Hybrid Deep Learning for Botnet Attack Detection in the Internet of Things Networks

Segun I. Popoola, Bamidele Adebisi, Mohammad Hammoudeh, Guan Gui, Haris Gacanin

Abstract—**Deep Learning (DL) is an efficient method for botnet attack detection. However, the volume of network traffic data and memory space required is usually large. It is, therefore, almost impossible to implement the DL method in memory-constrained IoT devices. In this paper, we reduce the feature dimensionality of large-scale IoT network traffic data using the encoding phase of Long Short-Term Memory Autoencoder (LAE). In order to classify network traffic samples correctly, we analyse the long-term inter-related changes in the low-dimensional feature set produced by LAE using deep Bidirectional Long Short-Term Memory (BLSTM). Extensive experiments are performed with the BoT-IoT dataset to validate the effectiveness of the proposed hybrid DL method. Results show that LAE significantly reduced the memory space required for large-scale network traffic data storage by** 91.89%**, and it outperformed state-of-the-art feature dimensionality reduction methods by** $18.92 - 27.03\%$**. Despite the significant reduction in feature size, the deep BLSTM model demonstrates robustness against model under-fitting and over-fitting. It also achieves good generalisation ability in binary and multi-class classification scenarios.**

Index Terms—**Internet of Things, botnet detection, dimensionality reduction, Long Short-Term Memory, autoencoder.**

I. Introduction

NOWADAYS, critical infrastructures such as power generation [1]–[4], communications [5]–[7], healthcare [8]–[10], manufacturing [11]–[13], transportation [14], [15], water treatment [16] and agriculture [17], [18] are interconnected in order to tap into the various benefits of the Internet of Things (IoT) [19]–[21]. The increased inter-connectivity and the use of Industrial Control Systems (ICS) renders smart critical infrastructures vulnerable to cyberattacks from terrorists or “hacktivists”. For instance, ICS and IoT devices have been proven to be easily hackable and remotely controllable to form IoT-based botnets [22], [23]. Successful exploitation of a single vulnerable IoT device can lead to leakage of sensitive information and serious security breaches in the wider IoT-enabled system [24]. This makes them an attractive target to Advanced Persistent Threats (APT) of diverse botnet attacks, especially when they are deployed in critical environments.

To secure connected IoT devices against complex botnet attacks, Machine Learning (ML) techniques have been employed to develop Network Intrusion Detection Systems (NIDS), e.g., [25]. Such NIDS can be installed at strategic points within an IoT network. Specifically, Deep Learning (DL), an advanced ML approach, offers a unique capability for automatic extraction of features from large-scale, high-speed network traffic generated by interconnected heterogeneous IoT devices [26]. Considering the resource-constraints in IoT devices, NIDS techniques used in classical computer networks are not efficient for botnet detection in IoT systems due to high computation and memory requirements [27]. In order to develop an efficient DL method for botnet detection in IoT networks, sufficiently large network traffic information is needed to guarantee efficient classification performance [28]. However, processing and analyzing high-dimensional network traffic data can lead to *curse of dimensionality* [29]. Also, training DL models with such high-dimensional data can cause *Hughes phenomena* [30]. High-dimensional data processing is complex and requires huge computational resources and storage capacity [31], [32]. IoT devices do not have sufficient memory space to store big network traffic data required for DL. Therefore, there is a need for end-to-end DL-based botnet detection method that will reduce high dimensionality of big network traffic features and also detect complex and recent botnet attacks accurately based on low-dimensional network traffic information.

Currently, Bot-IoT dataset [33] is the most relevant publicly available dataset for botnet attack detection in IoT networks because it: (a) has IoT network traffic samples; (b) captured complete network information; (c) has a diversity of complex IoT botnet attack scenarios; (d) contains accurate ground truth labels; and (e) provides massive volume of labeled data required for effective supervised DL. The original feature dimensionality[1] of the Bot-IoT dataset is 43, and the memory space required to store this network traffic data is 1.085 GB. So far, feature dimensionality reduction methods that have been applied to the Bot-IoT dataset were all based on feature selection techniques. These techniques include the filter

Manuscript received April 16, 2020; revised August 27, 2020.

This work is supported in part by Cyraatek Ltd UK, the Faculty of Science & Engineering, Manchester Metropolitan University, and in part by ENERGY-IQ project, a UK-Canada Power Forward Smart Grid Demonstrator project funded by The Department for Business, Energy and Industrial Strategy (BEIS) under Grant 7454460; and the NICE (Nigerian Intelligent Clean Energy) Marketplace project funded by the Department for International Development (DFID).

S. I. Popoola and B. Adebisi are with the Department of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom. E-mail: s.popoola@mmu.ac.uk; b.adebisi@mmu.ac.uk

M. Hammoudeh is with the Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom. E-mail: m.hammoudeh@mmu.ac.uk

G. Gui is with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, 210003 China. E-mail: guiguan@njupt.edu.cn

H. Gacanin is with the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany. E-mail: harisg@ice.rwth-aachen.de

[1]Feature dimensionality is the total number of network traffic features in a given dataset

method with Pearson Correlation and Entropy (PCE) [33], X-Mean clustering with Particle Swarm Optimisation (XMP) [34], and Information Gain (IFG) [35].

On the other hand, Long Short-Term Memory Autoencoder (LAE) is an effective DL method that produces a low-dimensional latent-space feature representation of a high-dimensional feature set at its hidden layer. To the best of our knowledge, this DL method has not been previously applied to reduce the dimensionality of the feature set in the Bot-IoT dataset. Also, deep Bidirectional Long Short-Term Memory (BLSTM) is a DL method that learns hierarchical feature representations and long-term inter-related changes directly from raw data using multiple hidden layers. However, this DL method has not been previously used to classify latent-space representation of network traffic features.

In this paper, we propose a hybrid DL framework, called LAE-BLSTM, for efficient botnet detection in IoT networks using LAE and deep BLSTM algorithms. The main contributions of this paper are as follows:

1) A single hidden-layered LAE is proposed for feature dimensionality reduction. This method reduces the dimensionality of large-scale IoT network traffic data and produces a low-dimensional latent-space feature representation at the hidden layer without loosing useful intrinsic network information;
2) A deep BLSTM is proposed for network traffic classification. This method analyses the long-term inter-related changes in low-dimensional feature set produced by LAE to distinguish botnet attack traffic from benign traffic in IoT networks;
3) Extensive experiments are performed with the Bot-IoT dataset to validate the effectiveness of LAE-BLSTM in binary and multi-class classification scenarios.
4) The performance of state-of-the-art optimisation algorithms was investigated and compared to ensure efficient feature dimensionality reduction and botnet attack detection in IoT networks.

The remaining parts of the paper is organized as follows: In Section II, we review related state-of-the-art methods for feature dimensionality reduction and network traffic classification. In Section III, we describe the hybrid DL framework (LAE-BLSTM) proposed for botnet detection in IoT networks. Extensive experiments are performed in Section IV to validate the effectiveness of LAE-BLSTM. Experimentation results are presented and discussed in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Although several datasets are available for network intrusion detection, they have various challenges, including lack of reliable labels, low attack diversity, redundancy of network traffic, and missing ground truth. For instance, KDD Cup99 and NSL-KDD datasets are popularly used, but they are outdated, and they do not reflect current normal and attack scenarios [36], [37]. The application of the DEFCON-8 dataset is limited because of the low number of benign traffic samples [36]. The attack scenarios in the UNIBS dataset are limited to DoS; the network traffic data are presented in packets with no extracted features, and the labels were not provided [38]. CAIDA datasets have no ground truth information about the attack samples [38]. Network traffic samples in the LBNL dataset were not labeled, and the features were not extracted from the packet files [38]. Attack samples in the UNSW-NB15 dataset were generated in a synthetic environment [39]. ISCX and CICIDS2017 datasets were generated based on the concept of profiling, and this can be due to their innate complexity. Also, the ground truth of these datasets is not available to enhance the labeling process. Not much information is given about the botnet scenarios that were used in most datasets [36], [38]–[40]. Also, IoT network traffic data was not included in related datasets [36], [39], [41]–[44].

Bot-IoT dataset [33] is the most relevant dataset that is publicly available for network-based botnet attack detection in IoT networks. To realise this dataset, an IoT network testbed was set up to generate benign and malicious network traffic using heterogeneous communication protocols which include User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), Internet Protocol version-6 ICMP (IPv6-ICMP), Internet Group Management Protocol (IGMP), and Reverse Address Resolution Protocol (RARP). The testbed setup comprised a variety of IoT devices, including a weather station, smart fridge, motion-activated lights, remotely-activated garage door, and smart thermostat. Also, millions of IoT botnet attack traffic samples were included in Bot-IoT. These attack traffic samples can be categorized into four IoT botnet scenarios, namely: DDoS, DoS, reconnaissance and information theft.

Feature dimensionality reduction is mostly achieved by applying either linear or non-linear transformation technique to high-dimensional feature set. Principal Component Analysis (PCA) [45] is one of the common linear transformation methods while kernel methods [46], spectral methods [47] and DL methods [48] employ non-linear transformation techniques. Autoencoder is an unsupervised DL method that produces latent-space representation of input data at the hidden layer. Different autoencoder architectures have been proposed to reduce the feature dimensionality in most popular network intrusion datasets. These methods were implemented and evaluated with the network traffic data in publicly available datasets which include KDD-Cup99 [49]–[56], NSL-KDD [49], [57]–[60], UNSW-NB15 [50], [59], [61] and CICIDS2017 [62]. Table I shows the autoencoder-based feature dimensionality reduction techniques in the literature. The original feature dimensionality, feature reduction method, new feature dimensionality, classifier and classification scenarios were provided. Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Network (RNN) and it has the capacity to learn long-term dependencies in network traffic features [63]–[65]. However, none of the proposed autoencoder-based methods in Table I was implemented nor validated with Bot-IoT dataset.

Different feature selection methods have been proposed to reduce the dimensionality of network traffic features in Bot-IoT dataset. Table II presents an overview of state-

TABLE I
DIMENSIONALITY REDUCTION OF NETWORK TRAFFIC FEATURES USING AUTOENCODER

Dataset	Ref.	Input features	Reduction method	Output features	Classifier	Classification scenarios
KDD-Cup99	[49]	41	Stacked deep autoencoder	28	RF	Binary, multi-class
	[50]	41	Stacked deep autoencoder	10	Softmax	Binary, multi-class
	[51]	122	Autoencoder	100	CNN, softmax	Multi-class
	[52]	41	Autoencoder	-	k-means	Binary
	[53]	41	Stacked autoencoder	13, 5	Decision tree	Binary
	[54]	41	Variational autoencoder	20	Dictionary	Binary
	[55]	41	Autoencoder	18	k-means	Multi-class
	[56]	39	Autoencoder	3	k-means	Binary
NSL-KDD	[49]	41	Stacked deep autoencoder	28	RF	Binary, multi-class
	[57]	122	Stacked sparse autoencoder	-	SVM	Binary, multi-class
	[58]	115	Stacked sparse autoencoder	10	Logistic	Binary, multi-class
	[59]	41	Deep autoencoder	3	Deep FFNN	Binary, multi-class
	[60]	52	Autoencoder	2	-	Multi-class
UNSW-B15	[50]	42	Stacked deep autoencoder	10	Softmax	Binary, multi-class
	[61]	207	Semi-supervised autoencoder	2	Decision tree	Binary, multi-class
	[59]	41	Deep autoencoder	3	Deep FFNN	Binary, multi-class
CICIDS2017	[62]	81	Stacked sparse autoencoder	64	RF	Binary, multi-class
Bot-IoT	**This paper**	**37**	**LAE**	**6**	**Deep BLSTM**	**Binary, multi-class**

TABLE II
DIMENSIONALITY REDUCTION OF NETWORK TRAFFIC FEATURES IN BOT-IOT DATASET

Ref.	Method	Feature size	Classifier	Classification scenarios
[33]	PCE	10	SVM RNN LSTM	Binary
[34]	XMP	10	SVM DNN C4.5 decision tree	Binary
[35]	IFG	13	C5 decision tree, One-class SVM	Multi-class
This paper	**LAE**	**6**	**Deep BLSTM**	**Binary, Multi-class**

of-the-art feature dimensionality reduction methods that are related to this paper. Koroniotis et al. [33] employed PCE method for feature selection. The authors reported that 10 optimal network traffic features were selected. These include *seq*, *stddev*, *N_IN_Conn_P_SrcIP*, *min*, *state_number*, *mean*, *N_IN_Conn_P_DstIP*, *drate*, *srate* and *max*. Support Vector Machine (SVM), Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) models were trained with the optimal features to perform binary classification in general and specific attack detection scenarios. Asadi et al. [34] identified optimal feature clusters and eliminated outliers using XMP technique. Most relevant network traffic features were selected based on the XMP method. The best binary classification performance was recorded when SVM, Dense Neural Network (DNN) and C4.5 decision tree classifiers were trained with 10 network traffic features. These network traffic features include *Proto*, *state_number*, *dur*, *mean*, *dpkts*, *drate*, *TnBPSrcIP*, *TnP_PSrcIP*, *TnP_Per_Dport* and *N_IN_Conn_P_DstIP*. Khraisat et al. [35] selected 13 network traffic features using information gain (entropy) method. These features include *dport*, *seq*, *dur*, *flgs_number*, *flgs*, *sport*, *N_IN_Conn_P_DstIP*, *srate*, *AR_P_Proto_P_Sport*, *daddr*, *TnBPDstIP*, *rate* and *AR_P_Proto_P_SrcIP*. An ensemble classifier, which comprised of C5 decision tree and One-Class SVM (OCSVM) models, was trained with the selected network traffic features to perform multi-label classification.

To ensure a fair comparison, feature dimensionality reduction methods that do not include benign network traffic traces and all the four botnet attack scenarios in the Bot-IoT dataset were not included in this paper. For instance, Soe et al. [66] did not consider the DoS attack scenario. Also, the performance of the method in detecting benign network traffic was not reported. In a similar work [67], the authors did not evaluate the performance of the proposed method. In another work [68], Guerra-Manzanares et al. did not evaluate the performance of the proposed method with the network traffic data in the Bot-IoT dataset.

In summary, the state-of-the-art methods in the related work focused on the selection of specific features from available network traffic information available in the Bot-IoT dataset. However, this approach may likely affect the efficiency of botnet attack detection in IoT networks because the classifiers will not have access to some relevant network information during training, validation, and testing. Consequently, the feature selection approach may lead to low botnet attack detection accuracy and a high false alarm rate in IoT networks. On the other hand, LAE reduces the dimensionality of big IoT network traffic data and produces a low-dimensional latent-space feature representation at the hidden layer without loosing useful intrinsic network information.

III. PROPOSED METHOD FOR BOTNET ATTACK DETECTION IN IOT NETWORKS

In this section, we propose a hybrid DL framework, named LAE-BLSTM, for efficient botnet attack detection in IoT networks. The description of LAE and deep BLSTM methods is presented in Algorithms 1 and 2, respectively. In this paper, boldface uppercase alphabets and boldface lower case alphabets represent matrices and column vectors respectively.

Algorithm 1: LAE algorithm

Input: $\mathbf{X}$
Initialization: $\mathbf{h}_0(j) = \mathbf{x}(j), \forall j \in [1, k]$
$h_d(0) = x(d),\ \forall d \in [1, n]$
Output: $\tilde{\mathbf{X}}$

for $d = 1$ *to* n **do**
  for $j = 1$ *to* k **do**
    $\mathbf{i}_j = \sigma_r \left(\mathbf{W}_{ix}\mathbf{x}_j + \mathbf{W}_{ih}\mathbf{h}_{j-1} + \mathbf{b}_i\right)$
    $\mathbf{f}_j = \sigma_r \left(\mathbf{W}_{fx}\mathbf{x}_j + \mathbf{W}_{fh}\mathbf{h}_{j-1} + \mathbf{b}_f\right)$
    $\tilde{\mathbf{c}}_j = \sigma_h \left(\mathbf{W}_{\tilde{c}x}\mathbf{x}_j + \mathbf{W}_{\tilde{c}h}\mathbf{h}_{j-1} + \mathbf{b}_{\tilde{c}}\right)$
    $\mathbf{c}_j = \mathbf{D}_i \odot \tilde{\mathbf{c}}_j + \mathbf{D}_f \odot \mathbf{c}_{j-1}$
    $\tilde{\mathbf{x}}_j = k_\phi(\mathbf{x}_j, \mathbf{c}_{j-1})$
  end
end
$\tilde{\mathbf{X}} = \left\{\left\{\tilde{\mathbf{x}}_{d,j}\right\}_{j=1}^{k}\right\}_{d=1}^{n}$
$L = \theta\left(\mathbf{X}, \tilde{\mathbf{X}}\right) = \left[\theta\left(\mathbf{X}_d, \tilde{\mathbf{X}}_d\right)\right]_{d=1}^{n}$

Algorithm 2: Deep BLSTM algorithm

Input: $\tilde{\mathbf{X}}$
Target: $\mathbf{y}$
Initialization: $\mathbf{h}_0(j) = \mathbf{x}(j), \forall j \in [1, u]$
$h_d(0) = x(d),\ \forall d \in [1, n]$
Output: $\tilde{\mathbf{y}}$

for $d = 1$ *to* n **do**
  for $j = 1$ *to* u **do**
    %% Forward direction
    $\overrightarrow{\mathbf{i}}_j = \sigma_r \left(\overrightarrow{\mathbf{W}}_{ix}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{ih}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_i\right)$
    $\overrightarrow{\mathbf{f}}_j = \sigma_r \left(\overrightarrow{\mathbf{W}}_{fx}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{fh}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_f\right)$
    $\overrightarrow{\tilde{\mathbf{c}}}_j = \sigma_h \left(\overrightarrow{\mathbf{W}}_{\tilde{c}x}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{\tilde{c}h}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_{\tilde{c}}\right)$
    $\overrightarrow{\mathbf{c}}_j = \overrightarrow{\mathbf{i}}_j \odot \overrightarrow{\tilde{\mathbf{c}}}_j + \overrightarrow{\mathbf{f}}_j \odot \overrightarrow{\mathbf{c}}_{j-1}$
    $\overrightarrow{\mathbf{o}}_j = \sigma_r \left(\overrightarrow{\mathbf{W}}_{ox}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{oh}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_o\right)$
    $\overrightarrow{\mathbf{h}}_{1,j} = \overrightarrow{\mathbf{o}}_j \odot \sigma_h\left(\overrightarrow{\mathbf{c}}_j\right)$
    $\overrightarrow{\mathbf{h}}_{2,j} = \sigma_h \left(\overrightarrow{\mathbf{W}}_{hx1}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh1}\overrightarrow{\mathbf{h}}_{1,j} + \overrightarrow{\mathbf{b}}_{h1}\right)$
    $\overrightarrow{\mathbf{h}}_{3,j} = \sigma_h \left(\overrightarrow{\mathbf{W}}_{hx2}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh2}\overrightarrow{\mathbf{h}}_{2,j} + \overrightarrow{\mathbf{b}}_{h2}\right)$
    $\overrightarrow{\mathbf{h}}_{4,j} = \sigma_h \left(\overrightarrow{\mathbf{W}}_{hx3}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh3}\overrightarrow{\mathbf{h}}_{3,j} + \overrightarrow{\mathbf{b}}_{h3}\right)$
  end
  %% Backward direction
  $\overleftarrow{\mathbf{i}}_j = \sigma_r \left(\overleftarrow{\mathbf{W}}_{ix}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{ih}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_i\right)$
  $\overleftarrow{\mathbf{f}}_j = \sigma_r \left(\overleftarrow{\mathbf{W}}_{fx}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{fh}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_f\right)$
  $\overleftarrow{\tilde{\mathbf{c}}}_j = \sigma_h \left(\overleftarrow{\mathbf{W}}_{\tilde{c}x}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{\tilde{c}h}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_{\tilde{c}}\right)$
  $\overleftarrow{\mathbf{c}}_j = \overleftarrow{\mathbf{i}}_j \odot \overleftarrow{\tilde{\mathbf{c}}}_j + \overleftarrow{\mathbf{f}}_j \odot \overleftarrow{\mathbf{c}}_{j-1}$
  $\overleftarrow{\mathbf{o}}_j = \sigma_r \left(\overleftarrow{\mathbf{W}}_{ox}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{oh}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_o\right)$
  $\overleftarrow{\mathbf{h}}_{1,j} = \overleftarrow{\mathbf{o}}_j \odot \sigma_h\left(\overleftarrow{\mathbf{c}}_j\right)$
  $\overleftarrow{\mathbf{h}}_{2,j} = \sigma_h \left(\overleftarrow{\mathbf{W}}_{hx1}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh1}\overleftarrow{\mathbf{h}}_{1,j} + \overleftarrow{\mathbf{b}}_{h1}\right)$
  $\overleftarrow{\mathbf{h}}_{3,j} = \sigma_h \left(\overleftarrow{\mathbf{W}}_{hx2}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh2}\overleftarrow{\mathbf{h}}_{2,j} + \overleftarrow{\mathbf{b}}_{h2}\right)$
  $\overleftarrow{\mathbf{h}}_{4,j} = \sigma_h \left(\overleftarrow{\mathbf{W}}_{hx3}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh3}\overleftarrow{\mathbf{h}}_{3,j} + \overleftarrow{\mathbf{b}}_{h3}\right)$
end
$\tilde{\mathbf{y}} = \sigma_y\left(\overrightarrow{\mathbf{W}}_y\overrightarrow{\mathbf{h}_4} + \overleftarrow{\mathbf{W}}_y\overleftarrow{\mathbf{h}_4} + \mathbf{b}_y\right)$ $\tilde{\mathbf{y}} = \left\{\tilde{\mathbf{y}}_d\right\}_{d=1}^{n}$
$\theta\left(\mathbf{y}, \tilde{\mathbf{y}}\right) = \left[\theta\left(\mathbf{y}_d, \tilde{\mathbf{y}}_d\right)\right]_{d=1}^{n}$

A. LSTM Autoencoder

Autoencoder is an unsupervised DL method which analyses the dynamic relationships between the features of high-dimensional data and produces a low-dimensional latent-space representation. Unlike the conventional Autoencoder, LAE employs LSTM units to model the long-term inter-related changes in network traffic features. In this subsection, LAE method is developed to reduce feature dimensionality of big IoT network traffic data and obtain a low-dimensional latent-space feature representation with minimum reconstruction error.

A sequence of network traffic features is represented by a three-dimensional matrix, $\mathbf{X} \in \mathbb{R}^{n\times 1\times k}$, in Equation 1:

$$\mathbf{X} = \left\{\left\{\mathbf{x}_{d,j}\right\}_{j=1}^{k}\right\}_{d=1}^{n}, \tag{1}$$

where $\{\mathbf{X}_d\}_{d=1}^{n} = \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, $k \in \mathbb{Z}^+$ is the number of network traffic features, and $n \in \mathbb{Z}^+$ is the total instances of network traffic available in the dataset. An instance of network traffic is represented by Equation 2:

$$\{\mathbf{x}_j\}_{j=1}^{k} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \tag{2}$$

where $\mathbf{x}_j \in \mathbb{R}^n$ is a network traffic feature vector.

The encoder part of a single hidden layered LSTM autoencoder [69] was used to compress the network traffic feature matrix given by Equation 2 with the aim of reducing its dimensionality without loosing the information contained in the original data. LSTM has the capability to learn long time-dependencies with its feedback connections and a recurrent memory unit that is controlled by three gates, namely, input gate, a forget gate and an output gate [70]. LAE accepts high-dimensional network traffic feature set, $\mathbf{X}$, and produces a low-dimensional latent-space representation, $\tilde{\mathbf{X}}$, at the hidden layer. The input gate vector ($\mathbf{i}_j$), forget gate vector ($\mathbf{f}_j$), memory cell state vector ($\mathbf{c}_j$), output gate vector ($\mathbf{o}_j$) and hidden state vector ($\mathbf{h}_j$) were formed based on the LAE algorithm presented in Algorithm 1. The dimension of the column vectors is represented by $\mathbf{i}_j, \mathbf{f}_j, \mathbf{c}_j, \mathbf{o}_j, \mathbf{h}_j \in \mathbb{R}^u$, where u is the number of LSTM hidden units that represent the desired network traffic feature dimensionality.

Weight matrices and bias vectors were obtained by training the LAE using the Back Propagation Through Time (BPTT) algorithm [71]. The weight matrices for the connections between the input and the recurrent gates are given as $\mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{\tilde{c}x}, \mathbf{W}_{ox} \in \mathbb{R}^{u\times n}$, and these are the weight matrices of input-to-input gate connection, input-to-forget gate connection, input-to-cell connection and input-to-output gate connection respectively. Similarly, weight matrices for connections between the recurrent gates and the hidden state are given as $\mathbf{W}_{ih}, \mathbf{W}_{fh}, \mathbf{W}_{\tilde{c}h}, \mathbf{W}_{oh} \in \mathbb{R}^{u\times u}$, and these are the weight matrices of input gate-to-hidden state connection, forget gate-to-hidden state connection, cell state-to-hidden state connection and output gate-to-hidden state connection respectively. On the other hand, $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_{\tilde{c}}, \mathbf{b}_o \in \mathbb{R}^u$ are the bias vectors of input gate, forget gate, cell state and output gate respectively. Recurrent activation function is a sigmoid function and it is represented by σ_r. Hidden layer activation function is represented by σ_h. Diagonal matrices, $\mathbf{D}_i$ and $\mathbf{D}_f$, are formed with input gate vector and forget gate vectors as represented by Equations 3 and 4 respectively:

$$\mathbf{D}_i = diag(\mathbf{i}) = \begin{bmatrix} i_1 & & \\ & \ddots & \\ & & i_h \end{bmatrix}, \tag{3}$$

$$\mathbf{D}_f = diag(\mathbf{f}) = \begin{bmatrix} f_1 & & \\ & \ddots & \\ & & f_h \end{bmatrix}. \tag{4}$$

For $\mathbf{x}_j$, the encoded output is given in Equation 5.

$$\tilde{\mathbf{x}}_j = k_\phi(\mathbf{x}_j, \mathbf{c}_{j-1}), \tag{5}$$

where $\mathbf{c}_{j-1}$ is the previous cell state vector and k_ϕ is the encoding function. Finally, the low-dimensional network traffic feature matrix is represented by Equation 6:

$$\tilde{\mathbf{X}} = \left\{ \left\{ \tilde{\mathbf{x}}_{d,j} \right\}_{j=1}^{u} \right\}_{d=1}^{n}. \tag{6}$$

B. Bidirectional LSTM

Conventional LSTM is unidirectional and it can only capture the dependence of the current state based on previous context [72]. Meanwhile, BLSTM has full access to both past and future sequential information using two LSTM hidden layers to scan input data sequence in positive and negative time directions respectively [73]. Therefore, a deep BLSTM method is developed to efficiently detect IoT botnet attack traffic by analysing the long-term inter-related changes in low-dimensional features produced by LAE.

Reduced network traffic feature set, $\tilde{\mathbf{X}}$, and its corresponding target vector, $\tilde{\mathbf{y}}$, were fed into a deep BLSTM model to produce input gate vectors ($\overrightarrow{\mathbf{i}}_j$, $\overleftarrow{\mathbf{i}}_j$), forget gate vectors ($\overrightarrow{\mathbf{f}}_j$, $\overleftarrow{\mathbf{f}}_j$), memory cell state vectors ($\overrightarrow{\mathbf{c}}_j$, $\overleftarrow{\mathbf{c}}_j$), output gate vectors ($\overrightarrow{\mathbf{o}}_j$, $\overleftarrow{\mathbf{o}}_j$) and hidden state vectors ($\overrightarrow{\mathbf{h}}_j$, $\overleftarrow{\mathbf{h}}_j$) based on the BLSTM algorithm provided in Algorithm 2. These column vectors are represented as $\overrightarrow{\mathbf{i}}_j$, $\overleftarrow{\mathbf{i}}_j$, $\overrightarrow{\mathbf{f}}_j$, $\overleftarrow{\mathbf{f}}_j$, $\overrightarrow{\mathbf{c}}_j$, $\overleftarrow{\mathbf{c}}_j$, $\overrightarrow{\mathbf{o}}_j$, $\overleftarrow{\mathbf{o}}_j$, $\overrightarrow{\mathbf{h}}_j$, $\overleftarrow{\mathbf{h}}_j$. Weight matrices ($\overrightarrow{\mathbf{W}}_{(\cdot)}$, $\overleftarrow{\mathbf{W}}_{(\cdot)}$) and bias vectors ($\overrightarrow{\mathbf{b}}_{(\cdot)}$, $\overleftarrow{\mathbf{b}}_{(\cdot)}$) were obtained by training the BLSTM using the BPTT algorithm. Recurrent activation function is a sigmoid function and it is represented by σ_r; hidden layer activation function is represented by σ_h; while output layer activation function is a softmax function and it is represented by σ_y. The parameters of the forward LSTM hidden layer are computed from the past to the present input data sequence while those of the backward LSTM hidden layer are calculated starting from the future to the present input data sequence. The LSTM hidden layers in the positive and negative time directions are jointly connected to the output layer. The difference between the predicted output of LAE-BLSTM, $\tilde{\mathbf{y}}$, and the target output, $\mathbf{y}$, is minimized in Equation 7.

$$\theta\left(\mathbf{y}, \tilde{\mathbf{y}}\right) = \left[\theta\left(\mathbf{y}_d, \tilde{\mathbf{y}}_d\right)\right]_{d=1}^{n}, \tag{7}$$

where θ is either a binary cross-entropy loss function for binary classification or a categorical cross-entropy loss function for multi-class classification scenarios.

IV. Feature Dimensionality Reduction and Network Traffic Classification

In this section, we implement the hybrid DL framework proposed in Section III (i.e. LAE-BLSTM) and perform extensive experiments with Bot-IoT [33] to validate its effectiveness for botnet attack detection in IoT networks. The overall architecture of LAE-BLSTM is shown in Fig. 1.

All the experiments performed in this paper leveraged Numpy, Pandas, Scikit-learn and Keras libraries in Python programming language. Python codes were written and implemented within Spyder Integrated Development Environment (IDE) running on Ubuntu 16.04 LTS workstation with the following specifications: Random Access Memory (32 GB), Processor (Intel Core i7-9700K CPU @ 3.60GHz × 8), Graphics (GeForce RTX 2080 Ti/PXCIe/SSE2) and 64-bit Operating System (OS).

A. Data pre-processing

Large network traffic data in Bot-IoT dataset was pre-processed to transform the features and the ground truth labels into appropriate formats for ease of computation. In this study, data pre-processing involves the following: (a) elimination of redundant network information; (a) random division of complete network traffic data into training, validation and testing sets; (b) selection of network traffic features and ground truth labels; (c) normalization or scaling of network traffic features; and (d) integer encoding of ground truth labels.

Redundant network traffic information (*pkSeqID*, *saddr*, *daddr*, *proto*, *state* and *flgs*) were removed from Bot-IoT. Therefore, only 37 out of the 43 network traffic features were selected to form high-dimensional network traffic feature set. The description of each of the 43 features are provided in [33]. The high-dimensional feature set was randomly divided into training set (70%), validation set (15%) and testing set (15%) with reference to most recent works covering diverse areas of application, e.g., [74]–[79]. Table III presents the distribution of data samples in training, validation and testing sets under binary and multi-class classification scenarios. Elements of the high-dimensional feature set were normalized to a range of $[0, 1]$ using min-max transformation method given by Eq. (8):

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}, \tag{8}$$

where $\mathbf{x}$ is a network traffic feature vector; while $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the minimum and maximum values of $\mathbf{x}$ respectively. This method retains the original distribution of network traffic features. The pre-processed high-dimensional network traffic feature set was named RAW-F.

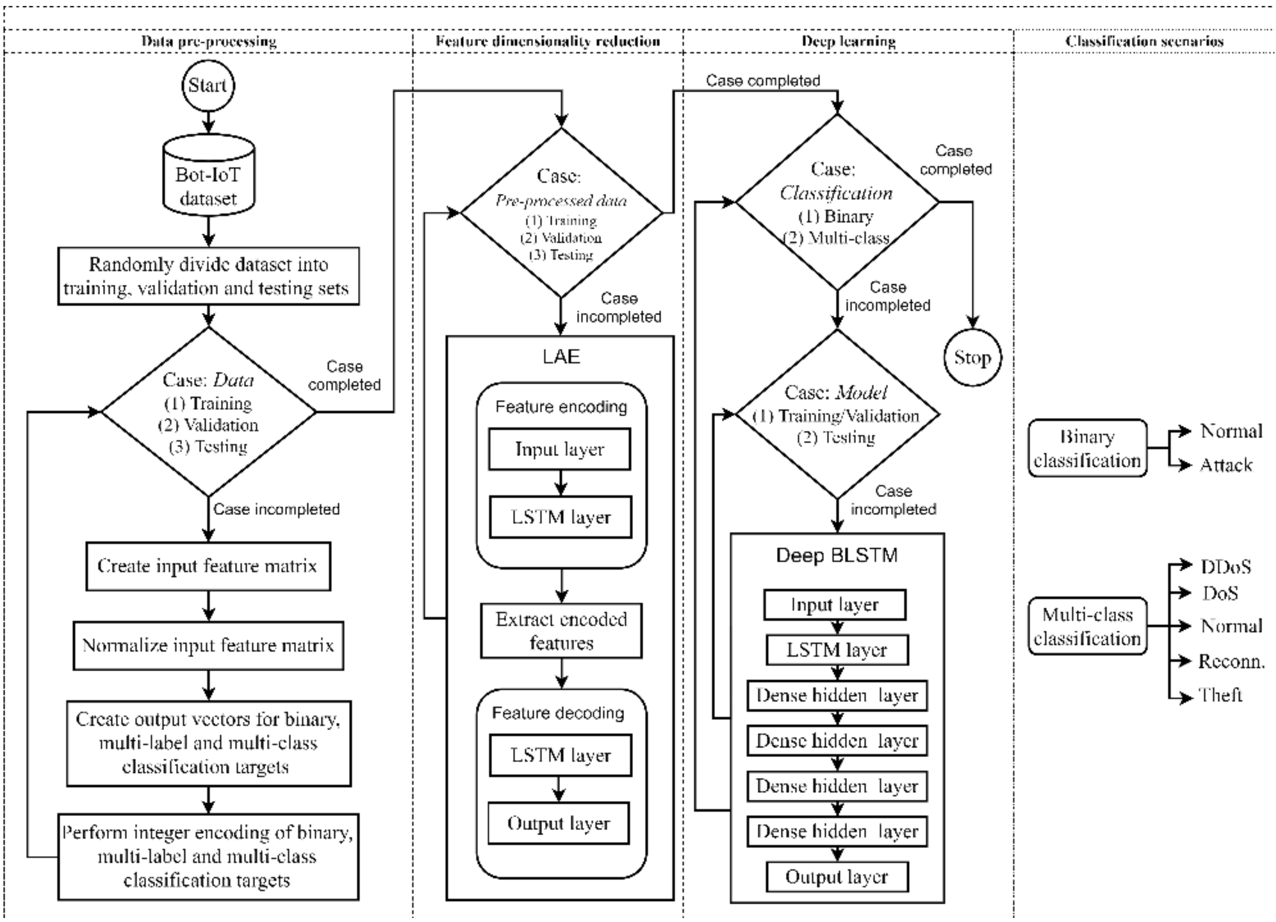

Fig. 1. LAE-BLSTM architecture for botnet attack detection in IoT networks

TABLE III
SAMPLE DISTRIBUTION OF IOT NETWORK TRAFFIC DATA

Classification scenario	Target	Training	Validation	Testing
Binary	Attack	2,567,548	550,303	550,194
	Normal	325	67	85
Multi-class	DDoS	1,348,654	288,809	289,161
	DoS	1,155,031	247,680	247,549
	Normal	325	67	85
	Reconn.	63,806	13,800	13,476
	Theft	57	14	8

On the other hand, binary and multi-class ground truth labels were converted into integer format for ease of computation. Specifically, binary ground truth labels i.e. *attack* and *normal* were represented by 0 and 1 respectively. Similarly, multi-class ground truth labels i.e. *DDoS*, *DoS*, *normal*, *reconnaissance* and *theft* were represented by 0, 1, 2, 3 and 4 respectively.

B. LAE for feature dimensionality reduction

LAE model was developed to reduce the feature dimensionality and the data size of RAW-F such that the limited available memory space in IoT devices will be sufficient for network traffic data storage. The internal architecture of LAE is shown in Fig. 1, and its working principles were discussed earlier in Section IIIA. This DL method employed the following hyperparameters: a single LSTM layer, 37 input neurons, six neurons at the LSTM layer, a Rectified Linear Unit (ReLU) activation function, Mean Square Error (MSE) as a loss function, a learning rate of 0.001, 10 epochs, and a batch size of 64. LAE models were trained with state-of-the-art optimisation algorithms that are available in Keras ML library[2]. These optimisation algorithms include Adaptive moment estimation (Adam) [80], Stochastic Gradient Descent with Nesterov momentum (SGD) [81], RMSprop [82], Adadelta [83], Adagrad [84], Adamax [80], Adam with Nesterov momentum (Nadam) [85], and Follow The Regularised Leader (FTRL) [86]. To ensure efficient feature dimensionality reduction, we investigated and compared the performance of LAE model when each of these optimisation algorithms was used. The performance evaluation was based on reconstruction loss, data size of low-dimensional features, and the rate of feature dimensionality reduction. Finally, the performance of the optimal LAE model was compared with that of three state-of-the-art feature dimensionality reduction methods i.e. PCE [33], XMP [34], and IFG [35]. The low-dimensional features produced by LAE, PCE [33], XMP [34], and IFG [35] methods are referred to as PCE-F, XMP-F, and IFG-F, respectively.

[2]https://keras.io/api/optimizers

C. Deep BLSTM for network traffic classification

Given the low-dimensional feature set, LAE-F, binary and multi-class deep BLSTM models were developed to correctly detect benign network traffic and botnet attack traffic in IoT networks. The network architecture of deep BLSTM model is shown in Fig. 1, and its principles of operation were previously explained in Section IIIB. In this paper, a deep BLSTM model comprised of a single LSTM layer with six input neurons, four dense hidden layers, and an output layer. The LSTM layer and the four dense hidden layers have 100 output neurons each. The output layer used a single neuron and five neurons for binary and multi-class classification, respectively. ReLU activation function was employed in all the layers of deep BLSTM model, except the output layer. A sigmoid activation function and a softmax activation function were employed at the output layer for binary and multi-class classification, respectively. The hyperparameters that were used to train deep BLSTM model include: a learning rate of 0.0001, 20 epochs, a batch size of 64, as well as a binary cross-entropy loss function and a categorical cross-entropy loss function for binary and multi-class classification, respectively. Deep BLSTM models were trained with Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimisation algorithms.

Table III shows that the distribution of network traffic samples in Bot-IoT dataset is highly imbalanced across the classes in both binary and multi-class classification scenarios. It has been experimentally proven that class imbalance affects the value and meaning of traditional classification performance metrics as they become bias in favour of the majority class [87]. Therefore, highly imbalanced data classification cannot be evaluated using traditional performance metrics. In a case where both classification successes and errors are to be considered, Matthews Correlation Coefficient (MCC) is the best null-biased performance metric for highly imbalanced data classification [87]. The value of MCC is calculated using Eq. (9):

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)\,(TP + FN)\,(TN + FP)\,(TN + FN)}}, \tag{9}$$

where TP is the number of IoT botnet attack samples that are correctly classified as IoT botnet attack traffic; FP is the number of normal samples that are misclassified as IoT botnet attack traffic; TN is the number of normal samples that are correctly classified as normal traffic; and FN is the number of IoT botnet attack samples that are misclassified as normal traffic.

For efficient network traffic classification, we investigated and compared the performance of deep BLSTM model when each of these optimisation algorithms was used. The performance of these models was evaluated with the training, validation and testing sets. We analysed the training loss, validation loss, and MCC values to determine the most efficient optimisation algorithm for deep BLSTM model. In binary classification scenario, the performance of the optimal deep BLSTM model was compared with that of four state-of-the-art classifiers i.e. SVM [33], RNN [33], LSTM [33], and VCDL [88]. In multi-class classification scenario, the performance of the optimal deep BLSTM model was compared with that of three state-of-the-art classifiers i.e. FNN [89], SNN [89], and C5-OCSVM [35].

V. Results and Discussion

In this section, we present and analyse the results of the experiments performed in Section IV to validate the effectiveness of LAE-BLSTM model for botnet attack detection in IoT networks. The performance of LAE-BLSTM model is compared with that of the state-of-the-art ML and DL models in binary and multi-class classification scenarios. In this context, LAE-BLSTM model is considered to be efficient if it meets all the following conditions:

1) *Relatively low memory space requirement for big network traffic data storage*. This is evaluated based on reconstruction loss, data size of low-dimensional features, and the rate of feature dimensionality reduction. Low reconstruction loss, low data size, and high reduction rate imply that relatively low memory space is required for big network traffic data storage.
2) *Robustness against model under-fitting and over-fitting*. Model under-fitting is evaluated based on the training loss and the MCC value obtained when deep BLSTM model was evaluated with the training set. Model over-fitting is evaluated based on the validation loss and MCC value obtained when deep BLSTM model was evaluated with the validation set. Low training loss and high MCC value on training set imply that deep BLSTM model is robust against model under-fitting. On the other hand, low validation loss and high MCC value on validation set imply that deep BLSTM model is robust against model over-fitting.
3) *Good generalization ability*. This is evaluated based on the MCC values obtained when deep BLSTM model was evaluated with the testing set. High MCC values on testing set implies that deep BLSTM model performs well on previously unseen network traffic data.

A. Results of feature dimensionality reduction

To determine the most suitable optimiser for efficient feature dimensionality reduction, we analyse the reconstruction losses generated when Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimisation algorithms were used to train LAE model. Fig. 2 shows that the use of Adam optimiser produced the lowest reconstruction loss (0.0041 – 0.0023) throughout the 10-epoch training period. Table IV shows that the use of Adam optimiser outperformed the use of the other seven optimisers by 39.55 – 89.33%.

Data sizes of the low-dimensional feature sets (PCE-F, XMP-F, IFG-F, and LAE-F) were compared with that of the original feature set (RAW-F) to evaluate the impact of feature dimensionality reduction methods. Table V shows that LAE and state-of-the-art methods significantly reduced the data size of RAW-F. However, LAE achieved the lowest data size (88.04 MB) with a reduction rate of 91.89%. This method outperformed XMP [34], IFG [35], and PCE [33] methods by 18.92%, 18.92%, and 27.03%, respectively. The

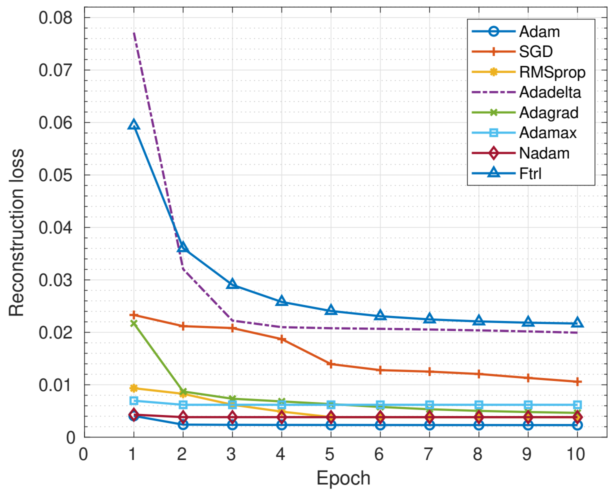

Fig. 2. Trends of reconstruction loss during the training of LAE

TABLE IV
RECONSTRUCTION LOSS AFTER THE TRAINING OF LAE MODEL

Optimiser	Reconstruction loss
Adam	**0.0023**
SGD	0.0106
RMSprop	0.0038
Adadelta	0.0199
Adagrad	0.0046
Adamax	0.0062
Nadam	0.0038
FTRL	0.0217

implementation of the DL method in memory-constraint IoT devices becomes more practicable for efficient botnet detection when the data size of the network traffic feature set is as small as possible.

B. Results of network traffic classification in binary scenario

To determine the most suitable optimiser for efficient network traffic classification in binary scenario, we analyse the training losses, validation losses, and MCC values obtained when deep BLSTM model was trained with Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimisation algorithms.

Fig. 3 shows the training losses of deep BLSTM model in binary classification scenario when each of the eight optimisation algorithms was used. In all cases, the training losses reduced as the number of epochs increased from 1 to 20. However, deep BLSTM model had the lowest training losses

TABLE V
NETWORK TRAFFIC FEATURE SETS

Feature set	Feature size	Data size (MB)	Reduction rate (%)
RAW-F	37	1085.88	-
PCE-F [33]	10	293.48	72.97
XMP-F [34]	10	293.48	72.97
IFG-F [35]	13	381.53	64.86
LAE-F	**6**	**88.04**	**91.89**

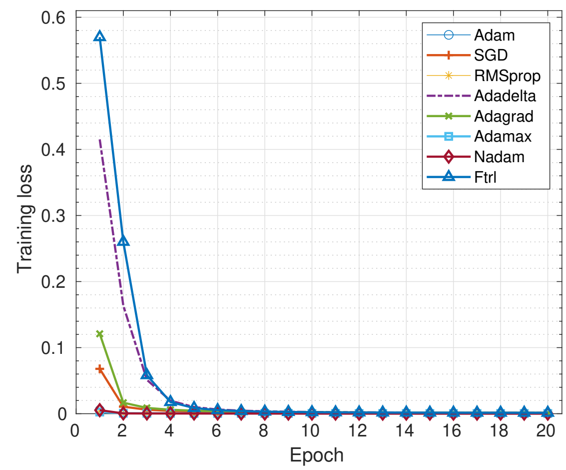

Fig. 3. Training losses of deep BLSTM model in binary classification scenario

when Nadam optimiser was employed. The values of training losses produced by Adam optimiser were very close to those of Nadam optimiser. At the end of the 20-epoch training, Nadam optimiser achieved a training loss of 8.19×10^{-5} while that of Adam optimiser was 8.39×10^{-5}. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax and FTRL optimisers produced relatively high training losses of 0.0011, 0.0006, 0.0013, 0.0013, 0.0003, and 0.0014, respectively.

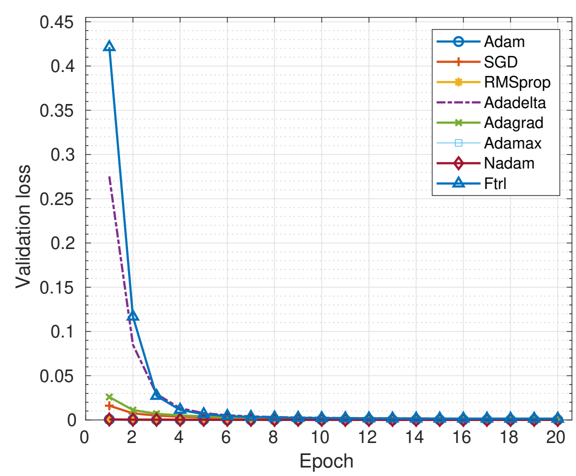

Fig. 4. Validation losses of deep BLSTM model in binary classification scenario

Fig. 4 shows the validation losses of deep BLSTM model in binary classification scenario when each of the eight optimisation algorithms was used. In all cases, the validation losses reduced as the number of epochs increased from 1 to 20. However, deep BLSTM model had the lowest validation losses when Nadam optimiser was employed. The values of validation losses produced by Adam optimiser were very close to those of Nadam optimiser. At the end of the 20-epoch validation, Nadam optimiser achieved a validation loss of 8.75×10^{-5} while that of Adam optimiser was 0.0001. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax and

TABLE VI
MCC VALUES OF DEEP BLSTM MODEL IN BINARY CLASSIFICATION SCENARIO

Model	Training (%)	Validation (%)	Testing (%)
Adam	91.67	89.55	93.03
SGD	null	null	null
RMSprop	63.79	62.29	52.36
Adadelta	null	null	null
Adagrad	null	null	null
Adamax	82.38	78.41	82.61
Nadam	**93.04**	**90.66**	**95.80**
FTRL	null	null	null

FTRL optimisers produced relatively high validation losses of 0.0011, 0.0008, 0.0013, 0.0013, 0.0003, and 0.0014, respectively.

Table VI presents the MCC values of deep BLSTM model in binary classification scenario when each of the eight optimisation algorithms was used. For SGD, Adadelta, Adagrad, and FTRL optimisers, deep BLSTM model correctly classified all the samples in the *attack* class as botnet attack traffic but it misclassified all the samples in the *normal* class as botnet attack traffic. In these cases, the values of TN and FN were zero. Based on Eq. (9), the MCC values were undefined when deep BLSTM model was evaluated with network traffic data in the training, validation, and testing sets. This shows that SGD, Adadelta, Adagrad, and FTRL optimisers could not handle the class imbalance problem in Bot-IoT dataset. The classifier was biased in favour of the *attack* (majority) class because the number of samples in this class is far greater than those in the *normal* class (see Table III). On the other hand, Nadam optimiser produced the highest MCC values when deep BLSTM model was evaluated with network traffic samples in the training, validation, and testing sets. Nadam optimiser achieved an overall MCC of 93.17%, and it outperformed Adam, RMSprop, and Adamax optimisers by 1.91%, 33.69% and 12.03%, respectively. Fig. 5 shows the confusion matrix of deep BLSTM model when Nadam optimiser was used for binary classification. The detection rate of normal traffic was 92.90% while that of botnet attack traffic was 100%.

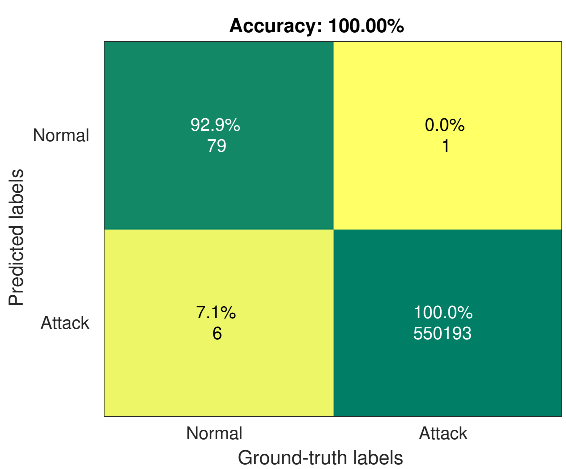

Fig. 5. Confusion matrix of deep BLSTM model in binary classification scenario

TABLE VII
PERFORMANCE IN ML/DL MODELS IN BINARY CLASSIFICATION SCENARIO

Model	MCC (%)
PCE-SVM [33]	3.14
PCE-RNN [33]	5.98
PCE-LSTM [33]	7.03
PCE-VCDL [88]	20.42
LAE-BLSTM	**93.17**

Finally, the performance of LAE-BLSTM model was compared with that of state-of-the-art models in binary classification scenario. Table VII shows that LAE-BLSTM model outperformed PCE-SVM [33], PCE-RNN [33], PCE-LSTM [33], and PCE-VCDL [88] models by 90.03%, 87.19%, 86.14%, and 72.75%, respectively.

C. Results of network traffic classification in multi-class scenario

To determine the most suitable optimiser for network traffic classification in multi-class scenario, we analyse the training losses, validation losses, and MCC values obtained when deep BLSTM model was trained with Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimisation algorithms.

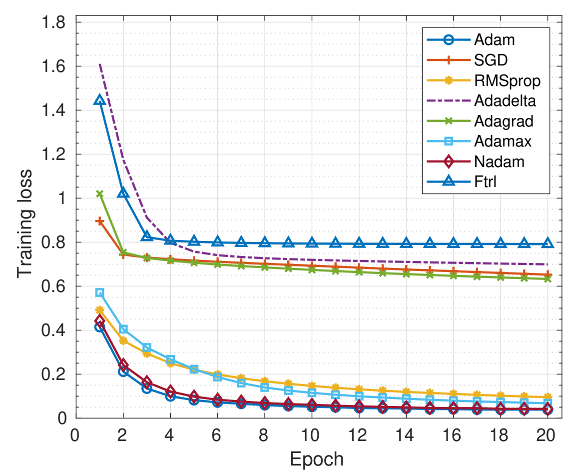

Fig. 6. Training losses of deep BLSTM model in multi-class classification scenario

Fig. 6 shows the training losses of deep BLSTM model in multi-class classification scenario when each of the eight optimisation algorithms was used. In all cases, the training losses reduced as the number of epochs increased from 1 to 20. However, deep BLSTM model had the lowest training losses when Adam optimiser was employed. The values of training losses produced by Nadam optimiser were very close to those of Adam optimiser. At the end of the 20-epoch training, Adam optimiser achieved a training loss of 0.0389 while that of Nadam optimiser was 0.0421. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax and FTRL optimisers produced relatively high training losses of 0.6524, 0.0953, 0.6992, 0.6328, 0.0680, and 0.7913, respectively.

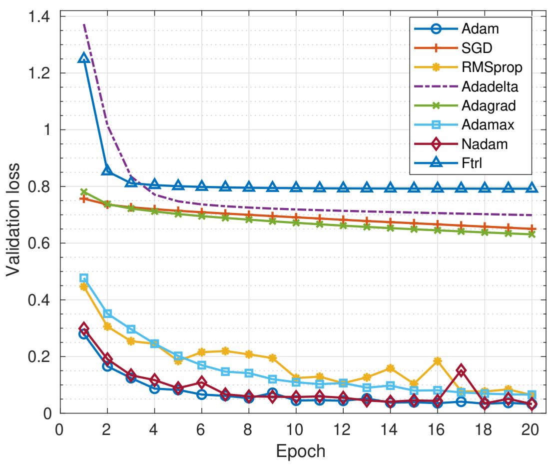

Fig. 7. Validation losses of deep BLSTM model in multi-class classification scenario

TABLE VIII
MCC VALUES OF DEEP BLSTM MODEL FOR TRAINING IN MULTI-CLASS CLASSIFICATION SCENARIO

Optimiser	DDoS	DoS	Norm	Recon	Theft	Mean
Adam	98.33	98.31	**93.34**	**99.96**	93.66	96.72
SGD	57.83	57.18	null	99.62	null	42.93
RMSprop	92.78	92.71	null	99.47	null	56.99
Adadelta	27.86	24.79	null	71.05	null	24.74
Adagrad	36.07	35.71	null	98.73	null	34.10
Adamax	94.98	94.94	73.05	99.82	null	72.56
Nadam	**98.97**	**98.96**	92.88	**99.96**	**93.68**	**96.89**
FTRL	19.24	15.00	null	98.10	null	26.47

Fig. 7 shows the validation losses of deep BLSTM model in multi-class classification scenario when each of the eight optimisation algorithms was used. In all cases, the validation losses reduced as the number of epochs increased from 1 to 20. However, deep BLSTM model had the lowest validation losses when Adam optimiser was employed. The values of validation losses produced by Nadam optimiser were very close to those of Adam optimiser. At the end of the 20-epoch validation, Adam optimiser achieved a validation loss of 0.0325 while that of Adam optimiser was 0.0326. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax and FTRL optimisers produced relatively high validation losses of 0.6503, 0.0640, 0.6988, 0.6310, 0.0656, and 0.7920, respectively.

TABLE IX
MCC VALUES OF DEEP BLSTM MODEL FOR VALIDATION IN MULTI-CLASS CLASSIFICATION SCENARIO

Optimiser	DDoS	DoS	Norm	Recon	Theft	Mean
Adam	98.28	98.26	**90.10**	**99.94**	**96.36**	96.59
SGD	57.96	57.31	null	99.63	null	42.98
RMSprop	92.76	92.70	null	99.49	null	56.99
Adadelta	28.11	25.02	null	71.00	null	24.83
Adagrad	36.18	35.82	null	98.73	null	73.21
Adamax	94.97	94.93	76.29	99.83	null	73.21
Nadam	**98.94**	**98.93**	89.83	99.93	**96.36**	**96.80**
FTRL	19.24	15.00	null	98.10	null	26.47

TABLE X
MCC VALUES OF DEEP BLSTM MODEL FOR TESTING IN MULTI-CLASS CLASSIFICATION SCENARIO

Optimiser	DDoS	DoS	Norm	Recon	Theft	Mean
Adam	98.33	98.31	91.34	99.94	100.00	97.58
SGD	57.78	57.14	null	99.55	null	42.90
RMSprop	92.74	92.68	null	99.44	null	56.97
Adadelta	27.76	24.72	null	70.53	null	24.60
Adagrad	35.96	35.61	null	98.54	null	34.02
Adamax	95.01	94.97	71.38	99.76	null	72.23
Nadam	**98.98**	**98.97**	**92.94**	**99.95**	**100.00**	**98.17**
FTRL	18.79	14.56	null	97.94	null	26.26

Furthermore, we evaluate the performance of deep BLSTM model when each of the eight optimisation algorithms was used in multi-class classification scenario. Table VIII presents the MCC values of deep BLSTM model when evaluated with network traffic samples in the training set. Adam optimiser achieved the highest MCC value in detecting *normal* traffic. Nadam optimiser achieved the highest MCC values in detecting *DDoS*, *DoS*, *reconnaissance*, and *theft* attacks. This optimiser outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimisers by 0.08%, 53.87%, 39.81%, 72.06%, 62.70%, 24.24%, and 62.70%, respectively. Table IX presents the MCC values of deep BLSTM model when evaluated with network traffic samples in the validation set. Nadam optimiser achieved the highest MCC values in all classes. This optimiser outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimisers by 0.21%%, 53.82%, 39.81%, 71.97%, 23.59%, 23.59%, and 70.33%, respectively. Table X presents the MCC values of deep BLSTM model when evaluated with network traffic samples in the testing set. Nadam optimiser achieved the highest MCC values in all classes. This optimiser outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimisers by 0.59%, 55.27%, 41.20%, 73.57%, 64.15%, 25.94%, and 71.91%, respectively.

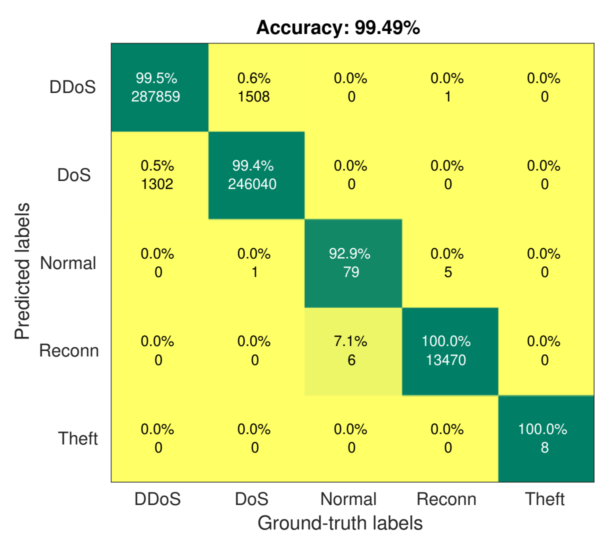

Fig. 8. Confusion matrix of deep BLSTM model in multi-class classification scenario

TABLE XI
PERFORMANCE IN ML/DL MODELS IN MULTI-CLASS CLASSIFICATION SCENARIO

Model	DDoS	DoS	Norm	Recon	Theft	Mean
PCE-FNN [89]	-	-	-	-	-	89.00
PCE-SNN [89]	-	-	-	-	-	85.00
IFG-CSVM [35]	99.38	99.53	88.91	44.76	99.32	86.38
LAE-BLSTM	98.96	98.95	91.89	99.95	96.68	**97.29**

Fig. 8 shows the confusion matrix of deep BLSTM model when Nadam optimiser was used for multi-class classification. The detection rates of *DDoS*, *DoS*, *normal*, *reconnaissance*, and *theft* were 99.5%, 99.4%, 92.90%, 100%, and 100%, respectively. Finally, the performance of LAE-BLSTM model was compared with that of state-of-the-art models in multi-class classification scenario. Table XI shows that LAE-BLSTM model outperformed PCE-FNN [89], PCE-SNN [89], and IFG-CSVM [35] models by 8.29%, 12.29%, and 10.91%, respectively.

VI. CONCLUSION

LAE-BLSTM, an hybrid DL method, was proposed for efficient botnet detection in IoT networks using LAE and deep BLSTM algorithms. The effectiveness of this method was validated by performing extensive experiments with the most relevant publicly available dataset (Bot-IoT) in binary and multi-class classification scenarios. Simulation results showed that LAE model achieved the highest data size reduction rate of 91.89%, outperforming XMP [34], IFG [35], and PCE [33] methods by 18.92%, 18.92%, and 27.03%, respectively. As the data size of big network traffic features becomes smaller, the implementation of DL method in memory-constraint IoT devices seems to be more practicable for efficient botnet detection. In addition, deep BLSTM model, which were trained to analyse the long-term inter-related changes in low-dimensional feature set produced by LAE, demonstrated robustness against model under-fitting and over-fitting as well as good generalization ability. Therefore, LAE-BLSTM has proven to be efficient for botnet attack detection in IoT networks.

REFERENCES

[1] K. Anoh, A. Ikpehai, D. Bajovic, O. Jogunola, B. Adebisi, D. Vukobratovic, and M. Hammoudeh, "Virtual microgrids: a management concept for peer-to-peer energy trading," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, 2018, pp. 1–5.

[2] A. O. Akmandor, Y. Hongxu, and N. K. Jha, "Smart, secure, yet energy-efficient, internet-of-things sensors," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 914–930, 2018.

[3] Z. Wang, Y. Liu, Z. Ma, X. Liu, and J. Ma, "Lipsg: Lightweight privacy-preserving q-learning based energy management for the iot-enable smart grid," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3935–3947, 2020.

[4] R. J. Tom, S. Sankaranarayanan, and J. J. Rodrigues, "Smart energy management and demand reduction by consumers and utilities in an iot-fog-based power distribution system," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7386–7394, 2019.

[5] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) towards 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[6] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5g era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.

[7] W. Feng, J. Wang, Y. Chen, X. Wang, N. Ge, and J. Lu, "Uav-aided mimo communications for 5g internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1731–1740, 2018.

[8] P. Verma and S. K. Sood, "Fog assisted-iot enabled patient health monitoring in smart homes," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789–1796, 2018.

[9] L. Liu, J. Xu, Y. Huan, Z. Zou, S.-C. Yeh, and L. Zheng, "A smart dental health-iot platform based on intelligent hardware, deep learning and mobile terminal," *IEEE journal of biomedical and health informatics*, vol. 24, no. 3, pp. 898–906, 2020.

[10] R. K. Pathinarupothi, P. Durga, and E. S. Rangan, "Iot-based smart edge for global health: Remote monitoring with severity detection and alerts transmission," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2449–2462, 2018.

[11] F. Tao, J. Cheng, and Q. Qi, "Iihub: An industrial internet-of-things hub toward smart manufacturing based on cyber-physical system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2271–2280, 2017.

[12] Y.-C. Lin, M.-H. Hung, H.-C. Huang, C.-C. Chen, H.-C. Yang, Y.-S. Hsieh, and F.-T. Cheng, "Development of advanced manufacturing cloud of things (amcot)- a smart manufacturing platform," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1809–1816, 2017.

[13] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined industrial internet of things in the context of industry 4.0," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7373–7380, 2016.

[14] X.-G. Luo, H.-B. Zhang, Z.-L. Zhang, Y. Yu, and K. Li, "A new framework of intelligent public transportation system based on the internet of things," *IEEE Access*, vol. 7, pp. 55 290–55 304, 2019.

[15] S. Chavhan, D. Gupta, B. Chandana, A. Khanna, and J. J. Rodrigues, "Iot-based context-aware intelligent public transport system in a metropolitan area," *IEEE Internet of Things Journal*, 2020.

[16] H. Serra, I. Bastos, J. L. de Melo, J. P. Oliveira, N. Paulino, E. Nefzaoui, and T. Bourouina, "A 0.9-v analog-to-digital acquisition channel for an iot water management sensor node," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 10, pp. 1678–1682, 2019.

[17] N. Ahmed, D. De, and I. Hussain, "Internet of things (iot) for smart precision agriculture and farming in rural areas," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018.

[18] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758–3773, 2018.

[19] M. Hammoudeh, G. Epiphaniou, S. Belguith, D. Unal, B. Adebisi, T. Baker, A. Kayes, and P. Watters, "A service-oriented approach for sensing in the internet of things: intelligent transportation systems and privacy use cases," *IEEE Sensors Journal*, 2020.

[20] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, "Internet of things: Evolution and technologies from a security perspective," *Sustainable Cities and Society*, vol. 54, p. 101728, 2020.

[21] A. Ikpehai, B. Adebisi, K. M. Rabie, K. Anoh, R. E. Ande, M. Hammoudeh, H. Gacanin, and U. M. Mbanaso, "Low-power wide area network technologies for internet-of-things: A comparative review," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2225–2240, 2018.

[22] L. Yin, X. Luo, C. Zhu, L. Wang, Z. Xu, and H. Lu, "Connspoiler: Disrupting c&c communication of iot-based botnet through fast detection of anomalous domain queries," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373–1384, 2020.

[23] S. Walker-Roberts, M. Hammoudeh, O. Aldabbas, M. Aydin, and A. Dehghantanha, "Threats on the horizon: understanding security threats in the era of cyber-physical systems," *The Journal of Supercomputing*, pp. 1–22, 2019.

[24] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[25] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, vol. 89, pp. 349 – 359, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18307532

[26] N. Koroniotis, N. Moustafa, and E. Sitnikova, "Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions," *IEEE Access*, vol. 7, pp. 61 764–61 785, 2019.

[27] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[28] Y. Wang, L. Guo, Y. Zhao, J. Yang, B. Adebisi, H. Gacanin, and G. Gui, "Distributed learning for automatic modulation classification in edge devices," *IEEE Wireless Communications Letters*, 2020.

[29] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[30] Z. Wang, B. Du, L. Zhang, L. Zhang, and X. Jia, "A novel semisupervised active-learning algorithm for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3071–3083, 2017.

[31] F. Luo, B. Du, L. Zhang, L. Zhang, and D. Tao, "Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image," *IEEE transactions on cybernetics*, vol. 49, no. 7, pp. 2406–2419, 2018.

[32] J. Peng, W. Sun, and Q. Du, "Self-paced joint sparse representation for the classification of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 1183–1194, 2018.

[33] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[34] M. Asadi, M. A. J. Jamali, S. Parsa, and V. Majidnezhad, "Detecting botnet by using particle swarm optimization algorithm based on voting system," *Future Generation Computer Systems*, vol. 107, pp. 95–111, 2020.

[35] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, p. 1210, 2019.

[36] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[38] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection." *IJ Network Security*, vol. 17, no. 6, pp. 683–701, 2015.

[39] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[40] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *International Conference on Contemporary Computing*. Springer, 2012, pp. 322–334.

[41] E. Alomari, S. Manickam, B. Gupta, P. Singh, and M. Anbar, "Design, deployment and use of http-based botnet (hbb) testbed," in *16th International Conference on Advanced Communication Technology*. IEEE, 2014, pp. 1265–1269.

[42] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE, 2006, pp. 967–974.

[43] S. Bhatia, D. Schmidt, G. Mohay, and A. Tickle, "A framework for generating realistic traffic for distributed denial-of-service attacks and flash events," *computers & security*, vol. 40, pp. 95–107, 2014.

[44] S. Behal and K. Kumar, "Detection of ddos attacks and flash events using information theory metrics–an empirical investigation," *Computer Communications*, vol. 103, pp. 18–28, 2017.

[45] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[46] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, pp. 1171–1220, 2008.

[47] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[48] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[49] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[50] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30 373–30 385, 2019.

[51] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[52] A. Dawoud, S. Shahristani, and C. Raun, "Dimensionality reduction for network anomalies detection: A deep learning approach," in *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer, 2019, pp. 957–965.

[53] F. C. Schuartz, M. Fonseca, and A. Munaretto, "Improving threat detection in networks using deep learning," *Annals of Telecommunications*, pp. 1–10, 2020.

[54] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33 353–33 361, 2018.

[55] X. Wang and L. Wang, "Research on intrusion detection based on feature extraction of autoencoder and the improved k-means algorithm," in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2017, pp. 352–356.

[56] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE, 2017, pp. 63–69.

[57] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[58] S. Gurung, M. K. Ghose, and A. Subedi, "Deep learning approach on network intrusion detection system using nsl-kdd dataset," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 11, no. 3, pp. 8–14, 2019.

[59] A.-H. Muna, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information security and applications*, vol. 41, pp. 1–11, 2018.

[60] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *2015 7th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2015, pp. 1–5.

[61] D. C. Ferreira, F. I. Vázquez, and T. Zseby, "Extreme dimensionality reduction for network attack visualization with autoencoders," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–10.

[62] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019.

[63] J. Yang, T. Li, G. Liang, W. He, and Y. Zhao, "A simple recurrent unit model based intrusion detection system with dcgan," *IEEE Access*, vol. 7, pp. 83 286–83 296, 2019.

[64] A. Liu and B. Sun, "An intrusion detection system based on a quantitative model of interaction mode between ports," *IEEE Access*, vol. 7, pp. 161 725–161 740, 2019.

[65] R.-H. Dong, X.-Y. Li, Q.-Y. Zhang, and H. Yuan, "Network intrusion detection model based on multivariate correlation analysis–long short-time memory network," *IET Information Security*, vol. 14, no. 2, pp. 166–174, 2020.

[66] Y. N. Soe, Y. Feng, P. I. Santosa, and R. Hartanto, "Towards a lightweight detection system for cyber attacks in the iot environment using corresponding features," *Electronics*, vol. 9, no. 1, p. 144, 2020.

[67] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Rule generation for signature based detection systems of cyber attacks in iot environments," *Bulletin of Networking, Computing, Systems, and Software*, vol. 8, no. 2, pp. 93–97, 2019.

[68] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "Hybrid feature selection models for machine learning based botnet detection in iot networks," in *2019 International Conference on Cyberworlds (CW)*. IEEE, 2019, pp. 324–327.

[69] F. Zhao, J. Feng, J. Zhao, W. Yang, and S. Yan, "Robust lstm-autoencoders for face de-occlusion in the wild," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 778–790, 2017.

[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[71] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[72] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[73] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[74] M. D. de Lima, J. d. O. R. e Lima, and R. M. Barbosa, "Medical data set classification using a new feature selection algorithm combined

with twin-bounded support vector machine," *Medical & Biological Engineering & Computing*, pp. 1–10, 2020.

[75] Y. Liu, J. He, H. Ma, and Y. Zhao, "Golden chip free trojan detection leveraging probabilistic neural network with genetic algorithm applied in the training phase," *Science China Information Sciences*, vol. 63, no. 2, p. 129401, 2020.

[76] A. F. Osman, N. M. Maalej, and K. Jayesh, "Prediction of the individual multi-leaf collimator positional deviations during dynamic imrt delivery priori with artificial neural network," *Medical Physics*, 2020.

[77] M. Zounemat-Kermani, D. Stephan, M. Barjenbruch, and R. Hinkelmann, "Ensemble data mining modeling in corrosion of concrete sewer: A comparative study of network-based (mlpnn & rbfnn) and tree-based (rf, chaid, & cart) models," *Advanced Engineering Informatics*, vol. 43, p. 101030, 2020.

[78] G. H. F. de Oliveira, S. C. Murray, L. C. C. Júnior, K. M. G. de Lima, C. d. L. M. de Morais, G. H. de Almeida Teixeira, and G. V. Môro, "Estimation and classification of popping expansion capacity in popcorn breeding programs using nir spectroscopy," *Journal of Cereal Science*, vol. 91, p. 102861, 2020.

[79] D. Tanikić, "Computationally intelligent optimization of metal cutting regimes," *Measurement*, vol. 152, p. 107358, 2020.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[81] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[82] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, 2012.

[83] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[84] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[85] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[86] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin *et al.*, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1222–1230.

[87] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognition*, vol. 91, pp. 216–231, 2019.

[88] B. A. NG and S. Selvakumar, "Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment," *Future Generation Computer Systems*, 2020.

[89] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in iot networks," *arXiv preprint arXiv:1905.05137*, 2019.

Segun I. Popoola received the B.Tech. degree (Hons.) in electronic and electrical engineering from the Ladoke Akintola University of Technology, Ogbomoso, Nigeria, and the M.Eng. degree in information and communication engineering from the Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria. He is currently pursuing the Ph.D. degree with the School of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, U.K. He was a Lecturer with the Department of Electrical and Information Engineering, Covenant University. His research interests include wireless communications, machine/deep learning, cybersecurity, and IoT. He is a Registered Engineer with the Council for the Regulation of Engineering in Nigeria (COREN).

Bamidele Adebisi received the B.S. degree in electrical engineering from Ahmadu Bello University, Zaria, Nigeria, in 1999, the M.S. degree in advanced mobile communication engineering, and the Ph.D. degree in communication systems from Lancaster University, Lancaster, U.K., in 2003 and 2009, respectively. He was a Senior Research Associate with the School of Computing and Communication, Lancaster University, from 2005 to 2012. He joined Manchester Metropolitan University, Manchester, U.K., in 2012, where he is currently a Professor in electrical and electronic engineering. He has been involving in several commercial and government projects focusing on various aspects of wireline and wireless communications. He is particularly interested in the research and development of communication technologies for electrical energy monitoring/management, transport, water, critical infrastructures protection, home automation, the IoTs, and cyber physical systems. He has several publications and a patent in the research area of data communications over power line networks and smart grid. He is a member of the IET.

Mohammad Hammoudeh is currently the Head of the MMU IoT Laboratory and a Senior Lecturer in computer networks and security with the School of Computing, Math and Digital Technology, Manchester Metropolitan University. He has been a researcher and publisher in the field of big sensory data mining and visualization. He is a highly proficient, experienced, and professionally certified cybersecurity professional, specializing in threat analysis, and information and network security management. His research interests include highly decentralized algorithms, communication, and cross-layered solutions to Internet of Things, and wireless sensor networks.

Guan Gui received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2012. From 2009 to 2014, he joined the Tohoku University as a Research Assistant as well as a Postdoctoral Research Fellow, respectively. From 2014 to 2015, he was an Assistant Professor with the Akita Prefectural University, Akita, Japan. Since 2015, he has been a Professor with Nanjing University of Posts and Telecommunications, Nanjing, China. His recent research interests include artificial intelligence, deep learning, non-orthogonal multiple access, wireless power transfer, and physical layer security. Dr. Gui has authored more than 200 IEEE Journal/Conference papers and won several best paper awards, and he is serving or served on the editorial boards of several journals. He was the recipient of the Top Editor Award of IEEE Transactions on Vehicular Technology in 2019.

Haris Gacanin received his Dipl.-Ing. degree in electrical engineering from the University of Sarajevo in 2000. In 2005 and 2008, respectively, he received M.Sc. and Ph.D. degrees from Tohoku University, Japan. He was with Tohoku University from 2008 until 2010, first as a Japan Society for Promotion of Science postdoctoral fellow and later as an assistant professor. Since 2010, he has been with Alcatel-Lucent (now Nokia), where he leads a research department at Nokia Bell Labs. His professional interest is related to the application of artificial intelligence to enable autonomous networking and design of mobile and wireless systems. He has 200+ scientific publications (journals, conferences, and patent applications) and invited/tutorial talks. He is a Senior Member of the Institute of Electronics, Information, and Communication Engineering (IEICE).