# Reconstruction-based LSTM-Autoencoder for Anomaly-based DDoS Attack Detection over Multivariate Time-Series Data

Yuanyuan Wei, Julian Jang-Jaccard, Fariza Sabrina, *Member, IEEE*, Wen Xu, Seyit Camtepe, *Senior Member, IEEE*, and Aeryn Dunmore

Abstract—A Distributed Denial-of-service (DDoS) attack is a malicious attempt to disrupt the regular traffic of a targeted server, service, or network by sending a flood of traffic to overwhelm the target or its surrounding infrastructure. As technology improves, new attacks have been developed by hackers. Traditional statistical and shallow machine learning techniques can detect superficial anomalies based on shallow data and feature selection, however, these approaches can not detect unseen DDoS attacks. In this context, we propose a reconstruction-based anomaly detection model named LSTM-Autoencoder (LSTM-AE) which combines two deep learning-based models for detecting DDoS attack anomalies. The proposed structure of long short-term memory (LSTM) networks provides units that work with each other to learn the long short-term correlation of data within a time series sequence. Autoencoders are used to identify the optimal threshold based on the reconstruction error rates evaluated on each sample across all time-series sequences. As such, a combination model LSTM-AE can not only learn delicate sub-pattern differences in attacks and benign traffic flows but also minimize reconstructed benign traffic to obtain a lower range reconstruction error, with attacks presenting a larger reconstruction error. In this research, we trained and evaluated our proposed LSTM-AE model on reflection-based DDoS attacks (DNS, LDAP, and SNMP). The results of our experiments demonstrate that our method performs better than other state-of-the-art methods, especially for LDAP attacks, with an accuracy of over 99%.

Index Terms—LSTM, Autoencoder, anomaly detection, multivariate analysis, time-series, DDoS Attack

## I. INTRODUCTION

**N**ETWORK traffic is increasing rapidly with the continued development of information and communication technology (ICT) due to advanced innovative technologies, including cloud computing, and big data. However, the rapid

Yuanyuan Wei is with the CybersecurityLab, Comp Sci/Info Tech, Massey University, Auckland, 0632, NEW ZEALAND (e-mail: y.wei1@massey.ac.nz).

Julian Jang-Jaccard is with the CybersecurityLab, Comp Sci/Info Tech, Massey University, Auckland, 0632, NEW ZEALAND (e-mail: j.jangjaccard@massey.ac.nz).

Fariza Sabrina is with the School of Engineering and Technology, Central Queensland University, Sydney NSW 2000, AUSTRALIA (e-mail: f.sabrina@cqu.edu.au).

Wen Xu is with the CybersecurityLab, Comp Sci/Info Tech, Massey University, Auckland, 0632, NEW ZEALAND (e-mail: w.xu2@massey.ac.nz).

Seyit Camtepe is with the CSIRO Data61, AUSTRALIA (e-mail: Seyit.Camtepe@data61.csiro.au).

Aeryn Dunmore is with the CybersecurityLab, Comp Sci/Info Tech, Massey University, Auckland, 0632, NEW ZEALAND (e-mail: a.dunmore@massey.ac.nz).

Manuscript received April 19, 2021; revised August 16, 2021.

proliferation of innovative technologies and communication infrastructure brings the potential for cyberattacks and other threats to Internet users. In the area of cyber security attacks, one of the most dangerous threats is a distributed denial-ofservice (DDoS) attack [1]–[3].

A DDoS attack is a form of network attack that attempts to overwhelm online services, websites, and web applications with malicious traffic from multiple compromised computer systems. It can also make simultaneous requests to the target server in order to exhaust the network resources and thereby deny normal online service to legitimate users or computer systems [4]–[6]. DDoS attacks are not only conducted against online services, web applications, and information infrastructure to cause downtime, but also to prevent legitimate users from purchasing products and using online services - such as emails, websites, and applications - and affecting program performance [6]. As a result of the COVID-19 lockdown in 2020, there has been an increase in attacks on education, online shopping, and office work, as a large number of people are now studying, working, and shopping online, giving hackers greater opportunities [7]. In [8] Azure Networking found there was a 25% increase in DDoS attacks in the first six months of 2021 when compared with the fourth quarter of 2020. Moreover, Azure mitigated approximately 35 thousand attacks against its global infrastructure in the last six months of 2021, which increased from 43% compared with the first six months of 2021. A white paper from Cisco [9] predicted that nearly 300 billion mobile applications would be downloaded by 2023 and that DDoS attacks would rise to 15.4 million globally by 2023.

DDoS detection is becoming an urgent need because of the sophistication and diversification of attacks. For instance, difficult-to-track attackers and unknown or new attack types occur continuously, for example, zero-day attacks [10], [11]. Detecting DDoS attacks becomes more and more difficult not only because a large proportion of attack traffic is similar to legitimate traffic, but also because of newer hybrid attack methods [3], [5]. Therefore, the detection and mitigation of DDoS attacks not only protects the network for legitimate users but also reduces financial loss for businesses [12]. In order to detect and mitigate DDoS attacks, statistical techniques have been proposed in [13] to identify DDoS attacks. Furthermore, some machine learning approaches for signature, threshold, and statistics-based measurements have been proposed to distinguish DDoS attack traffic [14], [15]. However, traditional statistical and machine learning can not detect previously unseen DDoS attacks [5]. Moreover, most traditional statistical and machine learning-based detection approaches require better-selected features or defined thresholds [3], [5]. In contrast to traditional detection techniques, deep learningbased DDoS attack detection - such as Convolutional Neural Networks (CNNs) [5], Recurrent Neural Networks (RNNs) [3], Autoencoders [12], and so forth - can offer better detection rates for DDoS network traffic [3]. However, some limitations in existing deep learning-based detection need to be addressed, for example, Autoencoder models are sensitive to the anomalies in the training stage, and RNNs can better address historical sequence data, but face the shortcoming of the vanishing gradient problem. To address these issues, we propose a reconstruction-based hybrid deep learning model that combines the capabilities of long short-term memory (LSTM) and Autoencoders (AE) for detecting DDoS attacks, using the state-of-the-art CICDDoS2019 datasets.

In this research, the LSTM model aims to solve the timeseries sequence problem of DDoS traffic flow, while the AE is used to calculate the reconstruction loss in order to define the threshold and detect DDoS attacks. In order to overcome the shortcoming caused by sensitivity to anomalies in the training process, we use only benign traffic from the DDoS dataset to train our model and minimize the reconstruction error. Furthermore, the LSTM can learn the time series sequence of DDoS network traffic continuously but learns the delicate difference between attacks and benign traffics based on the time window length section. A combination model LSTM-AE can learn delicate differences in sub-patterns between attacks and benign traffic while minimizing the reconstructed benign traffic to obtain a lower range reconstruction error. Our experimental results showed that the proposed LSTM-AE model achieves better performance in processing reconstruction-based timeseries data than other comparable proposed models. The main contributions of our proposed model are as follows:

## **Summary of Original Contributions**

- We propose a novel time-series anomaly detection architecture that leverages reconstruction-based LSTM-AE for efficient DDoS attack detection. In our proposed model, the LSTM networks are comprised of multiple LSTM units that work with each other to learn the long shortterm correlation of data within a time series sequence. An autoencoder is used to identify the optimal threshold based on the reconstruction error rates evaluated across all time-series sequences. This threshold is used to identify anomalies.
- We apply our proposed LSTM-AE model against the reflection-based DDoS attacks DNS, LDAP and SNMP. The model is trained on normal time-based traffic flow features using a subset of traffic flow information over a fixed-time window length.
- A novel anomaly score technique is proposed to calculate the MAE value of each traffic flow, which can be calculated flexibly based on different fixed-time window lengths.
- We performed tests on the state-of-the-art CICDDoS2019 dataset, and compared the performance of our proposed model with other similar approaches that use different

aspects of LSTM and/or AE. Our experimental results, based on the comprehensive set of evaluation criteria, demonstrate that our proposed model can effectively detect anomalies reaching a detection accuracy exceeding 99%.

The rest of this paper is structured as follows: Section II introduces related works in the field of DDoS attack detection. Section III introduces our methodology. Section IV illustrates the experimental setup and Section V details the analysis of our results evaluated on the various reflection-based attack types, including DNS, LDAP, and SNMP datasets. Section VI concludes the paper with the planned future works.

# II. RELATED WORK

In recent years, anomaly detection has attracted extensive attention in literature exploring machine learning techniques. In this paper, we review the issues of variable length of DDoS anomaly detection, areas closely related to our contributions.

Sharafaldin et al. [16] generated a dataset titled CICD-DoS2019 and classified benign traffic and attacks based on 4 machine learning methods, including ID3, Random Forest, Naïve Bayes, and Multinomial Logistic Regression. The evaluation result shows the highest accuracy from RF and ID3. Jia et al. [17] proposed an IoT DDoS defense technique named FlowGuard, and constructed LSTM and CNN techniques for DDoS identification and classification based on simulated data and CICDDoS2019 dataset to identify, classify, and mitigate DDoS attacks. They used a CNN to better classify all malicious flows, then employed the LSTM technique as an identification module. This was not only able to capture significant features of flows to identify benign samples but also to apply a softmax function on the output layer to distinguish between benign and malicious traffic. The above researches used flow-based statistical features, which were extracted from CICDDoS2019 dataset for DDoS attack detection.

Novaes et al. [18] introduced two scenarios for detecting anomalies and mitigating attacks on Software-Defined Networks (SDNs) using LSTM-FUZZY techniques. Firstly, the authors used an LSTM network semi-supervised learning technique to predict benign univariate time series behaviour of IP flows, followed by classifying attacks with a Fuzzy logic technique. There were two datasets used in this research, including the SDN dataset and the CICDDoS2019 dataset. Aydin et al. [19] proposed an LSTM-based system (LSTM-CLOUD) to detect and prevent DDoS attacks in a public cloud network environment through experimentation on CICDDoS2019 dataset. The authors built the LSTM models with two hidden layers, three dense and dropout layers, and used the sigmoid function to classify benign and DDoS attacks (anomalies). The model performed to a high accuracy rate of 99.83%, but this work only classified 3 attack types of attacks - UDP, MSSQL, SYN - as well as benign traffic samples.

Nezhad et al. [20] normalized two features (packets and source IP addresses) on a 1-minute time series interval by using a Box-Cox transformation. They then used a statistical time series analysis technique called ARIMA to predict the number of packets. Ergan et al. [21] introduced LSTM-based neural networks to detect anomalies in a time series in an unsupervised manner. The author employs an LSTM-based network technique to obtain the fixed-length sequence data. They utilized the OC-SCM and SVDD algorithms together with a scoring function to detect anomalies.

Salahuddin et al. [12] proposed a time-based Autoencoder technique named Chronos to detect DDoS traffic anomalies with aggregating features. One of the contributions in this research was the threshold they selected to highlight the efficiency of their anomaly detection system. They utilised threshold selection heuristic maximizes the F1 score. To detect anomalies effectively, they implemented various window sizes on different DDoS attacks. As a result, the proposed Chronos system achieves an F1-score of 99% for most attack types and over 95.86% for all attack type performance measurements by using two-time windows along with the selected heuristic threshold.

Fouladi et al. [22] used the CAIDA dataset to train and evaluate the K-SVD and BMP (Basic Matching Pursuit) algorithms and a SOM (Self-Organizing Map) model, using sparse coding and frequency domain for DDoS attack anomaly detection. They extracted normal time series data using a K-SVD algorithm and applied a BMP method to train and evaluate the benign data to estimate the sparse coefficients. They then used the normal data on the SOM model to obtain a SOM lattice. This enabled them to calculate the minimum Euclidean Distance between corresponding coefficients and use the SOM lattice to distinguish between normal and attack behaviors.

Although many researchers above have introduced statistical and machine learning techniques to detect anomalies in the DDoS datasets, it was also a challenging task worthy of further study, especially for distinguishing subtle differences in benign and attack traffics. In this research, we propose a hybrid deep learning reconstruction-based LSTM-Autoencoder model for anomaly-based DDoS attack detection. Our model uses a specific selection of features over multivariate timeseries data analysis. Compared to the above statistical and machine learning-based studies, the differences in this research can be detailed as follows: 1) Our LSTM-AE neural network model conducts training and testing in an unsupervised manner (without labels), this is combined with the reconstruction error used to detect anomalies with several different time window lengths; 2) Anomaly detection in our proposed model gave experimental results for that DDoS attack anomaly detection with high accuracy on the CICDDoS2019 time-series dataset; 3) This experiment performed better in terms of the performance matrix - including precision, recall, and F1-score - than benchmark studies on the same CICDDoS2019 dataset, such as [16], which used machine learning techniques (including Random forest, Naive Bayes, etc.) to detect DDoS attacks.

## **III. LSTM-AUTOENCODER ANOMALY DETECTION**

#### A. Overview of Our Framework

This section provides an overview of our reconstructionbased time-series anomaly detection system. One of the DDoS traffic characteristics is collecting correlated temporal sequences, and the deep learning technique of LSTM is a good algorithm to deal with this temporal problem. Thus, we adopted the LSTM model for the LSTM-Autoencoder neural network for the Encoder and Decoder stages as its mechanism better captures DDoS flow information by feeding each flow at each time step. We propose a hybrid machine learning model which combines an LSTM neural network and an autoencoder. We apply our model to the multivariate time-series dataset CICDDoS2019. As illustrated in Fig. 1, our LSTM-AE model builds the LSTM networks on the encoder and decoder schemes of an Autoencoder. The encoder obtains the sequence of the high-dimensional input data as a fixed-size vector. Using the memory cells of LSTM, the data processed by the encoder scheme retains the dependencies across multiple data points within a time-series sequence. This stage reduces the high-dimensional input vector representation into a lowdimensional representation. The decoder-LSTM reproduces the fixed-size input sequence from the reduced representation of the input data in the latent space, while reconstruction error rates determine the classification threshold. Fig. 1 depicts the operation of a model for DDoS anomaly detection. Phase 1 is data pre-processing based on the selected characterization of each traffic flow fed into the LSTM-AE neural network. Phase 2 is the training and testing process, in which the threshold is obtained based on minimizing the reconstruction error on benign traffic. The DDoS anomaly detection is the last phase in the diagram, which determines anomaly scores by calculating the reconstruction error of each traffic flow between the timeseries sequence's original input and the reconstructed output.

# B. Feature usage and Input Sequence Data

Fig 1 shows an overview of our proposed LSTM-AE model. To obtain the DDoS traffic flow data, we used the public dataset CICDDoS2019 [16] from the Canadian Institute for Cybersecurity. In [16], the extracted network traffic features are stored in a CSV file, and each CSV file includes different DDoS attack types (anomalies) and benign traffic. In this research, we used three reflection-based DDoS attack types to detect anomalies, specifically DNS, SNMP and LDAP. As shown in the first part of the DDoS traffic window in Fig 1, all DDoS traffic flows are separated into n subsets based on a selected window length.

As our LSTM-AE model for detecting anomalies depends on learning the network traffic patterns, it is important to use high correlation features to obtain high performance in measurements such as accuracy. Using the research on the state-of-the-art CICDDoS2019 DDoS attack dataset in [16], the author specifies the importance of selecting the top-n features which correspond to the weight and mean value of different attack types. In this research, we have also used these top-n features as depicted in [16] from the CICDDoS2019 dataset. As such 'Max Packet Length', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Average Packet Size', 'Min Packet Length' is used as selected features for our multivariate time-series anomaly detection. After feature selection, the input data has to be reshaped into a 2-dimensional vectors



Fig. 1: Overview of our proposed model

before the data is fed into the required LSTM encoder input layer. The original DDoS dataset is comprised of a series of time sequence  $[X_1, X_2, X_3, ..., X_n]$ . Each sequence X with a fixed T-length time window data  $[x_1, x_2, x_3, ..., x_t]$  is created where  $x_t \in \mathbb{R}^m$  represents an m-features input at timeinstance t. They are then reshaped into 2-D (2-dimensional) arrays, representing samples and time steps. For example, a sequence of the DDoS attack data is converted into a 2-D array where each dimension indicates the list of samples at 10 time steps with n features.

# C. LSTM-AE Model Architecture

1) Long Short Term Memory: The LSTM network is a variation of a Recurrent Neural Network (RNN) that addresses the gradient vanishing and exploding problems of RNNs and can process long term sequences between data samples at any given time from many history time steps. The architecture of the LSTM network is suitable for processing time series sequence data and provides the capability of forgetting the historical data from each memory cell before updating the memory cell with new data. As illustrated in Fig. 2(a), the LSTM network contains memory cells  $c_t$  and multiple gate units, including the forget gate  $f_t$ , the input gate  $i_t$  and the

output gate  $o_t$ . These three gate units control the state of memory cells. For example, at time step t, the forget gate decides which bits of the cell state are useful given both the previous hidden state and new input data. The LSTM network can omit insignificant information (values) from the cell state of the forget gate and can recognize significant information (values) to keep and update in the cell state. Using this network architecture, the LSTM model performs to a high standard when capturing long-term patterns in time series sequence data.



Fig. 2: Neural Networks architectures: (a) LSTM; (b) Autoencodre.

2) Autoencoder: An Autoencoder (AE) is an unsupervised neural network model that is not only used for feature selection and dimension reduction but also can be used for reconstruction-based Encoder-Decoders to detect anomalies. The typical architecture of an autoencoder is composed three components: an input layer, one or more hidden layer(s), and an output layer. The operations of an autoencoder for detecting anomalies can be divided into Encoding, Decoding, and Reconstruction Loss as illustrated in Fig. 2(b). The encoder compresses the high dimension input data and maps it to low dimensional representations h, in the bottleneck layer while the decoder decompresses the encoded representation and reconstructs to the output  $\hat{x}$ . Typically, the autoencoder is trained by utilizing the MAE equation as a loss function to minimize the reconstruction error between the output  $\hat{x_t}$  and input  $x_t$ .

3) LSTM-Autoencoder: We have addressed the issue of anomaly detection in capturing normal phenomena through time-based traffic flow from the DDoS attacks dataset. An LSTM-AE model need not only be applied to address the problem of feed-forward neural networks but also may be utilized to learn patterns in time-based sequence data, making them suitable for time-series anomaly detection [23], [24]. Our proposed LSTM-AE model includes an autoencoder that utilizes the LSTM network as a hidden layer in both the Encoder and Decoder schemes. This is followed by an Encoded Features layer and an output layer respectively, which calculate the reconstruction error to detect anomalies. The role of the LSTM network in our proposed scheme is to learn the patterns of data based on time-series DDoS signals. We combine this with an autoencoder to learn the best encoder-decoder scheme to detect anomalies. The output of the LSTM Encoder and Decoder are then compared with the original input data and the reconstruction error is backpropagated through the architecture to update the weights of the neural network.

Our LSTM-AE model employs an autoencoder that utilizes the LSTM network as a hidden layer in both Encoder and Decoder schemes followed by an Encoded Features layer and an output layer respectively. These calculate the reconstruction error to detect anomalies. The role of the LSTM network in our proposed scheme is to learn the patterns of data based on selected time window length sequences. Our proposed LSTM-AE is composed of six layers - an Input, an LSTM Encoder, an LSTM Decoder, a RepeatVector, a TimeDistributed and an Output layer. The LSTM Encoder obtains the sequence of high-dimensional input data as a fixed-size vector and compresses the input data into a low-dimensional hidden representation. Within an LSTM cell, for an input time series sequence data  $X_1 = [x_1, x_2, ..., x_t]$  where t represents the time steps, each  $x_t$  calculation is performed using the following:

$$f_t = \sigma(w_f[H_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(w_i[H_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = tanh(w_c[H_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot C_t \tag{4}$$

(5)

$$o_t = \sigma(w_o[H_{t-1}, x_t] + b_o)$$

$$H_t = o_t \odot tanh(C_t) \tag{6}$$

where

- $f_t$  represents the forget gate,  $i_t$ ,  $\tilde{C}_t$  and  $C_t$  represents the input gate, and  $o_t$  and  $H_t$  represents the output gate.
- w and b are the weights and the bias of the forget gate, input gate and output gate.
- H<sub>t-1</sub> and x<sub>t</sub> present the concatenation of the hidden state and the current input respectively.
- $\sigma$  is the activation function of each gate, and it outputs numbers in the range of [0, 1].
- $\bullet \ \odot$  represents element-wise multiplication.

Next, after compressing input data into low-dimensional representation until it reaches the latent space (encoded features), all data presentation can be repeated t times on the RepeatVector layer to feed into the LSTM Decoder layer. In this LSTM Decoder layer, the decoder scheme uses the same number of features (equal to the encoder features on the LSTM Encoder layer) to map the latent space representation back to a high-dimensional representation. We add a TimeDistributed layer in order to generate the output of the LSTM Decoder in time sequence.

Encoder: To illustrate the LSTM encoder stage, if the time step is set to 10 - seen LSTM cells working theory at the time step t (show in Fig 2a) - the input includes the previous output of the hidden state  $(h_{t_1})$  and the cell state  $(C_{t_1})$ , and the current input  $x_t$  while output includes the new hidden state  $(h_t)$  and the new cell state  $(C_t)$ , and an output  $y_t$ . At the time step t+1, the new input becomes a new input  $x_{t+1}$ for the next set of cells, and the output is obtained from the last hidden state and cell state of the LSTM cell. This then becomes the new input of hidden state  $(h_t)$  and the cell state  $(C_t)$  information. Note that we discard the output  $(y_t)$  of each LSTM cell at each time step t in the Encoder and preserve the initial state which is from the hidden state and cell state. Finally, the output of the last time step (10) is the hidden state  $(h_{10})$  and the cell state  $(c_{10})$ , which is a summary of the entire 10 time steps. Therefore, the input vector of LSTM encoder is  $10 \times 5$ , where 10 is the time steps (or time window length) and 5 is the number of features. If we set the number of input features (units) of LSTM to 16, after passing through the LSTM Encoder layer the size of the output vector is  $1 \times 16$ .

Our LSTM Encoder architecture (see Fig. 1: LSTM-Autoencoder neural network) is defined as follows:

- Each traffic flow in the Input layer has been reshaped into a 2D matrix. Note that each traffic flow is represented by a matrix  $n \times t$ , where n represents the traffic flows, and t represents the time steps. This form of a series of input data is able to capture DDoS traffic patterns based on the sliding window length.
- Each LSTM Encoder layer consists of 16 LSTM units with "tanh" activation functions.
- We added a dropout layer (0.2) to prevent over-fitting in the Encoder part.
- The encoded features have lower dimensions than the number of input features.

**Decoder:** Between the Encoder and Decoder layer, we added a RepeatVector Layer to create the copies of the  $1 \times 16$ 

vector equal to the number of time steps, which we called Encoded Features. For example, the size of time steps in our model is 10, therefore this layer will create 10 copies of the encoded features as a two-dimensional vector  $10 \times 16$ . The output of this layer becomes the input of the LSTM Decoder at each time step, which means each copy of the encoded features at each time step will be the input of the next set of LSTM cells. As depicted, the difference of LSTM network between the Encoder and Decoder is that the output of each LSTM cell at each time step  $(y_t)$  in the Decoder cannot be discarded and are outputted as  $y_t$ . There are two reasons to get the output of each of the LSTM cells: firstly, for the added layer, TimeDistributed, the input can be a 3dimensional vector, which means the output from the LSTM cells has to be a 3-dimensional vector; and second, to ensure the output of each time step is as close as possible to the input. Therefore, the LSTM Decoder representations obtained in lowdimensionality encoding are used as input in the decoder and there are utilised to reproduce the original input data using the LSTM network. This means the output from the last set of LSTM cells (copied t times) then becomes the input to the LSTM Decoder network.

Each  $1 \times 16$  set is fed as an input to the decoder which creates a 3 Layer network with 10 LSTM cell units. Each LSTM cell unit processes each  $1 \times 16$  encoded feature. These LSTM units produce an output that represents the result of the learning from the encoded feature where the output is multiplied with the  $1 \times 16$  vector created by the additional TimeDistribution layer. At the same time, each LSTM cell unit processed by the current LSTM cell which is passed to the next LSTM - with the exception of the last LSTM unit. The matrix multiplication between the output of each LSTM layer (L) ( $10 \times 16$ ) and the TimeDistributed layer ( $16 \times 1$ ) results in a vector with of size  $10 \times 1$  - the same as the size of the input.

The LSTM Decoder (see Fig. 1: LSTM-Autoencoder Neural Network) architecture is defined as follows:

- The encoded feature in the bottleneck layer will be the input of the LSTM Decoder.
- An LSTM Decoder layer consists of 16 LSTM units with "tanh" activation functions.
- We added a dropout layer (0.2) to prevent overfitting in the Decoder section.
- Typically, the output of LSTM Decoder has two outputs, including the output data (traffic flows)  $(O_t)$  and the new hidden state  $(H_t)$ . The output of  $H_t$  can be discarded. Thus, the output from the decoder part is the reconstructed feature of the same size and dimension as the input data.

The final aim of LSTM-AE is to reconstruct the input from the output, i.e.  $\hat{X}_1 \approx X_1$  where  $X_1$  indicates the input while  $\hat{X}_1$  indicates the output.

**LSTM-AE Training:** We divide the CICDDoS2019 dataset into a training set (70%), a validation set (10%), and a testing set (20%). Note that the training and validation set consists of benign samples only, in order to train the proposed LSTM-AE model. The testing set consists of both benign and attack samples for detecting anomalies. The architecture of the proposed LSTM-AE is designed to minimize the reconstruction error between the original input and the reconstructed output based on time series sequence traffic flows. This LSTM model aims to learn the patterns in the data. Our motivation for using the Encoder and Decoder scheme to detect anomalies is the fact that their working scheme is able to detect anomalies based on the low distribution of normal data. As such, our training data set consists of only normal sequence data and this is utilized for training the proposed LSTM-AE model. The LSTM-AE is taught normal traffic behavior, using the benign samples. Our proposed LSTM-AE is an Encoder-Decoder unsupervised learning model, and no label is provided in the training phase. The training process is depicted in Fig. 3(a). To address the problem of overfitting, dropout was added in both the encoder and decoder stages and set to 0.2. For the other settings in our model, "Adam" is the optimization method, "tanh" is used as the activation function, and "MAE" can be chosen as a loss function.



Fig. 3: Training and Testing Phase

## LSTM-AE Testing

Fig. 3(b) shows the details of how anomalies can be detected using the reconstruction-based anomaly detection method. Here, the maximum value of MAE from the training process can be designated as the threshold. A reconstruction error rate for each data point from the testing set is compared with this threshold. If the reconstruction loss value is greater than the threshold  $\eta$ , this data point is noted as an anomaly, otherwise, it is designated to be normal. This is shown in the following Equation 7.

$$X' = \begin{cases} X'_i \text{ is anomalies,} & \text{if } Score > \eta \\ X'_i \text{ is normal,} & \text{otherwise} \end{cases}$$
(7)

where X' indicates a reconstructed time-series,  $X'_i$  is a data point contained in the time-series, and a *score* is a result of a reconstruction loss function using MAE.

# D. Reconstruction-based anomaly detection

In order to effectively learn time series DDoS traffic behavior using reconstruction-based anomaly detection, our LSTM-AE model is trained with a dataset that contains only benign traffic flows from CICDDoS2019.



Fig. 4: Computing Reconstruction Error on multivariate Time Series

1) LSTM-AE for anomaly detection: An anomaly can be defined as an observation diverging from the majority of the data. A threshold can be set as a decision point to determine how much an observation deviates. Any observations that exceed the threshold are defined as anomalies. To better demonstrate the functionality of reconstruction-based time series anomaly detection, the LSTM-AE model can be applied to detect anomalies for each input sequence. This allows us to obtain the reconstruction error rates associated with the benign samples of the DDoS dataset. A backpropagation methodology is applied to adjust the weights and parameters of the model. We use the Mean Absolute Error (MAE) algorithm, as shown in Equation 8, as the reconstruction error (loss) function.

$$Loss(MAE) = \frac{\sum_{i=1}^{n} |x_i - \hat{x}_i|}{n}$$
(8)

where *n* indicates the total number of samples,  $x_i$  is the representation of the original input being fed to the encoder, and  $\hat{x}_i$  is the output produced by the decoder.

Once training is done and the reconstruction error is computed on all samples, the LSTM-AE model learns a low MAE and sets the maximum reconstruction error as a threshold. By contrast, if the testing set presents different behavior from the training process, the resulting MAE will be greater than the threshold and can be considered an anomaly. To the best of our knowledge of the CICDDoS2019 dataset, there are a few benign samples in each provided CSV file. In order to learn the normal behaviors of traffic flows, we extract all benign samples from the CICDDoS2019 dataset, and designate 80% of the benign traffic as the training set, while the remaining 20% of the benign traffic can be combined with different attack types to be used in the testing set.

2) Reconstruction Error threshold calculation: Our proposed reconstruction-based LSTM-AE model was trained in an unsupervised manner and aimed to minimize reconstruction error between input and output while using unlabeled data. In [25], the univariate time series LSTM-AE anomaly detection is performed using reconstruction error, shown in Fig. 4. The figure illustrates how to calculate reconstruction error for each sample contained in the different multivariate time series sequences. Suppose that there is a dataset containing 5 data samples with 2 features (shown as Input sequence data) which are 3 time-series sequences of  $[X_1, X_2, X_3]$  where each sequence contains 3 samples with 2 features over 3 different timesteps. For example, the first sequence  $X_1 \in \{[x_{1_1}, x_{1_2}], [x_{2_1}, x_{2_2}], [x_{2$  $[x_{3_1}, x_{3_2}]$ ,  $X_2 \in \{[x_{2_1}, x_{2_2}], [x_{3_1}, x_{3_2}], [x_{4_1}, x_{4_2}]\}$ , and  $X_3$  $\in \{[x_{3_1}, x_{3_2}], [x_{4_1}, x_{4_2}], [x_{5_1}, x_{5_2}]\}$ . Our model trains these 3 time series of sequences as inputs and constructs the outputs that map to each sequence  $X_1$ ,  $X_2$ , and  $X_3$ .

Using Equation 8, we can obtain the MAE value of each data sample for each feature. In order to obtain the MAE value of each data sample over multivariate features, we use the average MAE of each data sample for each feature. For example, in 3 time-series sequences of  $[X_1, X_2, X_3]$ . As such, we obtain the reconstruction error of each data sample of each feature. As can be seen in Fig. 4, with the assumption that each data sample has two features, we can calculate each data sample for each feature using time steps. This figure shows the reconstruction error calculation for each feature when the time step is 3. As an example of one of the data samples of feature 1  $(x_{3_1})$ , and the reconstruction error of  $x_{3_1}$ , the calculation is  $(|\hat{x_{3_1}} - x_{3_1}| + |\hat{x_{3_1}} - x_{3_1}| + |\hat{x_{3_1}} - x_{3_1}|)/3$ , and is defined as  $x'_{3_1}$ . Similarly, the reconstruction error of feature 2  $(x_{3_2})$  is  $(|\hat{x_{3_2}} - x_{3_2}| + |\hat{x_{3_2}} - x_{3_2}| + |\hat{x_{3_2}} - x_{3_2}|)/3$ , and is defined as  $x'_{3_2}$ . Note that the reconstructed data sample for each feature can be slightly different, which means each  $x_{3_1}$  in  $X_1$ ,  $X_2$ , and  $X_3$  are different. The  $x_{3_2}$  is presented with the same definition. After we obtain the reconstruction error of these two features, the total reconstruction error of  $x_3$  is calculated by using the average value of features 1 and 2, illustrated as  $MAE_{x_3} = (x'_{3_1} + x'_{3_2})/2$  where 2 is presented two features of  $x_3$ . When we obtain all the average reconstruction errors for each data sample on the training set, the maximum average reconstruction error is set as the threshold. During testing, any samples whose reconstruction error goes beyond this maximum average reconstruction error are therefore labeled as anomalies.

#### IV. DATA AND METHODOLOGIES

#### A. CICDDoS2019 Dataset

In this study, we use the CICDDoS2019 [16] dataset which is widely used for DDoS attack detection and classification. The dataset contains a large number of up-to-date realistic DDoS attack samples as well as benign samples. The total number of records contained in the CICDDoS2019 dataset is depicted in Table I.

The CICDDoS2019 dataset contains different DDoS attack types that exploit a wide range of network and application protocols. In our study, we used DDoS attack types (i.e.,

TABLE I: The number of records in CICDDoS2019

dataset	total	benign	malicious
Training day	50,063,112	56,863	50,006,249
Testing day	20,364,525	56,965	20,307,560

DNS) and benign traffic samples to train and test our proposed LSTM-AE model. The dataset is broken down as follows:

- **Benign**: Benign traffic based on HTTP, HTTPS, FTP, SSH, and email protocols.
- Attacks: These DDoS attacks cover two different categories, Reflection-based and Exploitation-based. In terms of our CICDDoS2019 dataset, any traffic included in MSSQL, SSDP, NTP, TFTP, DNS, LDAP, NetBIOS, and SNMP is categorised as a reflection-based attack. Traffic labeled as SYN, UDP, and UDP-lag in CIDDDoS2019 belongs to the exploitation-based category.

All CICDDoS2019 data samples from the training day set are depicted in Table II. Note that all data samples can be counted from different "CSV" files, which are collected and saved based on their various attack types. Note that the "WebDDoS" attack was collected and saved together with the "UDPLag" attack file.

TABLE II: Three Reflection-based Attack Types on Training Day

Attack Type	Malicious	Benign	Total Flow count	Attack times
DNS	5,071,011	3,402	5,074,413	10:52 - 11:05
LDAP	2,179,930	1,612	2,181,542	11:22 - 11:32
SNMP	5,159,870	1,507	5,161,377	12:12 - 12:23

The high-level description of the nature of the DDoS attack used in our study is summarised in Table III.

In this dataset, there are 88 features in total, and the best top 5 features of each attack type and benign traffics have been used based on the RandomForestRegressor class of scikit-learn which can calculate the importance of each feature in the dataset [16]. Table IV shows the top 5 important features which are employed in this research as well as a brief description.

## B. Data Pre-processing

In this section, we discuss the methodologies we used to process our dataset in order to feed it into our proposed LSTM-AE model.

1) Data Cleaning: The original dataset contained 88 features. As suggested by [16], they depicted the top 5 most important features of each attack type and benign samples based on weight and mean value calculated. According to these top 5 significant features which were selected, we also use these top-5 features as our features among different attack types. For example, we used DNS, LDAP, and SNMP as our analysis attack types, which provide three attack types (anomalies) as well as benign (normal) samples. We chose these three attack types because they shared the same top 5 important features based on their weight and the mean value calculation. There are five important features to be used in this

Attack Type	Attack Description
DNS Attack	DNS attacks are a type of amplification DDoS attack that exploits Domain Name Servers and exhausts the bandwidth of the victims. This attack can overwhelm the victims and make them inaccessible.
LDAP Attack	LDAP attacks are a DDoS attack associated with exploiting Lightweight Directory Access Protocol (LDAP) protocol. The attacker sends a massive number of LDAP requests to the vulnerable LDAP servers by pretending to be a legitimate LDAP client using spoofed IP addresses. The LDAP server becomes too busy to create responses for attackers and becomes unable to respond to real LDAP clients.
SNMP Attack	SNMP attacks are a volumetric DDoS attack that stands for Simple Network Management Protocol (SNMP), and these attacks aim to generate a large number of SNMP attacks utilizing a spoofing IP address that directed to the victims from multiple networks.

TABLE IV: Selected features in CICDDoS2019

Feature Name	Description
Max Packet Length	The maximum length of a flow
Fwd Packet Length Max	The maximum size of packets in the forward direction
Fwd Packet Length Min	The minimum size of packets in the forward direction
Average Packet Size	The average size of packets
Min Packet Length	The minimum length of a flow

research: "Max Packet Length", "Fwd Packet Length Max", "Fwd Packet Length Min", "Average Packet Size", and "Min Packet Length".

2) Label Encoding: We substituted the categorical labels for deep models as they only operate on float/numerical values. One categorical value which was converted was the attack label (benign or an attack type). For label encoding, "0" indicates a benign (normal) sample, while "1" indicates an attack type (anomalies).

3) Data Normalization: There are several widely used methods to perform feature scaling, including Z Score, standardization, and normalization. As proposed by [26], we utilize the MinMax-based normalization for our feature scaling. This method maps the original range of each feature into a new range using Equation (9), as follows:

$$Z_i = \frac{Z_i - min}{max - min} \tag{9}$$

where  $Z_i$  denotes all the normalized numeric values ranging between [0-1]; max and min denote the maximum and minimum values from all data points.

#### V. EXPERIMENTAL RESULTS

TABLE V: Implementation environment specification

Unit	Description
Processor	$3.4 \text{GH}_z$ Inter Core i5
RAM	16GB
OS	MacOS Big Sur 11.4
Packages used	tensorflow 2.0.0, sklearn 0.24.1

#### A. Experiment Setup

Our experiments were carried out using the system setup shown in Table V.

The hyperparameters used in the training phase are described (with the values for each parameter) complete with description in Table VI.

TABLE VI: LSTM-AE Training Parameters

Hyperparameters	Values	Descriptions
Learning rate Dropout	0.001	Learning speed (within range 0.0 and 1.0) No. of neurons ignored
Batch size Epoch	64 30	No. of samples in one fwd/bwd pass No. of one fwd/bwd pass of all samples

#### B. Performance Matrix

To evaluate the performance of our model, we used the following metrics: classification accuracy, precision, recall, and F1 score. Table VII illustrates the confusion matrix.

TABLE VII: Confusion Matrix

Total Population	Predicted Condition		
Total Topulation	Normal	Anomaly	
Actual Condition	Normal	TN	FP
Actual Collution	Anomaly	FN	TP

where;

- True Positive (TP) indicates an anomalous data point correctly classified as anomalous.
- True Negative (TN) indicates a normal data point correctly classified as normal.
- False Positive (FP) indicates a normal data point incorrectly classified as anomalous.
- False Negative (FN) indicates an anomalous data point incorrectly classified as normal.

Based on the aforementioned terms, the evaluation metrics are calculated as follows:

$$TPR(TruePositiveRate/Recall) = \frac{TP}{TP + FN}$$
(10)

$$FPR(FalsePositiveRate) = \frac{FP}{FP + TN}$$
 (11)

$$Precision = \frac{TP}{TP + FP}$$
(12)

$$F1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)$$
(13)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
(14)

The Area Under the Curve (AUC) computes the area under the Receiver Operating Characteristics (ROC) curve which is plotted using the true positive rate on the y-axis and the false



Fig. 5: Performance of Anomaly Detection on different DDoS attack types using the Confusion Matrix

positive rate on the x-axis over different thresholds. Mathematically, the AUC is computed as shown in Equation (15).

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{TN + FP}$$
(15)

# C. Results and Evaluations

1) Training and testing dataset: The CICDDoS2019 data collection was based on millisecond intervals. We first select 0.5% of the data based on the original millisecond time interval. However, after selection, there is a large number of malicious (anomalies) samples with only a few benign samples selected. As mentioned previously, our LSTM-AE model only uses benign samples in the training stage. Thus, to solve this issue, we first extracted all the benign samples from the training day collection and separated them into training (80%) and testing (20%) sets. For the model training, we only use 80% benign samples to train, while the rest of the benign samples can be used in the testing set. Moreover, We used 0.5% of the original CICDDoS2019 dataset for each attack type from the training day collection for testing as it was not feasible to use the full dataset due to performance considerations. Note that each testing set includes benign samples and different attack types.

We experimented with three attack types in the LSTM-AE model to get initial anomaly detection results based on the top 5 most important features for each attack type. Table IX shows the initial anomaly detection results for the three attack types: DNS, LDAP, and SNMP. The results show that the anomaly detection for LDAP attacks performed the best in terms of accuracy, precision, recall, and F1-score, based on the mentioned top 5 features.

Figure 5 illustrates the number of records classified for anomaly detection performance using the three DDoS attack types.

2) Influence of time window length: Table VIII shows the performance of all three attack types based on different time window lengths. The results show that performance was best for detecting LDAP attacks over diverse time window lengths,

with over 99% in accuracy and precision, recall, and F1-score, compared with other attack types as shown in Table VIII. As can be seen in Table VIII, as the time window length increased, the accuracy decreased for all three attack types. The highest accuracy was achieved at over 96% for a time window length of 10. We also compared the results with the baseline of other machine learning techniques from [16], in which the same features are used for testing purposes. In the comparison table (Table IX), the evaluation results show our model performed impressively in terms of accuracy, precision, recall, and F1-score.

*3) Results of batch size:* The testing results were evaluated based on different batch sizes while dropout, learning rate, and epoch remained constant.

*DNS attack:* For DNS attacks, the highest accuracy was 96.08% and required a relatively brief computational time (approximately 12s per epoch) with the batch size set to 64. We found that the smallest batch size value tested - size 10 - resulted in the lowest accuracy (88.83%) and required a relatively high computation time at approximately 42s per epoch.

*SNMP attack:* In the case of SNMP attacks as in Table XI, analyzing the performances for each batch size demonstrated that the best result was achieved when the batch size was increased to 64. Comparing this with a smaller batch size of 10, all performance metrics showed significant improvement at the larger batch size, particularly computational time. As can be seen in Table XI, a batch size of 10 (38s) takes approximately three times longer than a batch size of 64 (12s), per epoch.

LDAP attack: The performance of LDAP attacks for each hyperparameter of batch size is presented in Table XII. When analyzing the performance of each batch size, we can see that the LDAP attacks are detected with impressive performance across the different batch sizes, achieving over 98%. However, a batch size of 10 (42s) takes significantly longer (over three times greater) than a batch size of 10 (12s) per epoch.

4) Results of learning rate: The learning rate of the "Adam" optimizer is set at three candidate rates: 0.01, 0.001, and

TABLE VIII: Three Attack Types' Performance based on different time window lengths

Attack Type	w = 10ms			w = 50ms			w = 100ms					
	Acc	Pre	Re	F1	Acc	Pre	Re	F1	Acc	Pre	Re	F1
DNS	96.08	99.99	94.30	97.06	95.78	99.99	93.85	96.82	95.83	99.99	93.92	96.86
LDAP	99.96	99.99	99.93	99.96	98.68	99.99	97.28	98.61	99.77	99.99	97.47	98.71
SNMP	96.89	99.99	95.49	97.69	96.86	99.99	95.45	97.67	96.82	99.99	95.40	97.64

TABLE IX: Performance Comparison between the experimental results and Three Attack Types on the CICDDoS2019 dataset

Alaamit	Confusion Matrix				
Algorit	Acc	Pre	Re	F1	
ID3 [	-	0.78	0.65	0.69	
<b>RF</b> [1	-	0.77	0.56	0.62	
Naive Bay	-	0.41	0.11	0.05	
Logistic regre	-	0.25	0.02	0.04	
	DNS	0.96	0.99	0.94	0.97
Our model	LDAP	0.99	0.99	0.99	0.99
	SNMP	0.96	0.99	0.95	0.97

TABLE X: Batch Size Performance Metrics for Detecting DNS Attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
10	88.83	100	83.71	91.13	91.58	$42\pm1.24$
32	93.49	100	90.51	95.02	95.26	$12\pm1.03$
64	96.08	99.99	94.30	97.06	97.14	$12\pm0.94$

0.00001. This was tested over the three different attack types, while dropout and epoch values remained constant.

DNS attack: The results of the DNS attacks are shown in Table XIII, with different learning rates. The highest performance metrics for accuracy, recall, and F1-score was achieved at 96.08%, 94.30%, and 97.06% respectively. These results were all using the learning rate value of 0.001. On the other hand, the smallest learning value at 0.00001 resulted in the lowest accuracy (95.48%), recall (93.41%), and F1-score (96.70%). While selecting different learning rates has an impact on the results, the time taken is not much different per epoch.

*SNMP attack:* In examining the performance of detection on the SNMP attack in Table XIV, we can see that the best performance for accuracy was achieved at 96.89% using the learning rate of 0.001. Moreover, the results of all performance metrics decrease slightly as the learning rate changed from 0.001 to 0.00001. The processing time did not vary significantly due to the changes in the learning rate.

LDAP attack: As shown in Table XV, the LDAP attack performed the best over our experiments when compared to the other attack types. The results of detection overall metrics - accuracy, precision, recall, F1, and AUC-ROC - provided the smallest variation over LDAP attacks using the different learning rates, all scores being over 99%. The processing times showed there was a slightly greater cost using the smallest learning rate of 0.00001.

The impact on the different learning rate configurations presents little variation, but a learning rate of 0.001 gave the best results for accuracy. Similarly, the lowest value for the

TABLE XI: Batch Size Performance Metrics for Detecting SNMP Attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
10	88.47	99.99	83.29	90.88	91.64	$38 \pm 1.85$
32	96.84	99.98	95.44	97.66	97.71	$14 \pm 1.31$
64	96.89	99.99	95.49	97.69	97.75	$12\pm0.96$

TABLE XII: Batch Size Performance Metrics for Detecting LDAP Attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
10	98.82	99.99	97.57	98.76	98.78	$42 \pm 1.93$
32	98.55	99.99	97.02	98.48	98.51	$14\pm0.80$
64	99.96	99.99	99.93	99.96	99.96	$12\pm0.82$

TABLE XIII: Learning Rate Performance Metrics for Detecting DNS Attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
0.001	96.08	99.99	94.30	97.06	97.14	$10\pm0.99$
0.0001	95.80	99.99	93.88	96.84	96.94	$10\pm0.84$
0.00001	95.48	99.99	93.41	96.59	96.70	$10\pm1.05$

TABLE XIV: Learning Rate Performance Metrics for Detecting SNMP Attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
0.001	96.89	99.99	95.49	97.69	97.75	$12\pm0.96$
0.0001	96.82	99.99	95.39	97.63	97.69	$12\pm0.99$
0.00001	96.88	99.99	95.48	97.68	97.74	$12\pm1.05$

learning rate took a long time to converge, which results in a significantly longer time for computation per epoch.

TABLE XV: Learning Rate Performance Metrics for Detecting LDAP Attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$ /epoch
0.001	99.96	99.99	99.93	99.96	99.96	$11\pm0.91$
0.0001	99.94	99.99	99.89	98.94	99.95	$11\pm0.90$
0.00001	99.96	99.99	99.93	99.96	99.96	$12\pm0.94$

Figure 6 shows the AUC-ROC for anomaly detection over our three DDoS attack types using different time window length selections. Note that all three attack types achieved over 97% for the AUC, and that the highest value for the AUC-ROC was achieved for a time window length of 10. Our experimental evidence shows that the proposed model offers



significant potential to detect DDoS attacks effectively.

Table XVI shows the performance comparison for our proposed LSTM-AE model with other similar methods from shallow machine learning and deep learning-based approaches. As the results demonstrate, our approach offered the best performance of LDAP in terms of all aspects of evaluation metrics, reaching an average of 99.96% accuracy while precision, recall, and F1-score all remain very competitive at 99.99%, 99.93%, and 99.96% respectively.

## VI. CONCLUSION

In this study, we demonstrate that DDoS attacks can be detected with high accuracy using a combination of multiple deep learning-based techniques. Our proposed reconstructionbased LSTM-AE anomaly detection model leverages the benefits of an LSTM model and an Autoencoder in order to detect DDoS traffic anomalies. We use LSTM networks consisting of multiple LSTM units that work with each other to learn the long short-term correlation of DDoS traffic within a time series sequence. An Autoencoder is employed to identify the optimal threshold based on the reconstruction error rates. This can be used to identify anomalies in traffic. We have demonstrated the impact of different window lengths for classifying anomalies over different DDoS attack types. Our proposed model offers potential as an effective DDoS defense tool to assist in detecting a massively growing number of DDoS anomalies. Our model has been comprehensively and extensively tested against three different DDoS attack types. The evaluation results demonstrate high levels of performance on different time window lengths over many performance metrics including precision (99%), recall (99%), F1-score (99%), and accuracy (99%). Our model performed best for the LDAP attack detection case against all performance metrics, exceeding 99% and outperforming other state-of-the-art methods.

## REFERENCES

 S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, and J. Iqbal, "A deep cnn ensemble framework for efficient ddos attack detection in software defined networks," *Ieee Access*, vol. 8, pp. 53 972–53 983, 2020.

- [2] K. S. Sahoo, S. K. Panda, S. Sahoo, B. Sahoo, and R. Dash, "Toward secure software-defined networks against distributed denial of service attack," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 4829–4874, 2019.
- [3] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in 2017 IEEE international conference on smart computing (SMARTCOMP). IEEE, 2017, pp. 1–8.
- [4] S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, "Network flow based iot botnet attack detection using deep learning," in *IEEE INFO-COM 2020-IEEE conference on computer communications workshops* (*INFOCOM WKSHPS*). IEEE, 2020, pp. 189–194.
- [5] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.
- [6] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown ddos attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016.
- [7] P. P. M. Alethea Toh, Anupam Vij and S. Pasha, "Azure ddos protection—2020 year in review," 2021. [Online]. Available: https://azure. microsoft.com/en-us/blog/azure-ddos-protection-2020-year-in-review/
- [8] Microsoft, "Microsoft digital defense report," 2021.
- [9] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, 2020.
- [10] S. Yeom, C. Choi, and K. Kim, "Lstm-based collaborative source-side ddos attack detection," *IEEE Access*, vol. 10, pp. 44 033–44 045, 2022.
- [11] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A hybrid cnn-lstm based approach for anomaly detection systems in sdns," in *The 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–7.
- [12] M. A. Salahuddin, V. Pourahmadi, H. A. Alameddine, M. F. Bari, and R. Boutaba, "Chronos: Ddos attack detection using time-based autoencoder," *IEEE Transactions on Network and Service Management*, 2021.
- [13] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in 10th IEEE International Conference on Network Protocols, 2002. Proceedings. IEEE, 2002, pp. 312–321.
- [14] S. Alzahrani and L. Hong, "Detection of distributed denial of service (ddos) attacks using artificial intelligence on cloud," in 2018 IEEE World Congress on Services (SERVICES). IEEE, 2018, pp. 35–36.
- [15] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *Ieee Access*, vol. 7, pp. 41 525–41 550, 2019.
- [16] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in 2019 International Carnahan Conference on Security Technology (ICCST). IEEE, 2019, pp. 1–8.
- [17] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: an intelligent edge defense mechanism against iot ddos attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.
- [18] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proenca, "Long shortterm memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83765–83781, 2020.

Paper	Techniques	Accuracy	Precision	Recall	F1-score
Novaes et al. [18]	LSTM-Fuzzy	-		93.13	
Con et al [27]	DDoSNet 24 features	-	91.12	72.91	74.00
	DDosNet 82 features + FS	-	91.16	79.41	79.39
Elsayed et al. [28]	DDosNet (RNN+AE)	99	99	99	99
liao et al. [29]	K-means + Active Learning Method	90	-	-	95
Alghazzawi et al. [30]	CNN + BiLSTM	94.52	94.74	92.04	93.44
Jia et al. [17]	LSTM	98.9	99.47	99.31	99.35
Gniewkowski et al. [31]	LSTMED	89.7	44.8	88.3	59.4
Sayed et al. [32]	LSTM(scenario II)	88.5	88.1	87.8	87.0
This paper	LSTM-AE		99.99	99.93	99.96

TABLE XVI: Comparison of CICDDoS2019 with Different Algorithms

- [19] H. Aydın, Z. Orman, and M. A. Aydın, "A long short-term memory (lstm)-based distributed denial of service (ddos) detection and defense system design in public cloud network environment," *Computers & Security*, vol. 118, p. 102725, 2022.
- [20] S. M. T. Nezhad, M. Nazari, and E. A. Gharavol, "A novel dos and ddos attacks detection algorithm using arima time series model and chaotic system in computer networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 700–703, 2016.
- [21] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE transactions on neural networks and learning* systems, vol. 31, no. 8, pp. 3127–3141, 2019.
- [22] R. F. Fouladi, O. Ermiş, and E. Anarim, "Anomaly-based ddos attack detection by using sparse coding and frequency domain," in 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). IEEE, 2019, pp. 1–6.
- [23] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using 1stm based autoencoder," in *Proceedings of the* 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, 2020, pp. 37–45.
- [24] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.
- [25] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder based anomaly detection for indoor air quality time series data," arXiv preprint arXiv:2204.06701, 2022.
- [26] S. ur Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, Z. Jalil, and A. K. Bashir, "Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru)," *Future Generation Computer Systems*, vol. 118, pp. 453–466, 2021.
- [27] D. C. Can, H. Q. Le, and Q. T. Ha, "Detection of distributed denial of service attacks using automatic feature selection with enhancement for imbalance dataset," ACIIDS 2021, 2021.
- [28] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in 2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks" (WoWMoM). IEEE, 2020, pp. 391–396.
- [29] N. Liao and X. Li, "Traffic anomaly detection model using k-means and active learning method," *International Journal of Fuzzy Systems*, pp. 1–19, 2022.
- [30] D. Alghazzawi, O. Bamasag, H. Ullah, and M. Z. Asghar, "Efficient detection of ddos attacks using a hybrid deep learning model with improved feature selection," *Applied Sciences*, vol. 11, no. 24, p. 11634, 2021.
- [31] M. Gniewkowski, H. Maciejewski, and T. Surmacz, "Anomaly detection techniques for different ddos attack types," in *International Conference* on Dependability and Complex Systems. Springer, 2022, pp. 63–78.
- [32] M. I. Sayed, I. M. Sayem, S. Saha, and A. Haque, "A multi-classifier for ddos attacks using stacking ensemble deep neural network," in 2022 International Wireless Communications and Mobile Computing (IWCMC). IEEE, 2022, pp. 1125–1130.