IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 2, February, 2026, pp. 531-539
Journal homepage: http://iieta.org/journals/jesa

A Lightweight Deep Learning-Based Split Learning Scheme for Intrusion Detection in Multi-Controller Software-Defined Networking Internet of Things Environments

Mohammed Hameed

Computer Science Department, College of Education, University of Mustansiriyah, Baghdad 10071, Iraq

Corresponding Author Email: mohameed.majed@uomustansiriyah.edu.iq

Copyright: ©2026 The author. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/jesa.590222

Received: 6 December 2025
Revised: 3 February 2026
Accepted: 18 February 2026
Available online: 28 February 2026

Keywords:
intrusion detection systems, Internet of Things, Software-Defined Networking, deep learning, InSDN

ABSTRACT

The rapid development of the Internet of Things (IoT) has raised significant security concerns and increased the need for next-generation technologies, such as Software-Defined Networking (SDN), to improve network management and operation. However, conventional centralized intrusion detection systems (IDSs) often face challenges related to privacy constraints and the limited computational capabilities of IoT devices. To address these issues, this study proposes an IDS based on Split Learning (SL) for multi-controller SDN-IoT environments. The proposed design distributes the computational workload of an eight-layer deep neural network (DNN) between resource-constrained IoT hosts, which execute the first two layers, and the SDN controller, which processes the remaining six layers. In this way, raw network traffic does not leave the local device; instead, only smashed data in an abstract form is transmitted, thereby enhancing privacy preservation. The proposed system was evaluated using the InSDN dataset, and stratified sampling based on Shannon entropy was employed to ensure a balanced training set. Experimental results show that the model achieves an accuracy of approximately 99.5%, while the loss reaches a minimum value of 0.02. The system also demonstrates strong resilience against volumetric attacks, such as distributed denial-of-service (DDoS) and probing attacks, while maintaining a lightweight architecture suitable for IoT devices. The baseline model size is 106.67 KB, including 31.38 KB for the head model and 75.29 KB for the tail model. Overall, the proposed model provides an effective balance among intrusion detection performance, data privacy, and efficient resource utilization in modern converged networks.

1. INTRODUCTION

Software-Defined Networking (SDN) is an attractive network technology paradigm. It has overcome various obstacles and constraints of the conventional networking infrastructures. SDN is a management framework that offers various advantages and represents a significant development in network management [1]. The development of Internet-connected devices is only increasing network traffic. The changing demands and requirements of the modern computer networks have been fulfilled through SDN. SDN has helped improve the security and operational efficiency of computer networks. It facilitated the management of networks. It provides a software-centric, centralized network management strategy that renders computer networks dynamic, easily configurable, programmable, efficient, and dependable. SDN enables efficient network and traffic monitoring. Despite its numerous advantages, SDN also presents several weaknesses. The weaknesses are in the centralized methodology. SDN comprises a tiered architecture divided across three planes. The initial layer of SDN is the application plane. The second layer is the control plane, also referred to as the control layer. The third plane of SDN is the data plane, commonly referred to as the infrastructure layer [2]. Every layer of SDN is susceptible to hackers. SDN is a significant technology that separates the forwarding process of network packets from the routing process. The SDN control plane is often referred to as the brain, so attackers frequently attempt to compromise it. If adversaries successfully compromise the control plane, they can manipulate the entire network.

Machine learning (ML) has demonstrated efficacy in combating cyberattacks over the past couple of decades. Currently, ML is a significant solution in cybersecurity. In this period, the quantity of devices has proliferated and continues to grow significantly. With the expansion of the digital realm, the incidence of cyberattacks also increases. We require an automobile solution to combat cyberattacks. ML offers this type of solution [3]. ML algorithms can identify an illegal device attempting to connect to the network. They can detect the presence of a compromised device and identify a new form of malware based on existing signatures [4]. The centralized server cannot audit device data due to privacy constraints, as such an audit would require transferring sensitive data across the network. The inherent data distribution in ionic applications can significantly degrade distributed training performance, leading to reduced model accuracy and training instability. The second important reason for the heterogeneous learning environment in the context of IoT systems is the

variability in the computing and connectivity resources of IoT devices. This heterogeneity manifests in various dimensions, including differences in processor capabilities, memory capacity, network connectivity, and bandwidth. The vast range of computational capabilities presents considerable hurdles for training a universal model. Devices with constrained processing capabilities or memory, commonly referred to as stragglers, might significantly impede the overall training process. Moreover, devices with inadequate network connectivity or limited bandwidth struggle to efficiently transfer data or model updates [5].

In this work, an IDS-based SL framework has been proposed as a lightweight deep neural network (DNN)-based multi-controller SDN environment. The proposed system is designed and tested for three clients, each containing two layers of the DNN model, while the other six layers are implemented on the SDN controller server. The main contribution of this work is its ability to detect different types of IoT attacks, such as distributed denial-of-service (DDoS), Prob, web attacks, and others, with high accuracy while maintaining minimal client-side layers. The problem is specified as a multi-class intrusion detection challenge on the InSDN data set, which is focused on nine categories of cyber-attacks. The main limitations that we discuss are the lack of edge computing resources and the necessity to avoid uploading any raw data by IoT devices, hence maintaining privacy. We have clearly defined our baseline models, i.e., a centralized DNN, a small on-box model, and an SL architecture, to allow a clear comparative analysis.

The rest of this paper is organized as follows: Section two demonstrates the state-of-the-art related work. Section three shows the methodology and theoretical background for the proposed method in this work. Section four illustrates the proposed IDS-based SDN system, while section five presents the main results and their analysis. Finally, section six introduces the main conclusion and future work.

2. RELATED WORK

The increasing security threats in converged SDN and Internet of Things (IoT) environments have prompted significant research into the development of effective intrusion detection systems (IDSs), primarily emphasizing the use of deep learning and ML methodologies. Proposals for IDSs based on deep learning research suggest innovative architectures tailored to the distinct attributes of Software-Defined Networking Internet of Things (SDN-IoT) systems.

The practical validation of IDS components is exemplified by Ariffin et al. [6], who presented a proof-of-concept for the accuracy assessment of a Snort IDS implemented on an SDN-IoT testbed utilizing a Ryu Controller and the UNSW-NB15 dataset. Analysis of DL and feature fusion, alongside primary research and review papers, consolidates contemporary trends and issues. Elsayed et al. [7] presented a Secured Automatic Two-level Intrusion Detection System (SATIDS). This system utilizes an enhanced Long Short-Term Memory (LSTM) model engineered for multi-level classification: distinguishing between benign and attack traffic, categorizing attack types, and identifying sub-attack types. The SATIDS model achieved high accuracy on two modern datasets, with 96.35% and 99.73% on ToN-IoT and InSDN, respectively. A new HSAFS-OCAE model proposed by Maray et al. [8] would maximize intrusion detection in SDN-enabled IoT systems. Since DDoS attacks in IoT setups are widespread, several studies aim to detect them specifically and compare them.

Ataa et al. [9] introduced Ryu-IDS, an IDS for contemporary networks that employs DL and a foundational Ryu controller. They proposed a technique divided into three distinct phases and based on a CNN-LSTM architecture. Wahab et al. [10] proposed a Multi-Class IDS tailored for DDoS attacks in IoT, evaluating CNNs, DNNs, and Transformer-based models for multi-class classification using the CIC IoT 2023 dataset. Their findings indicated that the DNN model attained the greatest binary classification accuracy of 99.2%. Thalapala et al. [11] analyzed MLADAD, a study on conventional ML techniques, including Logistic Regression, Decision Tree, and Random Forest, for detecting DDoS attacks in SDN-IoT networks using the CSE-CIC-IDS2018 dataset. They determined that Random Forest and Decision Tree had the highest accuracy for the job. Comparative analyses are essential for evaluating new models and datasets. Dhirar and Hamad [12] conducted an extensive comparison of a newly created IDS dataset with established benchmarks (InSDN, BoT-IoT, and ToN-IoT). This study employed four deep learning architectures (CNN, LSTM, RNN, and DNN) for assessment, revealing that the DNN model exhibited consistent performance throughout the evaluated datasets.

Table 1. Summary of related work comparison

Ref.	Key Contribution / Technique	Network Environment	Dataset(s)	Key Performance Metric (e.g., Accuracy)
[6]	IDS accuracy testing using Snort IDS on an SDN-IoT	SDN-IoT Testbed	UNSW-NB15	Accurate in false positive and false negative tests
[7]	Secured IDS (SATIDS) based on improved LSTM	IoT and SDN	ToN-IoT, InSDN	96.35% (ToN-IoT), 99.73% (InSDN)
[8]	Deep Learning Driven IDS (HSAFS-OCAE)	SDN-Enabled IoT	Not explicitly stated in the snippet	Not explicitly stated in the snippet
[9]	Proposes (SDN/IoT) using Deep Learning (CNN-LSTM)	SDN and IoT	Not explicitly stated in the snippet	Not explicitly stated in the snippet
[10]	Multi-Class IDS using (CNN, DNN, Transformer)	IoT	CIC IoT 2023	DNN: 99.2% (Binary Classification)
[11]	Analysis of ML algorithms (LR, DT, RF) for DDoS detection	SDN-IoT	CSE-CIC-IDS2018	Random Forest and Decision Tree show the best accuracy
[12]	Comparison of an IDS dataset using (CNN, LSTM, RNN, DNN)	SDN-IoT	InSDN, BoT-IoT, ToN-IoT	DNN showed consistent performance

Note: IDS = intrusion detection systems; SDN-IoT = Software-Defined Networking Internet of Things; LSTM = Long Short-Term Memory

Table 1 compares the related work across different metrics and their main results. Compared with the current work, these studies [7, 12] use the same dataset, InSDN, and achieve good results (around 99% accuracy). In contrast, this work achieves almost the same results using SL, which is applicable in a limited-resource network such as an IoT network.

3. THEORETICAL BACKGROUND

Before discussing the suggested model, one must understand the main strategy and approach to be implemented in this work. These choices were made based on a review of past studies that investigated the process of building an efficient IDS system and evaluated the methodologies used. The next section outlines the technologies and procedures involved in deploying a NIDS on an SDN network.

3.1 Software-Defined Network

To build connections among network nodes and enable data exchange across the network, traditional networking architectures use switches and routers. The technology is a conventional networking system that is prone to confidentiality breaches and external attacks. SDN is another networking model that isolates the data delivery process from specific devices to improve the efficiency of a centralized system [13]. The organizational structure of this paradigm is split into multiple planes, the purpose of which were: (i) data plane, which handles the forwarding of packets; (ii) control plane, which defines routing using a flow table which outlines rules that are followed when packets are to be effectively handled; and the application plane, which is a set of services offered to the users.

Although that, the decentralization of SDN can spawn new vulnerabilities. The controller may be weakened by minimizing the communication bandwidth between its layers of infrastructure, including the OpenFlow switch and the SDN controller. However, SDN has the potential to improve network security by enabling the installation of security applications, such as IDS, which can detect potential threats. Furthermore, it is important to understand that the flow rules can be dynamically changed in response to changing demands by virtue of the programmability and control that SDN provides, which are absent in traditional networking systems [14]. This is an issue that should be included in any in-depth analysis. The traditional SDN architecture is shown in Figure 1.

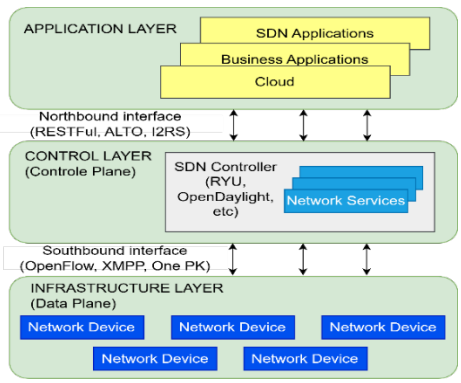


Figure 1. Software-Defined Network (SDN) network architecture [15]

3.2 Intrusion detection systems

IDSs are crucial for safeguarding systems by monitoring and analyzing network traffic to identify security breaches and threats using signature-based or anomaly-based approaches. The original method relies on existing network patterns and thus fails to identify fresh threats. In contrast, the latter technique analyzes certain characteristics of network traffic, facilitating the detection of anomalies from standard network behavior that may indicate possible attacks; a concise comparison is shown in Table 2 [16]. Nonetheless, certain drawbacks were noted, such as the inability to detect encrypted packets and a possibly elevated rate of false positives, requiring human intervention to adjust the anomaly indicators, which eventually resulted in an inefficient security solution [17].

Table 2. Detection technique comparison [16]

Attribute	Detection Type	
	Signature	Anomaly
Alarm_rate	Low	High
Speed	Hight	Low
Flexibility	Low	High
Reliability	High	Moderate
Scalability	Low	High
Robustness	Low	High

3.3 Split Learning

The neural network is divided into components, with one portion retained by a client and the other by a server. The client executes the early layers using its local data and transmits only the intermediate activations, referred to as smashed data, to the server. The server continues the computation and loss evaluation, then performs backpropagation. The resultant gradient data is transmitted to the client to refresh its local layers. Formally, a model f can be split into two functions [18].

$$f(x) = f_s(f_c(x)) \tag{1}$$

where,

- $f_c(\cdot)$ represents the layers executed on the client side.
- $f_s(\cdot)$ represents the layers executed on the server side.
- x is the input data.

During the forward pass:

$$a = f_c(x) \tag{2}$$

Here, a (the smash data) is transmitted to the server, which computes the prediction:

$$\hat{y} = f_s(a) \tag{3}$$

This gradient $\nabla_{a\mathcal{L}}$ is sent back to the client, which updates its local parameters via:

$$w_{c\leftarrow} w_{c-} \eta \nabla_{w_c} \mathcal{L}(y, \hat{y}) \tag{4}$$

While the server updates its parameters as:

$$w_{s\leftarrow} w_{s-} \eta \nabla_{w_s} \mathcal{L}(y, \hat{y}) \quad (5)$$

w_c and w_s indicate the client and the server weights, respectively, and e signifies the learning rate. On the one hand, SL helps reduce the computational costs of resource-constrained devices; on the other hand, it also presents risks such as activation leakage and privacy breaches caused by unwanted data disclosure, and communication overhead due to the transfer of activations and gradients.

4. PROPOSED INTRUSION SYSTEM DESIGN

This part presents the design of the proposed SL-IDS for multi-controller SDN-IoT networks. In the current work, the Ryu controller is applied; Ryu is an open-source Python-based SDN controller, which is developed by Nippon Telegraph and Telephone (NTT). The controller enables the management of flows judicially based on its modular design and APIs to accommodate a wide range of protocols, which includes OpenFlow (versions 1.0-1.5), Netconf, and OF-config. Its architectural stratification includes three different layers, which are the application layer (implementing network logic), the control layer (SDN services, APIs), and the infrastructure layer (physical and virtual devices). The controller provides RESTful northbound APIs without losing compatibility with a wide range of southbound protocols, thus allowing to control of switches like Open vSwitch. Experiments have shown that Ryu is able to attain one of the lowest response times among modern-day controllers, which provides it with better latency behavior.

The SDN paradigm has an intrinsic architectural weakness due to a centralized control plane. This centralization is not only creating a critical single point of failure, but also creates built-in scalability limits. These structural vulnerabilities are exacerbated by extreme security risks, especially the DDoS attacks aimed at the communication bandwidth of the controller as a result of organized packet-flooding efforts. Therefore, effective countering of a single point of failure and the effective management of large multi-domain networks require a distributed strategy.

In order to solve these issues, the Multi-Controller SDN architecture suggests a practical model, in which the controllers are in charge of a specific area. This scalable and robust Multi-Controller SDN design offers increased scalability, elasticity and fault tolerance, which makes it an interesting choice in the complex and large-scale network. Figure 2 shows the design of the suggested system and identifies several attack sources. The proposed architecture uses the Mininet simulator running on Linux to build a hierarchical tree topology that can segment and manage a network effectively. The topology has two controllers, one for each of the two OVS switches. These switches then connect to two additional OVS instances, forming four subdomains. Each subdomain has three hosts and one access point (AP), and the APs are connected to three stations and three sensors representing an IoT environment. An IoT-based SDN dataset is used to evaluate the proposed learning schemes. It is important to understand its structure and class distribution because these factors influence the model's training, performance, and classification feasibility. Table 3 summarizes the main features of the dataset. The IoT clients have been distributed between the two domains such that one domain has two clients and the other has one. By doing this, the attacks will be demonstrated among different domains and different controllers to ensure scalability. Changing thses topologies will not affect the training process or deployment of the models and even when increasing the number of clientss since the SDN in its nature is scalable and can manage these changes only by software configuration.

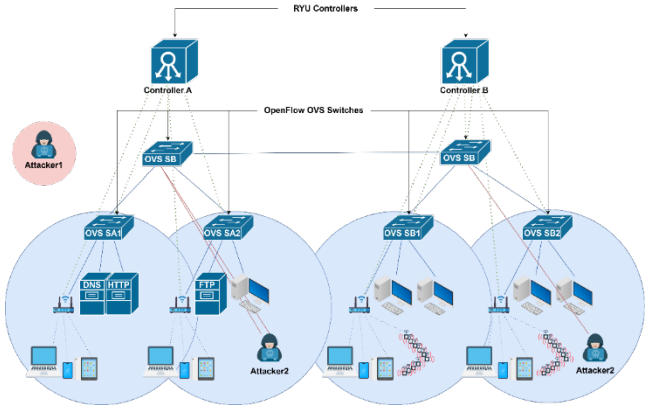


Figure 2. Proposed multi-controller Software-Defined Network (SDN) system-based intrusion detection systems (IDS)

Table 3. In the Software-Defined Network (SDN) dataset details

Parameters	Description
Dataset	InSDN
No. of Records	343889
No. of Features	84
Classes	Probe, DDoS, Normal, DoS, DDoS (SYN Flood), BFA, Web-Attack, BOTNET, U2R

Each dataset was further divided into 70% training, 20% testing, and 10% validation data. The training data was split into three parts based on the number of clients, using Shannon entropy-based stratified sampling to achieve statistical balance across evaluation subsets. For this, Shannon entropy is used to obtain the variances of various local datasets from each federated learning client. For a given dataset of length n and with k attack classes, where ci is the number of samples in class i, the class balance is described using the following formula [19, 20]:

$$Entropy = \frac{-\sum_{i=1}^{k} p_i \log p_i}{\log k} \quad (6)$$

where, $p_i = \frac{c_i}{n}$.

This approach maintains the class distribution of the original dataset. Table 4 shows entropy calculation for that distribution. The stratification approach attains maximum entropy (100%) by ensuring uniform proportions of attack classes within each partition, thereby eliminating sampling bias and accurately reflecting real-world attack prevalence. Table 4 shows the InSDN dataset parameters, while Figure 3 shows the different attack distributions in the dataset. Table 5 shows the hyperparameters of the SL model used in this work. Algorithm 1 shows the SL sequential procedure over the three clients.

Table 4. Entropy calculation for the class distribution

Label	Estimated Count (ni)	Percentage (Pi)
Probe	98,000	28.26%
DDoS (Combined)	74,000 + 48,500 = 122,500	35.32%
Normal	68,500	19.75%
DoS	54,000	15.57%
BFA	2,000	0.58%
Web-Attack	1,000	0.29%
BOTNET	700	0.20%
U2R	100	0.03%

Table 5. Hyperparameters of the proposed Split Learning (SL) model

Parameter	Value
Number of Clients	3
Epochs	25
Batch Size	32
Client Learning Rate	0.001
Server Learning Rate	0.001
Number of Layers	8
Client Head Layers	2
Server Tail Layers	6
Loss function	Categorical cross entropy
Optimizer	Adam
Activation function (hidden layers)	ReLU
Activation function (output layer)	Softmax
Normalization function	Min-Max
Label encoding	One-hot encoder

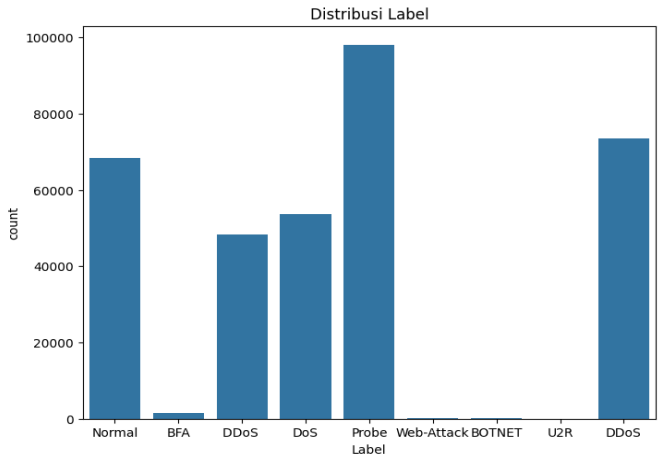


Figure 3. Attack distribution

Figure 4 shows the proposed supervised learning architecture for intrusion detection in an SDN-IoT. This setup provides a way to distribute computations by splitting an 8-layer DNN across resource-constrained IoT devices and the centralized SDN controllers. The technology works through a cooperative four-step process that ensures data privacy and reduces the processing load on IoT devices. The hosts, including the IoT devices such as sensors, cellphones, and PCs, manage their own localized databases. They only process the first two layers of the DNN and not the full eight-layer model. The preliminary identity of unprocessed data is processed on the host local to the data. SL framework processes the feedforward and backpropagation algorithm sequentially, i.e., the first client applies feedforward and computes the smashed data and uploads it to the server, then the second client and so on. In the server side each samshed for a specific client is processed also sequentially. This results in "smashed data" and an intermediate representation (feature map) that masks the original input data, thus allowing increased privacy. The hosts send the corrupted data to the SDN controller on the northbound interface. This is far lighter than sending raw datasets or even the full model parameters, and hence can be optimal for IoT devices with limited power capacity. The SDN controller is the main processing center that supports the other six layers of the DNN. The controller performs the forward pass with the processed information and calculates the error using a loss function, specifically categorical cross-entropy, since the model performs multi-class classification.

Algorithm 1. Split Learning

Inputs: K clients, total rounds T, local epochs E, learning rate η. **Initialize:** Client weights W_c^0, Server weights W_s^0. For each round $t = 1,2, \dots, R$:

- **Forward Pass:**

1. Client k computes smashed data:

$$A_k = f(X_k; W_{c,k}).$$

2. Client sends A_k and labels Y_k to the **Main Server sequentially**.
3. Main Server computes the output:

$$\hat{Y}_k = f(A_k; W_s).$$

4. Main Server calculates the loss:

$$L = \ell(\hat{Y}_k, Y_k).$$

- **Backpropagation process:**

5. Main Server computes gradients for the server-side: ∇W_s and gradients with respect to the smashed data:

$$g_{A,k} = \frac{\partial L}{\partial A_k}.$$

6. Main Server updates server weights:

$$W_s \leftarrow W_s - \eta \nabla W_s.$$

7. Main Server sends $g_{A,k}$ back to Client k.
8. Client k computes client-side gradients using the chain rule:

$$\nabla W_{c,k} = g_{A,k} \cdot \frac{\partial A_k}{\partial W_{c,k}}$$

9. Client k updates local weights:

$$W_{c,k} \leftarrow W_{c,k} - \eta \nabla W_{c,k}.$$

The controller begins with the backpropagation calculations, with the goal of changing the six-layer weights. To facilitate learning of client-side layers, the controller sends the gradients back to the IoT hosts. The hosts make use of these gradients to adjust the weights of their local two layers. This ends one training round without the server accessing the unprocessed local data. The main advantages of IoT with SDN

can be described as follows:

i. Lightweight for IoT: Needing only two layers of computing at the host of devices allows constrained CPU and RAM devices to be in a complex train IDS.
ii. Privacy-Conserving: Unprocessed sensitive network traffic; only the abstracted "smashed data" gets sent. Content-area network data stays on the local network device; only the system that posts it to the network knows the processed data.
iii. SDN Efficiency: Use of the SDN controller as a server allows for a wide picture of network security posture, which helps the IDS to get an idea about the trend for many hosts simultaneously.

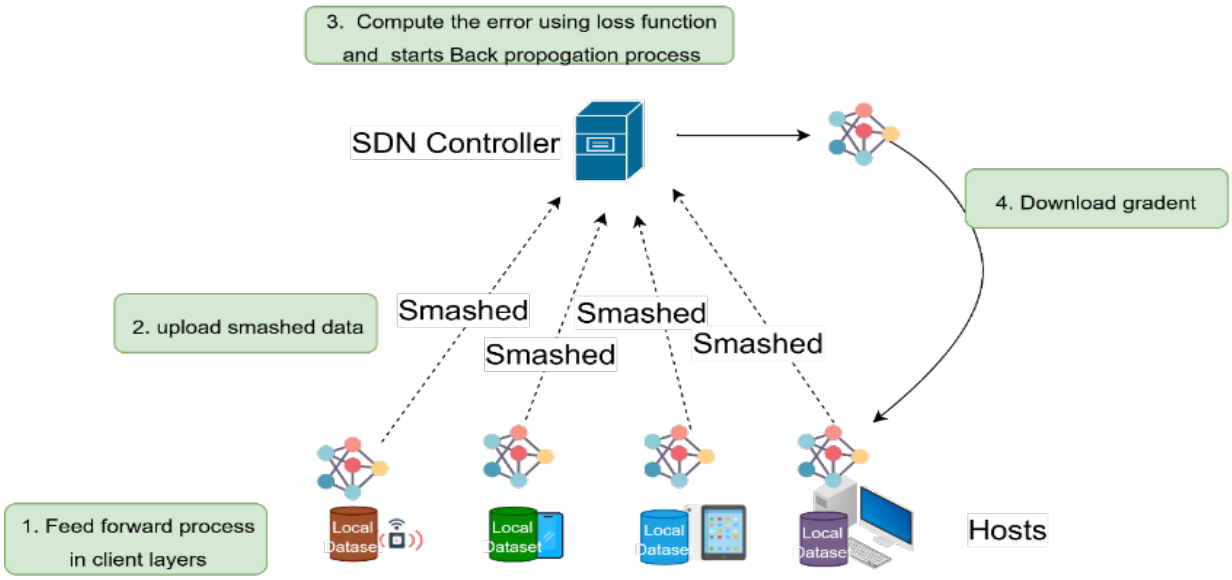


Figure 4. Proposed Split Learning (SL) model based on Software-Defined Network (SDN)

5. RESULTS AND DISCUSSION

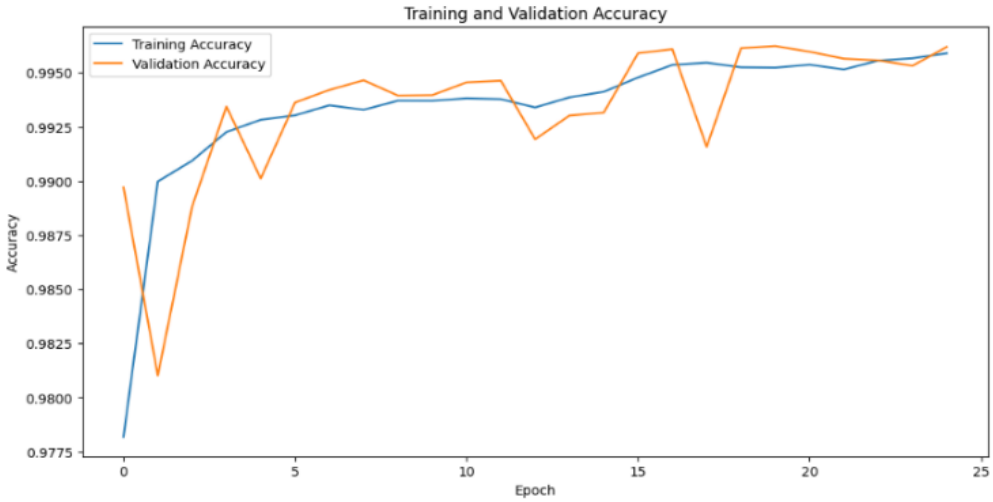
The model was advanced using TenFlow on Linux operating through Mininet network emulator. To reduce the imbalance in the classes, we used Shannon entropy-based stratified sampling, thus having a statistically balanced training sample in all three client subsets. We have also recorded the total training time and hardware requirements, namely, a GPU with 12GB of memory, of the IoT hosts and the SDN controller. Figure 4 shows the proposed supervised learning architecture for an IDS in an SDN-enabled IoT context. This configuration distributes the computational load of the 8-layer DNN across limited-resource IoT devices and a centralized SDN controller. The technology works through a cooperative four-step procedure that protects data privacy while limiting the processing load on IoT devices. The hosts, including IoT devices such as sensors, as well as cellphones and PCs, manage their own databases, called local databases. They only handle the first two layers of the DNN rather than the full eight-layer model. Each host performs the preliminary forward pass on its own processor for local unprocessed data. This results in "smashed data," an intermediate representation (in the form of a feature map) that obscures the original input data, thereby increasing privacy. The hosts pass the corrupted data to the SDN Controller via the northbound interface. This is much lighter than sending the raw datasets or the entire model parameters and, thus, best suited for low-power IoT devices. The SDN controller is the main processing center that fits in the other six layers of the DNN. This part dealt with the results of training and testing model performance. To assess the effectiveness of a classification model, the confusion matrix and classification report compare actual target values with expected results.

The accuracy and loss curves in Figures 5(a) and (b) indicate an excellent training process. The model has an exceptional accuracy of more than 99.5% percent. The similarity between the training and validation accuracy curves indicates that the allocation of 2 client layers and 6 server layers is an effective way to express the essential characteristics without overfitting. Notwithstanding the intricacies of SL (where gradients have to be transmitted from the server to the client), the training loss reduces continually and converges to a minimum value (below 0.02). There is a huge rise in validation loss at epoch 17. In an SDN-based SL situation, this is often blamed on a transient mismatch in feature-extraction synchronization or on an atypical, unrepresentative dataset. Nonetheless, the model shows rapid recovery, demonstrating its robustness.

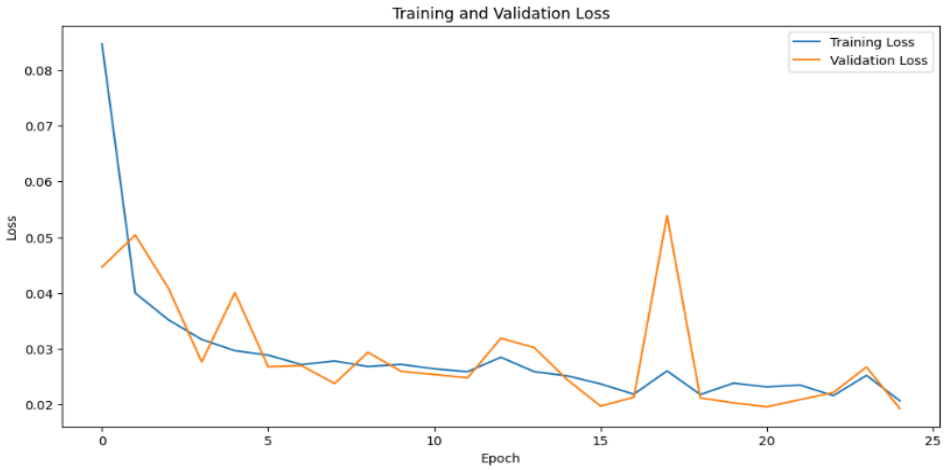
Figure 6 shows the IDS's effectiveness in distinguishing between different forms of network traffic, as shown in the confusion matrix. Most predictions fall on the diagonal, indicating accurate classifications across all attack categories. The model displays excellent performance in the face of common high-traffic attacks in SDN systems, such as Probe (29,319 correct) and DDoS (more than 36,000 correct across two categories). The Web-Attack class has worse performance (27 valid classifications vs. 31 misclassifications as DoS). This indicates an unbalanced dataset, with insufficient Web-Attack samples during the training phase.

This specific setup is exceptionally appropriate to an SDN context. By keeping only two layers on the client, presumably the SDN switches or the lightweight agents, the computational load on the data plane has been reduced. SL ensures that network traffic data from the edge (client side) is not lost or deleted, and that only "smashed data" (output from the 2nd layer) is sent to the SDN controller (server). The high accuracy

indicates that these two layers are sufficient for feature abstraction without losing crucial security information. The 6 levels of the server (SDN controller) perform the crucial work of identifying complex patterns such as Botnets and Probes while keeping network visibility at the center.



(a)



(b)

Figure 5. (a) Split Learning (SL) DNN model accuracy; (b) Split Learning (SL) DNN model loss

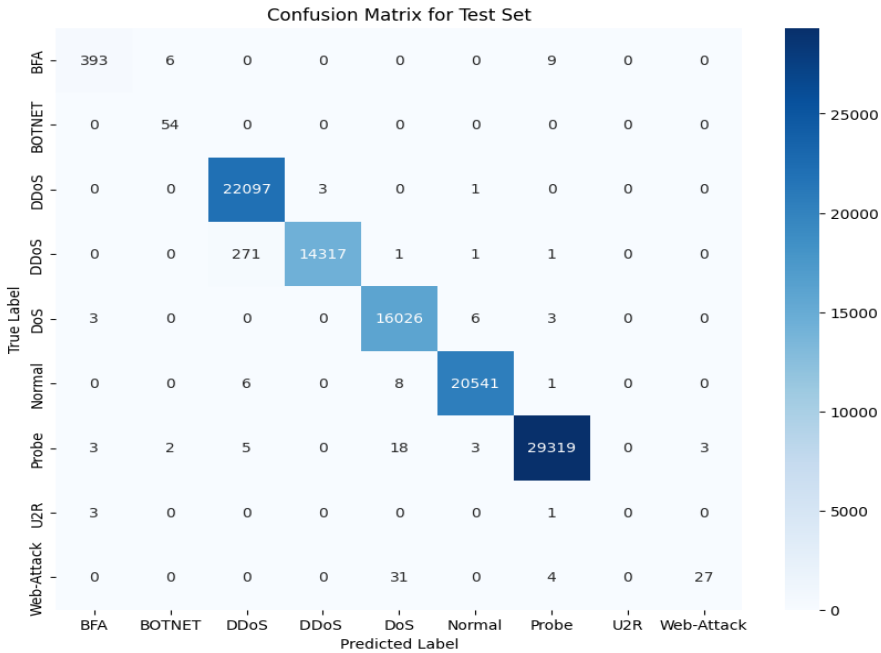


Figure 6. Multiclass confusion matrix

The classification report demonstrates that the model has impressive performance in most of the classes, that is, DDoS, DoS, Normal, and Probe, with almost perfect precision, recall, and F1-scores, which is due to the high level of training aid that the classes have. On the other hand, the report presents excessive challenges with the minority classes, the Web-Attack class recalls a score of 0.44, which means that over half of the real attacks were not captured, and the U2R class fails completely, with a score of 0.00, which can be explained by a small number of supports the class has (only four instances), which is not enough to make the model internalize the discriminative patterns. This disparity is seen in the aggregate measures, as accuracy and weighted average nominally are perfect, but they are skewed by the high-performing majority classes, and the macro-averaged score of 0.83 provides a more honest assessment of how this model performs heterogeneously across all groups. In addition, the two distinct entries of DDoS, adding to a support of 36,692, concurs the anomaly of duplication of labels as shown in the distribution chart; they should be combined to ensure a single and consistent evaluation of the relevant attack vector.

Table 6. Models size and trainable parameters

Model	No. of Parameters			Size (KB)
Client 1(head)	7968			31.12
Client 2(head)	7968			31.12
Client 3(head)	8032			31.38
Server (tail)	19273			75.29
Classification Report				
	Precision	**Recall**	**F1-Score**	**Support**
BFA	0.98	0.96	0.97	408
BOTNET	0.87	1.00	0.93	54
DDoS	0.99	1.00	0.99	22101
DDoS	1.00	0.98	0.99	14591
DoS	1.00	1.00	1.00	16038
Normal	1.00	1.00	1.00	20556
Probe	1.00	1.00	1.00	29353
U2R	0.00	0.00	0.00	4
Web-Attack	0.90	0.44	0.59	62
Accuracy			1.00	103167
Macro avg	0.86	0.82	0.83	103167
Weighted avg	1.00	1.00	1.00	103167

The results present cutting-edge performance for an IDS. The model is extraordinarily accurate and stable, and it also achieves the SL paradigm, balancing privacy with detection effectiveness. The split between a 2-layer client and a 6-layer server is carefully optimized to defend against and mitigate volumetric attacks (DDoS/DoS), a major threat to SDN controllers. Nevertheless, the principal U2R and Web-Attack findings imply that the "smashed data" sent from the client may lack the granular packet information required to detect low-volume and application-layer attacks. Table 6 shows the models size and number of parameters where the model has been split into three partions (heads) with two layers and single model in the server (tail) with six layers.

6. CONCLUSION

This research demonstrates the effectiveness of an SL paradigm for securing SDN-enabled IoT networks. While providing the SDN controller with a consolidated picture of the network's security posture, the proposed IDS reduces the processing required of individual IoT devices by decoupling the deep learning model across the data and control planes. The viability of distributed feature extraction without compromising detection performance is demonstrated by the model achieving an accuracy of 99.5% while training loss is less than 0.02. Although the system demonstrates state-of-the-art results in recognizing high-traffic threats such as DDoS and Probes, performance variances in low-volume categories, such as Web-Attacks, indicate that additional refinement of the "smashed data" abstraction is required to capture more granular packet information. The overall solution this split-architecture approach offers is scalable, protects users' privacy, and is extremely accurate. It is designed to meet the ever-changing security requirements of complex modern computer networks.

A central SDN controller will be defined mainly as an honest-but-curious one, that is, as an agent that faithfully adheres to the SL protocol but can also be interested in trying to understand sensitive local information about the information it obtains. To add to this, it will be critically discussed in a newly introduced Limitations paragraph that looks at inherent split-learning vulnerabilities. These are activation inversion, where an attacker can reconstruct original input feature of the intermediate layer, and attacks on attribute inference, which attack auxiliary information. It is going to be pointed out that the abstraction of data at high-level feature maps, i.e., the smashed data, is one of the major defense mechanisms. The proposed strategy, as the first feature-extraction layers are placed as close to the IoT devices as possible, allows for adequate transformation of the data exchanged across the SDN IoT network and significantly reduces the threat of attackers gaining access to personal user data or recreating the original patterns of traffic.

REFERENCES

[1] Lee, S., Woo, S., Kim, J., Nam, J., Yegneswaran, V., Porras, P., Shin, S. (2022). A framework for policy inconsistency detection in software-defined networks. IEEE/ACM Transactions on Networking, 30(3): 1410-1423. https://doi.org/10.1109/TNET.2022.3140824

[2] Priyadarsini, M., Bera, P. (2021). Software defined networking architecture, traffic management, security, and placement: A survey. Computer Networks, 192: 108047. https://doi.org/10.1016/j.comnet.2021.108047

[3] Sarker, I.H., Kayes, A.S.M., Badsha, S., Alqahtani, H., Watters, P., Ng, A. (2020). Cybersecurity data science: An overview from machine learning perspective. Journal of Big Data, 7(1): 41. https://doi.org/10.1186/s40537-020-00318-5

[4] Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A.V., Di Franco, F. (2023). The role of machine learning in cybersecurity. Digital Threats: Research and Practice, 4(1): 1-38. https://doi.org/10.1145/3545574

[5] Wu, D., Ullah, R., Harvey, P., Kilpatrick, P., Spence, I., Varghese, B. (2022). Fedadapt: Adaptive offloading for IoT devices in federated learning. IEEE Internet of Things Journal, 9(21): 20889-20901. https://doi.org/10.1109/JIOT.2022.3176469

[6] Ariffin, S.H., Le Chong, J., Latif, N.M.A.A., Abd Malik, N.N.N., Baharudin, M.A., Syed-Yusof, S.K., Yusof, K.M. (2022). Intrusion detection system (IDS) accuracy testing for software defined network Internet of Things

(SDN-IoT) testbed. ELEKTRIKA-Journal of Electrical Engineering, 21(3): 23-27. https://doi.org/10.11113/elektrika.v21n3.361

[7] Elsayed, R.A., Hamada, R.A., Abdalla, M.I., Elsaid, S.A. (2023). Securing IoT and SDN systems using deep-learning based automatic intrusion detection. Ain Shams Engineering Journal, 14(10): 102211. https://doi.org/10.1016/j.asej.2023.102211

[8] Maray, M., Mesfer Alshahrani, H., A Alissa, K., Alotaibi, N., et al. (2022). Optimal deep learning driven intrusion detection in SDN-enabled IoT environment. Computers, Materials & Continua, 74(3): 6587-6604. http://doi.org/10.32604/cmc.2023.034176

[9] Ataa, M.S., Sanad, E.E., El-khoribi, R.A. (2025). Ryu-IDS: Intrusion detection system for modern networks. https://doi.org/10.21203/rs.3.rs-6092259/v1

[10] Wahab, S.A., Sultana, S., Tariq, N., Mujahid, M., Khan, J.A., Mylonas, A. (2025). A multi-class intrusion detection system for DDoS attacks in IoT networks using deep learning and transformers. Sensors, 25(15): 4845. https://doi.org/10.3390/s25154845

[11] Thalapala, V., Reddy, A.K., Guravaiah, K. (2025). MLADAD: Machine learning algorithms analysis on DDoS attack detection in SDN-IoT networks. Procedia Computer Science, 260: 1121-1128. https://doi.org/10.1016/j.procs.2025.03.297

[12] Dhirar, H., Hamad, A. (2025). Comparative evaluation of a novel IDS dataset for SDN-IoT using deep learning models against InSDN, BoT-IoT, and ToN-IoT. Measurement: Digitalization, 4: 100015. https://doi.org/10.1016/j.meadig.2025.100015

[13] Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y., Mounir, S. (2023). A comprehensive survey on SDN security: Threats, mitigations, and future directions. Journal of Reliable Intelligent Environments, 9(2): 201-239. https://doi.org/10.1007/s40860-022-00171-8

[14] Khorseed, W.S., Hamad, A.H. (2024). Inter and intra domain DDoS attack mitigation for software defined network based on Hyperledger Fabric blockchain technology. Ingenierie des Systemes d'Information, 29(1): 301-311. https://doi.org/10.18280/isi.290130

[15] Dhirar, H., Hamad, A. (2025). Internet of things software-defined network intrusion detection dataset: Inter-and intra-domain. Al-Khwarizmi Engineering Journal, 21(3): 35-47. https://doi.org/10.22153/kej.2025.08.001

[16] Pradhan, M., Nayak, C.K., Pradhan, S.K. (2020). Intrusion Detection System (IDS) and their types. In Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications, pp. 481-497. https://doi.org/10.4018/978-1-5225-9866-4.ch026

[17] Dhirar, H., Hamad, A.H. (2025). Federated deep learning intrusion detection system on software defined-network based Internet of Things. IAES International Journal of Artificial Intelligence, 14(4): 3109-3120. http://doi.org/10.11591/ijai.v14.i4.pp3109-3120

[18] Samikwa, E., Di Maio, A., Braun, T. (2024). DFL: Dynamic federated split learning in heterogeneous IoT. IEEE Transactions on Machine Learning in Communications and Networking, 2: 733-752. https://doi.org/10.1109/TMLCN.2024.3409205

[19] Bonachela, J.A., Hinrichsen, H., Munoz, M.A. (2008). Entropy estimates of small data sets. Journal of Physics A: Mathematical and Theoretical, 41(20): 202001. https://doi.org/10.1088/1751-8113/41/20/202001

[20] Abdalah, R.W., Abdulateef, O.F., Hamad, A.H. (2025). A predictive maintenance system based on industrial Internet of Things for multim achine multiclass using a deep neural network. Journal Européen des Systèmes Automatisés, 58(2): 373-381. https://doi.org/10.18280/jesa.580218