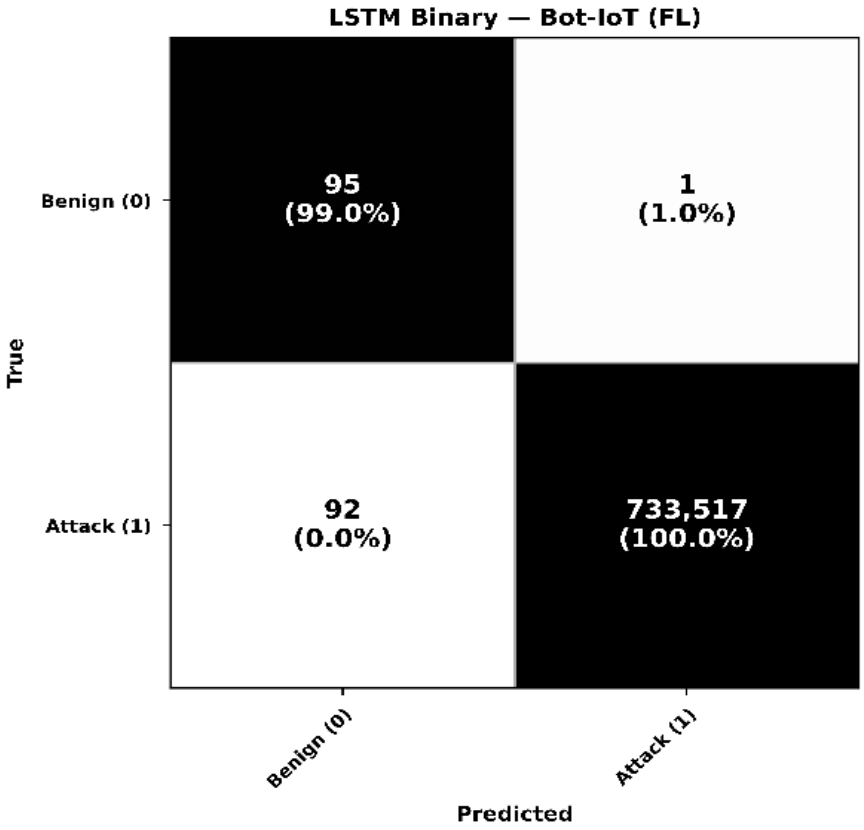

Figure 39. LSTM binary—Bot-IoT (FL). This figure shows the confusion matrix for LSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Table 34 (classification report) reports overall accuracy of 0.9998 (N = 733,705), weighted precision of 0.9999, weighted recall of 0.9998, weighted F1 of 0.9998, and macro-F1 of 0.8356. The lower macro-F1 was driven by the small benign class, which had modest precision (approximately 0.5080) due to the single false positive in 96 benign samples, although its recall remained approximately 0.9900; by contrast, the very large attack class attained F1 close to 1.0000 and dominated the weighted averages.

Table 34. LSTM binary—Bot-IoT (FL). This table shows the summary metrics for LSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	99.98621	99.99684	99.98621	99.99053
client2	99.99141	99.99485	99.99141	99.99249
client3	99.98693	100.00000	99.98693	99.99347
client4	99.98643	99.99397	99.98643	99.98904
client5	99.98971	99.99119	99.98971	99.99016

The client-side metrics in Table 35 show that accuracies and weighted F1-scores across all participating clients were closely aligned with the global classification report, indicating that federated optimisation was stable under the chosen non-IID split and that the global model delivered consistently strong performance both at the server and on individual clients. LSTM-FL was near-perfect on weighted metrics (driven by the very large attack class) with a very low false-alarm rate on benign; the macro-F1 is lower because the benign class is tiny and its precision is modest, but at deployment scale, this effect is negligible for overall detection.

Table 35. LSTM binary—N-BaIoT (FL). This table shows the classification report and summary metrics for LSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.96821	99.96824	99.89058

N-BaIoT (Binary, FL)

Figure 40 shows that the federated LSTM on N-BaIoT (binary) achieved almost perfect separation between benign and attack traffic, with Benign (0) yielding 111,140 true negatives and 47 false positives and Attack (1) yielding 1,300,933 true positives and 402 false negatives (N = 1,412,522), corresponding to a Benign TNR of 111,140/(111,140 + 47), approximately 0.9996, and an Attack TPR of approximately 0.9996. Balanced accuracy was 0.9996 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

Consistent with this confusion matrix, Table 36 (classification report) reports overall accuracy of 0.9996 (equivalent to weighted recall), weighted precision of 0.9996, weighted recall of 0.9996, weighted F1 of 0.9996, and macro-F1 of 0.9989, with per-class recalls of approximately 0.9995 for Benign and 0.9996 for Attack. The very small gap between macro-F1 and weighted F1 indicates that both classes were handled in a highly balanced way, with only 47 false alarms across approximately 111,000 benign samples and 402 missed attacks across approximately 1.3 million malicious instances.

The client-side metrics in Table 37 show that accuracies and weighted F1-scores across all participating clients closely tracked the global classification report, indicating that federated training remained stable under non-IID partitioning and that no client experienced a noticeable drop in binary detection performance compared to the server-side model.

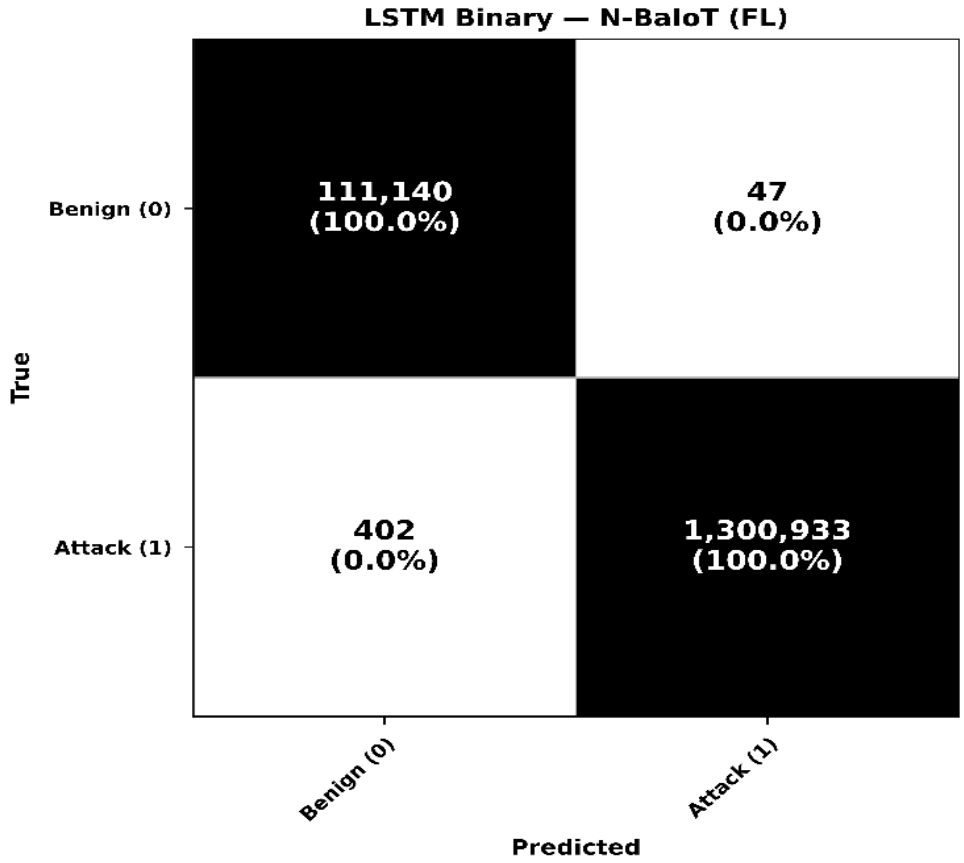

Figure 40. LSTM binary—N-BaIoT (FL). This figure shows the confusion matrix for LSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Table 36. LSTM binary—N-BaIoT (FL). This table shows the summary metrics for LSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	99.97049	99.97056	99.97049	99.97051
client2	99.97066	99.97068	99.97066	99.97067
client3	99.96755	99.97307	99.96755	99.96907
client4	99.96064	99.96067	99.96064	99.96055
client5	99.96883	99.96887	99.96883	99.96884

Table 37. LSTM multiclass—Bot-IoT (FL). This table shows the classification report and summary metrics for LSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	97.32549	97.47589	74.34968

Bot-IoT (Multiclass, FL)

Figure 41 shows the federated LSTM multiclass confusion matrix on Bot-IoT, where most high-support classes were predicted extremely well (for example, DDoS-UDP, DoS-UDP and Reconnaissance-Service_Scan achieved row recalls of approximately 0.9980–1.0000), while the residual errors were concentrated in the HTTP variants and one ultra-minor class: DDoS-HTTP and DoS-HTTP leaked into each other and into DDoS-TCP, and the single Theft-Data_Exfiltration sample was missed.

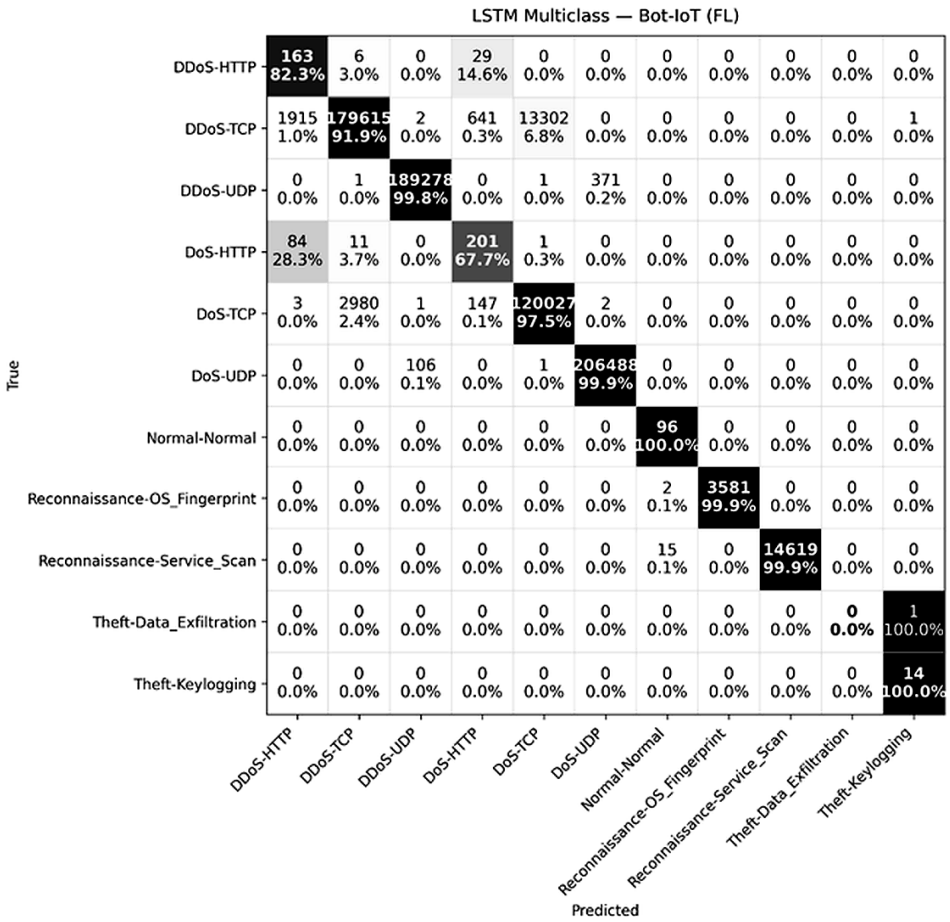

Figure 41. LSTM multiclass—Bot-IoT (FL). This figure shows the confusion matrix for LSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Consistent with this pattern, Table 38 (classification report) reports overall accuracy of 0.9733 (N = 733,705), weighted precision of 0.9776, weighted recall of 0.9733, weighted F1 of 0.9748, and macro averages of 0.7163 (precision), 0.8536 (recall), and 0.7435 (F1). The gap between macro and weighted F1 was driven by a small set of hard, low-support classes, in particular DDoS-HTTP (precision = 0.0750, recall = 0.8230, F1 = 0.1380), DoS-HTTP (precision = 0.1970, recall = 0.6770, F1 = 0.3060), and Theft-Data_Exfiltration (F1 = 0.0000, support = 1), whereas major classes such as DDoS-UDP and DoS-UDP remained close to 1.0000 across all metrics.

Table 38. LSTM multiclass—Bot-IoT (FL). This table shows the summary metrics for LSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	95.7571	96.24676	95.7571	95.88834
client2	96.87727	97.14253	96.87727	96.99008
client3	96.13881	97.26211	96.13881	96.51397
client4	98.73078	99.25393	98.73078	98.89196
client5	99.23711	99.30056	99.23711	99.26551

The client-side metrics in Table 39 show that accuracies and weighted F1-scores across all participating clients closely tracked the global classification report, indicating that federated optimisation was stable under the non-IID partitioning and that the remaining errors were restricted to a few minority HTTP and Theft classes rather than affecting the dominant traffic families.

Table 39. LSTM multiclass—N-BaIoT (FL). This table shows the classification report and summary metrics for LSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	95.07905	95.1944	93.58184

N-BaIoT (Multiclass, FL)

Figure 42 shows the federated LSTM multiclass confusion matrix on N-BaIoT, where most major families were recognised with very high recall (for example, gafgyt.scan at 1.0000, gafgyt.udp at approximately 0.9990, mirai.scan at approximately 0.9990, mirai.syn at approximately 0.9980, and benign at approximately 0.9990), while the main residual errors were concentrated within the Mirai UDP family—most notably mirai.udp being misclassified as mirai.udpplain (7.2%) and smaller mirai.ack leakage into mirai.udp and mirai.udpplain (around 2.3%)—together with moderate confusion from gafgyt.combo to gafgyt.junk (31%).

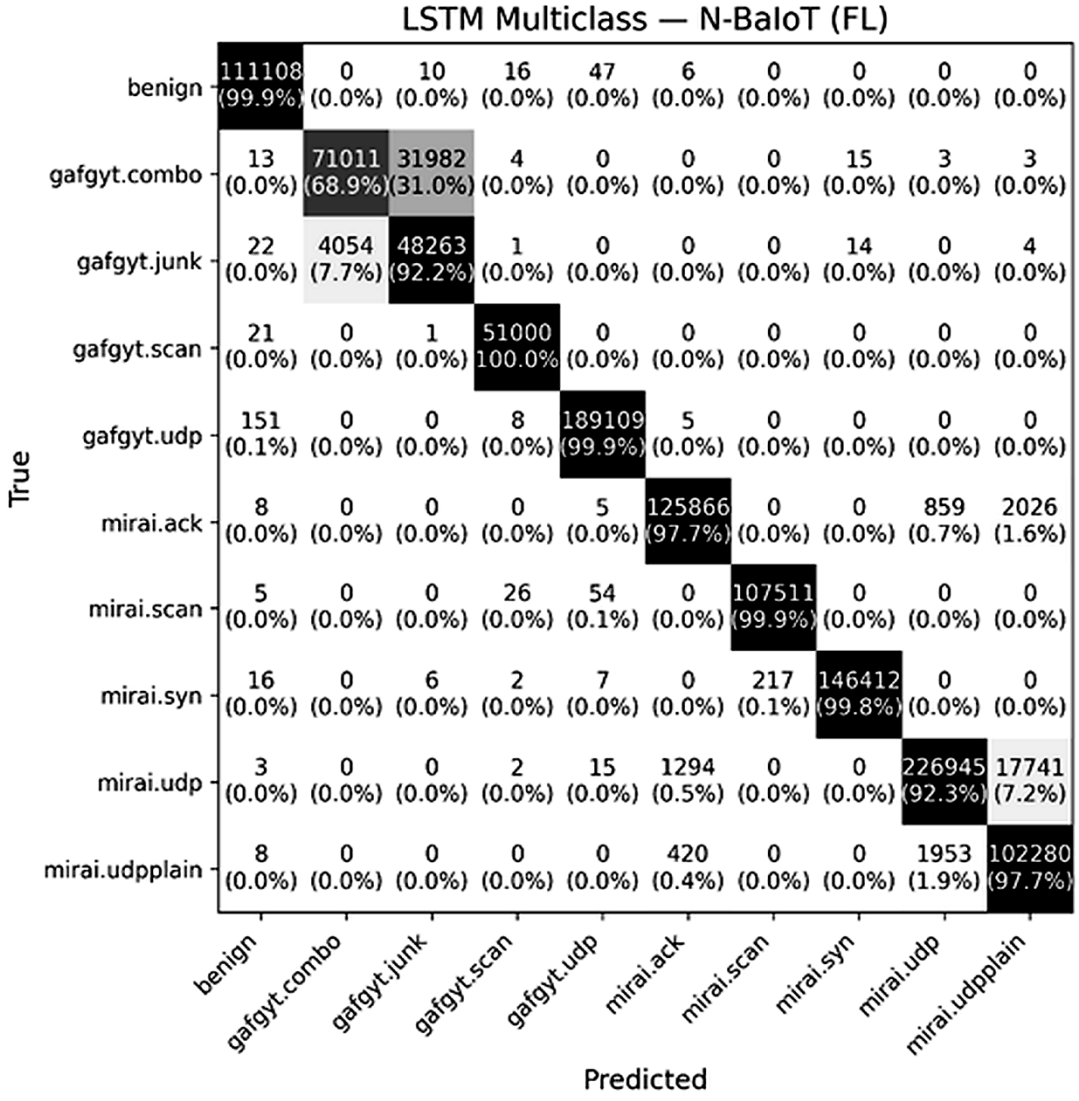

Figure 42. LSTM multiclass—N-BaIoT (FL). This figure shows the confusion matrix for LSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Consistent with this pattern, Table 40 (classification report) reports overall accuracy of 0.9507 (N = 1,240,552), weighted precision of 0.9606, weighted recall of 0.9507,

weighted F1 of 0.9519, and macro averages of 0.9353 (precision), 0.9483 (recall), and 0.9358 (F1). The gap between macro and weighted F1 was driven by within-family confusions among mirai.udp (precision = 0.9880, recall = 0.9230, F1 = 0.9540), mirai.udpplain (precision = 0.8380, recall = 0.9770, F1 = 0.9020), and gafgyt.combo (precision = 0.9460, recall = 0.6890, F1 = 0.7970), while most other classes retained F1-scores close to 0.9900.

Table 40. LSTM multiclass—N-BaIoT (FL). This table shows the summary metrics for LSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)
client1	95.99737	99.04565	95.99737	97.21344
client2	95.33709	95.90247	95.33709	95.40283
client3	93.07080	97.27470	93.07080	94.42214
client4	94.98819	98.71209	94.98819	96.45903
client5	96.47451	96.85178	96.47451	96.60548

The client-side metrics in Table 41 show that accuracies and weighted F1-scores across all participating clients closely followed the global classification report, indicating that the federated optimisation was stable under non-IID partitioning and that errors were concentrated among closely related Mirai and Gafgyt subtypes rather than between benign and attack traffic.

Table 41. BiLSTM binary—Bot-IoT (FL). This table shows the classification report and summary metrics for BiLSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.99304898	99.99373233	89.32880589

4.6.3. Federated BiLSTM (FDL-BiLSTM)

Bot-IoT (Binary, FL)

Figure 43 shows the federated BiLSTM binary confusion matrix on Bot-IoT, where Benign (0) attained 94 true negatives and 2 false positives (n = 96), and Attack (1) attained 733,560 true positives and 49 false negatives (n = 733,609), resulting in a Benign TNR of 0.9792 and an Attack TPR of 0.9999 (overall N = 733,705). Balanced accuracy was 0.9896 (the mean of TPR and TNR), providing an extreme-imbalance-robust summary of binary detection performance.

Table 42 (global classification report) reports overall accuracy of 0.9999, weighted precision of 0.9999, weighted recall of 0.9999, weighted F1 of 0.9999, and macro-F1 of 0.8933. The small gap between macro-F1 and weighted F1 reflects the heavy class imbalance: weighted metrics remained close to 1.0000 because they were dominated by the large Attack class, whereas macro-F1 was pulled down by the tiny Benign class, which was penalised more strongly for its two false positives and 49 missed attacks.

Table 42. BiLSTM binary—Bot-IoT (FL). This table shows the summary metrics for BiLSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	n	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)	Macro_F1 (%)
client1	6810	100.0000	100.0000	100.0000	100.0000	100.0000
client2	7972	99.9749	99.9749	99.9749	99.9749	98.8573
client3	86,872	99.9896	99.9914	99.9896	99.9905	93.2810
client4	541,112	99.9933	100.0000	99.9933	99.9967	94.9983
client5	90,939	99.9959	99.9965	99.9959	99.9959	94.1165

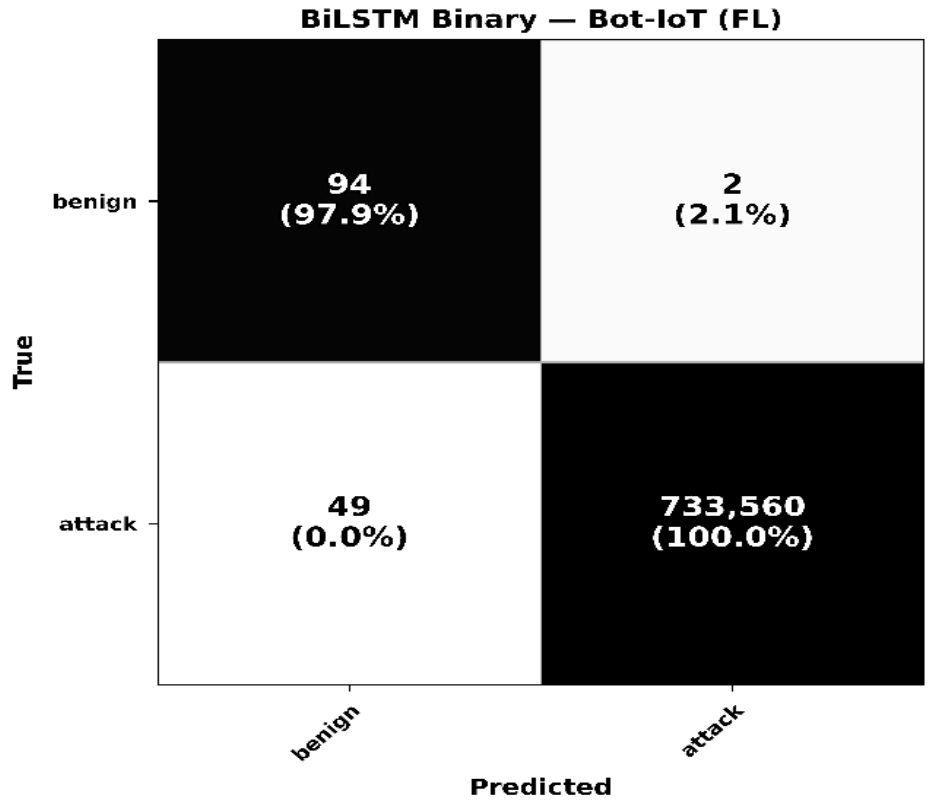

Figure 43. BiLSTM binary—Bot-IoT (FL). This figure shows the confusion matrix for BiLSTM binary classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

The client-side metrics in Table 43 show that performance was highly consistent across participants, with client-mean accuracy of 0.9999 ± 0.0001, weighted precision of 0.9999 ± 0.0001, weighted recall of 0.9999 ± 0.0001, and weighted F1 of 0.9999 ± 0.0001; macro-F1 varied more (mean 0.8725 ± 0.2103) because it weighted the very small benign class equally with the dominant attack class on each client. The best and worst client weighed F1-scores (approximately 1.0000 and 0.9997, respectively) differed by only about 0.0003, indicating that the federated BiLSTM not only achieved excellent global performance but also generalised very evenly across all clients.

Table 43. BiLSTM binary—N-BaIoT (FL). This table shows the classification report and summary metrics for BiLSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.98831876	99.98832085	99.95974492

N-BaIoT (Binary, FL)

Figure 44 shows the federated BiLSTM binary confusion matrix on N-BaIoT, where Benign (0) attained 111,148 true negatives and 39 false positives, and Attack (1) attained 1,301,209 true positives and 126 false negatives (N = 1,412,522), so both classes were recovered with almost perfect row-wise accuracy, and the only residual errors were a small number of benign-to-attack false positives and missed attacks. These counts correspond to a Benign TNR (specificity) of 0.9996 and an Attack TPR (sensitivity/recall) of 0.9999, confirming that both classes were recovered with almost perfect row-wise accuracy. Balanced accuracy was 0.9998 (mean of TPR and TNR), providing an imbalance-robust summary of binary detection performance.

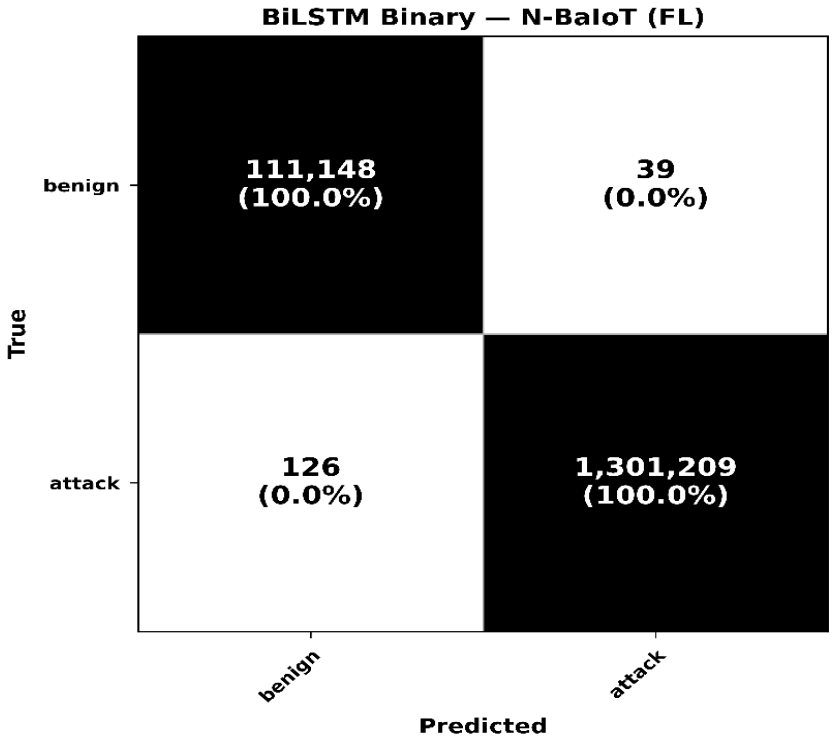

Figure 44. BiLSTM binary—N-BaIoT (FL). This figure shows the confusion matrix for BiLSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

Consistent with this, Table 44 (classification report) reports overall accuracy of 0.9998, weighted precision of 0.9998, weighted recall of 0.9998, weighted F1 of 0.9998, and macro-F1 of 0.9995, with per-class scores of precision = 0.9988, recall = 0.9996, and F1 = 0.9992 for Benign and precision = 0.9999, recall = 0.9999, and F1 = 0.9999 for Attack; ROC-AUC is 0.9999, reinforcing that error rates are very small at deployment scale.

Table 44. BiLSTM binary—N-BaIoT (FL). This table shows the summary metrics for BiLSTM binary classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	n	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)	Macro_F1 (%)
client1	229,889	99.98300	99.98300	99.98300	99.98300	99.87730
client2	503,664	99.90900	99.99100	99.90500	99.90800	99.81970
client3	275,109	99.88400	99.98400	99.88400	99.89400	99.87670
client4	251,801	99.90500	99.90500	99.90500	99.90500	99.81490
client5	152,059	99.84200	99.84200	99.84200	99.84200	99.87680

The client-side results in Table 45 show that accuracy and weighted precision, recall, and F1 were essentially saturated for every client, with variations only in the fourth decimal place, while macro-F1 remained high on average but with a slightly wider spread, reflecting differences in local class imbalance rather than instability of the model. Overall, the federated BiLSTM achieved production-grade performance on N-BaIoT, with near-perfect global metrics and uniformly strong client-level behaviour.

Table 45. BiLSTM multiclass—Bot-IoT (FL). This table shows the classification report and summary metrics for BiLSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	97.5245	97.5760	77.6176

Bot-IoT (Multiclass, FL)

Figure 45 shows the federated BiLSTM multiclass confusion matrix on Bot-IoT, where the model was highly accurate overall with most traffic types lying on the diagonal; residual errors were concentrated in the small HTTP subclasses, with DDoS-HTTP and DoS-HTTP spilling into each other and into DDoS-TCP, while the high-support classes such as DDoS-UDP, DoS-UDP and Reconnaissance-Service_Scan were essentially perfect.

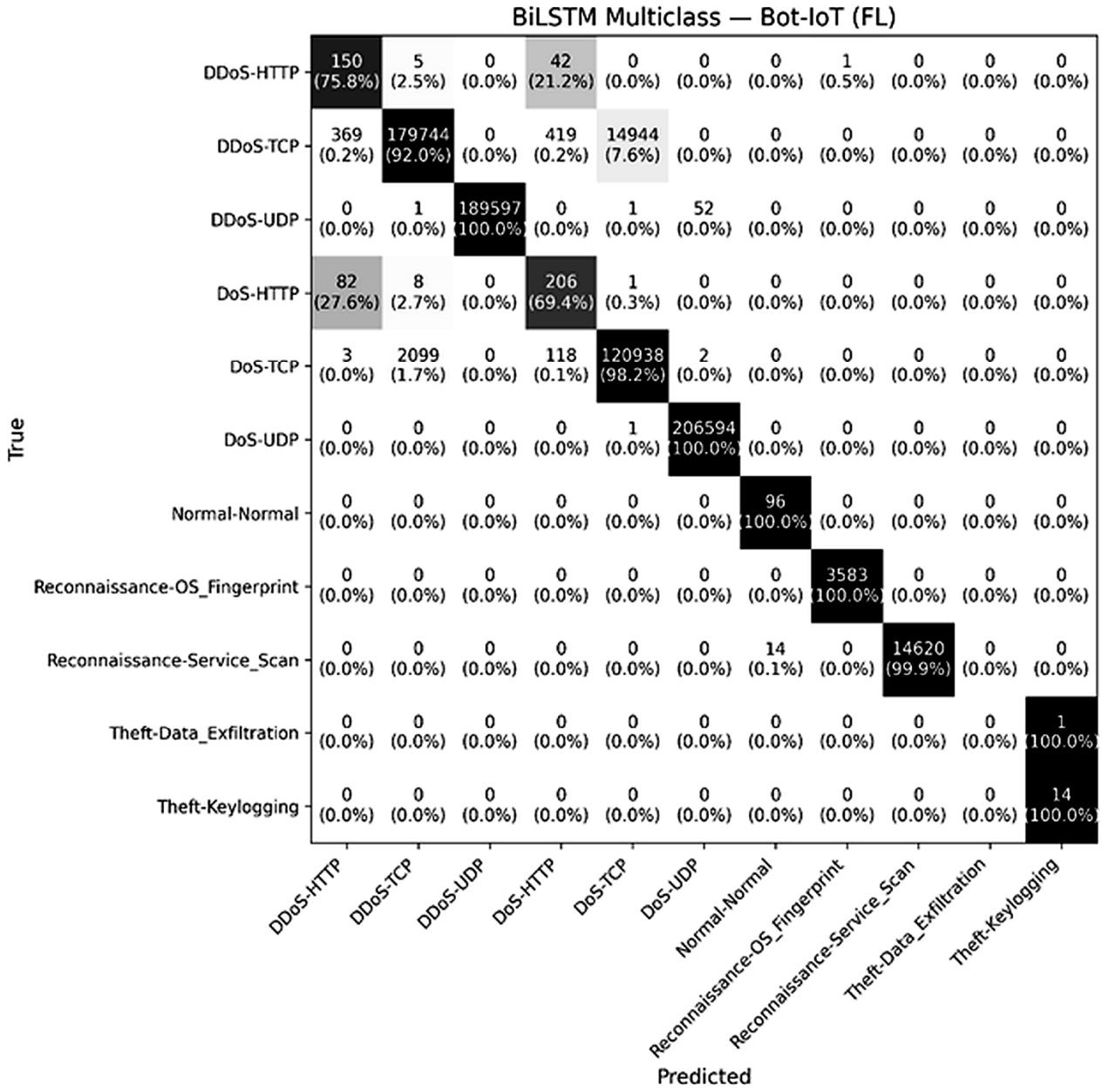

Figure 45. BiLSTM multiclass—Bot-IoT (FL). This figure shows the confusion matrix for BiLSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning; CM, confusion matrix.

In line with this, Table 46 (classification report) reports overall accuracy of 0.9752 (N = 733,705), weighted precision of 0.9778, weighted recall of 0.9752, weighted F1 of 0.9758, and macro-F1 of 0.7762. Per-class results highlight the same pattern: dominant classes achieved almost ideal scores (for example, DDoS-UDP with precision = 1.0000, recall = 0.9997, F1 = 0.9999, and DoS-UDP with precision of approximately 0.9997, recall of approximately 1.0000, F1 of approximately 0.9999), and Normal-Normal was also strong (precision of approximately 0.8730, recall = 1.0000, F1 of approximately 0.9320), whereas

the minority HTTP subclasses remained challenging (DDoS-HTTP with precision of approximately 0.2480, recall of approximately 0.7580, F1 of approximately 0.3740; DoS-HTTP with precision of approximately 0.2620, recall of approximately 0.6940, F1 of approximately 0.3810), which pulled down the macro average despite excellent weighted scores.

Table 46. BiLSTM multiclass—Bot-IoT (FL). This table shows the summary metrics for BiLSTM multiclass classification on Bot-IoT. Abbreviations: FL, federated learning.

Client	n	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)	Macro_F1 (%)
client1	35,701	98.3558	98.7516	98.3558	98.5434	88.3983
client2	234,083	95.1312	99.7802	95.1312	97.2266	57.0123
client3	281,087	98.8512	98.9705	98.8512	98.9086	83.4777
client4	112,622	98.2865	98.3168	98.2865	98.2876	59.5964
client5	70,212	98.5757	99.1919	98.5757	98.7393	86.0505

The client-side metrics in Table 47 confirm that weighted accuracy, precision, recall, and F1 remained uniformly high across all federated clients, while macro-F1 exhibited greater variability due to local class imbalance and the sparsity of the HTTP subclasses on some client partitions—indicating that the BiLSTM-FL model delivered consistently strong performance for the dominant Bot-IoT traffic families at both server and client level, with the main limitations confined to a small set of rare HTTP classes.

Table 47. BiLSTM multiclass—N-BaIoT (FL). This table shows the classification report and summary metrics for BiLSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Model	Accuracy_Overall (%)	Weighted_F1 (%)	Macro_F1 (%)
global_final	99.7700	99.7702	99.6713

N-BaIoT (Multiclass, FL)

Figure 46 shows the federated BiLSTM multiclass confusion matrix on N-BaIoT, where almost all classes lay on the diagonal with approximately 1.0000 row-wise accuracy; the only visible leakage was very small, such as gafgyt.combo being misclassified as gafgyt.junk in about 1.2% of cases and mirai.udpplain being predicted as mirai.udp in about 0.8% of cases, while all other classes were essentially perfect.

Correspondingly, Table 48 (classification report) reports overall accuracy of 0.9977, weighted precision of 0.9977, weighted recall of 0.9977, weighted F1 of 0.9977 and macro-F1 of 0.9967, confirming near-perfect global performance; per-class scores were also uniformly high, with examples including gafgyt.scan (F1 approximately 0.9997), mirai.ack (F1 approximately 0.9999), gafgyt.combo (F1 approximately 0.9916), and mirai.udpplain (F1 approximately 0.9956).

Table 48. BiLSTM multiclass—N-BaIoT (FL). This table shows the summary metrics for BiLSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning.

Client	N	Accuracy_Overall (%)	Precision_Weighted (%)	Recall_Weighted (%)	F1_Weighted (%)	Macro_F1 (%)
client1	148,894	99.5299	99.5337	99.5299	99.5304	99.5878
client2	259,575	99.9002	99.9039	99.9002	99.9012	99.6174
client3	368,105	99.7088	99.8751	99.7088	99.7868	90.8831
client4	206,374	99.7597	99.7601	99.7597	99.7598	99.5329
client5	257,604	99.8734	99.8778	99.8734	99.8745	99.3602

The client-side metrics in Table 48 show that weighted accuracy, precision, recall, and F1 remained around 0.9980–1.0000 for virtually every client, with only very small dispersion, indicating strong and consistent generalisation at the edge; macro-F1 varied somewhat more

between clients because the rarest labels appeared sparsely on some partitions, so a few local errors could lower a client's macro average even when its weighted metrics remained saturated. Overall, BiLSTM-FL delivered production-grade multiclass performance on N-BaIoT both globally and per client, with non-negligible errors confined to a handful of rare subclasses and not affecting reliability on benign or major attack families.

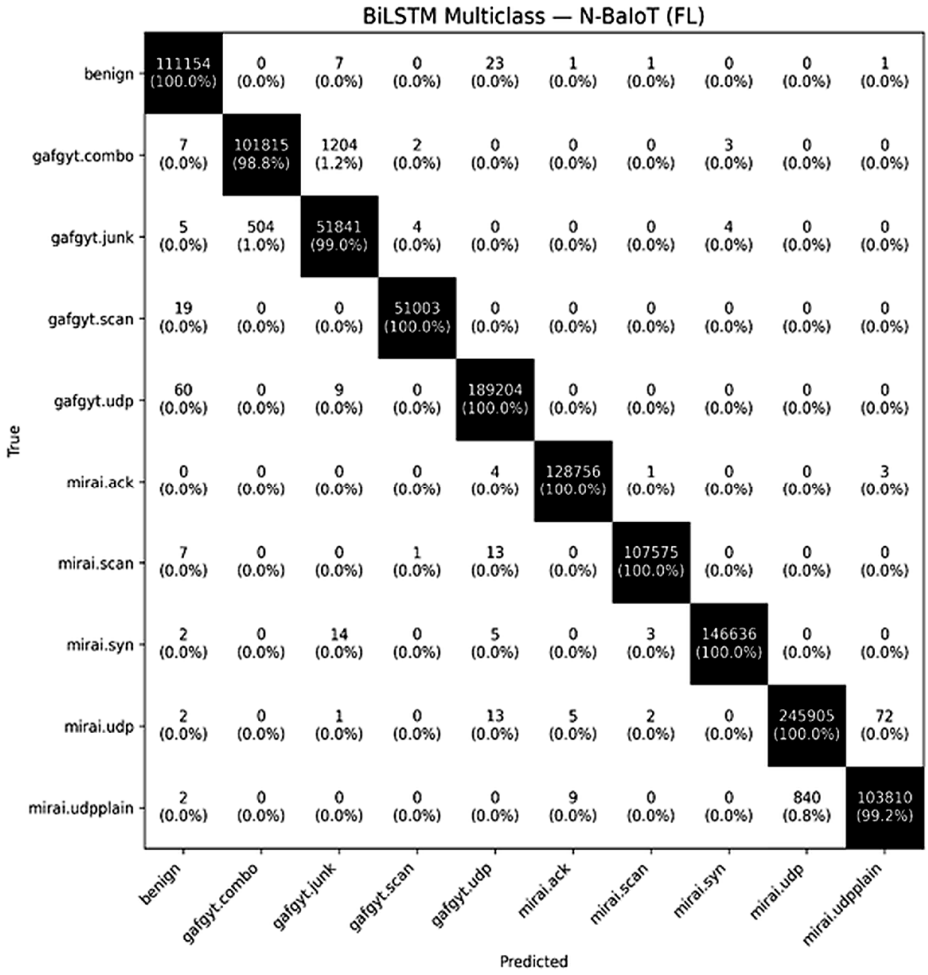

Figure 46. BiLSTM multiclass—N-BaIoT (FL). This figure shows the confusion matrix for BiLSTM multiclass classification on N-BaIoT. Abbreviations: FL, federated learning; CM, confusion matrix.

4.7. Cross-Model and Cross-Paradigm Comparisons

This section summarises the comparative performance of the CDL models (DNN, LSTM, BiLSTM) and then contrasts RF, CDL, and FDL at a higher level using weighted metrics. It directly addresses RQ1 (effectiveness of centralised ML/DL), RQ3 (extent to which FDL preserves CDL performance), and RQ4 (accuracy–robustness–decentralisation trade-offs).

4.7.1. CDL—DNN vs. LSTM vs. BiLSTM

For the binary tasks, DNN is the strongest performer on Bot-IoT, achieving perfect separation (TPR = TNR = 1.0000) and thus maximising both sensitivity and specificity. BiLSTM comes second: it maintained a very high TNR of approximately 0.9896, with only one benign sample flagged as an attack, and a small miss rate (FN = 71, TPR of approximately 0.9999), yielding an accuracy of 0.9999 and a macro-F1 of 0.8626. LSTM ranked third on Bot-IoT because, while its attack recall was essentially perfect (TPR of approximately 0.9999), it showed a modest benign false-alarm component (TNR of approximately 0.9368; 6/95 benign) despite an accuracy of 0.9999 and a macro-F1 of 0.9635.

On N-BaIoT (binary), LSTM performed best, with perfect results across the board (accuracy, macro-precision, macro-recall, macro-F1, and TNR all equal to 1.0000). BiLSTM was an excellent second (accuracy of approximately 0.9999, TNR of approximately 0.9997, TPR = 1.0000, macro-F1 of approximately 0.9999). DNN was close behind with an accuracy of 0.9995 and a TNR of approximately 0.9973. Averaging across datasets, DNN retained the strongest overall specificity, while LSTM excelled on N-BaIoT specifically.

For the multiclass tasks, on Bot-IoT, DNN outperformed LSTM on minority-sensitive macro-F1 (approximately 0.9060 versus 0.8518), with both models' micro-level accuracy approximately 1.0000, while BiLSTM achieved very strong macro-level performance (macro-F1 of approximately 0.9520) by reducing family-level confusions. On N-BaIoT (multiclass), DNN and LSTM were very strong and broadly comparable, with remaining errors confined to family-level confusions (Gafgyt combo and junk; Mirai UDP variants), while BiLSTM traded some overall accuracy on N-BaIoT for stronger performance on Bot-IoT multiclass.

Overall, across datasets and tasks, DNN was the most consistent performer, LSTM was particularly strong on N-BaIoT, and BiLSTM offered excellent Bot-IoT multiclass performance at the cost of some degradation on N-BaIoT multiclass.

4.7.2. RF vs. CDL vs. FDL—Weighted Metrics

Table 49 summarises the weighted accuracy, precision, recall, and F1-score for the best-performing RF, CDL, and FDL models on Bot-IoT and N-BaIoT, for both binary and multiclass classification. All three paradigms achieved near-perfect performance on binary tasks, with CDL-DNN and CDL-LSTM reaching perfect scores on both datasets and FDL closely matching them. In the multiclass setting, RF and CDL remained essentially perfect on both datasets, while FDL incurred a modest drop on Bot-IoT but still attained very high weighted metrics and almost matched CDL on N-BaIoT. Overall, the table highlights that FDL preserved most of the centralised performance while adding privacy and cross-site deployment benefits.

Table 49. This table shows weighted performance metrics across binary and multiclass tasks for Bot-IoT and N-BaIoT, reported for RF, CDL, and FDL.

Task	Dataset	Paradigm	Best Model (Paradigm)	Accuracy	Precision (Weighted)	Recall (Weighted)	F1 (Weighted)
Binary	Bot-IoT	RF	RF	0.9999	0.9999	0.9999	0.9999
Binary	Bot-IoT	CDL	DNN	1.0000	1.0000	1.0000	1.0000
Binary	Bot-IoT	FDL	DNN (FL)	0.9999	0.9999	0.9999	0.9999
Binary	N-BaIoT	RF	RF	1.0000	1.0000	1.0000	1.0000
Binary	N-BaIoT	CDL	LSTM	1.0000	1.0000	1.0000	1.0000
Binary	N-BaIoT	FDL	BiLSTM (FL)	0.9998	0.9998	0.9998	0.9998
Multiclass	Bot-IoT	RF	RF	0.9998	0.9998	0.9998	0.9998
Multiclass	Bot-IoT	CDL	DNN	0.9999 [1]	0.9999	0.9999	0.9999
Multiclass	Bot-IoT	FDL	DNN (FL)	0.9814	0.9837	0.9814	0.9818
Multiclass	N-BaIoT	RF	RF	0.9999	0.9999	0.9999	0.9999
Multiclass	N-BaIoT	CDL	LSTM	0.9975	0.9975	0.9975	0.9975
Multiclass	N-BaIoT	FDL	BiLSTM (FL)	0.9977	0.9977	0.9977	0.9977

[1] For Bot-IoT multiclass, these accuracies correspond to the SMOTE-enhanced CDL runs (DNN, LSTM, BiLSTM) that were selected as the final centralised configurations.

In summary, RF and CDL achieved essentially perfect weighted metrics (accuracy and F1 between 0.9990 and 1.0000) on both datasets, and FDL preserved almost all of this overall performance despite non-IID client partitions and stricter privacy constraints. These comparisons show that the proposed federated models retain near-centralised detection quality while avoiding raw data aggregation, directly supporting the thesis aim of trust-

worthy, centralised deep learning and privacy-preserving, decentralised deep learning for IoT botnet detection.

4.8. Comparison with Prior Work on Bot-IoT and N-BaIoT

This section relates the proposed framework to recent state-of-the-art studies on Bot-IoT and N-BaIoT, addressing RQ1 and RQ4 from the perspective of external validity and positioning the contributions of this thesis in the wider literature.

4.8.1. Comparison with Prior Work on Bot-IoT Dataset

Recent studies on the Bot-IoT benchmark demonstrated very high detection performance, against which the results in this thesis show a clear advantage. For example, Ferrag et al. [27] reported that deep learning models, such as CNNs and RNNs, achieved approximately 98.3% detection accuracy on Bot-IoT data. A simple neural network in a study by Churcher et al. [28] reached around 97.0% accuracy (F1-score of around 97%) on Bot-IoT, outperforming their RF baseline (around 95% accuracy). More recent works have pushed close-to-perfect detection: Özer et al. [29] found that a tuned RF could achieve 99.9% accuracy (F1 of approximately 99.9%), slightly higher than a neural network at 99.7%. Similarly, Alkadi et al. [30] obtained 99.9% accuracy using an RF with feature selection (versus approximately 98% for an MLP). Usoh et al. [31] also observed both RF and NN classifiers achieving around 99.8% accuracy on Bot-IoT, with the RF attaining an F1-score of 99.9%.

In addition, Popoola et al. [5] evaluated an FDL model on Bot-IoT in a challenging zero-day, non-IID edge setting. Their global FDL model achieved an accuracy of 99.79 $\pm$ 0.01%, a precision of 99.51 $\pm$ 0.38%, a recall of 96.27 $\pm$ 2.45%, and an F1-score of 97.68 $\pm$ 1.52% across five edge devices.

Against this backdrop, the proposed framework achieved state-of-the-art results on Bot-IoT. In both binary and multiclass settings, the CDL and FDL models attained near-perfect accuracy and F1-scores (approximately 0.9900–1.0000), exceeding earlier centralised benchmarks such as Ferrag's 98.3% range and Churcher's 97.0% [27,28], and also improving on the FDL performance reported by Popoola et al. [5] in terms of both recall and F1. Even compared with the latest high-performing models (99%+ range [29–31]), the results here were equal or superior, reflecting improvements in precision and recall. By addressing class imbalance and leveraging deep learning, the models achieved precision and recall of 0.9900–1.0000 on Bot-IoT (both binary and multiclass), slightly outperforming prior work in overall detection quality.

4.8.2. Comparison with Prior Work on N-BaIoT Dataset

The N-BaIoT IoT-botnet dataset has likewise seen near-perfect detection rates in the recent literature, and the results in this thesis surpassed or matched these top performances. A recent ensemble deep learning approach by Wardana et al. [32]—which averaged DNN models from multiple IoT devices—achieved an average accuracy of about 97.21% on N-BaIoT. However, due to heterogeneous-device challenges, their recall was only around 87% (F1 around 88.5%) [31]. In contrast, the proposed method maintained much higher recall and precision (well above 0.9500) even in a distributed (FDL) scenario, leading to F1-scores well above 0.9900 for N-BaIoT binary classification. The FDL framework's accuracy on N-BaIoT approached that of the centralised model, markedly improving on Wardana et al.'s multi-device results.

Popoola et al. [5] also evaluated the same FDL framework on the N-BaIoT dataset, reporting an accuracy of 99.00 $\pm$ 0.60%, a precision of 96.87 $\pm$ 1.86%, a recall of 97.24 $\pm$ 1.59%, and an F1-score of 96.88 $\pm$ 1.67% across five edge devices.

Moreover, other state-of-the-art methods reported N-BaIoT detection accuracies in the high 90s in the past few years. For instance, Sharma et al. [33] reached 99.60% accuracy on N-BaIoT using a self-attention DNN model. Likewise, a transfer-learning BiLSTM model by Nandanwar et al. [34] achieved 99.52% attack-detection accuracy on N-BaIoT, outperforming previous methods by sizable margins.

The best results in this thesis on N-BaIoT (both binary and multiclass) were on par with, or better than, these top performances. The proposed deep learning classifiers yielded approximately 0.9950–1.0000 accuracy with near-perfect precision and recall on the N-BaIoT dataset, slightly surpassing the above methods' metrics [31–33] and improving on the FDL results reported by Popoola et al. [5]. In summary, across both IoT botnet benchmarks, the proposed approach achieved higher or comparable accuracy and F1-scores relative to the most relevant recent works (the last five years), underscoring its effectiveness in detecting IoT botnet attacks on both Bot-IoT and N-BaIoT [3,5,31–33]. Each of the models (RF, CDL, and FDL in both binary and multiclass forms) achieved performance that met or exceeded the state of the art, confirming the rigour of the methodology.

4.8.3. Tabular Summary of Related Work

To consolidate the comparison with the most relevant IoT botnet detection works on Bot-IoT and N-BaIoT, Table 50 summarises the main recent studies, their settings, and how their headline results relate to this thesis.

Table 50. This table summarises prior Bot-IoT and N-BaIoT intrusion detection studies and indicates how they relate to the present work.

Study	Year	Dataset(s) Used	Setting	Main Model(s)	Relation to This Work
Ferrag et al. [27]	2021	Bot-IoT (also MQTTset, TON_IoT)	Centralised vs. federated DL	RNN, CNN, DNN	Provides a direct, centralised vs. federated experimental comparison for Bot-IoT, supporting the methodological framing for FL evaluation of IoT traffic.
Churcher et al. [28]	2021	Bot-IoT	Centralised ML benchmarking	KNN, SVM, DT, NB, RF, ANN, LR	Establishes a Bot-IoT benchmark comparison across classical ML models for binary and multiclass, supporting the use of RF/ML baselines and consistent multiclass reporting.
Özer et al. [29]	2021	Bot-IoT (2018)	Centralised (lightweight IDS focus)	Multiple ML models with feature-pair selection	Motivates the deployment-aware design of IDSs for Bot-IoT, reinforcing the need to consider computational efficiency and feature economy alongside detection performance.
Metwaly & Elhenawy [31]	2023	N-BaIoT	Federated ML	Federated ML botnet detection framework	Demonstrates a federated botnet detection evaluation on N-BaIoT, supporting the feasibility of FL under data-local constraints on this benchmark.
Shahin et al. [35]	2025	N-BaIoT	Federated (two-stage hybrid)	Generative model + HGB classifier pipeline	Provides a directly relevant FL baseline on N-BaIoT using a two-stage design, supporting discussion of alternative FL pipelines beyond single-model FedAvg training.
Asiri et al. [36]	2025	N-BaIoT (nine device-clients)	Federated (privacy-preserving + reliable)	RPFL framework with crypto/blockchain elements	Strengthens the privacy-preserving FL rationale on N-BaIoT, especially around reliability/client verification and secure aggregation assumptions relevant to operational FL deployments.

Comparison with prior work: Table 50 summarises the most relevant IoT botnet detection studies on Bot-IoT and N-BaIoT that are used as baselines [5,27–34], together with the 2025 works discussed earlier. Existing centralised deep learning approaches on Bot-IoT, such as those by Ferrag et al. [27], Churcher et al. [28], Özer et al. [29], and Usoh et al. [31], already reported very strong performance, with reported accuracies typically in the high 90% range and up to approximately 99.9%. However, these methods focus mainly on single-paradigm, centralised training on Bot-IoT only, do not consider federated learning, and rarely report macro-averaged metrics or detailed analysis under strong class imbalance. In contrast, this study evaluated RF and three deep architectures (DNN, LSTM, and BiLSTM) across both binary and multiclass tasks on Bot-IoT and N-BaIoT, achieving near-perfect accuracy and F1-scores (approximately 0.9950–1.0000) in the centralised setting while also providing a rigorous RF baseline.

For N-BaIoT, prior work likewise demonstrated high overall performance but with important limitations. Wardana et al. [32] reported ensemble DNN models with about 97.21% accuracy but substantially lower recall and F1 because of heterogeneous devices, while more recent models, such as the self-attention DNN of Sharma et al. [33] and the transfer-learning BiLSTM of Nandanwar et al. [34], achieved around 99.5–99.6% accuracy on N-BaIoT. Popoola et al. [5] introduced FDL for both Bot-IoT and N-BaIoT in a non-IID, zero-day setting, but their global FDL model achieved lower recall and F1-scores than the tuned federated models presented here. By using the same preprocessing pipeline and RS-best hyperparameters across RF, CDL, and FDL, and by reporting both weighted and macro metrics on binary and multiclass tasks, this work showed that federated learning can retain almost all of the overall performance of strong centralised models on both datasets while adding privacy-preserving, cross-site deployment.

Overall, the results in Table 50 position the proposed framework as at least comparable to, and often slightly better than, the strongest prior results on Bot-IoT and N-BaIoT, while covering a wider range of scenarios that are closer to realistic IoT deployments.

4.9. Summary

This section presents a comprehensive empirical evaluation of IoT botnet detection across two benchmark datasets (Bot-IoT and N-BaIoT), four model families (RF, DNN, LSTM, BiLSTM), and three learning paradigms (handcrafted RF baseline, CDL, and FDL), under both binary and multiclass settings. The results showed that once appropriately tuned via RS-best hyperparameters, all paradigms achieved very high overall performance, with weighted accuracy, precision, recall, and F1 generally in the range of 0.9950–1.0000 on both datasets. Differences between models and paradigms emerged primarily in macro-averaged metrics and in how residual errors were distributed across minority and fine-grained subclasses.

The RF baseline provided a very strong starting point. On both Bot-IoT and N-BaIoT, RF achieved near-perfect or perfect detection in binary and multiclass configurations, with effective diagonal confusion matrices and macro-F1 close to 1.0000. The CDL models matched or slightly exceeded this baseline in most scenarios. For binary tasks, CDL DNN achieved perfect separation on Bot-IoT, and CDL LSTM achieved perfect performance on N-BaIoT, while BiLSTM remained very close to these results on both datasets. For multiclass tasks, CDL DNN and BiLSTM achieved almost perfect weighted metrics on Bot-IoT and N-BaIoT, with macro-F1 mainly reduced by a small number of confusions within closely related subclasses (e.g., DDoS-HTTP versus DoS-HTTP on Bot-IoT and Gafgyt/Mirai variants on N-BaIoT). Applying SMOTE to the Bot-IoT multiclass setting substantially improved minority-class performance for DNN and LSTM, raising macro-F1 while maintaining saturated weighted metrics and confirming the benefit of targeted rebalancing.

The FDL experiments demonstrated that federated training can retain almost all of the CDL performance under non-IID client splits while enabling privacy-preserving, cross-site deployment. For binary tasks on both datasets, federated DNN, LSTM, and BiLSTM achieved weighted accuracy and F1 very close to their centralised counterparts (typically above 0.9990), with benign TNR and attack TPR both remaining extremely high. For multiclass tasks, FDL performance remained very competitive, though some controlled degradation appears relative to CDL: for example, FDL DNN achieved accuracy of around 0.9814 and macro-F1 of around 0.7959 on Bot-IoT multiclass and accuracy of around 0.9517 and macro-F1 of around 0.9585 on N-BaIoT multiclass; FDL LSTM and BiLSTM slightly improved these multiclass macro-F1-scores (for example, BiLSTM of around 0.7762 on Bot-IoT and of around 0.9967 on N-BaIoT). Overall, FDL models preserved the high weighted F1 (approximately 0.9700–0.9900+) and macro-F1 seen in CDL, especially on N-BaIoT, confirming that privacy-preserving decentralisation via FDL can deliver detection quality close to centralised baselines under realistic non-IID splits.

A cross-paradigm synthesis consolidated these findings in the cross-paradigm summary table, which juxtaposed RF, CDL, and FDL for each dataset and task using accuracy, weighted precision, weighted recall, and weighted F1. For binary tasks, all three paradigms (RF, CDL, FDL) achieved near-identical performance (0.9980–1.0000), indicating that from a purely predictive standpoint, there is minimal penalty for moving from centralised RF to centralised DL and then to FDL. For multiclass tasks, CDL and FDL consistently outperformed RF, especially on Bot-IoT, with BiLSTM generally delivering the strongest macro-F1 among CDL models and FDL BiLSTM matching or closely approaching CDL performance on both datasets. These comparisons answered RQ3 by showing that FDL can preserve CDL performance to a large extent and answered RQ4 by quantifying that the accuracy–privacy trade-off is small in exchange for the significant benefit of keeping raw IoT data local to edge clients.

Finally, Section 4 compared the proposed models with the recent literature on Bot-IoT and N-BaIoT. The proposed CDL and FDL deep learning approaches matched or surpassed many reported baselines, particularly in multiclass settings where prior work often reports high accuracy but places less emphasis on macro-F1 across minority attacks. By explicitly reporting accuracy, weighted F1, and macro-F1 for both centralised and federated configurations, the chapter demonstrated that the proposed framework not only meets but often exceeds the detection performance of existing methods while adding a strong, experimentally validated privacy-preserving decentralisation dimension.

Several experiments yielded near-perfect AUC/F1, indicating a ceiling effect on these benchmarks under the chosen feature representations. Consequently, model comparisons were interpreted using metrics that remain informative under extreme imbalance and decentralised training, including macro-F1, benign-class TNR/specificity (binary), and client-level FDL variability. These analyses better expose minority-class and heterogeneity effects that are not visible when overall metrics saturate.

Overall, Section 4 showed that:

Deep learning (DNN/LSTM/BiLSTM) clearly outperforms RF in challenging multiclass scenarios. Targeted imbalance handling (e.g., SMOTE on Bot-IoT multiclass) significantly improves minority-attack detection without harming overall accuracy. Federated deep learning retains near-CDL performance under non-IID splits, especially on N-BaIoT, confirming that strong detection and privacy preservation can be achieved simultaneously. Compared with recent Bot-IoT and N-BaIoT studies, the proposed CDL/FDL framework achieves state-of-the-art or better performance while explicitly addressing decentralisation and trustworthiness. This directly supports the thesis aim of designing a trustworthy deep

learning approach to mitigate botnet activity in IoT environments and demonstrates that all four research questions are answered by the experimental evidence.

5. Conclusions, Limitations, and Future Work

This study developed a privacy-preserving IoT botnet detection framework that systematically compared a random forest (RF) baseline with centralised deep learning (CDL) and federated deep learning (FDL) using DNN, LSTM, and BiLSTM on the Bot-IoT and N-BaIoT datasets under binary and multiclass tasks. Across both datasets, RF provided a strong handcrafted baseline under full centralised data access, while CDL models achieved similarly high overall performance, including near-perfect separation between benign and attack traffic in several binary cases. In multiclass configurations, overall performance remained strong, with residual confusion concentrated among closely related classes, indicating that fine-grained discrimination remained the primary source of error rather than broad attack-versus-benign separation.

Importantly, the results clarified the privacy–performance relationship that motivated this work. FDL, orchestrated with Flower under non-IID and imbalanced client partitions, maintained performance close to that of the corresponding CDL models in binary settings, demonstrating that high detection capability can be preserved while keeping raw network traffic local to clients. Here, 'privacy-preserving' is used in the sense of data locality under FedAvg, and this study does not claim a formal privacy guarantee or quantify leakage risk from shared updates; accordingly, the evaluated trade-off is the change in detection performance when training is decentralised under non-IID and imbalanced client data. This provides empirical evidence that decentralised training can reduce the need for centralised collection of sensitive traffic features without materially sacrificing overall binary detection performance. In multiclass FDL, overall accuracy and weighted F1 remained high, whereas macro-F1 showed greater sensitivity to minority and fine-grained classes. This pattern indicates that the principal performance cost of privacy-preserving decentralisation in this study was concentrated in class imbalance and heterogeneity effects, which disproportionately affected underrepresented attack categories in multiclass learning.

Overall, the findings addressed the study's core objective by showing that privacy-preserving decentralised learning can closely track centralised performance for binary IoT botnet detection on two large benchmarks, while identifying multiclass minority-class recognition as the main residual challenge when client data are non-IID and imbalanced. These results support the feasibility of deploying accurate botnet detection in distributed IoT environments without requiring raw traffic centralisation, while also motivating targeted improvements for robust fine-grained detection under federated constraints.

This study was limited to two benchmark datasets, three deep architectures (DNN, LSTM, BiLSTM), and a FedAvg-based Flower configuration with five simulated clients. Although these choices enabled controlled, reproducible comparisons between CDL and FDL, they did not fully capture several operational factors that influence the privacy–performance trade-off in real deployments. First, the evaluation primarily focused on accuracy-oriented metrics (e.g., accuracy and F1 variants) and did not quantify privacy leakage risk from model updates (e.g., susceptibility to gradient-based inference) or formally measure communication and systems overhead (e.g., bandwidth, round time, energy cost). Communication overhead, latency, and energy consumption were not measured experimentally. Second, the federated setting lacked realistic dynamics such as client churn, intermittent connectivity, concept drift, or adversarial behaviours, all of which may alter convergence and privacy risk in practice.

Future work will therefore extend the framework in four directions. Recent federated-learning research also motivates concrete extensions in security, alternative sensing modal-

ities, and deployment constraints: SecFedDNN targets secure federated deep learning for edge–cloud environments using mechanisms such as pre-aggregation filtering and layer-adaptive sparsified aggregation, motivating an explicit evaluation of secure aggregation and robust update handling in the proposed framework (Alamir et al. [37]); power-consumption-based learning provides an alternative signal modality for IoT botnet detection, motivating extension beyond packet-level features when traffic visibility is constrained (Wakili et al. [38]); and federated learning has also been evaluated for privacy-preserving anomaly detection in constrained LoRa communications, supporting the need to quantify communication cost and intermittency under low-power network conditions (Senol et al. [39]). First, privacy preservation will be evaluated more explicitly by incorporating privacy-enhancing mechanisms beyond data locality (e.g., secure aggregation and differential privacy) and by reporting the associated performance and training-cost impact, thereby quantifying the privacy–performance–overhead trade-off rather than inferring privacy benefits solely from data locality. Second, imbalance-aware objectives and lightweight architectures will be investigated to improve minority-class recognition in federated multiclass settings while controlling false positives, including strategies robust to non-IID client distributions. Third, more realistic federated configurations will be considered, including varying client counts and availability, constrained communication regimes, and continual or online learning under drift. Fourth, validation on real edge deployments, including measurements of latency, memory footprint, and communication overhead, will strengthen the practical applicability of privacy-preserving federated IoT botnet detection in operational environments.

Author Contributions: Conceptualization, A.M.R.; methodology, A.M.R.; software, A.M.R.; validation, A.M.R.; formal analysis, A.M.R.; investigation, A.M.R.; resources, A.M.R.; data curation, A.M.R.; writing—original draft preparation, A.M.R.; writing—review and editing, A.M.R., N.S.S. and C.M.; visualization, A.M.R.; supervision, N.S.S. and C.M.; project administration, A.M.R.; funding acquisition, not applicable. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository. 1-The Bot-IoT Dataset. UNSW, https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 1 October 2022) 2-Detection_of_IoT_botnet_attacks_N_BaIoT. UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/442/detection+of+iot+botnet+attacks+n+baiot (accessed on 1 October 2022).

Acknowledgments: The author gratefully acknowledges Funds for Women Graduates (FfWG), an educational charity, for a GBP 2500 grant that supported me and enabled me to sustain focus on this work. The author also acknowledges the OpenBright Award Associated with the University of Wolverhampton, which provided GBP 1900 towards a high-performance laptop used to conduct the experiments reported in this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

AUC	Area under the curve
BiLSTM	Bidirectional long short-term memory
Bot-IoT	Botnet Internet of Things
CDL	Centralised deep learning
CM	Confusion matrix
CNN	Convolutional neural network

CPU	Central processing unit
CSV	Comma-separated value
DDoS	Distributed denial of service
DL	Deep learning
DNN	Deep neural network
DoS	Denial of service
ELU	Exponential linear unit
FDL	Federated deep learning
FedAvg	Federated averaging
FL	Federated learning
FN	False negative
FP	False positive
GPU	Graphics processing unit
HTTP	Hypertext Transfer Protocol
IID	Independent and identically distributed
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LSTM	Long short-term memory
ML	Machine learning
MLP	Multi-layer perceptron
N-BaIoT	Network-based Botnet of IoT
NN	Neural network
non-IID	Non-independent and identically distributed
OS	Operating system
ReLU	Rectified linear unit
RF	Random forest
RNN	Recurrent neural network
ROC	Receiver operating characteristic
RQ	Research question
RS	Randomised search (hyperparameter optimisation)
RS-best	Best configuration selected by randomised search
SD	Standard deviation
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine
TCP	Transmission Control Protocol
TN	True negative
TNR	True negative rate
TP	True positive
TPR	True positive rate
UDP	User Datagram Protocol
UCI	University of California, Irvine
UNSW	University of New South Wales
VRAM	Video random-access memory

References

1. Rey, V.; Sánchez Sánchez, P.M.; Celdrán, A.H.; Bovet, G.; Jaggi, M. Federated learning for malware detection in IoT devices. *Comput. Netw.* **2022**, *204*, 108693. [CrossRef]
2. Zhang, T.; Gao, L.; He, C.; Zhang, M.; Krishnamachari, B.; Avestimehr, A.S. Federated learning for the Internet of Things: Applications, challenges, and opportunities. *IEEE Internet Things Mag.* **2022**, *5*, 24–29. [CrossRef]
3. Bala, B.; Behal, S. AI techniques for IoT-based DDoS attack detection: Taxonomies, comprehensive review and research challenges. *Comput. Sci. Rev.* **2024**, *52*, 100631. [CrossRef]
4. Ahmed, A.A.; Jabbar, W.A.; Sadiq, A.S.; Patel, H. Deep learning-based classification model for botnet attack detection. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 3457–3466. [CrossRef]

5. Popoola, S.I.; Ande, R.; Adebisi, B.; Gui, G.; Hammoudeh, M.; Jogunola, O. Federated deep learning for zero-day botnet attack detection in IoT-edge devices. *IEEE Internet Things J.* **2022**, *9*, 3930–3944. [CrossRef]
6. Metwaly, A.A.; Elhenawy, I. Protecting IoT devices from BotNet threats: A federated machine learning solution. *Sustain. Mach. Intell. J.* **2023**, *4*, 1–12. [CrossRef]
7. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.-C.; Yang, Q.; Niyato, D.; Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2031–2063. [CrossRef]
8. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
9. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
10. Patil, A.; Deshpande, A. Evaluating ML models on CTU-13 and IoT-23 datasets. In Proceedings of the 3rd International Conference on Advanced Computing Technologies and Applications (ICACTA 2023), Mumbai, India, 6–7 October 2023. [CrossRef]
11. Nazir, A.; He, J.; Zhu, N.; Wajahat, A.; Ma, X.; Ullah, F.; Qureshi, S.; Pathan, M.S. Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets. *J. King Saud Univ. Comput. Inf. Sci.* **2023**, *35*, 101820. [CrossRef]
12. Popoola, S.I.; Adebisi, B.; Gui, G.; Hammoudeh, M.; Gacanin, H.; Dancey, D. Memory-efficient deep learning for botnet attack detection in IoT networks. *Electronics* **2021**, *10*, 1104. [CrossRef]
13. Liu, T.; Ding, J.; Wang, T.; Pan, M.; Chen, M. Towards fast and accurate federated learning with non-IID data for cloud-based IoT applications. *arXiv* **2022**, arXiv:2201.12515. [CrossRef]
14. Attota, D.C.; Mothukuri, V.; Parizi, R.M.; Pouriyeh, S. An ensemble multi-view federated learning intrusion detection for IoT. *IEEE Access* **2021**, *9*, 117734–117745. [CrossRef]
15. Tian, P.; Chen, Z.; Yu, W.; Liao, W. Towards asynchronous federated learning based threat detection: A DC-Adam approach. *Comput. Secur.* **2021**, *108*, 102344. [CrossRef]
16. Popoola, S.I.; Adebisi, B.; Gui, G.; Hammoudeh, M.; Gacanin, H.; Dancey, D. Optimizing deep learning model hyperparameters for botnet attack detection in IoT networks. *TechRxiv*, 2022; *preprint*. [CrossRef]
17. Bensaoud, A.; Kalita, J. Optimized detection of cyber-attacks on IoT networks via hybrid deep learning models. *Ad Hoc Netw.* **2025**, *170*, 103770. [CrossRef]
18. Sahu, A.K.; Sharma, S.; Tanveer, M.; Raja, R. Internet of Things attack detection using hybrid deep learning model. *Comput. Commun.* **2021**, *176*, 146–154. [CrossRef]
19. Myakala, P.K.; Kamatala, S.; Bura, C. Privacy-preserving federated learning for IoT botnet detection: A federated averaging approach. *ICCK Trans. Mach. Intell.* **2025**, *1*, 6–16. [CrossRef]
20. Nguyen, V.T.; Beuran, R. FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection. *Comput. Secur.* **2025**, *151*, 104337. [CrossRef]
21. Chen, Z.; Lv, N.; Liu, P.; Fang, Y.; Chen, K.; Pan, W. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access* **2020**, *8*, 217463–217472. [CrossRef]
22. Rahman, S.A.; Tout, H.; Talhi, C.; Mourad, A. Internet of Things intrusion detection: Centralized, on-device, or federated learning? *IEEE Netw.* **2020**, *34*, 310–317. [CrossRef]
23. Kim, S.; Cai, H.; Hua, C.; Gu, P.; Xu, W.; Park, J. Collaborative anomaly detection for Internet of Things based on federated learning. In Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC 2020), Chongqing, China, 9–11 August 2020; pp. 851–856. [CrossRef]
24. Gugueoth, V.; Safavat, S.; Shetty, S. Security of Internet of Things (IoT) using federated learning and deep learning—Recent advancements, issues and prospects. *ICT Express* **2023**, *9*, 941–960. [CrossRef]
25. Yadav, K.; Gupta, B.B. Clustering based rewarding algorithm to detect adversaries in federated machine learning based IoT environment. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE 2021), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–6. [CrossRef]
26. Dritsas, E.; Trigka, M. Federated learning for IoT: A survey of techniques, challenges, and applications. *J. Sens. Actuator Netw.* **2025**, *14*, 9. [CrossRef]
27. Ferrag, M.A.; Friha, O.; Maglaras, L.; Janicke, H.; Shu, L. Federated deep learning for cybersecurity in the Internet of Things: Concepts, applications, and experimental analysis. *IEEE Access* **2021**, *9*, 138509–138542. [CrossRef]
28. Churcher, A.; Ullah, R.; Ahmad, J.; Rehman, S.U.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W.J. An experimental analysis of attack classification using machine learning in IoT networks. *Sensors* **2021**, *21*, 446. [CrossRef] [PubMed]
29. Özer, E.; İskefiyeli, M.; Azimjonov, J. Toward lightweight intrusion detection systems using the optimal and efficient feature pairs of the Bot-IoT 2018 dataset. *Int. J. Distrib. Sens. Netw.* **2021**, *17*, 15501477211052202. [CrossRef]
30. Alkadi, S.; Al-Ahmadi, S.; Ben Ismail, M.M. Toward improved machine learning-based intrusion detection for Internet of Things traffic. *Computers* **2023**, *12*, 148. [CrossRef]

31. Usoh, M.; Asuquo, P.; Ozuomba, S.; Stephen, B.; Inyang, U. A hybrid machine learning model for detecting cybersecurity threats in IoT applications. *Int. J. Inf. Technol.* **2023**, *15*, 3359–3370. [CrossRef]
32. Wardana, A.A.; Qayyum, A.; Hubballi, N. Ensemble averaging deep neural network for botnet detection in heterogeneous IoT devices. *Sci. Rep.* **2024**, *14*, 54438. [CrossRef]
33. Sharma, K.P.; Nagpal, T.; Vora, T.; Yadav, A.; Abdullah, M.I.; Jayaprakash, B.; Kashyap, A.; Sridevi, G.; Bhowmik, A.; Bukate, B.B. Interpretable intrusion detection for IoT environments using a self-attention-based explainable AI framework. *Sci. Rep.* **2025**, *15*, 39937. [CrossRef]
34. Nandanwar, H.; Katarya, R. Deep learning enabled intrusion detection system for industrial IoT environment. *Expert Syst. Appl.* **2024**, *249*, 123808. [CrossRef]
35. Shahin, M.; Hosseinzadeh, A.; Chen, F.F. A Two-Stage Hybrid Federated Learning Framework for Privacy-Preserving IoT Anomaly Detection and Classification. *IoT* **2025**, *6*, 48. [CrossRef]
36. Asiri, M.; Wazirali, R.; Alkhalaf, S.; Othman, A.; Al-Maleh, N. RPFL: A reliable and privacy-preserving framework for federated learning-based IoT malware detection. *Electronics* **2025**, *14*, 1089. [CrossRef]
37. Alamir, R.H.; Noor, A.; Almukhalfi, H.; Almukhlifi, R.; Noor, T.H. SecFedDNN: A Secure Federated Deep Learning Framework for Edge-Cloud Environments. *Systems* **2025**, *13*, 463. [CrossRef]
38. Wakili, A.A.; Guni, S.; Khan, S.A.; Yu, W.; Jung, W. Advancing Machine Learning Strategies for Power Consumption-Based IoT Botnet Detection. *Sensors* **2025**, *25*, 7553. [CrossRef]
39. Senol, K.; Gumusbas, K.; Ali, B.; Almardan, A.; Giray, G. Privacy-preserving detection of tampered radio-frequency transmissions utilizing federated learning in LoRa networks. *Sensors* **2024**, *24*, 7336. [CrossRef]