Computers & Security 148 (2025) 104109

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

An enhanced Deep-Learning empowered Threat-Hunting Framework for software-defined Internet of Things

Prabhat Kumar [a,*], Alireza Jolfaei [b], A.K.M Najmul Islam [a]

[a] Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta 53850, Finland
[b] Cyber Security and Networking College of Science and Engineering, Flinders University, Adelaide, Australia

ARTICLE INFO

Keywords:
Deep learning
Internet of Things
Intrusion detection system
Software-defined networking

ABSTRACT

The Software-Defined Networking (SDN) powered Internet of Things (IoT) offers a global perspective of the network and facilitates control and access of IoT devices using a centralized high-level network approach called Software Defined-IoT (SD-IoT). However, this integration and high flow of data generated by IoT devices raises serious security issues in the centralized control intelligence of SD-IoT. Motivated by the aforementioned challenges, we present a new Deep-Learning empowered Threat Hunting Framework named DLTHF to protect SD-IoT data and detect (binary and multi-vector) attack vectors. First, an automated unsupervised feature extraction module is designed that combines data perturbation-driven encoding and normalization-driven scaling with the proposed Long Short-Term Memory Contractive Sparse AutoEncoder (LSTMCSAE) method to filter and transform dataset values into the protected format. Second, using the encoded data, a novel Threat Detection System (TDS) using Multi-head Self-attention-based Bidirectional Recurrent Neural Networks (MhSaBiGRNN) is designed to detect cyber threats and their types. In particular, a unique TDS strategy is developed in which each time instances is analyzed and allocated a self-learned weight based on the degree of relevance. Further, we also design a deployment architecture for DLTHF in the SD-IoT network. The framework is rigorously evaluated on two new SD-IoT data sources to show its effectiveness.

1. Introduction

In recent years, the exponential proliferation of smart sensors and demand for inter/intra-connections of devices have promoted the emergence of the Internet of Things (IoT) (Mishra et al., 2020). IoT is primarily used in healthcare, engineering, industry, transportation, and agriculture, requiring a higher degree of protection, robustness, versatility, scalability, and management of networks (He et al., 2021; Soleymani et al., 2022). Moreover, IoT devices are resource-constrained and come with application-specific integrated circuits (*i.e.*, pre-programmed with different complex rules) and, therefore, cannot be pre-programmed with multiple rules (Shafiq et al., 2020). Thus, traditional networks are becoming more sophisticated due to the new application-specific requirements of IoT (Rafique et al., 2020). In order to promote the scalability and flexibility in resource and network management of traditional networks, a new paradigm called Software-Defined IoT (SD-IoT) has been proposed to incorporate Software-Defined Networking (SDN) into IoT (Bera et al., 2017). The SDN has emerged as an essential next-generation networking paradigm. SDN has changed the traditional Internet multifaceted network design by separating the control plane and the data plane (Gadallah et al., 2024). As a result, the control plane has become more intelligent, programmable, and innovative and has access to all components of the underlying SD-IoT network where resources and traffic are effectively managed (Karmakar et al., 2020). Despite the several benefits of SD-IoT, security is the major challenge associated with and further prevents the faster realization of its applications (Singh et al., 2020). Firstly, security issues arise due to the centralized control intelligence of SD-IoT (Nichelini et al., 2023). An attacker can easily launch and capture the centralized controller by performing and introducing sophisticated threats and attacks such as Denial of Service (DoS), Distributed DoS (DDoS), and botnet, and can inject false flow rules and thereby choke the entire network (Shaji et al., 2023), Wu et al. (2021). Therefore, in order to protect functionality, maintain integrity, and protect against the evolving latest cyber threats, an effective security mechanism should be an inherent component of the SD-IoT architecture (Singh et al., 2021).

The vulnerabilities of current cyber defense, including intrusion detection/prevention systems and firewalls, are demonstrated by recent cyber threats, as these systems are mostly designed using heuristic and static attack patterns and, therefore, cannot identify new forms

* Corresponding author.
E-mail addresses: prabhat.kumar@lut.fi (P. Kumar), alireza.jolfaei@flinders.edu.au (A. Jolfaei), najmul.islam@lut.fi (A.K.M Najmul Islam).

https://doi.org/10.1016/j.cose.2024.104109
Received 1 May 2024; Received in revised form 15 August 2024; Accepted 7 September 2024
Available online 13 September 2024

of threats (Al-Hawawreh et al., 2020). Threat Intelligence (TI) involves a proactive defense strategy that gathers information from multiple sources about cyber threats, including context, indicators, consequences, and actionable guidance regarding current or emerging threats. This information is used to improve the process of detecting attacks (Moustafa et al., 2018; Tounsi and Rais, 2018). In order to construct a TI-driven Threat Detection System (TDS), most current solutions have concentrated on statistical, data mining, and Machine Learning (ML) approaches (Mittal et al., 2019). Moreover, the TI model presented in Montasari et al. (2021), Sentuna et al. (2021), Usman et al. (2021), Deliu et al. (2018) has either used a manual labeling process or suffered from high false alarm rate, low detection rate, and lack of generalization capabilities has also made evolving cyber threats challenging to detect using these approaches. In addition, there are still a number of issues that need to be addressed (see also Section 2).

1.1. Motivation

To the best of the authors' knowledge, previous TI solutions are mostly based on ML techniques and have not been applied in the SD-IoT environment. Most of them use manual labeling processes and have a high false alarm rate and low accuracy with a lack of generalization capabilities. Additionally, few cyber security solutions based on DL techniques used control plane data to detect threats. The sampling of these datasets, on the other hand, will take a significant amount of network resources. Thereby, it is still a challenge to design a threat modeling and identification framework for SD-IoT with high accuracy, detection, and low false alarm rate.

1.2. Key contribution

In this paper, we present a new secure network framework for the analysis of emerging multi-vector threats in the SD-IoT network using deep learning techniques. The key contributions are as follows:

- A novel deep-learning empowered threat-hunting framework named "DLTHF" is proposed. The DLTHF is comprised of two primary modules. First, a feature extraction module is designed. In this, we have combined data perturbation-driven encoding and normalization-driven scaling with Long Short-Term Memory Contractive Sparse AutoEncoder (LSTMCSAE) method to filter and convert real data into a new format, and that can also be used in an unsupervised automated manner.
- Second, a DL-based TDS module is proposed using the Multi-head Self-attention-based Bidirectional Recurrent Neural Networks (MhSaBiGRNN) model. A unique TDS strategy is developed in which each time instance is analyzed and allocated a self-learned weight based on the degree of relevance. This allows the model to adaptively focus on the more significant input features in threat detection.
- The proposed DLTHF framework is rigorously evaluated using two different SD-IoT data sources, namely InSDN-Dataset and SDN-Dataset. The performance is compared with RNN-based methods, and the result indicates an increased overall accuracy by 1.5% with low inference time.
- Finally, to mitigate the dynamic and heterogeneous network of SD-IoT, a detailed discussion on the deployment architecture for the proposed "DLTHF" framework in SD-IoT control and data plane is also suggested.

The remainder of the paper is as follows. The background and related work is reviewed in Section 2. The functional components and deployment architecture of the DLTHF framework are presented in Section 3. In Section 4, performance is evaluated, followed by conclusion and future direction in Section 5.

2. Background and related work

In this section, we have first discussed the threat model for SD-IoT and then presented various recent security solutions with their limitations for SDN and IoT networks.

2.1. Threat models in SD-IoT

As a starting point for investigating cyber threats and their intent, the easiest approach to protect SD-IoT is to conduct threat modeling. It is a systematic form of strategic analysis about identifying SD-IoTs' most important assets, their data flows, and boundaries of trust, identifying possible threats and attacks, and designing effective countermeasures for security and threat detection (Al-Hawawreh et al., 2020). Threat modeling can be accomplished by analyzing the entire structure and examining each plane, considering the layered structure of the SD-IoT architecture, which involves an application, control, data, and device plane (Algamdi et al., 2023).

The application plane reflects the SDN application logic and services using mobile and web clients that are used to track, control, and monitor physical IoT devices. Social media interaction, phishing, and web applications are the majority of the threats and attacks associated with this plane (Singh et al., 2021). The control plane is responsible for the management of network-related functionalities and controls the overall network. This plane consists of an SDN controller as a network component. The programmable nature and single-point dependency of an SDN controller make it vulnerable and a potential choice for attackers to launch cyber-attacks. Once the SDN controller is compromised, the entire network becomes vulnerable to various threats and attacks. Man-in-the-middle, control message infinite loop attacks, internal storage misuse attacks, control message drop attacks, and flooding attacks such as DoS and DDoS are attack examples of this plane (Dayal and Srivastava, 2023).

In the layered structure of SD-IoT, the data plane acts as the Internet layer. Devices such as routers and gateways are responsible for data transfer and local data caching by deciding the flow rules, and the controller controls data processing and network access (Algamdi et al., 2023). False data injection, DoS, ransomware, jamming, and control flow attacks are the most significant attacks related to this plane (Mishra et al., 2020). The lowest layer of the SD-IoT layered structure is a device plane, similar to the IoT architecture, which consists of various sensors and actuators. These devices lack individual flows or monitoring mechanisms. However, they are monitored by the centralized controllers that are responsible for distributing flow rules across these devices (Bera et al., 2017). Fake node injection, energy manipulation attack, and malicious code injection are a few common attacks found in both planes (Singh et al., 2020). Considering various recent research on SD-IoT, we identify that control and data segments are the most vulnerable planes. The reason is that the SDN controller manages and controls the entire SD-IoT network (Karmakar et al., 2020). On the other hand, with the help of data plane interaction between the device and other layers or components of SD-IoT is performed (Algamdi et al., 2023). In addition, the majority of these devices have limited computation power and use open access channels, *i.e.*, the Internet, for communication with and no self-security. As a result, these segments are the most attacker-focused, and their effect can impact the working of the entire SD-IoT structure. Therefore, it is the prime objective of this study to provide adequate safety countermeasures for these two segments (Singh et al., 2020).

2.2. Related studies

Various research proposals were suggested to improve the security of the SD-IoT network. For instance, Mishra et al. (2020) surveyed various threats and attacks in SDN and IoT and suggested various solutions using ML and DL. Rafique et al. (2020) reviewed various challenges

Table 1
Summary of state-of-the-art techniques.

Author	Technique used	Application area	Use of SDN-related datasets	Deployment technique	Unsupervised feature selection technique	Technology limitation
He et al. (2021)	Hybrid Variational Bayes Algorithm	IoT	×	×	×	Evaluated the model using obsolete KDD 99 dataset and achieved low accuracy
Al-Hawawreh et al. (2020)	Gated Recurrent Neural Network	IoT-SAGS	×	✓	✓	No analysis performed in terms of time required to predict attacks
Wang et al. (2021)	Probability statistics based Anomaly Detection Model (PADM)	Wireless Sensors Network	×	×	✓	Evaluated the model using dataset with less features and instances and obtained low accuracy
Keshk et al. (2020)	Long Short-Term Memory	Smart Power Networks	×	×	✓	Expensive in terms of time to detect unknown attacks
Assis et al. (2020)	Gated Recurrent Units	SD-IoT	×	✓	×	Evaluated only in binary-class scenario and used non-SDN datasets for SD-IoT ecosystem

and attacks on the centralized control plane of SDN. Similarly, authors in Bera et al. (2017), Karmakar et al. (2020), Mittal et al. (2019), Wang et al. (2021) surveyed various architectures, applications, and ML and DL-based TI schemes. Although several studies (Al-Hawawreh et al., 2020; Keshk et al., 2020; Bilal and Pack, 2019) for threat hunting for SDN, IoT and its applications have been designed and implemented, it is still a challenge to develop an efficient and reliable TDS with encoded data that retains high accuracy and detection rate. For instance, Moustafa et al. (2018) proposed a TI scheme for CPS, Cloud, Fog computing and IoT based on beta mixture-hidden Markov models. The performance was evaluated for power systems by using the UNSW-NB15 datasets. Zhang et al. (2019) suggested a TDS for social IoT. In order to acquire threat information relevant to the malicious activity of target accounts, the method uses the Support Vector Machine (SVM) to evaluate qualitative data in TI to predict malicious account behavior. Assis et al. (2020) introduced an SDN security system designed to enhance network security through two main components: threat detection and mitigation. This system employs the Gated Recurrent Units (GRU) deep learning technique to analyze single IP flow records for timely detection of cyber threats. Kumar et al. (2021) presented a privacy-preserving driven attack system in SDIoT-Fog. This model uses Sparse AutoEncoder (SAE) for privacy-preservation and Deep Neural Network (DNN) to detect and evaluate the effectiveness of the suggested privacy-preservation scheme. Alshra'a and Jochen (2021) used a One-Dimensional Convolutional Neural Network (1D-CNN) to protect the SDN controller. He et al. (2021) developed a lightweight variational Bayes algorithm aimed at detecting DDoS attacks within IoT networks. Their approach was validated using the KDD99 dataset, demonstrating its effectiveness in a controlled environment. Al-Hawawreh et al. (2020) proposed a Threat Intelligence (TI) framework that incorporates deep learning to advance threat detection capabilities. This framework includes a TI-Deep Pattern Extractor, which utilizes a deep sparse auto-encoder, and a Threat Detection System (TDS) based on Gated Recurrent Neural Networks (GRNN). Noor et al. (2018) detailed a machine learning-based system that processes tactics, techniques, and procedures (TTPs). This system utilizes information gain for feature selection to highlight prominent TTPs, which are then analyzed using Association Rule Mining (ARM) for threat detection. Table 1 summarizes the limitations of these advanced techniques, offering insights into potential areas for improvement in the field of network security.

To the best of authors knowledge, previous TI solutions are mostly based on ML techniques and have not been applied in SD-IoT environment. Most of them uses manual labeling process and have high false alarm rate and low accuracy with lack of generalization capabilities. Additionally, few cyber security solutions based on DL techniques used control plane data to detect threats. The sampling of these datasets, on the other hand, will take a significant amount of network resources. Thereby, it is still a challenge to design a threat modeling and identification framework for SD-IoT with high accuracy, detection and low false alarm rate.

3. The proposed DLTHF framework

In this section, we discuss the proposed feature extraction and threat detection modules. The proposed framework is shown in Fig. 1, and its working is explained below.

3.1. Proposed feature extraction module

In this module, a novel approach to feature extraction is introduced. It involves combining a data perturbation-driven encoding (PdE) technique, which is essentially label encoding, and a normalization-driven scaling (NdS) method, which employs min–max scaling. These techniques are integrated with an unsupervised generative deep learning architecture known as Long Short-Term Memory Contractive Sparse AutoEncoder (LSTMCSAE). The proposed approach allows an automated unsupervised feature extraction mechanism to obtain better and more resilient low-dimensional features from raw network data sources. We have explained the preliminaries and the proposed module below: Long Short-Term Memory (LSTM) is a deep learning technique designed to identify and utilize long-term relationships in data. This method generates its outputs by integrating data from previous timestamps with current inputs. Structurally, LSTM comprises layers of neurons adept at retaining information over extended periods, which enables them to grasp long-term data dependencies effectively. LSTMs selectively retain or discard past information using three specialized gates: the input gate, the forget gate, and the output gate. These gates manage the flow of information by allowing new data to enter, removing irrelevant data, and producing the final output, respectively. A concise overview of how these gates operate is provided below (Keshk et al., 2020).

$$\mathcal{F}_{\mathcal{T}} = \sigma\left(\mathcal{X}_{\mathcal{T}}\mathcal{W}_{\mathcal{F}} + \mathcal{W}_{\mathcal{F}}\mathcal{H}_{\mathcal{T}-1} + \mathcal{B}_{\mathcal{F}}\right), \tag{1}$$

$$\mathcal{Z}_{\mathcal{T}} = \tanh\left(\mathcal{X}_{\mathcal{T}}\mathcal{W}_{\mathcal{Z}} + \mathcal{W}_{\mathcal{Z}}\mathcal{H}_{\mathcal{T}-1} + \mathcal{B}_{\mathcal{Z}}\right), \tag{2}$$

$$\mathcal{C}_{\mathcal{T}} = \mathcal{Z}_{\mathcal{T}} \odot I_{\mathcal{T}} + \mathcal{C}_{\mathcal{T}-1} \odot \mathcal{F}_{\mathcal{T}}, \tag{3}$$

$$I_{\mathcal{T}} = \sigma\left(\mathcal{X}_{\mathcal{T}}\mathcal{W}_{I} + \mathcal{W}_{I}\mathcal{H}_{\mathcal{T}-1} + \mathcal{B}_{I}\right), \tag{4}$$

$$O_{\mathcal{T}} = \sigma\left(\mathcal{X}_{\mathcal{T}}\mathcal{W}_{O} + \mathcal{W}_{O}\mathcal{H}_{\mathcal{T}-1} + \mathcal{B}_{O}\right), \tag{5}$$

$$\mathcal{H}_{\mathcal{T}} = \tanh\left(\mathcal{C}_{\mathcal{T}}\right) \odot O_{\mathcal{T}}. \tag{6}$$

where, the previous hidden state is denoted by $\mathcal{H}_{\mathcal{T}-1}$ and the current input by $\mathcal{X}_{\mathcal{T}}$, where the subscripts $\mathcal{T}$ and $\mathcal{T}-1$ represent the current and previous time steps, respectively. The model includes various elements: $\mathcal{D}$ is the input vector, $\mathcal{H}$ is the output vector, $\mathcal{F}$, $\mathcal{I}$, and $\mathcal{O}$ denote the forget, input, and output gates, respectively, and $\mathcal{C}$ stands for the cell state. Operations within the model, such as element-wise multiplications, are carried out using the Hadamard product, represented by $\odot$, while the sigmoid function is symbolized by σ. The model's weight matrices are $\mathcal{W}_{\mathcal{F}}$, $\mathcal{W}_{\mathcal{Z}}$, $\mathcal{W}_{\mathcal{I}}$, and $\mathcal{W}_{\mathcal{O}}$, with corresponding biases $\mathcal{B}_{\mathcal{F}}$, $\mathcal{B}_{\mathcal{Z}}$, $\mathcal{B}_{\mathcal{I}}$, and $\mathcal{B}_{\mathcal{O}}$, which facilitate the gates' and cell state's functionality in processing data over time.

(1) Contractive Sparse AutoEncoder (CSAE): The CSAE is an unsupervised method and one of the types of AutoEncoder (AE). The AE first consist an encoder to convert the input $\mathcal{X}_{\mathcal{T}}$ into a hidden form $\mathcal{Y}_{\mathcal{T}}$ via a deterministic affine transformation matrix with nonlinearity and a decoder that uses the variable $\mathcal{Y}_{\mathcal{T}}$ from encoder to recreate the output $\widehat{\mathcal{X}_{\mathcal{T}}}$. The operations of the encoder and decoder are shown below:

$$\mathcal{Y}_{\mathcal{T}} = \mathcal{F}\left(\mathcal{W}_1 \mathcal{X}_{\mathcal{T}} + \mathcal{B}_1\right) \tag{7}$$

$$\widehat{\mathcal{X}_{\mathcal{T}}} = \mathcal{F}'\left(\mathcal{W}_2 \mathcal{Y}_{\mathcal{T}} + \mathcal{B}_2\right) \tag{8}$$

where, $\mathcal{W}_1$ represents the weight matrix that connects the input vector $\mathcal{X}_{\mathcal{T}}$ to the hidden representation $\mathcal{Y}_{\mathcal{T}}$, while $\mathcal{W}_2$ denotes the weights connecting the hidden representation $\mathcal{Y}_{\mathcal{T}}$ to the reconstructed output $\widehat{\mathcal{X}}_{\mathcal{T}}$. Similarly, $\mathcal{B}_1$ and $\mathcal{B}_2$ are the bias terms for these transformations, respectively. The reconstructed output $\widehat{\mathcal{X}}_{\mathcal{T}}$ serves as the AE's approximation of the original input $\mathcal{X}_{\mathcal{T}}$. The AE is trained to optimize these parameters, specifically $\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}$, and $\widehat{\mathcal{B}_2}$, in order to minimize the reconstruction error. This optimization process involves adjusting the parameters to reduce the cost function tailored for the given training set.

$$\begin{aligned}\left\{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}\right\} &= \underset{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}}{\operatorname{argmin}} \left\{\mathcal{L}_{AE}(\mathcal{W}, \mathcal{B})\right\} \\ &= \underset{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}}{argmin} \left\{ \frac{1}{2\mathcal{N}} \sum_{\mathcal{T}=1}^{\mathcal{N}} L\left(\mathcal{X}_{\mathcal{T}}, \widehat{\mathcal{X}_{\mathcal{T}}}\right) \right. \\ &\qquad \left. + \frac{\lambda}{2} \sum_{r=1}^{2} \| \mathcal{X}_{\mathcal{T}} \|_F^2 \right\}\end{aligned} \tag{9}$$

where $\mathcal{N}$ indicates total number of training samples, loss function is denoted by $L\left(\mathcal{X}_{\mathcal{T}}, \widehat{\mathcal{X}_{\mathcal{T}}}\right)$ and λ is regularization term used for model generalization. The loss function is mainly calculated using square error or cross-entropy functions. Further, we can add a sparsity term to the AE loss function. This step will make AE sparse, *i.e.*, Sparse AutoEncoder (SAE) and its loss function is calculated as follows:

$$\mathcal{L}_{SAE}(\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}) = \mathcal{L}_{AE}(\mathcal{W}, \mathcal{B}) + \eta \sum_{J=1}^{M} KL\left(\varrho \parallel \hat{\varrho}_J\right) \tag{10}$$

where the sparse penalty item's weight is determined by the constant η and is taken as small as 0.05. The hidden layers in SAE is suppressed by using KL divergence loss function and is determined as:

$$\sum_{J=1}^{M} KL\left(\varrho \parallel \hat{\varrho}_J\right) = \sum_{J=1}^{M} \left\{ (1-\varrho) \log\left(\frac{1-\varrho}{1-\hat{\varrho}}\right) + \varrho \log \frac{\varrho}{\hat{\varrho}_j} \right\} \tag{11}$$

The mean activation value of the jth neuron in the hidden layer during training is denoted as $\hat{\varrho}_J$, and it is associated with the sparsity parameter, represented by ϱ. Furthermore, the cost function of the Contractive Sparse AutoEncoder (CSAE) is expounded upon in Kumar et al. (2022). Finally, an explicit regularizer for the input data in the discussed cost function, denoted as $\mathcal{X}_{\mathcal{T}}$, is the Jacobian matrix $\mathcal{J}_{\mathcal{F}}\left(\mathcal{X}_{\mathcal{T}}\right)$, aiding in reducing the model's sensitivity to minor changes in the input data. This regularization is limited to the model training phase and does not affect the operational behavior of the network. Additionally, the

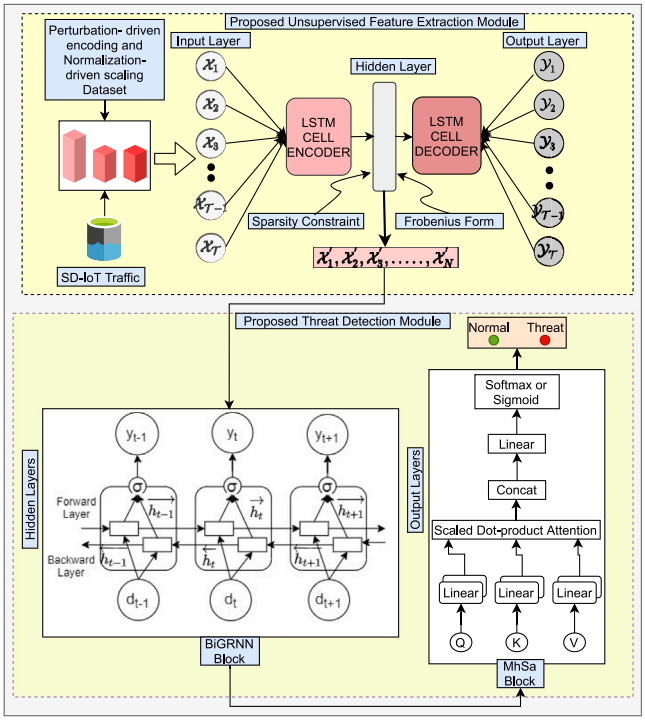


Fig. 1. The proposed DLTHF framework.

cost function of the Contractive Sparse AutoEncoder (CSAE) is detailed in Kumar et al. (2022).

$$\begin{aligned}\left\{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}\right\} &= \underset{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}}{\operatorname{argmin}} \left\{\mathcal{L}_{CSAE}(\mathcal{W}, \mathcal{B})\right\} \\ &= \underset{\widehat{\mathcal{W}_1}, \widehat{\mathcal{W}_2}, \widehat{\mathcal{B}_1}, \widehat{\mathcal{B}_2}}{argmin} \left\{ \frac{1}{2\mathcal{N}} \sum_{\mathcal{T}=1}^{\mathcal{N}} L\left(\mathcal{X}_{\mathcal{T}}, \widehat{\mathcal{X}_{\mathcal{T}}}\right) \right. \\ &\quad + \frac{\lambda}{2} \sum_{r=1}^{2} \| \mathcal{J}_{\mathcal{F}}\left(\mathcal{X}_{\mathcal{T}}\right) \|_F^2 \\ &\quad \left. + \eta \sum_{J=1}^{M} KL\left(\varrho \parallel \hat{\varrho}_J\right) \right\}\end{aligned} \tag{12}$$

where $\| \mathcal{J}_{\mathcal{F}}\left(\mathcal{X}_{\mathcal{T}}\right) \|_{\mathcal{F}}^2$ represents frobenius norm.

(2) Long Short-Term Memory Contractive Sparse AutoEncoder (LSTM-CSAE)

The LSTMCSAE model integrates the strengths of LSTM for temporal data processing with the CSAE's capabilities in dimensionality reduction and regularization. The approach combines LSTM outputs with CSAE encoding and decoding mechanisms.

- **LSTM to CSAE Integration** The LSTM module outputs a hidden state $\mathcal{H}_{\mathcal{T}}$ at each timestep $\mathcal{T}$, which is fed as the input to the CSAE encoder:

$$\mathcal{Y}_{\mathcal{T}} = \sigma\left(\mathcal{W}_{\text{enc}} \mathcal{H}_{\mathcal{T}} + \mathcal{B}_{\text{enc}}\right) \tag{13}$$

  where σ is the sigmoid activation function, $\mathcal{W}_{\text{enc}}$ are the weights of the encoder, and $\mathcal{B}_{\text{enc}}$ is the bias of the encoder.
- **CSAE Encoding and Decoding** The encoder transforms the LSTM output into a lower-dimensional representation, which the decoder then attempts to reconstruct back to the original output:

$$\widehat{\mathcal{H}_{\mathcal{T}}} = \sigma\left(\mathcal{W}_{\text{dec}} \mathcal{Y}_{\mathcal{T}} + \mathcal{B}_{\text{dec}}\right) \tag{14}$$

  where $\mathcal{W}_{\text{dec}}$ are the weights of the decoder, and $\mathcal{B}_{\text{dec}}$ is the bias of the decoder.

Algorithm 1 Proposed LSTMCSAE algorithm for feature extraction.

procedure LSTMCSAE(Unlabeled_dataset $\mathcal{X}=\{X_i\}_{i=1}^{\mathcal{N}}$, $Epochs = \mathcal{E}p$)
  Perform $\mathcal{PdE}$ on $\mathcal{X}$.
  Perform $\mathcal{NdS}$ on $\mathcal{X}$.
  Initialize $\mathcal{E}p$ =1 and $\mathcal{E}p_{max}$ in LSTMCSAE.
  for $\mathcal{E}p = 1 \rightarrow \mathcal{E}p_{max}$ **do**
    Perform encoding using LSTM Eqs. (1) to (6) and compute $\mathcal{H}$, $\mathcal{F}$, $\mathcal{I}$, $\mathcal{C}$, and $\mathcal{O}$ using BPTT.
    Call CSAE on $\mathcal{X}$.
    Encode $\mathcal{X}$ using variable $\mathcal{Y}_{\mathcal{T}}$ of Eq. (7).
    Update gradients g to update encoders weight.
    Add η using Eq. (10).
    Compute KL divergence loss function using Eq. (11).
    Compute $\| \jmath_{\mathcal{F}}(\mathcal{X}_{\mathcal{T}}) \|_F^2$ using Eq. (12).
    Compute the cost function of CSAE using Eq. (12).
    Minimize $\widehat{\mathcal{W}_{\text{enc}}}, \widehat{\mathcal{W}_{\text{dec}}}, \widehat{\mathcal{B}_{\text{enc}}}, \widehat{\mathcal{B}_{\text{dec}}}$ using $Adam$ optimizer.
  end for
return Extracted feature set $\widehat{\mathcal{X}}' = \{\mathcal{X}_i'\}_{i=1}^{\mathcal{N}}$.
end procedure

- **Combined Loss Function** The loss function for LSTMCSAE integrates terms for reconstruction error, regularization through the contractive Jacobian, and sparsity through KL divergence:

$$\mathcal{L}_{\text{LSTMCSAE}} = \sum_{\mathcal{T}=1}^{\mathcal{N}} \left(\| \mathcal{X}_{\mathcal{T}} - \widehat{\mathcal{X}}_{\mathcal{T}} \|_2^2 + \lambda \| \jmath_{\mathcal{F}}(\mathcal{Y}_{\mathcal{T}}) \|_F^2 + \eta \text{KL}(\varrho \parallel \hat{\varrho}) \right) \quad (15)$$

  where, $\mathcal{X}_{\mathcal{T}}$ and $\widehat{\mathcal{X}}_{\mathcal{T}}$ are the original and reconstructed data at time $\mathcal{T}$. λ and η are regularization coefficients. $\jmath_{\mathcal{F}}(\mathcal{Y}_{\mathcal{T}})$ represents the Jacobian matrix of the encoder's output with respect to its input, emphasizing sensitivity to input variations. $\text{KL}(\varrho \parallel \hat{\varrho})$ is the KL divergence for sparsity enforcement on the activations within the autoencoder.
- **Optimization Process** The parameters of the LSTMCSAE model are optimized using Back Propagation Through Time (BPTT) and the Adam optimizer:

$$\{\mathcal{W}_{\text{enc}}, \mathcal{W}_{\text{dec}}, \mathcal{B}_{\text{enc}}, \mathcal{B}_{\text{dec}}\} = \underset{\mathcal{W}_{\text{enc}}, \mathcal{W}_{\text{dec}}, \mathcal{B}_{\text{enc}}, \mathcal{B}_{\text{dec}}}{\text{argmin}} \mathcal{L}_{\text{LSTMCSAE}} \quad (16)$$

  The functionality of the proposed LSTMCSAE approach is mentioned in Algorithm 1 .

3.2. Proposed threat detection system module

The extracted features are used by the proposed DL-based Threat Detection System (TDS) module. The TDS model is designed using the Multi-head Self-attention-based Bidirectional Recurrent Neural Networks (MhSaBiGRNN) model. This method enables the model to adaptively focus on the more important input properties in threat detection. Each time an instance is examined, a self-learned weight based on the degree of relevance is assigned, and this process is repeated. The working of the proposed LSTMCSAE approach is mentioned in Algorithm 2. The preliminaries for our approach are explained below:

(1) Bidirectional Recurrent Neural Networks (BiRNN): The cyclic connections in Recurrent Neural Networks (RNNs) are used to extract contextual information from sequential input. Given the extracted feature set from LSTMCSAE module *i.e.*, $\widehat{\mathcal{X}}' = \{\mathcal{X}_i'\}_{i=1}^{\mathcal{N}}$ *i.e.*, $\widehat{\mathcal{X}}' = (\mathcal{X}_1', \mathcal{X}_2' \dots, \mathcal{X}_{\mathcal{N}}')$ from timestep $\mathcal{T} = 1, 2, \dots \mathcal{N}$, the RNN updates the current hidden state $\mathcal{H}_{\mathcal{T}}$ by using the current time information $\widehat{\mathcal{X}}_{\mathcal{T}}'$ and the previous hidden state $\mathcal{H}_{\mathcal{T}-1}$, as follows (Popoola et al., 2021);

$$\mathcal{H}_{\mathcal{T}} = \forall \left(\mathcal{W}_{\widehat{\mathcal{X}}'\mathcal{H}} \widehat{\mathcal{X}}_{\mathcal{T}}' + \mathcal{W}_{\mathcal{H}\mathcal{H}} \mathcal{H}_{\mathcal{T}-1} + \mathcal{B}_{\mathcal{H}} \right), \quad (17)$$

where Å, $\mathcal{W}_*$ and $\mathcal{B}_{\mathcal{H}}$ denotes activation function, weight matrix, and bias term of the hidden layer, respectively. Due to the RNN's unidirectional construction, it is only able to retrieve the past context of each time step. The BiRNN, on the other hand, employs two distinct hidden layers to enable complete access to both the sequence's prior and subsequent contexts in both a forward and a backward direction. The bidirectional process can be formulated as:

$$\overrightarrow{\mathcal{H}_{\mathcal{T}}} = \forall \left(\overrightarrow{\mathcal{W}_{\widehat{\mathcal{X}}'\overrightarrow{H}}} \widehat{\mathcal{X}}_{\mathcal{T}}' + \overrightarrow{\mathcal{W}_{\mathcal{H}\overrightarrow{\mathcal{H}}}} \overrightarrow{\mathcal{H}_{\mathcal{T}-1}} + \overrightarrow{\mathcal{B}_{\mathcal{H}}} \right), \quad (18)$$

$$\overleftarrow{\mathcal{H}_{\mathcal{T}}} = \forall \left(\overleftarrow{\mathcal{W}_{\widehat{\mathcal{X}}'\overleftarrow{H}}} \widehat{\mathcal{X}}_{\mathcal{T}}' + \overleftarrow{\mathcal{W}_{\mathcal{H}\overleftarrow{\mathcal{H}}}} \overleftarrow{\mathcal{H}_{\mathcal{T}+1}} + \overleftarrow{\mathcal{B}_{\mathcal{H}}} \right), \quad (19)$$

where, $\rightarrow$, $\leftarrow$stand for the forward and backward processes without contact. Where: Årepresents the activation function, typically a sigmoid or tanh function, applied to the linear combinations of inputs and states. $\overrightarrow{\mathcal{W}_{\widehat{\mathcal{X}}'\overrightarrow{H}}}$ and $\overleftarrow{\mathcal{W}_{\widehat{\mathcal{X}}'\overleftarrow{H}}}$ are the forward and backward weight matrices for the connections from input features to the hidden states. $\overrightarrow{\mathcal{W}_{\mathcal{H}\overrightarrow{H}}}$ and $\overleftarrow{\mathcal{W}_{\mathcal{H}\overleftarrow{H}}}$ are the forward and backward weight matrices for the connections between hidden states at consecutive time steps. $\overrightarrow{\mathcal{B}_{\mathcal{H}}}$ and $\overleftarrow{\mathcal{B}_{\mathcal{H}}}$ are the bias vectors for the hidden states in the forward and backward directions, respectively. $\widehat{\mathcal{X}}_{\mathcal{T}}'$ denotes the input vector at time step $\mathcal{T}$. These hidden states are combined using the following equation:

$$\mathcal{H}_{\mathcal{T}} = \overrightarrow{\mathcal{H}_{\mathcal{T}}} \bigoplus \overleftarrow{\mathcal{H}_{\mathcal{T}}} \quad (20)$$

(2) Multi-head Self-attention (MhSa): The MhSa approach enables a model to efficiently recover the dependencies in the input sequence while simultaneously focusing on data in several representation subspaces at various points. This method creates representation subspaces by splitting the model into heads. Its many heads enable it to gather information from various angles and include various viewpoints in order to comprehend the input sequence. Given the inputs queries $\mathcal{Q}$, keys $\mathcal{K}$, and values $\mathcal{V}$, this procedure is formalized as follows (Jin et al., 2020):

$$MhSa(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = Concat\left(head_1, head_2, \dots, head_{\mathcal{H}}\right) \mathcal{W}^{\mathcal{O}} \quad (21)$$

$$head_{\mathcal{I}} = Attention\left(\mathcal{Q}\mathcal{W}_{\mathcal{I}}^{\mathcal{Q}}, \mathcal{K}\mathcal{W}_{\mathcal{I}}^{\mathcal{K}}, \mathcal{V}\mathcal{W}_{\mathcal{I}}^{\mathcal{V}}\right) \quad (22)$$

$$Attention(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = Softmax\left(\frac{\mathcal{Q}\mathcal{K}^{\mathcal{T}}}{\sqrt{d_{\mathcal{K}}}}\right)\mathcal{V} \quad (23)$$

where $\mathcal{W}_{\mathcal{I}}^{\mathcal{Q}} \in \mathcal{R}^{d_{model} \times d_k}$, $\mathcal{W}_{\mathcal{I}}^{\mathcal{K}} \in \mathcal{R}^{d_{model} \times d_k}$, $\mathcal{W}_{\mathcal{I}}^{\mathcal{V}} \in \mathcal{R}^{d_{model} \times d_v}$ denotes trainable parameters and the dimensions of the keys and the values are represented by d_k, d_v, respectively.

(3) Multi-head Self-attention-based Bidirectional Recurrent Neural Networks (MhSaBiGRNN) model: The proposed TDS module is designed based on the MhSaBiGRNN technique that identifies the threat patterns in the SD-IoT traffic using the intelligence provided by the LSTMCSAE method. The mathematical processes (Eqs. (17) to (23)) are repeated in multiple timesteps by the proposed MhSaBiGRNN technique on the extracted patterns (*i.e.*, depending on the output). In order to perform binary classification, the output layer uses a sigmoid function to decide the action that can be taken $\widehat{\mathcal{Y}}$ in relation to the sequence of patterns $\widehat{\mathcal{X}}'$. For this, we compute the loss $\mathcal{L}$ for $\mathcal{S}$ batch size between actual $\mathcal{Y}$ and predicted outputs $\widehat{\mathcal{Y}}$.

$$\mathcal{L}(\mathcal{Y}, \widehat{\mathcal{Y}}) = \frac{1}{\mathcal{S}} \sum_{i=1}^{\mathcal{S}} \left(\widehat{\mathcal{Y}^i} - \mathcal{Y}^i \right)^2 \quad (24)$$

In addition to the binary classification task, we have also designed the proposed TDS to identify different attack types in the SD-IoT network. This is done by employing a softmax function to the BiGNN output layer. Let us assume that we obtain $\widehat{\mathcal{X}}' = \{\mathcal{X}_i'\}_{i=1}^{\mathcal{N}}$ *i.e.*, $\widehat{\mathcal{X}}' = (\mathcal{X}_1', \mathcal{X}_2' \dots, \mathcal{X}_{\mathcal{N}}')$ as input from feature extraction module with $\mathcal{N}$ samples. Then, we express the training set as, $\{\mathcal{X}_1', \mathbb{C}_1\}, \{\mathcal{X}_2', \mathbb{C}_2\}, \dots \{\mathcal{X}_{\mathcal{N}}', \mathbb{C}_{\mathcal{K}}\}$, where label $\mathbb{C}_i \in \{1, 2, \dots, k\}$. The softmax function is calculated as,

$$\mathcal{J} = -\sum_{i=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{K}} I\left(\mathbb{C}_i = j\right) \log \left(\frac{e^{\mathbb{C}_i w_j + b_j}}{\sum_{j=1}^{\mathcal{K}} e^{\mathbb{C}_i w_j + b_j}} \right) \quad (25)$$

Algorithm 2 Proposed MhSaBiGRNN algorithm for threat detection

Input: Training set: $\widehat{X}' = \{X_i'\}_{i=1}^{\mathcal{N}}$; Testing set X; $Epochs = \mathcal{E}p$; batch size $\mathcal{S}$; class label $\mathbb{C}_i \in \{1, 2, \ldots, k\}$.
Initialization $\mathcal{E}p$ =1, $\mathcal{E}p_{max}$ and $\forall : \mathcal{T} \in [1, 2, \ldots, \mathcal{N}]$.
Output: Predicted threat values of X.
repeat
  for $\mathcal{E}p = 1 \rightarrow \mathcal{E}p_{max}$ **do**
    for Number of $\mathcal{S}$ **do**
      for $\mathcal{T} = 1 \rightarrow \mathcal{N}$ **do**
        RNN computes $(\overrightarrow{\mathcal{H}_1}, \overrightarrow{\mathcal{H}_2}, \ldots, \overrightarrow{\mathcal{H}_{\mathcal{T}-1}})$ via forward pass mentioned in Eq. (14).
      end for
      for $\mathcal{T} = \mathcal{N} \rightarrow 1$ **do**
        RNN computes $(\overleftarrow{\mathcal{H}_1}, \overleftarrow{\mathcal{H}_2}, \ldots, \overleftarrow{\mathcal{H}_{\mathcal{T}+1}})$ via backward pass mentioned in Eq. (15).
      end for
      The hidden sates are combined to retrieve BiRNN using Eq. (16).
      Hidden layer of BiRNN is obtained.
      The MhSa layer assigns corresponding weights for time instances according to the Eqs. (17)–(19).
      Compute $\mathcal{L}$ using Eq. (20) for binary classification.
      Use *Adam* as convergence optimizer.
    end for
  end for
until Convergence of Eq. (20) is reached
Input X to trained MhSaBiGRNN model.
while *True* **do**
  if $\mathbb{C}$ == 0 **then**
    return Threat
  else
    return Normal
  end if
end while
repeat
To perform Multiclassification ▷ This is comment
  Add Softmax layer using Eqs. (21) and (22).
  Compute $\mathcal{L}$ using Eq. (23) for multi-classification.
  Use *Adam* as convergence optimizer.
until Convergence of Eq. (23) is reached
Input X to trained MhSaBiGRNN model.
while *True* **do**
  Predict the value of $\mathbb{C}$
  return *Threat Type*
end while

where the boolean function is denoted by $I\left(\phi_i = j\right)$ and it returns the value 1 for all $\mathbb{C}_i = j$; otherwise it returns 0. w_j and b_j denotes the weight vector and threshold value for the jth class. According to the underlying assumption of the loss function $\mathcal{J}$, if the sample d_i relates to the jth class, the value of $\mathbb{C}_i w_j + b_j$ should be high, increasing the likelihood that the sample $\mathbb{C}_i$ denotes the jth class. Given the intricacy of the formula above, the derivative expression could be challenging. Thus the matrix formula for the softmax function is written as follows:

$$\mathcal{J} = -\sum_{i=1}^{\mathcal{N}} \log\left(\frac{\exp\left(\mathbb{C}_i'\mathcal{W}\right)\mathcal{Y}_i^{\mathcal{T}}}{\exp\left(\mathbb{C}_i'\mathcal{W}\right)\vec{1}^{\mathcal{T}}}\right) \tag{26}$$

In order to satisfy the equivalent threshold in the above Eq. (26), $\mathbb{C}_i' = [\mathbb{C}_i, 1]$ pads a value 1. $\vec{1} \in \mathbb{R}^{1\times k}$ is the row vector and all its value corresponds to 1. The one hot encoder vector form is $\mathcal{Y}_i$. The $\exp\left(\mathbb{C}_i'W\right)$ is represented by $e^{\mathbb{C}_i'W}$. Finally, we employ $\mathbb{C}$-way cross-entropy loss, which provides the probability across $\mathbb{C}$ class labels and calculates the loss for each prediction at each timestamp using the below formula:

$$\mathcal{L} = \frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}}\sum_{\mathbb{C}=1}^{\mathbb{C}} \mathcal{Y}_{i\mathbb{C}} \log\left(\widehat{\mathcal{Y}}_{i\mathbb{C}}\right) \tag{27}$$

Table 2
Sample distributions of InSDN-Dataset.

Class	Type	InSDN-Dataset	
		Training	Testing
Normal	Normal	48045	20 379
Attack	DoS	36789	15 682
	DDoS	33864	14 549
	Probe	25246	11 126
	Brute-Force Attack (BFA)	798	312
	Web-Attack	139	53
	Botnet	121	43

Table 3
Sample distributions of SDN-Dataset.

Class	Type	SDN-Dataset	
		Training	Testing
Normal	Normal	24679	10 321
Attack	DoS	24362	10 638
	DDoS	24476	10 524
	Port Scanning	24641	10 359
	OS fingerprinting	24430	10 570
	Fuzzing	24412	10 588

where batch size is denoted with $\mathcal{N}$, $\mathbb{C}$ is number of classes, $\mathcal{Y}$ is actual and $\widehat{\mathcal{Y}}$ is predicted class labels.

3.3. New deployment architecture for proposed framework

The systematic deployment architecture for the proposed framework includes four main components; (*a*) Device plane, (*b*) Data plane, (*c*) Control plane, and (*d*) Virtualized Northbound applications, as illustrated in Fig. 2.

(a) *Device plane*: It consists of various IoT devices (*i.e.*, the combination of sensor and actuator) that generates a huge amount of data in real time (Bera et al., 2017). IoT offloads their computational intensive tasks to nearby fog computing devices for processing and storage in an SD-IoT network.

(b) *Data plane*: It handles the infrastructure of the physical network, including fog nodes, routers, switches, access points, and other SDN agents. Fog nodes are powerful nodes that include data analysis servers, industrial computers, fog-computing servers, and so on. Fog nodes in the proposed deployment architecture resolve the resource-limitation problem in IoT (Rafique et al., 2020). In SD-IoT, the controller manages these devices using the Control-Data Plane Interface (C-DPI), *i.e.*, OpenFlow protocols (Karmakar et al., 2020). The proposed DLTHF framework can be installed as Software as a Service (SaaS) at each choke point of the network. Thus, in fog devices and other applications, the network traffic between IoT devices and applications can be positively monitored and examined.

(c) *Control plane*: It is an intelligent, centralized entity having visibility over its network domain. SDN Controller is responsible for making the network's routing decisions (Mishra et al., 2020). Thus, in SDN, decisions are made remotely instead of on each individual router. The southbound Application Programming Interfaces (APIs) are configured to allow connectivity between SDN-enabled switches of the data plane and the SDN Controller. In addition, any commercial SDN controller, such as OpenDaylight, Floodlight, POX, etc., can be customized and can act as an expanded module of our implemented DSNF framework.

(d) *Virtualized Northbound applications*: For a logically unified controller to make coordinated decisions, it includes several user applications that communicate to the controller to obtain abstraction using the northbound API protocol.

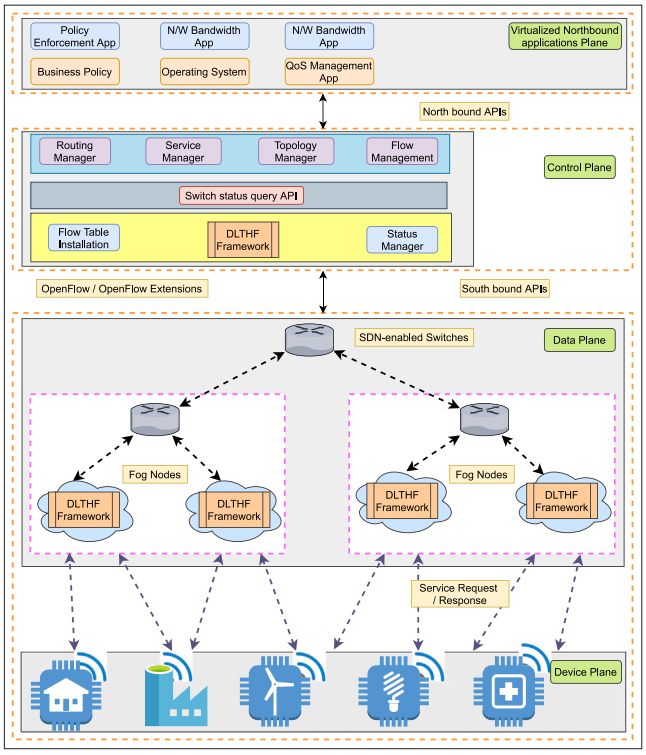


Fig. 2. Deployment architecture of DLTHF framework in SD-IoT.

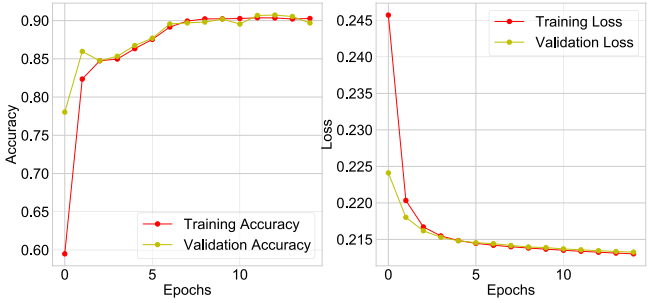


Fig. 3. Comparative Analysis of Accuracy and Loss for the LSTMCSAE Technique on the InSDN-Dataset (Binary).

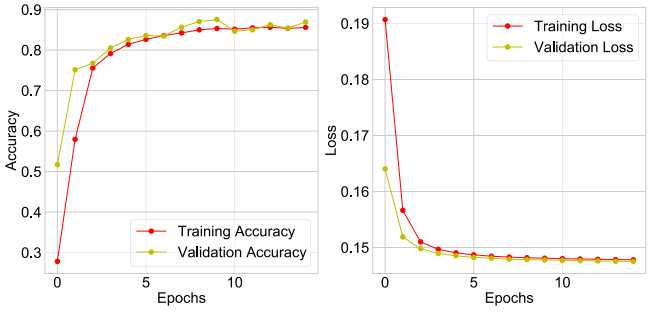


Fig. 4. Comparative Analysis of Accuracy and Loss for the LSTMCSAE Technique on the InSDN-Dataset (Binary).

4. Experimental study and result discussion

4.1. Datasets description

To evaluate the performance of the proposed DLTHF framework, we utilized two distinct open sources of SDN datasets: namely InSDN-Dataset (Elsayed et al., 2020; Elsayed, 2020) and SDN-Dataset (Sarica and Angin, 2020; Sarica, 2020). Both datasets can be used in binary and multi-class threat detection tasks. The InSDN-Dataset has 83 features and 1 output class with different attacks. It contains various attacks that are commonly found in SDN-IoT environments, such as DoS, DDoS, probe, brute-force attacks, web attacks, and botnet attacks. The distribution of attack and normal instances are shown in Table 2. Similarly, the SDN-Dataset has 32 features and 1 output class with different attack groups. It also contains various attacks that are commonly found in SDN-IoT environments, such as DoS, DDoS, port scanning, OS fingerprinting and fuzzing attacks. The distribution of attack and normal instances are shown in Table 3. The details of features and attack groups are explained in more detail in articles (Elsayed et al., 2020) and Sarica and Angin (2020) for InSDN-Dataset and SDN-Dataset.

4.2. Evaluation metrics

The evaluation's parameters and associated calculation methods are presented below, with $\mathcal{T}p$ denoting a true positive, $\mathcal{T}n$ a true negative, $\mathcal{F}p$ a false positive, and $\mathcal{F}n$ a false negative. Accuracy ($\mathcal{AC}$), Detection Rate ($\mathcal{DR}$), Precision ($\mathcal{PR}$), False Alarm Rate ($\mathcal{FAR}$) and $\mathcal{F}1$ score can be determined using the aforementioned formula *i.e.*, $\mathcal{AC} = \frac{\mathcal{T}p+\mathcal{T}n}{\mathcal{T}p+\mathcal{T}n+\mathcal{F}p+\mathcal{F}n}$; $\mathcal{DR} = \frac{\mathcal{T}p}{\mathcal{T}p+\mathcal{F}n}$; $\mathcal{PR} = \frac{\mathcal{T}p}{\mathcal{T}p+\mathcal{F}p}$; $\mathcal{FAR} = \frac{\mathcal{F}p}{\mathcal{F}p+\mathcal{T}n}$; $\mathcal{F}1 = \frac{2\mathcal{T}p}{2\mathcal{T}p+\mathcal{F}p+\mathcal{F}n}$. A Confusion Matrix $\mathcal{CM}$ is used to determine these four metrics ($\mathcal{T}p$, $\mathcal{T}n$, $\mathcal{F}n$, $\mathcal{F}p$), and the leading diagonal components of the $\mathcal{CM}$ represent the number of records that were properly predicted. For instance, the element $\mathcal{CM}_{i,j}$ indicates the number of entries that are genuinely from class i but were misclassified as belonging to class j. The $\mathcal{T}p$ and $\mathcal{F}p$ are used by the ROC curve to show classification performance. The $\mathcal{T}p$ and $\mathcal{F}p$ are greater and lower, respectively, depending on the size of the area under the ROC curve (AUC). One-tailed paired t-tests were employed to determine whether the differences between the proposed approach and benchmarks were statistically significant.

4.3. Experimental environment

The DLTHF framework is designed on a Tyrone PC having Intel CPUs @ 2.20 GHz (2 processors) and 128 GB RAM. Deep-learning architecture is designed using the TensorFlow library Keras. The R programming language is used to implement perturbation-based encoding and normalization techniques. In this experiment, a random sample from both datasets of 70% is used for training and 30% for testing. To optimize the model's hyperparameters, we employed a combination of grid search and random search techniques. This approach allowed us to systematically explore the hyperparameter space to find the most effective settings. The proposed DLTHF framework is a combination of LSTMCSAE and MhSaBiGRNN, where LSTMCSAE is trained for 15 epochs and three (64, 32, 5) hidden layers, sigmoid as activation function and adam Optimizer. In addition, LSTMCSAE incorporates dropout with a rate of 0.5 in between layers to prevent overfitting. The MhSaBiGRNN consists of three (64, 32, 16) hidden layers, ReLU activation function, adam Optimizer, and a batch size of 64. For binary classification, binary cross-entropy and for multi-class classification, categorical cross-entropy were used as a loss function. The training is monitored using a validation split of 10% from the training dataset to ensure that the model does not overfit and generalizes well to new data. Additionally, early stopping is employed to halt training when the validation loss ceases to decrease, thereby preventing overfitting and optimizing computational resources.

4.4. Scenario 1: Analysis of TDS in binary-class scenario

In this subsection, the results of the proposed DLTHF framework in terms of binary classification are discussed. The primary function of

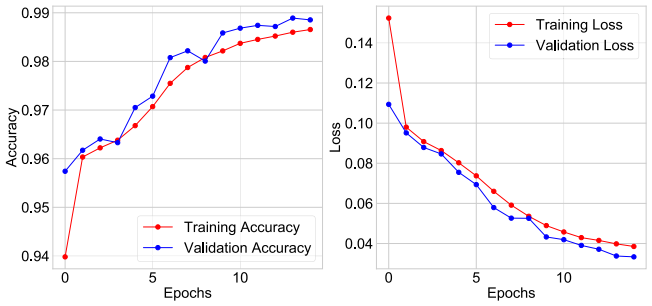


Fig. 5. Comparative Analysis of Accuracy and Loss for the MhSaBiGRNN Technique on the SDN-Dataset (Binary).

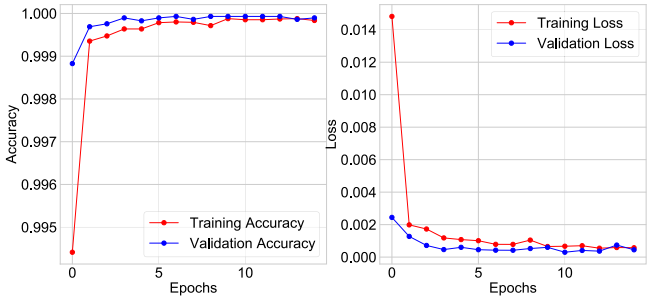
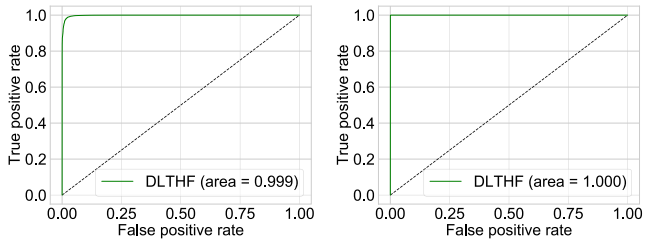


Fig. 6. Comparative Analysis of Accuracy and Loss for the MhSaBiGRNN Technique on the InSDN-Dataset (Binary).

the TDS model is to categorize traffic into legitimate and threat types, which is known as the binary classification problem. It is critical to quickly and precisely identify malicious activity in the SD-IoT network. In this step of the experiment, a series of experiments are conducted using SDN-Dataset and InSDN-Dataset to evaluate the effectiveness of the proposed framework rigorously.

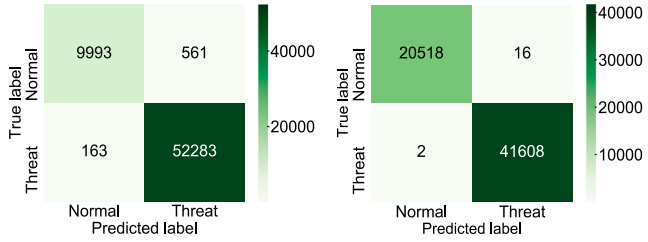
First, before training the classifier, we applied data perturbation-driven encoding and normalization-driven scaling to both datasets. The pre-processed datasets are used by the proposed unsupervised LSTMCSAE technique to reduce feature dimensions. The training and validation accuracy vs loss parameters are used to check the impact LSTMCSAE technique on both datasets. Figs. 3 and 4 show the training accuracy and loss on the SDN-Dataset and InSDN-Dataset, respectively. It is seen that the accuracy value gradually increases with an increase in the number of epochs. We have performed our experiments for 15 epochs and have achieved training and validation accuracy close to 90% with SDN-Dataset and close to 88% with InSDN-Dataset. Similarly, it can be seen that the loss value of each epoch gradually decreased and finally converged close to 0% with both datasets. From these experiments, we can conclude that the LSTMCSAE technique has learned both datasets well. This implies that the rate of convergence is fast, and the error value is small. In general, both datasets with unbalanced distributions show good convergence of the LSTMCSAE technique. Once again, it demonstrates that our technique performs better during training on the two imbalanced SDN attack datasets. However, the main objective of the LSTMCSAE technique is to reduce the dimension. The reduced features obtained are used by the proposed MhSaBiGRNN-based TDS for threat detection.

We have reduced the feature dimensions from 32 to 5 and 82 to 5 in SDN-Dataset and InSDN-Dataset, respectively. These reduced features are learned by the proposed MhSaBiGRNN-based TDS. In Figs. 5 and 6, we have shown the training and validation accuracy vs loss values on the SDN-Dataset and InSDN-Dataset, respectively. For 15 epochs, we have trained and analyzed the results. It is seen that the proposed approach has obtained training and validation accuracy close to 99% with SDN-Dataset and InSDN-Dataset, respectively. On



(a) The ROC Curve of DLTHF using SDN-Dataset (b) The ROC Curve of DLTHF using InSDN-Dataset

Fig. 7. ROC curve obtained from DLTHF framework (Binary).



(a) The confusion matrix of DLTHF using SDN-Dataset (b) confusion matrix of DLTHF using InSDN-Dataset

Fig. 8. The confusion matrix obtained from DLTHF framework (Binary).

Table 4
Result obtained from DLTHF framework based on various performance metrics.

Source	$\mathcal{AC}$	$\mathcal{DR}$	$\mathcal{PR}$	$\mathcal{F1}$	$\mathcal{FAR}$	$\mathcal{LOSS}$
SDN-Datset	98.85%	99.68%	98.93%	99.31%	5.31%	0.033%
InSDN-Dataset	99.97%	99.99%	99.96%	99.97%	0.077%	4.5016e−04%

Terms & Abbreviations: $\mathcal{AC}$: Accuracy; $\mathcal{DR}$: Detection Rate; $\mathcal{PR}$: Precision Rate; $\mathcal{FAR}$: False Alarm Rate.

the other hand, the loss value of each epoch gradually decreased and finally converged to close to 0% with both datasets. The MhSaBiGRNN-based TDS achieved the best performance after ten epochs of training due to its special structure design. The combination of MhSa and BiGRNN has first assigned a self-learned weight based on the degree of relevance, and further, its ability to access the feature series in both directions has learned more comprehensive dependent information for threat detection, thereby increasing the overall accuracy.

We have analyzed the performance in terms of the ROC curve for the proposed framework shown in Fig. 7. A TDS is considered to be effective when its ROC curve runs parallel to the x-axis until point (1,0) and parallel to the y-axis until point (1,1). This is due to the fact that point (1,0) indicates a position in which the TDS is at its best, having the lowest (0) $\mathcal{F}p$ and the highest (1) $\mathcal{T}p$. In this experiment, Fig. 7(a) and Fig. 7(b) show the ROC curves corresponding to the proposed DLTHF framework using both datasets. The AUC values are 0.999 and 1.0 using SDN-Dataset and InSDN-Dataset, respectively.

In addition to the aforementioned evaluation metrics, we have used $\mathcal{CM}$ to evaluate the performance of the DLTHF. $\mathcal{CM}$ is an excellent tool that provides statistics on a TDS model's performance in binary and multiclass classifications. In Fig. 8, a sample $\mathcal{CM}$ for a binary classification is shown, with the true label in the left-most column and the predicted label in the bottom-most row. The amount of real attack data records as shown in Figs. 8(a) and 8(b) is equal to 52,446 and 41,610 for SDN-Dataset and InSDN-Dataset, respectively. Furthermore, out of 52,446 threat data of SDN-Dataset, 52,283 of them were correctly categorized as attack data, and 163 of them were incorrectly classified as attack data. Similarly, with InSDN-Dataset, out of 41,610 threat data, 41,608 of them are correctly categorized as attack data, and 2 of them are incorrectly classified as attack data. So, from above $\mathcal{CM}$ numerical

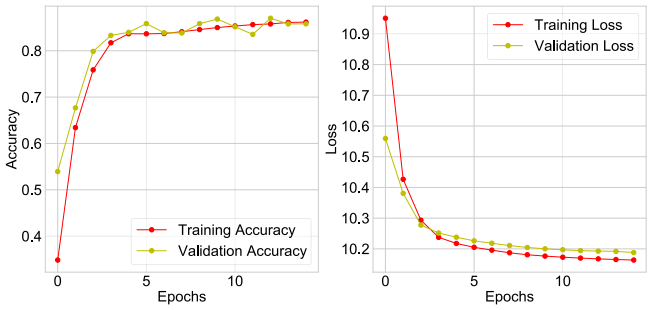


Fig. 9. Comparative Analysis of Accuracy and Loss for the LSTMCSAE Technique on the SDN-Dataset (Multi).

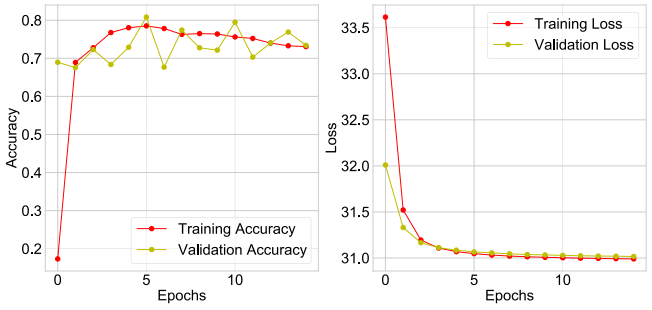


Fig. 10. Comparative Analysis of Accuracy and Loss for the LSTMCSAE Technique on the InSDN-Dataset (Multi).

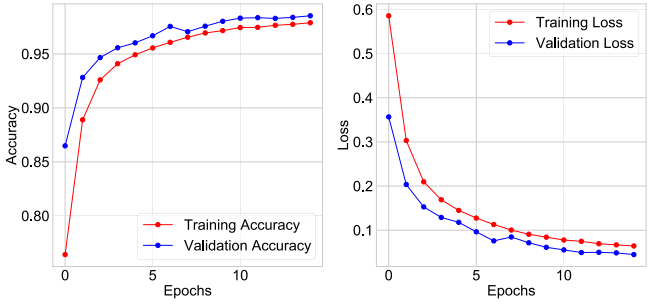


Fig. 11. Comparative Analysis of Accuracy and Loss for the MhSaBiGRNN Technique on the SDN-Dataset (Multi).

values, we can confirm that the proposed approach has shown better results with both datasets.

Finally, we have used overall $\mathcal{AC}$, $\mathcal{DR}$, $\mathcal{PR}$, $\mathcal{F1}$, $\mathcal{FAR}$ and $\mathcal{LOSS}$ metrics to evaluate the performance of the proposed approach. Table 4 shows numerical values obtained for SDN-Dataset and InSDN-Dataset using the DLTHF framework. It is worth noting that the proposed approach using SDN-Dataset has obtained higher values *i.e.*, 98.85% $\mathcal{AC}$, 99.68% $\mathcal{DR}$, 98.93% $\mathcal{PR}$, 99.31% $\mathcal{F1}$, and reduced $\mathcal{FAR}$ to 5.31% and $\mathcal{LOSS}$ to 0.033%. Similarly, we see that using InSDN-Dataset, the proposed DLTHF framework has achieved higher values for above metrics *i.e.*, 99.97% $\mathcal{AC}$, 99.99% $\mathcal{DR}$, 99.96% $\mathcal{PR}$, 99.97% $\mathcal{F1}$, and reduced $\mathcal{FAR}$ to 0.077% and $\mathcal{LOSS}$ to 4.5016e−04%. Therefore, we can see that the proposed approach has achieved high numerical values for the above metrics and performed well by reducing $\mathcal{FAR}$ and $\mathcal{LOSS}$.

4.5. Scenario 2: Analysis of TDS in multi-class scenario

In this subsection, we evaluate our framework's effectiveness in multi-class scenarios using designated metrics. Specifically, Figs. 9 and 10 depict the accuracy versus loss for the LSTMCSAE model on the SDN-Dataset and InSDN-Dataset, respectively. The simultaneous

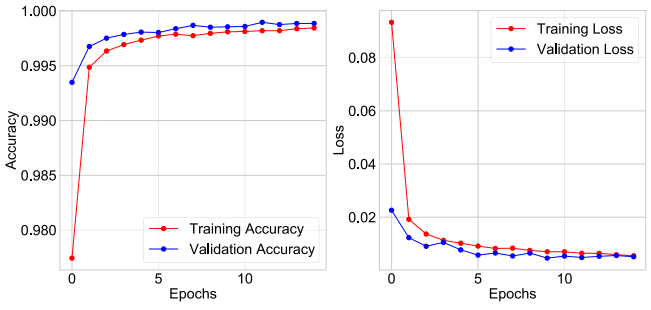


Fig. 12. Comparative Analysis of Accuracy and Loss for the MhSaBiGRNN Technique on the InSDN-Dataset (Multi).

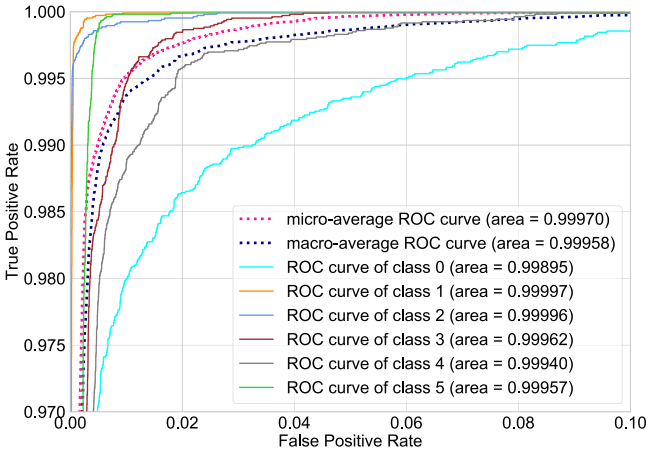


Fig. 13. The ROC Curve of DLTHF using SDN-Dataset (Multi).

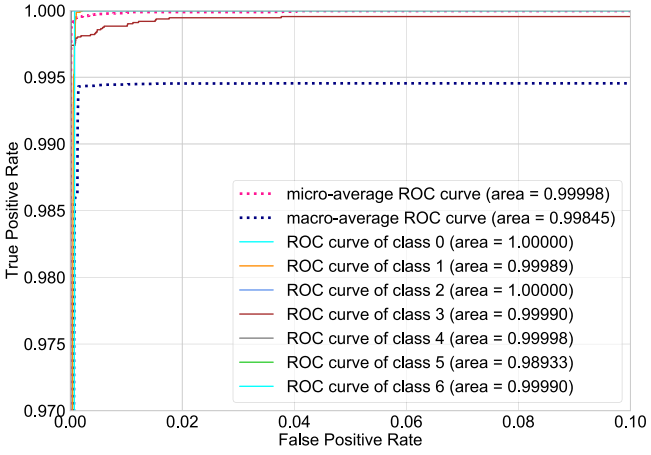


Fig. 14. The ROC Curve of DLTHF using InSDN-Dataset (Multi).

increase in training and validation accuracy indicates that the model is robust, showing no signs of variance issues and demonstrating strong generalization capabilities on the test dataset. Notably, the accuracy measurements for the SDN-Dataset show consistent improvement, ultimately reaching convergence at 86.18% for training accuracy and 85.81% for validation accuracy. In contrast, the training and validation losses gradually drop and converge at 10.16% and 10.18%, respectively. Similar to other datasets, the InSDN-Dataset exhibits a steady rise in training and validation accuracy that converges at 73.07% and 73.40%, respectively, and a steady decline in training and validation loss that converges at 30.99% and 31.01%, respectively. The proposed LSTMCSAE technique is trained to reduce the features set from 32 to 8

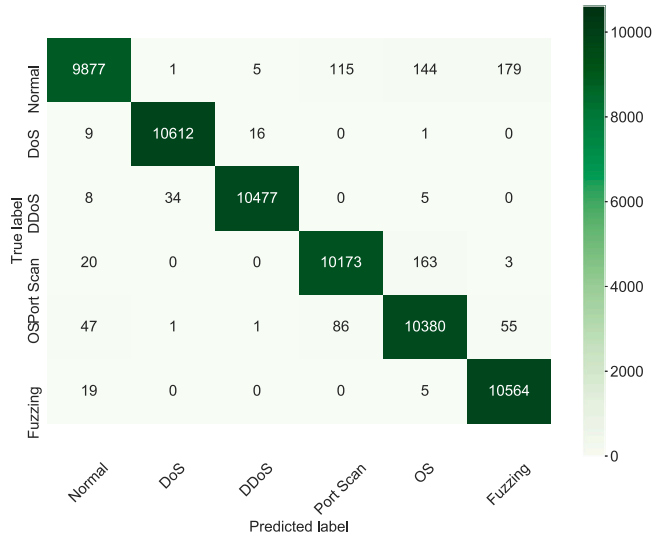


Fig. 15. The confusion matrix of DLTHF using SDN-Dataset (Multi).

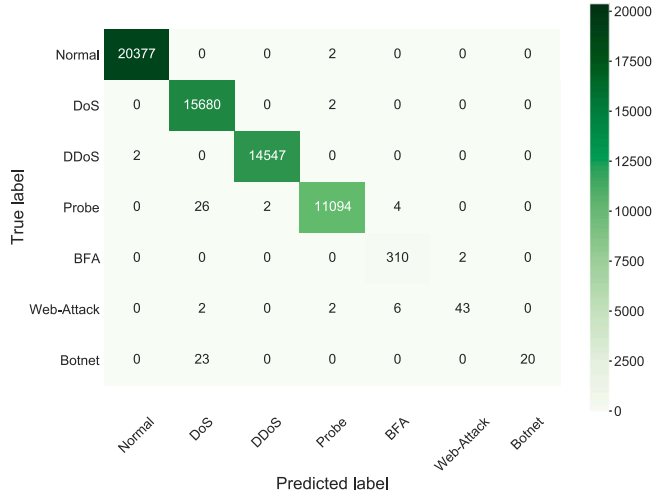


Fig. 16. The confusion matrix of DLTHF using InSDN-Dataset (Multi).

and 83 to 8 for SDN-Dataset and InSDN-Dataset, respectively. Finally, the reduced features are used by the proposed MhSaBiGRNN-based TDS to identify different types of threats in the network.

The accuracy vs loss for training and validation of MhSaBiGRNN-based TDS in the multi-class scenario is shown in Fig. 11 and Fig. 12 for SDN-Dataset and InSDN-Dataset, respectively. It can be seen that with SDN-Dataset, the training and validation accuracy has increased and is 97.89% and 98.54%. Similarly, the loss has been reduced to 0.061% and 0.044% for training and validation sets. With InSDN-Dataset, accuracy values are 99.84% and 99.89%. The loss with the training set is 0.0055%, and with validation, it is 0.0051%. This indicates that the proposed approach has learned the reduced features well and should perform the same with training sets.

In the multi-class scenario, we have also analyzed the performance of the DLTHF framework in terms of the ROC curve. Fig. 13 and Fig. 14 show the AUC value for different groups, *i.e.*, normal and threat types for SDN-Dataset and InSDN-Dataset, respectively. It is seen that the AUC values lie between 1 and 0.99 for all instances present in both datasets. Moreover, with SDN-Dataset, the micro-average AUC value is 0.99, and the macro-average AUC value is 0.99. Similarly, with

Table 5
Class-wise prediction (%) obtained from DLTHF using SDN-Dataset (multi).

Metrics	Normal	DoS	DDoS	Port Scan	OS	Fuzzing
$\mathcal{PR}$	98.96	99.66	99.79	98.06	97.02	97.80
$\mathcal{DR}$	95.69	99.75	99.55	98.20	98.20	99.77
$\mathcal{F}1$	97.30	99.70	99.67	98.13	97.61	98.77
$\mathcal{FAR}$	0.0019	0.0006	0.0004	0.0038	0.0060	0.0045

Table 6
Class-wise prediction (%) obtained from DLTHF using InSDN-Dataset (multi).

Metrics	Normal	DoS	DDoS	Probe	BFA	Web attack	Botnet
$\mathcal{PR}$	99.99	99.67	99.98	99.94	96.87	95.55	100.00
$\mathcal{DR}$	99.99	99.98	99.98	99.71	99.35	81.13	46.51
$\mathcal{F}1$	99.99	99.83	99.98	99.82	98.10	87.75	63.49
$\mathcal{FAR}$	0.000047	0.001097	0.000042	0.000117	0.000161	0.000032	0.0

InSDN-Dataset, the micro-average AUC value is 0.99, and the macro-average AUC value is 0.99. We have also plotted $\mathcal{CM}$ in the multi-class scenario for DLTHF. As shown in Fig. 15 and Fig. 16, the proposed framework has identified most of the instances of different attacks and also the normal group for SDN-Dataset and InSDN-Dataset, respectively. However, the presence of false positives, particularly in complex attack scenarios like Web-attack and Botnet attacks, can be attributed to the inherent challenges of differentiating these from benign activities. Finally, we have analyzed the performance of the proposed DLTHF in terms of class-wise prediction. Table 5 and Table 6 shows the numerical values for $\mathcal{PR}$, $\mathcal{DR}$, $\mathcal{F}1$ and $\mathcal{FAR}$ using SDN-Dataset and InSDN-Dataset, respectively. It is seen that the proposed DLTHF framework has obtained 95%–99% values for $\mathcal{PR}$, $\mathcal{DR}$, $\mathcal{F}1$ and reduced $\mathcal{FAR}$ close to 0% for all classes. Similarly, with InSDN-Dataset, the framework has achieved 46%–99% of these values and reduced $\mathcal{FAR}$ close to 0%.

4.6. Comparison with state-of-the-art methods in binary and multi-class scenario

The two SD-IoT datasets are used to compare the accuracy and inference time of the proposed framework with RNN-based methods published in some recent articles (Keshk et al., 2020; Assis et al., 2020; Popoola et al., 2021; Alkadi et al., 2020) (in the binary and multi-class scenario). We have not compared training time, as it can be done offline before deploying the framework. Table 7 shows the comparison with RNN-based methods. It is seen that the proposed approach has achieved higher accuracy compared to LSTM, GRU, BiGRU, and BiLSTM, as indicated by the p-values ($p < 0.05$ for binary and $p < 0.01$ for multi-class scenarios). Additionally, the inference time is also less compared to these models. The main reason for less inference time is due to the proposed feature extraction module. The proposed module has reduced the features to lower numbers, and thus, generalization has become efficient for DLTHF. However, the computation time can be further optimized by using Graphic Processing Unit (GPU) servers in real-time implementation.

5. Conclusion and future work

In this article, we designed an enhanced deep-learning empowered threat-hunting framework for SD-IoT named "DLTHF". Specifically, we first proposed a feature extraction module by combining data perturbation-driven encoding and normalization-driven scaling with the long Short-Term Memory Contractive Sparse AutoEncoder technique to filter and transform dataset values into the protected format. This approach can be used in an unsupervised manner. In the second module, a new threat detection system was proposed based on the Multi-head Self-attention-based Bidirectional Recurrent Neural Networks model. We also suggested a deployment architecture for the

Table 7
Comparison of the proposed approach with DL-based methods in the binary and multi-class scenario.

Methods	Scenario	Dataset	AC (%)	Inference time (s)
DNN (Kumar et al., 2021)	Binary	SDN-Dataset	90.26***	10.29
		InSDN-Dataset	88.09***	14.91
	Multi	SDN-Dataset	89.16***	11.63
		InSDN-Dataset	88.17**	18.11
1D-CNN (Alshra'a and Jochen, 2021)	Binary	SDN-Dataset	94.25***	14.14
		InSDN-Dataset	95.22***	21.02
	Multi	SDN-Dataset	93.36***	18.27
		InSDN-Dataset	97.96**	25.41
LSTM (Keshk et al., 2020)	Binary	SDN-Dataset	97.76***	5.14
		InSDN-Dataset	97.18***	11.02
	Multi	SDN-Dataset	95.22***	8.11
		InSDN-Dataset	97.96**	13.48
GRU (Assis et al., 2020)	Binary	SDN-Dataset	97.28***	8.47
		InSDN-Dataset	97.63**	13.55
	Multi	SDN-Dataset	97.01***	11.54
		InSDN-Dataset	97.99**	15.40
BiGRU (Popoola et al., 2021)	Binary	SDN-Dataset	97.29***	9.04
		InSDN-Dataset	97.99**	14.50
	Multi	SDN-Dataset	96.14**	13.04
		InSDN-Dataset	96.99***	17.40
BiLSTM (Alkadi et al., 2020)	Binary	SDN-Dataset	96.83***	10.11
		InSDN-Dataset	97.97**	12.51
	Multi	SDN-Dataset	96.39***	10.43
		InSDN-Dataset	98.22*	14.44
Proposed DLTHF	Binary	SDN-Dataset	98.85	2.94
		InSDN-Dataset	99.97	5.43
	Multi	SDN-Dataset	98.54	2.88
		InSDN-Dataset	99.88	5.47

Terms & Abbreviations: LSTM: Long Short-Term Memory; GRU: Gated Recurrent Units; BiGRU: Bidirectional Gated Recurrent Units; BiLSTM: Bidirectional Long Short-Term Memory.

* $p < 0.05$; indicates the level of statistical significance of the results compared to the baseline

** $p < 0.01$; indicates the level of statistical significance of the results compared to the baseline.

*** $p < 0.001$. indicates the level of statistical significance of the results compared to the baseline.

DLTHF framework in both the data plane and control plane. The proposed threat detection system was tested using two publicly available SD-IoT datasets, namely, SDN-Dataset and InSDN-Dataset. To validate the efficacy of the proposed threat detection system, rigorous testing was conducted using two widely accessible SD-IoT datasets, namely, SDN-Dataset and InSDN-Dataset. The proposed threat detection system demonstrates robust threat detection capabilities and achieves low inference times compared to deep learning benchmarks. In the future, the proposed DLTHF framework will be used in an SDN-based Intelligent Transportation System (ITS) environment to enhance the security and privacy of ITS data. Also, the DLTHF framework is expected to be applied with next-generation techniques using Explainable AI (XAI) and quantum computing to further prove its potential for real-life applications in terms of energy.

CRediT authorship contribution statement

Prabhat Kumar: Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Alireza Jolfaei:** Writing – original draft, Visualization, Supervision. **A.K.M Najmul Islam:** Writing – original draft, Visualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Al-Hawawreh, M., Moustafa, N., Garg, S., Hossain, M.S., 2020. Deep learning-enabled threat intelligence scheme in the Internet of Things Networks. IEEE Trans. Netw. Sci. Eng. 1.

Algamdi, H., Aujla, G.S., Jindal, A., Trehan, A., 2023. Intrusion detection in critical SD-IoT ecosystem. In: 2023 IEEE International Conference on Communications Workshops. ICC Workshops, IEEE, pp. 1559–1564.

Alkadi, O., Moustafa, N., Turnbull, B., Choo, K.R., 2020. A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. IEEE Internet Things J. 1.

Alshra'a, A., Jochen, S., 2021. One-dimensional convolutional neural network for detection and mitigation of ddos attacks in SDN. In: International Conference on Machine Learning for Networking. Springer, pp. 11–28.

Assis, M.V., Carvalho, L.F., Lloret, J., Proença Jr., M.L., 2020. A GRU deep learning system against attacks in software defined networks. J. Netw. Comput. Appl. 102942.

Bera, S., Misra, S., Vasilakos, A.V., 2017. Software-defined networking for Internet of Things: A survey. IEEE Internet Things J. 4 (6), 1994–2008.

Bilal, M., Pack, S., 2019. Secure distribution of protected content in information-centric networking. IEEE Syst. J. 14 (2), 1921–1932.

Dayal, N., Srivastava, S., 2023. Analyzing effective mitigation of DDoS attack with software defined networking. Comput. Secur. 130, 103269.

Deliu, I., Leichter, C., Franke, K., 2018. Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, pp. 5008–5013.

Elsayed, M.S., 2020. InSDN-dataset. Online. Available: http://iotseclab.ucd.ie/datasets/SDN/. (Accessed 25 Apr 2022).

Elsayed, M.S., Le-Khac, N.-A., Jurcut, A.D., 2020. Insdn: A novel SDN intrusion dataset. IEEE Access 8, 165263–165284.

Gadallah, W.G., Ibrahim, H.M., Omar, N.M., 2024. A deep learning technique to detect distributed denial of service attacks in software-defined networks. Comput. Secur. 137, 103588.

He, W., Liu, Y., Yao, H., Mai, T., Zhang, N., Yu, F.R., 2021. Distributed variational Bayes-based in-network security for the Internet of Things. IEEE Internet Things J. 8 (8), 6293–6304.

Jin, Y., Tang, C., Liu, Q., Wang, Y., 2020. Multi-head self-attention-based deep clustering for single-channel speech separation. IEEE Access 8, 100013–100021.

Karmakar, K.K., Varadharajan, V., Nepal, S., Tupakula, U., 2020. SDN enabled secure IoT architecture. IEEE Internet Things J. 1.

Keshk, M., Turnbull, B., Moustafa, N., Vatsalan, D., Choo, K.R., 2020. A privacy-preserving-framework-based blockchain and deep learning for protecting smart power networks. IEEE Trans. Ind. Inform. 16 (8), 5110–5118.

Kumar, P., Kumar, R., Garg, S., Kaur, K., Zhang, Y., Guizani, M., 2022. A secure data dissemination scheme for IoT-based e-health systems using AI and blockchain. In: GLOBECOM 2022 - 2022 IEEE Global Communications Conference. pp. 1397–1403.

Kumar, P., Tripathi, R., P. Gupta, G., 2021. P2IDF: A privacy-preserving based intrusion detection framework for software defined Internet of Things-Fog (SDIoT-Fog). In: Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking. ICDCN '21, Association for Computing Machinery, pp. 37–42.

Mishra, P., Biswal, A., Garg, S., Lu, R., Tiwary, M., Puthal, D., 2020. Software defined internet of things security: Properties, state of the art, and future research. IEEE Wirel. Commun. 27 (3), 10–16.

Mittal, S., Joshi, A., Finin, T., 2019. Cyber-all-intel: An ai for security related threat intelligence. arXiv preprint arXiv:1905.02895.

Montasari, R., Carroll, F., Macdonald, S., Jahankhani, H., Hosseinian-Far, A., Daneshkhah, A., 2021. Application of artificial intelligence and machine learning in producing actionable cyber threat intelligence. In: Digital Forensic Investigation of Internet of Things (IoT) Devices. Springer, pp. 47–64.

Moustafa, N., Adi, E., Turnbull, B., Hu, J., 2018. A new threat intelligence scheme for safeguarding industry 4.0 systems. IEEE Access 6, 32910–32924.

Nichelini, A., Pozzoli, C.A., Longari, S., Carminati, M., Zanero, S., 2023. Canova: a hybrid intrusion detection framework based on automatic signal classification for can. Comput. Secur. 128, 103166.

Noor, U., Anwar, Z., Noor, U., Anwar, Z., Rashid, Z., 2018. An association rule mining-based framework for profiling regularities in tactics techniques and procedures of cyber threat actors. In: 2018 International Conference on Smart Computing and Electronic Enterprise. ICSCEE, pp. 1–6.

Popoola, S.I., Ande, R., Fatai, K.B., Adebisi, B., 2021. Deep bidirectional gated recurrent unit for botnet detection in smart homes. In: Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics. Springer, pp. 29–55.

Rafique, W., Qi, L., Yaqoob, I., Imran, M., Rasool, R.U., Dou, W., 2020. Complementing IoT services through software defined networking and edge computing: A comprehensive survey. IEEE Commun. Surv. Tutor. 22 (3), 1761–1804.

Sarica, A.K., 2020. SDN-dataset. Available: https://github.com/AlperKaan35/SDN-Dataset. Online (Accessed 25 Apr 2022).

Sarica, A.K., Angin, P., 2020. A novel SDN dataset for intrusion detection in IoT networks. In: 2020 16th International Conference on Network and Service Management. CNSM, IEEE, pp. 1–5.

Sentuna, A., Alsadoon, A., Prasad, P., Saadeh, M., Alsadoon, O.H., 2021. A novel enhanced naïve bayes posterior probability (ENBPP) using machine learning: Cyber threat analysis. Neural Process. Lett. 53 (1), 177–209.

Shafiq, M., Tian, Z., Bashir, A.K., Du, X., Guizani, M., 2020. IoT malicious traffic identification using wrapper-based feature selection mechanisms. Comput. Secur. 94, 101863.

Shaji, N.S., Jain, T., Muthalagu, R., Pawar, P.M., 2023. Deep-discovery: Anomaly discovery in software-defined networks using artificial neural networks. Comput. Secur. 132, 103320.

Singh, M., Aujla, G.S., Singh, A., Kumar, N., Garg, S., 2021. Deep-learning-based blockchain framework for secure software-defined industrial networks. IEEE Trans. Ind. Inform. 17 (1), 606–616.

Singh, P., Kaur, A., Aujla, G.S., Batth, R.S., Kanhere, S., 2020. DaaS: Dew computing as a service for intelligent intrusion detection in edge-of-things ecosystem. IEEE Internet Things J. 1.

Soleymani, S.A., Goudarzi, S., Anisi, M.H., Cruickshank, H., Jindal, A., Kama, N., 2022. TRUTH: Trust and authentication scheme in 5G-IIoT. IEEE Trans. Ind. Inform. 19 (1), 880–889.

Tounsi, W., Rais, H., 2018. A survey on technical threat intelligence in the age of sophisticated cyber attacks. Computers security 72, 212–233.

Usman, N., Usman, S., Khan, F., Jan, M.A., Sajid, A., Alazab, M., Watters, P., 2021. Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. Future Gener. Comput. Syst. 118, 124–141.

Wang, F., Li, R., Wang, H., Zhu, H., Xiong, N., 2021. TS-PADM: Anomaly detection model of wireless sensors based on spatial-temporal feature points. Wirel. Commun. Mob. Comput. 2021.

Wu, X., Xu, X., Bilal, M., 2021. Toward privacy protection composition framework on Internet of Vehicles. IEEE Consum. Electron. Mag. 11 (6), 32–38.

Zhang, H., Yi, Y., Wang, J., Cao, N., Duan, Q., 2019. Network attack prediction method based on threat intelligence for IoT. Multimedia Tools Appl. 78 (21), 30257–30270.

Prabhat Kumar received his Ph.D. degree in Information Technology, National Institute of Technology Raipur, Raipur, India, under the prestigious fellowship of Ministry of Human Resource and Development (MHRD) funded by the Government of India in 2022. Thereafter, he worked with Indian Institute of Technology Hyderabad, India as a Post-Officer under project "Development of Indian Telecommunication Security Assurance Requirements for IoT devices". He is currently working as Post-Doctoral Researcher with the Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta, Finland. He has many research contributions in Machine Learning, Deep Learning, Federated Learning, Big Data Analytics, Cybersecurity, Blockchain, Cloud Computing, Internet of Things and Software Defined Networking. He has authored or coauthored over 50+ publications in high-ranked journals and conferences, including 15+ IEEE TRANSACTIONS paper. One of his Ph.D. publications was recognized as a top cited article by WILEY in 2020–21. He is also an IEEE Member.

Alireza Jolfaei is an Associate Professor of Networking and Cyber Security in the College of Science and Engineering at Flinders University, Adelaide, Australia. He is a Senior Member of the IEEE and a Distinguished Speaker of the ACM. His main research interest is in Cyber-Physical Systems Security. He has published over 100 papers, which appeared in peer-reviewed journals, conference proceedings, and books. Before Flinders University, he has been a faculty member with Macquarie University, Federation University, and Temple University in Philadelphia, PA, USA. He received the prestigious IEEE Australian council award for his research paper published in the IEEE Transactions on Information Forensics and Security. Dr. Jolfaei is the IEEE Consumer Technology Publication Board member and the Editor-in-Chief of the Consumer Technology Society World Newsletter. He has served as the Regional Chair of the IEEE Technology and Engineering Management Society's Membership Development and Activities for Australia. He has served as a program coChair, a track Chair, a session Chair, and a Technical Program Committee member, for major conferences, including IEEE TrustCom and IEEE ICCCN.

A.K.M. Najmul Islam is a Full Professor at LUT University, Finland. He conducts cross-disciplinary research in digitalization and its impact on citizens, organizations, and society. He is a docent of Information Systems at Tampere University. Islam's publication has appeared in top Information Systems outlets such as Journal of Strategic Information Systems, European Journal of Information Systems, and Information Systems Journal. He has published in other highly ranked interdisciplinary journals such as IEEE Access, Computers & Education, Technological Forecasting and Social Change, International Journal of Information Management, Information Technology & People, Computers in Human Behavior, Computers in Industry, Internet Research, Communications of the AIS, among others. He is currently serving as a Senior Editor for Information Technology & People journal.