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

Segun I. Popoola, Bamidele Adebisi, *Senior Member, IEEE*, Mohammad Hammoudeh, *Senior Member, IEEE*, Guan Gui, *Senior Member, IEEE*, and Haris Gacanin, *Senior Member, IEEE*

***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 Internet-of-Things (IoT) devices. In this article, 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 analyze 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 data set 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 underfitting and overfitting. It also achieves good generalisation ability in binary and multiclass classification scenarios.**

***Index Terms*—Autoencoder, botnet detection, dimensionality reduction, Internet of Things (IoT), long short-term memory (LSTM).**

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 interconnectivity and the use of industrial control systems (ICSs) render 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 (NIDSs), 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 data set [33] is the most relevant publicly available data set for botnet attack detection in IoT networks because it: 1) has IoT network traffic samples; 2) captured complete network information; 3) has a diversity of complex IoT botnet attack scenarios; 4) contains accurate ground

Manuscript received April 16, 2020; revised September 7, 2020; accepted October 21, 2020. Date of publication October 27, 2020; date of current version March 5, 2021. This work was supported in part by the Cyraatek Ltd., U.K., the Faculty of Science & Engineering, Manchester Metropolitan University; in part by the ENERGY-IQ Project, a U.K.–Canada Power Forward Smart Grid Demonstrator Project funded by the Department for Business, Energy and Industrial Strategy under Grant 7454460; and in part by the Nigerian Intelligent Clean Energy Marketplace Project funded by the Department for International Development. *(Corresponding author: Bamidele Adebisi.)*

Segun I. Popoola and Bamidele Adebisi are with the Department of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester M1 5GD, U.K. (e-mail: s.popoola@mmu.ac.uk; b.adebisi@mmu.ac.uk).

Mohammad Hammoudeh is with the Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, U.K. (e-mail: m.hammoudeh@mmu.ac.uk).

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

Haris Gacanin is with the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, 52062 Aachen, Germany (e-mail: harisg@ice.rwth-aachen.de).

Digital Object Identifier 10.1109/JIOT.2020.3034156

truth labels; and 5) provides massive volume of labeled data required for effective supervised DL. The original feature dimensionality[1] of the Bot-IoT data set 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 data set were all based on feature selection techniques. These techniques include the filter method with Pearson correlation and entropy (PCE) [33], X-Mean clustering with particle swarm optimization (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 data set. 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 article, 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 article 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 analyzes the long-term interrelated 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 data set to validate the effectiveness of LAE-BLSTM in binary and multiclass 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 remainder of this article 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 this article in Section VI.

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

II. Related Work

Although several data sets 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 data sets 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 data set is limited because of the low number of benign traffic samples [36]. The attack scenarios in the UNIBS data set are limited to DoS; the network traffic data are presented in packets with no extracted features, and the labels were not provided [38]. CAIDA data sets have no ground-truth information about the attack samples [38]. Network traffic samples in the LBNL data set were not labeled, and the features were not extracted from the packet files [38]. Attack samples in the UNSW-NB15 data set were generated in a synthetic environment [39]. ISCX and CICIDS2017 data sets were generated based on the concept of profiling, and this can be due to their innate complexity. Also, the ground truth of these data sets is not available to enhance the labeling process. Not much information is given about the botnet scenarios that were used in most data sets [36], [38]–[40]. Also, IoT network traffic data were not included in related data sets [36], [39], [41]–[44].

Bot-IoT data set [33] is the most relevant data set that is publicly available for network-based botnet attack detection in IoT networks. To realize this data set, 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 ARP (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 nonlinear 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 nonlinear 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 data sets. These methods were implemented and evaluated with the network traffic data in publicly available data sets that 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,

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**

new feature dimensionality, and classifier and classification scenarios were provided. Long short-term memory (LSTM) is a variant of the 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 data set.

Different feature selection methods have been proposed to reduce the dimensionality of network traffic features in Bot-IoT data set. Table II presents an overview of state-of-the-art feature dimensionality reduction methods that are related to this article. Koroniotis *et al.* [33] employed PCE method for feature selection. The authors reported that ten 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), RNN, and 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 ten 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 IFG (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 multilabel classification.

To ensure a fair comparison, feature dimensionality reduction methods that do not include benign network traffic

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

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**

traces and all the four botnet attack scenarios in the Bot-IoT data set were not included in this article. 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 data set.

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 data set. 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.

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: $\tilde{\mathbf{X}}$
1  for $d = 1$ to $n$ do
2      for $j = 1$ to $k$ do
3          $\mathbf{i}_j = \sigma_r(\mathbf{W}_{ix}\mathbf{x}_j + \mathbf{W}_{ih}\mathbf{h}_{j-1} + \mathbf{b}_i)$
4          $\mathbf{f}_j = \sigma_r(\mathbf{W}_{fx}\mathbf{x}_j + \mathbf{W}_{fh}\mathbf{h}_{j-1} + \mathbf{b}_f)$
5          $\tilde{\mathbf{c}}_j = \sigma_h(\mathbf{W}_{\tilde{c}x}\mathbf{x}_j + \mathbf{W}_{\tilde{c}h}\mathbf{h}_{j-1} + \mathbf{b}_{\tilde{c}})$
6          $\mathbf{c}_j = \mathbf{D}_i \odot \tilde{\mathbf{c}}_j + \mathbf{D}_f \odot \mathbf{c}_{j-1}$
7          $\tilde{\mathbf{x}}_j = k_\phi(\mathbf{x}_j, \mathbf{c}_{j-1})$
8      end
9  end
10 $\tilde{\mathbf{X}} = \left\{\{\tilde{\mathbf{x}}_{d,j}\}_{j=1}^{k}\right\}_{d=1}^{n}$
11 $L = \theta\left(\mathbf{X}, \tilde{\mathbf{X}}\right) = \left[\theta\left(\mathbf{X}_d, \tilde{\mathbf{X}}_d\right)\right]_{d=1}^{n}$
```

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 article, boldface uppercase alphabets and boldface lower case alphabets represent matrices and column vectors, respectively.

A. LSTM Autoencoder

Autoencoder is an unsupervised DL method that analyzes 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 interrelated changes in network traffic features. In this section, 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 3-D matrix, $\mathbf{X} \in \mathbb{R}^{n\times 1\times k}$, in

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

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 data set. An instance of network traffic is represented by

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

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 (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

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: $\tilde{\mathbf{y}}$
1  for $d = 1$ to $n$ do
2      for $j = 1$ to $u$ do
3          %% Forward direction
4          $\overrightarrow{\mathbf{i}}_j = \sigma_r\left(\overrightarrow{\mathbf{W}}_{ix}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{ih}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_i\right)$
5          $\overrightarrow{\mathbf{f}}_j = \sigma_r\left(\overrightarrow{\mathbf{W}}_{fx}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{fh}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_f\right)$
6          $\overrightarrow{\tilde{\mathbf{c}}}_j = \sigma_h\left(\overrightarrow{\mathbf{W}}_{\tilde{c}x}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{\tilde{c}h}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_{\tilde{c}}\right)$
7          $\overrightarrow{\mathbf{c}}_j = \overrightarrow{\mathbf{i}}_j \odot \overrightarrow{\tilde{\mathbf{c}}}_j + \overrightarrow{\mathbf{f}}_j \odot \overrightarrow{\mathbf{c}}_{j-1}$
8          $\overrightarrow{\mathbf{o}}_j = \sigma_r\left(\overrightarrow{\mathbf{W}}_{ox}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{oh}\overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_o\right)$
9          $\overrightarrow{\mathbf{h}}_{1,j} = \overrightarrow{\mathbf{o}}_j \odot \sigma_h(\overrightarrow{\mathbf{c}}_j)$
10         $\overrightarrow{\mathbf{h}}_{2,j} = \sigma_h\left(\overrightarrow{\mathbf{W}}_{hx1}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh1}\overrightarrow{\mathbf{h}}_{1,j} + \overrightarrow{\mathbf{b}}_{h1}\right)$
11         $\overrightarrow{\mathbf{h}}_{3,j} = \sigma_h\left(\overrightarrow{\mathbf{W}}_{hx2}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh2}\overrightarrow{\mathbf{h}}_{2,j} + \overrightarrow{\mathbf{b}}_{h2}\right)$
12         $\overrightarrow{\mathbf{h}}_{4,j} = \sigma_h\left(\overrightarrow{\mathbf{W}}_{hx3}\tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh3}\overrightarrow{\mathbf{h}}_{3,j} + \overrightarrow{\mathbf{b}}_{h3}\right)$
13     end
14     %% Backward direction
15     $\overleftarrow{\mathbf{i}}_j = \sigma_r\left(\overleftarrow{\mathbf{W}}_{ix}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{ih}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_i\right)$
16     $\overleftarrow{\mathbf{f}}_j = \sigma_r\left(\overleftarrow{\mathbf{W}}_{fx}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{fh}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_f\right)$
17     $\overleftarrow{\tilde{\mathbf{c}}}_j = \sigma_h\left(\overleftarrow{\mathbf{W}}_{\tilde{c}x}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{\tilde{c}h}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_{\tilde{c}}\right)$
18     $\overleftarrow{\mathbf{c}}_j = \overleftarrow{\mathbf{i}}_j \odot \overleftarrow{\tilde{\mathbf{c}}}_j + \overleftarrow{\mathbf{f}}_j \odot \overleftarrow{\mathbf{c}}_{j-1}$
19     $\overleftarrow{\mathbf{o}}_j = \sigma_r\left(\overleftarrow{\mathbf{W}}_{ox}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{oh}\overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_o\right)$
20     $\overleftarrow{\mathbf{h}}_{1,j} = \overleftarrow{\mathbf{o}}_j \odot \sigma_h(\overleftarrow{\mathbf{c}}_j)$
21     $\overleftarrow{\mathbf{h}}_{2,j} = \sigma_h\left(\overleftarrow{\mathbf{W}}_{hx1}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh1}\overleftarrow{\mathbf{h}}_{1,j} + \overleftarrow{\mathbf{b}}_{h1}\right)$
22     $\overleftarrow{\mathbf{h}}_{3,j} = \sigma_h\left(\overleftarrow{\mathbf{W}}_{hx2}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh2}\overleftarrow{\mathbf{h}}_{2,j} + \overleftarrow{\mathbf{b}}_{h2}\right)$
23     $\overleftarrow{\mathbf{h}}_{4,j} = \sigma_h\left(\overleftarrow{\mathbf{W}}_{hx3}\tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh3}\overleftarrow{\mathbf{h}}_{3,j} + \overleftarrow{\mathbf{b}}_{h3}\right)$
24 end
25 $\tilde{\mathbf{y}} = \sigma_y\left(\overrightarrow{\mathbf{W}}_y\overrightarrow{\mathbf{h}_4} + \overleftarrow{\mathbf{W}}_y\overleftarrow{\mathbf{h}_4} + \mathbf{b}_y\right)$ $\tilde{\mathbf{y}} = \{\tilde{\mathbf{y}}_d\}_{d=1}^{n}$
26 $\theta(\mathbf{y}, \tilde{\mathbf{y}}) = \left[\theta(\mathbf{y}_d, \tilde{\mathbf{y}}_d)\right]_{d=1}^{n}$
```

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 backpropagation 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, respectively, by

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

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

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

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

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

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

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 analyzing the long-term interrelated 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

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

where θ is either a binary cross-entropy loss function for binary classification or a categorical cross-entropy loss function for multiclass 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 article 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 @ 3.60 GHz $\times$ 8), Graphics (GeForce RTX 2080 Ti/PXCIe/SSE2) and 64-b operating system (OS).

A. Data Preprocessing

Large network traffic data in the Bot-IoT data set were preprocessed to transform the features and the ground-truth labels into appropriate formats for ease of computation. In this study, data preprocessing involves the following: 1) elimination of redundant network information; 2) random division of complete network traffic data into training, validation, and testing sets; 3) selection of network traffic features and ground-truth labels; 4) normalization or scaling of network traffic features; and 5) 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 is 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 multiclass classification scenarios. Elements of the high-dimensional feature set were normalized to a range of [0, 1] using min–max transformation method given by

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

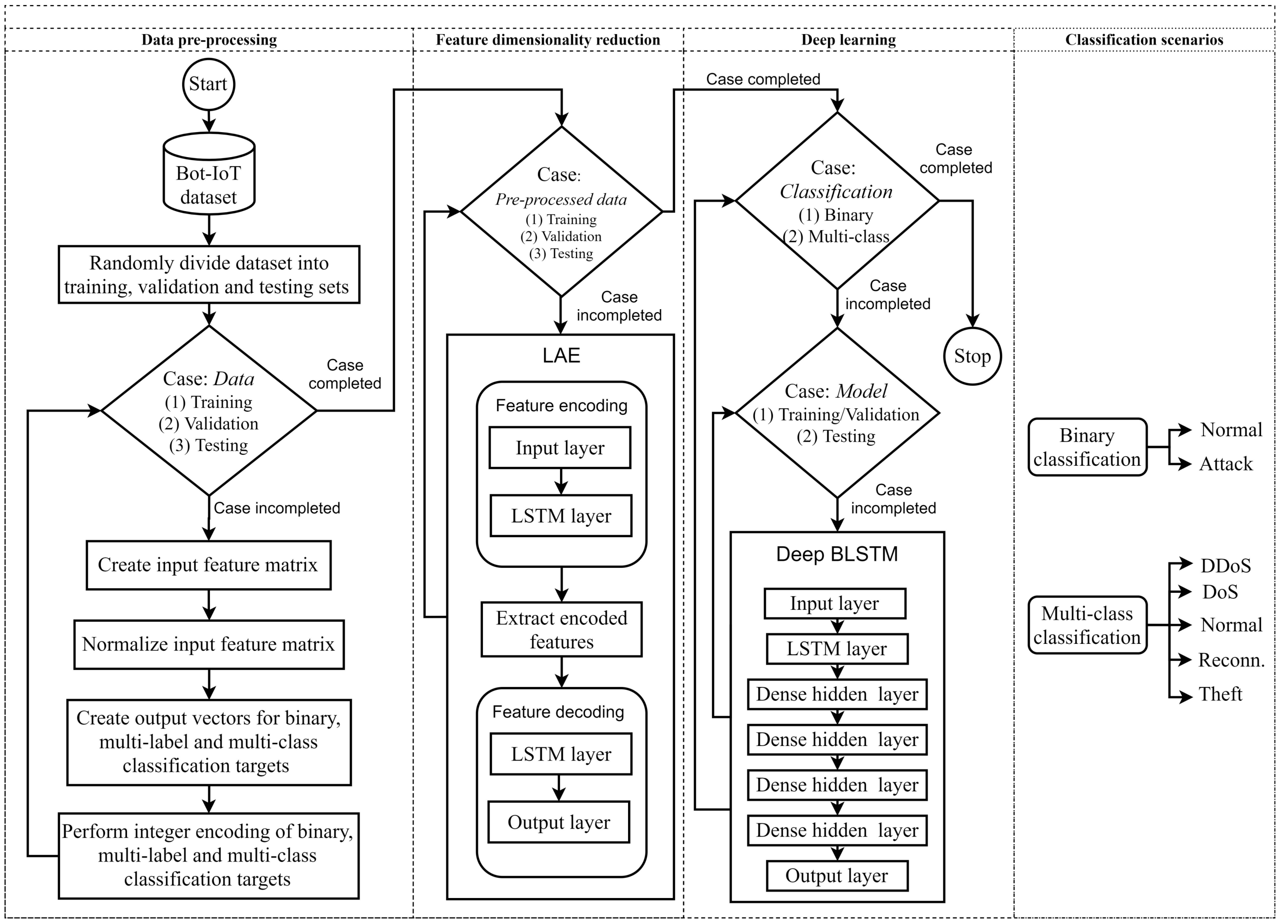
Data pre-processing
Feature dimensionality reduction
Deep learning
Classification scenarios
Start
Bot-IoT dataset
Randomly divide dataset into training, validation and testing sets
Case: Data (1) Training (2) Validation (3) Testing
Case completed
Case incompleted
Create input feature matrix
Normalize input feature matrix
Create output vectors for binary, multi-label and multi-class classification targets
Perform integer encoding of binary, multi-label and multi-class classification targets
Case: Pre-processed data (1) Training (2) Validation (3) Testing
LAE
Feature encoding
Input layer
LSTM layer
Extract encoded features
Feature decoding
LSTM layer
Output layer
Case: Classification (1) Binary (2) Multi-class
Stop
Case: Model (1) Training/Validation (2) Testing
Deep BLSTM
Input layer
LSTM layer
Dense hidden layer
Dense hidden layer
Dense hidden layer
Dense hidden layer
Output layer
Binary classification
Normal
Attack
Multi-class classification
DDoS
DoS
Normal
Reconn.
Theft

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

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 preprocessed high-dimensional network traffic feature set was named RAW-F.

On the other hand, binary and multiclass 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, multiclass 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 III-A. 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 optimization 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 regularized leader (FTRL) [86]. To ensure efficient feature dimensionality reduction, we investigated and compared the performance of LAE model when each of these optimization 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.

C. Deep BLSTM for Network Traffic Classification

Given the low-dimensional feature set, LAE-F, binary, and multiclass deep BLSTM models were developed to correctly detect benign network traffic and botnet attack traffic in IoT networks. The network architecture of the deep BLSTM model is shown in Fig. 1, and its principles of operation were

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

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

previously explained in Section III-B. In this article, 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 multiclass 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 multiclass 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 multiclass classification, respectively. Deep BLSTM models were trained with Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimization algorithms.

Table III shows that the distribution of network traffic samples in Bot-IoT data set is highly imbalanced across the classes in both binary and multiclass 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 favor 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

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

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 analyzed the training loss, validation loss, and MCC values to determine the most efficient optimization 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 the multiclass 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 analyze the results of the experiments performed in Section IV to validate the effectiveness of the LAE-BLSTM model for botnet attack detection in IoT networks. The performance of the LAE-BLSTM model is compared with that of the state-of-the-art ML and DL models in binary and multiclass 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 Underfitting and Overfitting:* Model underfitting is evaluated based on the training loss and the MCC value obtained when deep BLSTM model was evaluated with the training set. Model overfitting 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 underfitting. On the other hand, low validation loss and high MCC value on validation set imply that deep BLSTM model is robust against model overfitting.
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 optimizer for efficient feature dimensionality reduction, we analyze the reconstruction losses generated when Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and FTRL optimization algorithms were used to train LAE model. Fig. 2 shows that the use of Adam optimizer produced the lowest reconstruction loss (0.0041–0.0023) throughout the 10-epoch training period. Table IV shows that the use of Adam optimizer outperformed the use of the other seven optimizers 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

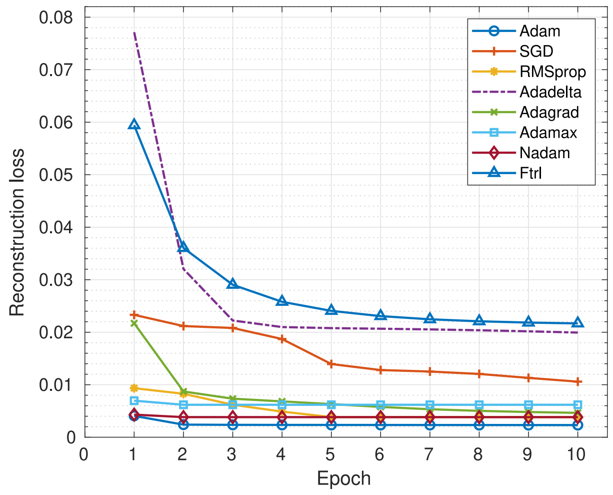

Fig. 2. Trends of reconstruction loss during the training of LAE.

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

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**

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 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 optimizer for efficient network traffic classification in binary scenario, we analyze 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 optimization algorithms.

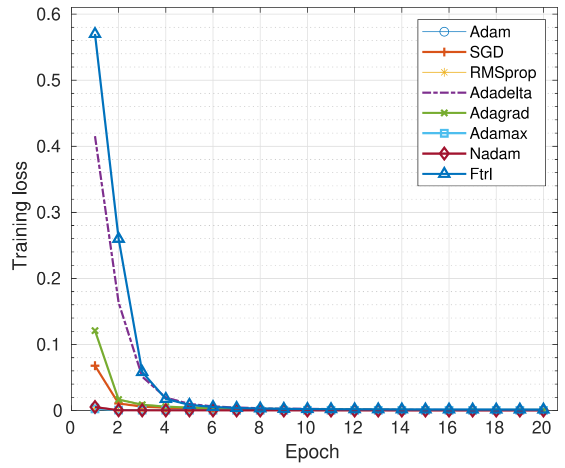

Fig. 3. Training losses of deep BLSTM model in binary classification scenario.

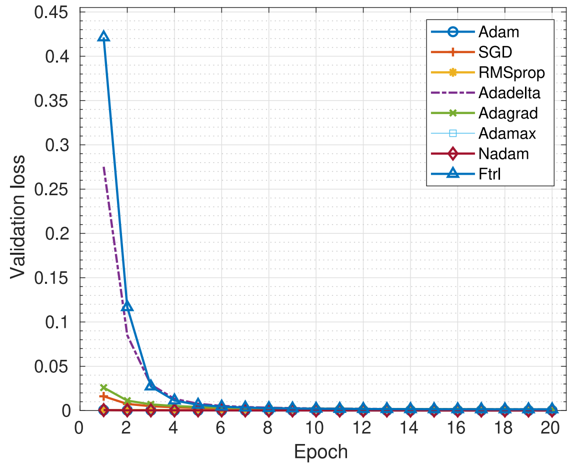

Fig. 4. Validation losses of deep BLSTM model in binary classification scenario.

Fig. 3 shows the training losses of the deep BLSTM model in the 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 when Nadam optimizer was employed. The values of training losses produced by Adam optimizer were very close to those of Nadam optimizer. At the end of the 20-epoch training, Nadam optimizer achieved a training loss of 8.19×10^{-5} while that of Adam optimizer was 8.39×10^{-5}. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers produced relatively high training losses of 0.0011, 0.0006, 0.0013, 0.0013, 0.0003, and 0.0014, respectively.

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 optimizer was employed. The values of validation losses produced by Adam optimizer were very close to those of Nadam optimizer. At the end of the 20-epoch validation, Nadam optimizer achieved a validation loss

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

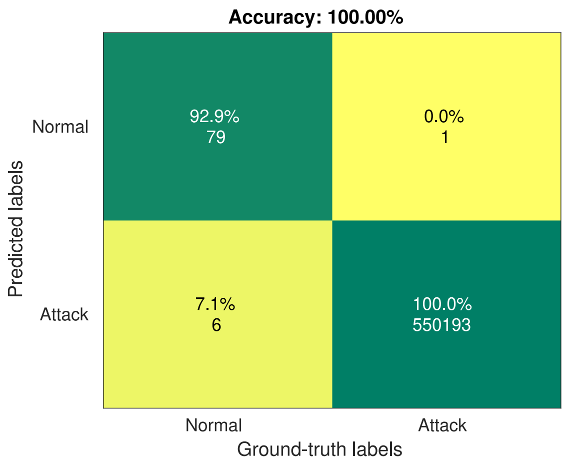

Fig. 5. Confusion matrix of deep BLSTM model in binary classification scenario.

of 8.75×10^{-5} while that of Adam optimizer was 0.0001. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers 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 the binary classification scenario when each of the eight optimization algorithms was used. For SGD, Adadelta, Adagrad, and FTRL optimizers, 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 (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 optimizers could not handle the class imbalance problem in Bot-IoT data set. The classifier was biased in favor 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 optimizer produced the highest MCC values when deep BLSTM model was evaluated with network traffic samples in the training, validation, and testing sets. Nadam optimizer achieved an overall MCC of 93.17%, and it outperformed Adam, RMSprop, and Adamax optimizers by 1.91%, 33.69%, and 12.03%, respectively. Fig. 5 shows the confusion matrix of deep BLSTM model when Nadam optimizer was used for binary classification. The detection rate of

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**

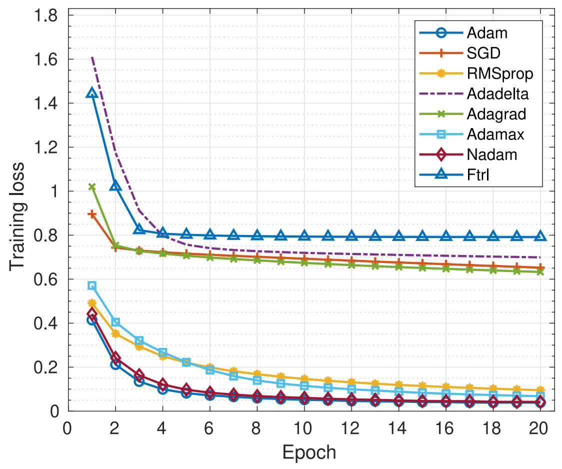

Fig. 6. Training losses of deep BLSTM model in multiclass classification scenario.

normal traffic was 92.90% while that of botnet attack traffic was 100%.

Finally, the performance of the LAE-BLSTM model was compared with that of state-of-the-art models in the 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 Multiclass Scenario*

To determine the most suitable optimizer for network traffic classification in multiclass scenario, we analyze 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 shows the training losses of deep BLSTM model in multiclass classification scenario when each of the eight optimization 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 optimizer was employed. The values of training losses produced by Nadam optimizer were very close to those of Adam optimizer. At the end of the 20-epoch training, Adam optimizer achieved a training loss of 0.0389 while that of Nadam optimizer was 0.0421. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers produced relatively high training losses of 0.6524, 0.0953, 0.6992, 0.6328, 0.0680, and 0.7913, respectively.

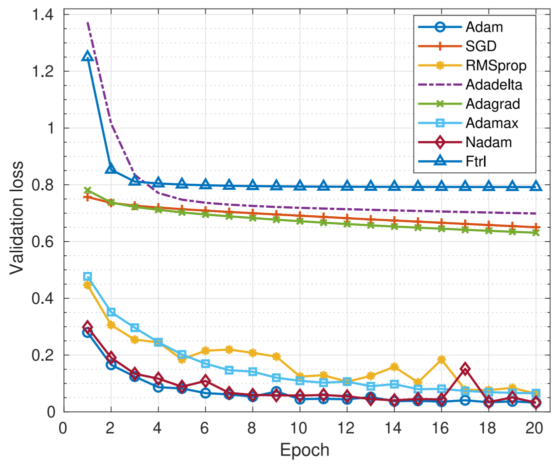

Fig. 7. Validation losses of deep BLSTM model in multiclass classification scenario.

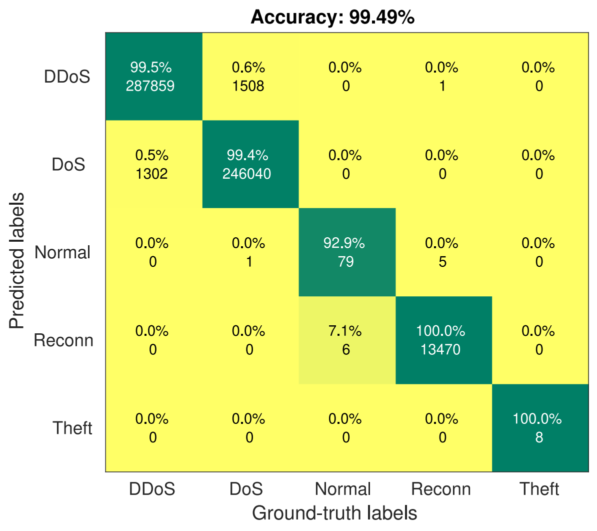

Fig. 8. Confusion matrix of deep BLSTM model in multiclass classification scenario.

Fig. 7 shows the validation losses of deep BLSTM model in multiclass classification scenario when each of the eight optimization 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 optimizer was employed. The values of validation losses produced by Nadam optimizer were very close to those of Adam optimizer. At the end of the 20-epoch validation, Adam optimizer achieved a validation loss of 0.0325 while that of Adam optimizer was 0.0326. On the other hand, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers produced relatively high validation losses of 0.6503, 0.0640, 0.6988, 0.6310, 0.0656, and 0.7920, respectively.

Furthermore, we evaluate the performance of the deep BLSTM model when each of the eight optimization algorithms was used in multiclass classification scenario. Table VIII presents the MCC values of deep BLSTM model when evaluated with network traffic samples in the training set. Adam optimizer achieved the highest MCC value in detecting *normal* traffic. Nadam optimizer achieved the highest MCC values in detecting *DDoS*, *DoS*, *reconnaissance*, and *theft* attacks. This

TABLE VIII
MCC VALUES OF DEEP BLSTM MODEL FOR TRAINING IN MULTICLASS 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

TABLE IX
MCC VALUES OF DEEP BLSTM MODEL FOR VALIDATION IN MULTICLASS 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 MULTICLASS 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

optimizer outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers by 0.08%, 53.87%, 39.81%, 72.06%, 62.70%, 24.24%, and 62.70%, respectively. Table IX presents the MCC values of the deep BLSTM model when evaluated with network traffic samples in the validation set. Nadam optimizer achieved the highest MCC values in all classes. This optimizer outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers by 0.21%%, 53.82%, 39.81%, 71.97%, 23.59%, 23.59%, and 70.33%, respectively. Table X presents the MCC values of the deep BLSTM model when evaluated with network traffic samples in the testing set. Nadam optimizer achieved the highest MCC values in all classes. This optimizer outperformed Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and FTRL optimizers by 0.59%, 55.27%, 41.20%, 73.57%, 64.15%, 25.94%, and 71.91%, respectively.

Fig. 8 shows the confusion matrix of the deep BLSTM model when Nadam optimizer was used for multiclass 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 the LAE-BLSTM model was compared with that of state-of-the-art models in the multiclass classification scenario. Table XI

TABLE XI
PERFORMANCE IN ML/DL MODELS IN MULTICLASS 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**

shows that the 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 data set (Bot-IoT) in binary and multiclass 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 analyze the long-term interrelated changes in low-dimensional feature set produced by LAE, demonstrated robustness against model underfitting and overfitting, 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 *et al.*, "Virtual microgrids: A management concept for peer-to-peer energy trading," in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst.*, 2018, pp. 1–5.

[2] A. O. Akmandor, Y. Hongxu, and N. K. Jha, "Smart, secure, yet energy-efficient, Internet-of-Things sensors," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 914–930, Oct./Dec. 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 Things J.*, vol. 7, no. 5, pp. 3935–3947, May 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 Things J.*, vol. 6, no. 5, pp. 7386–7394, Oct. 2019.

[5] L. Chettri and R. Bera, "A comprehensive survey on Internet of Things (IoT) towards 5G wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[6] M. R. Palattella *et al.*, "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 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 Things J.*, vol. 6, no. 2, pp. 1731–1740, Apr. 2019.

[8] P. Verma and S. K. Sood, "Fog assisted-IoT enabled patient health monitoring in smart homes," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1789–1796, Jun. 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 J. Biomed. Health Informat.*, vol. 24, no. 3, pp. 898–906, Mar. 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 Things J.*, vol. 6, no. 2, pp. 2449–2462, Apr. 2019.

[11] F. Tao, J. Cheng, and Q. Qi, "IIHub: An industrial Internet-of-Things hub toward smart manufacturing based on cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2271–2280, May 2018.

[12] Y.-C. Lin *et al.*, "Development of advanced manufacturing Cloud of Things (AMCoT)—A smart manufacturing platform," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1809–1816, Jul. 2017.

[13] J. Wan *et al.*, "Software-defined industrial Internet of Things in the context of industry 4.0," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7373–7380, Oct. 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. 55290–55304, 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 Things J.*, vol. 7, no. 7, pp. 6023–6034, Jul. 2020.

[16] H. Serra *et al.*, "A 0.9-V analog-to-digital acquisition channel for an IoT water management sensor node," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 10, pp. 1678–1682, Oct. 2019.

[17] N. Ahmed, D. De, and I. Hussain, "Internet of Things (IoT) for smart precision agriculture and farming in rural areas," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4890–4899, Dec. 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 Things J.*, vol. 5, no. 5, pp. 3758–3773, Oct. 2018.

[19] M. Hammoudeh *et al.*, "A service-oriented approach for sensing in the Internet of Things: Intelligent transportation systems and privacy use cases," *IEEE Sensors J.*, early access, Mar. 18, 2020, doi: 10.1109/JSEN.2020.2981558.

[20] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, "Internet of Things: Evolution and technologies from a security perspective," *Sustain. Cities Soc.*, vol. 54, Mar. 2020, Art. no. 101728.

[21] A. Ikpehai *et al.*, "Low-power wide area network technologies for Internet-of-Things: A comparative review," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2225–2240, Apr. 2019.

[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 Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1373–1384, Feb. 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," *J. Supercomput.*, vol. 76, pp. 2643–2664, Apr. 2020.

[24] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *Proc. 26th USENIX Security Symp. (USENIX Security)*, 2017, pp. 1093–1110.

[25] I. Ghafir *et al.*, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Gener. Comput. Syst.*, vol. 89, pp. 349–359, Dec. 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. 61764–61785, 2019.

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

[28] Y. Wang *et al.*, "Distributed learning for automatic modulation classification in edge devices," *IEEE Wireless Commun. Lett.*, early access, Aug. 14, 2020, doi: 10.1109/LWC.2020.3016822.

[29] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton Univ. 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 Trans. Geosci. Remote Sens.*, vol. 55, no. 6, pp. 3071–3083, Jun. 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 Trans. Cybern.*, vol. 49, no. 7, pp. 2406–2419, Jul. 2019.

[32] J. Peng, W. Sun, and Q. Du, "Self-paced joint sparse representation for the classification of hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1183–1194, Feb. 2019.

[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 Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 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 Gener. Comput. Syst.*, vol. 107, pp. 95–111, Jun. 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 *Proc. Int. Conf. Inf. Syst. Security Privacy*, 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 *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.

[38] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *Int. J. Netw. 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 *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[40] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *Proc. Int. Conf. Contemp. Comput.*, 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 *Proc. IEEE 16th Int. Conf. Adv. Commun. Technol.*, 2014, pp. 1265–1269.

[42] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning techniques to identify botnet traffic," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, 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," *Comput. Security*, vol. 40, pp. 95–107, Feb. 2014.

[44] S. Behal and K. Kumar, "Detection of DDoS attacks and flash events using information theory metrics—An empirical investigation," *Comput. Commun.*, vol. 103, pp. 18–28, May 2017.

[45] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometr. Intell. Lab. Syst.*, vol. 2, no. 1–3, pp. 37–52, 1987.

[46] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Stat.*, vol. 36, no. 3, 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, "Learning deep architectures for Ai," *Found. Trends Mach. Learn.*, 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 Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 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. 30373–30385, 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. 42210–42219, 2019.

[52] A. Dawoud, S. Shahristani, and C. Raun, "Dimensionality reduction for network anomalies detection: A deep learning approach," in *Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl.*, 2019, pp. 957–965.

[53] F. C. Schuartz, M. Fonseca, and A. Munaretto, "Improving threat detection in networks using deep learning," *Ann. Telecommun.*, vol. 75, nos. 3–4, pp. 133–142, 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. 33353–33361, 2018.

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

[56] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, 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. 52843–52856, 2018.

[58] S. Gurung, M. K. Ghose, and A. Subedi, "Deep learning approach on network intrusion detection system using NSL-KDD dataset," *Int. J. Comput. Netw. Inf. Security*, 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," *J. Inf. Security Appl.*, vol. 41, pp. 1–11, Aug. 2018.

[60] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *Proc. 7th Conf. Inf. Knowl. Technol. (IKT)*, 2015, pp. 1–5.

[61] D. C. Ferreira, F. I. Vázquez, and T. Zseby, "Extreme dimensionality reduction for network attack visualization with autoencoders," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 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. 83286–83296, 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. 161725–161740, 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 Inf. 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," *Bull. Netw. Comput. Syst. Softw.*, 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 *Proc. IEEE Int. Conf. Cyberworlds (CW)*, 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 Trans. Image Process.*, vol. 27, no. 2, pp. 778–790, Feb. 2018.

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

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

[72] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, 2005.

[73] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 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," *Med. Biol. Eng. Comput.*, vol. 58, no. 3, pp. 519–528, 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," *Sci. China Inf. Sci.*, vol. 63, no. 2, 2020, Art. no. 129401.

[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," *Med. Phys.*, vol. 47, no. 4, pp. 1421–1430, 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," *Adv. Eng. Informat.*, vol. 43, Jan. 2020, Art. no. 101030.

[78] G. H. F. de Oliveira *et al.*, "Estimation and classification of popping expansion capacity in popcorn breeding programs using NIR spectroscopy," *J. Cereal Sci.*, vol. 91, Jan. 2020, Art. no. 102861.

[79] D. Tanikić, "Computationally intelligent optimization of metal cutting regimes," *Measurement*, vol. 152, Feb. 2020, Art. no. 107358.

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

[81] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 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, pp. 1–31, 2012.

[83] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012. [Online]. Available: arXiv:1212.5701.

[84] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.

[85] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, vol. 1, 2016, pp. 2013–2016.

[86] H. B. McMahan *et al.*, "Ad click prediction: A view from the trenches," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 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 Recognit.*, vol. 91, pp. 216–231, Jul. 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 Gener. Comput. Syst.*, vol. 113, pp. 255–265, Dec. 2020.

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

Segun I. Popoola received the B.Tech. degree (Hons.) in electronic and electrical engineering from the Ladoke Akintola University of Technology, Ogbomosho, Nigeria, in 2014, and the M.Eng. degree in information and communication engineering from the Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria, in 2018. 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.

Mr. Popoola is a Registered Engineer with the Council for the Regulation of Engineering in Nigeria.

Bamidele Adebisi (Senior Member, IEEE) received the B.S. degree in electrical engineering from Ahmadu Bello University, Zaria, Nigeria, in 1999, and 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. In 2012, he joined Manchester Metropolitan University, Manchester, U.K., where he is currently a Professor of 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.

Prof. Adebisi is a member of IET.

Mohammad Hammoudeh (Senior Member, IEEE) received the B.Sc. degree in computer communications from Arts Sciences and Technology University, Beirut, Lebanon, in 2004, the M.Sc. degree in advanced distributed systems from the University of Leicester, Leicester, U.K., in 2006, and the Ph.D. degree in computer science from the University of Wolverhampton, Wolverhampton, U.K., in 2008.

He joined Manchester Metropolitan University, Manchester, U.K., in 2009, where he is currently a Professor (Chair) of Cyber Security and the Head of the CfACS Internet of Things Lab, Department of Computing and Mathematics. His research interests are in the communication and security protocols of highly decentralised systems and their applications to smart critical infrastructure.

Dr. Mohammad has been awarded above £1.5 million in competitive research funding for research projects with industrial partners. He is a Fellow of the Higher Education Academy, U.K.

Guan Gui (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2012.

In 2009, he joined Tohoku University, Sendai, Japan, as a Research Assistant, where he was a Postdoctoral Research Fellow in 2014. 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, nonorthogonal multiple access, wireless power transfer, and physical layer security. He has authored more than 200 IEEE journal/conference papers.

Dr. Gui was a recipient of the Top Editor Award of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2019. He won several best paper awards. He is serving or served on the editorial boards of several journals.

Haris Gacanin (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from the University of Sarajevo, Sarajevo, Bosnia and Herzegovina, in 2000, and the M.Sc. and Ph.D. degrees from Tohoku University, Sendai, Japan, in 2005 and 2008, respectively.

From 2008 to 2010, he was with Tohoku University, 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 (currently, Nokia), Boulogne-Billancourt, France, where he leads a Research Department with Nokia Bell Labs, Murray Hill, NJ, USA. His professional interest is related to the application of artificial intelligence to enable autonomous networking and design of mobile and wireless systems. He has over 200 scientific publications (journals, conferences, and patent applications) and invited/tutorial talks.

Dr. Gacanin is a Senior Member of the Institute of Electronics, Information, and Communication Engineering.