Stacked Recurrent Neural Network for Botnet Detection in Smart Homes

Segun I. Popoola[a], Bamidele Adebisi[a,*], Mohammad Hammoudeh[b], Haris Gacanin[c], Guan Gui[d]

[a] *Department of Engineering, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom.*
[b] *Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom.*
[c] *Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany.*
[d] *College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China.*

Abstract

Internet of Things (IoT) devices in Smart Home Network (SHN) are highly vulnerable to complex botnet attacks. In this paper, we investigate the effectiveness of Recurrent Neural Network (RNN) to correctly classify network traffic samples in the minority classes of highly imbalanced network traffic data. Multiple layers of RNN are stacked to learn the hierarchical representations of highly imbalanced network traffic data with different levels of abstraction. We evaluate the performance of Stacked RNN (SRNN) model with Bot-IoT dataset. Results show that SRNN outperformed RNN in all classification scenarios. Specifically, SRNN model learned the discriminating features of highly imbalanced network traffic samples in the training set with better representations than RNN model. Also, SRNN model is more robust and it demonstrated better capability to effectively handle over-fitting problem than RNN model. Furthermore, SRNN model achieved better generalisation ability in detecting network traffic samples of the minority classes.

Keywords: Internet of Things, botnet detection, network traffic, deep learning, recurrent neural network, smart home.

1. Introduction

Smart home is an integral part of smart grid, which utilizes bidirectional flow of electricity and information to form a distributed energy network. Compared to traditional grid, smart grid promises to deliver more efficient and effective power management, scalability, better system reliability, improved resilience, cost-effectiveness and clean energy production. This is accomplished by introducing Distributed Energy Resources (DER), modern control systems and advanced networking and communication technologies. Intelligent electrical appliances and smart meters in homes heavily rely on Internet of Things (IoT) technologies for bidirectional Machine-to-Machine (M2M) communication with one another and other components of smart grid over the Internet. This interconnection in home environment is referred to as Smart Home Network (SHN). Today, heterogeneity of communication protocols, distributed nature and proliferation of IoT systems have exposed SHN to serious cyber-attack vulnerabilities. Therefore, the confidentiality, integrity and availability of IoT devices, online applications and sensitive data generated in SHN can be easily compromised.

Botnet has become a major cybersecurity threat to effective functioning of IoT-enabled networked infrastructures [1]. A botnet is made up of several compromised devices (known as bots), which are remotely controlled by a botmaster through a Command and Control (C&C) server. Advanced botnets employ Peer-to-Peer (P2P) network topology to avoid a single point of failure and evade detection. Vormayr *et al* [1] extensively discussed the topologies, attack scenarios, protocols and complexity of 18 known botnets from network communication perspective. In September 2016, a new botnet named *Mirai* launched a Distributed Denial of Service (DDoS) attack of up to 1.1 Tbps [2]. Also, it is now clear that IoT botnet of high wattage devices can disrupt the power grid [3]. Smart grids are vulnerable to MAnipulation of Demand (MAD) attacks [4]. In the same vein, hackers can easily infect IoT devices in SHN with malware to join an existing botnet or form a new one. Specifically, botnets can be used to launch complex attacks such as Denial of Service (DoS), DDoS, Service Scanning (SS), Operating System (OS) fingerprinting, Data Ex-filtration (DE) and keylogging (KL) against SHN [5, 6]. Therefore, it is very important to protect SHN against possible botnet attacks.

Sophisticated mechanisms such as encryption, authentication and network access control were primarily designed to protect traditional computer networks against cyber-attacks. Unfortunately, these mechanisms are not effective for IoT network security due to limited computation, communication and energy resources in IoT devices [7]. Machine Learning (ML) and Deep Learning (DL) methods were developed for botnet detection in IoT networks [6]. In these methods, behavioural analysis was

*Corresponding author
Email addresses: segun.i.popoola@stu.mmu.ac.uk (Segun I. Popoola), b.adebisi@mmu.ac.uk (Bamidele Adebisi), m.hammoudeh@mmu.ac.uk (Mohammad Hammoudeh), harisg@ice.rwth-aachen.de (Haris Gacanin), guiguan@njupt.edu.cn (Guan Gui)

conducted to classify network traffic as either *normal* or *anomalous* depending on certain discriminating features [8].

Similar to most real-life classification tasks [9, 10], network traffic data in SHN follows a long tail distribution [6]. For instance, samples of network traffic generated by botnets will be far more than those generated by real IoT devices in case of DDoS attack. In such case, the combined network traffic data available for ML and DL model training will be imbalanced. In ML and DL methods, training data is considered to be highly imbalanced when the Imbalance Ratio (IR)[1] is greater than 10:1 [11]. Class imbalance problem affects the effectiveness of state-of-the-art ML and DL methods such that the models are biased in favour of class(es) with majority samples [12]. That is, a large percentage of samples in the minority class(es) may not be correctly classified. In the literature, Recurrent Neural Network (RNN) is one of the state-of-the-art methods proposed for botnet detection [6, 13, 14, 15]. However, this method may not be most suitable to handle class imbalance problem that is often encountered in the process of developing ML/DL models for botnet detection in real-life SHN.

In this paper, botnet detection in SHN is formulated as a network traffic classification problem. We seek to improve the effectiveness of RNN architecture for more accurate classification of highly imbalanced network traffic data in SHN. The main contributions of this paper are as follows:

1. Multiple layers of Recurrent Neural Network (RNN) are stacked to learn hierarchical representations of highly imbalanced network traffic data with different levels of abstraction. The proposed deep learning architecture is named Stacked Recurrent Neural Network (SRNN);
2. A methodology is proposed for deep learning model development and experimentation. RNN and SRNN models are developed with highly imbalanced samples in a typical SHN traffic dataset (Bot-IoT [6]) to validate the performance of the proposed framework in binary and multi-class classification scenarios;
3. We evaluate and compare the learning efficiency of RNN, SRNN, and state-of-the-art ML/DL architectures based on training cross-entropy loss while the robustness of the deep learning models against overfitting was assessed based validation cross-entropy loss;
4. Lastly, the generalisation performance of RNN and SRNN models was evaluated based on accuracy, precision, recall, F1 score, False Positive Rate (FPR), Negative Predictive Value (NPV), Area Under receiver operating characteristic Curve (AUC), Geometric Mean (GM) and Matthews Correlation Coefficient (MCC).

The remaining parts of the paper is organised as follows: In Section 2, we review related state-of-the-art ML and DL. In Section 3, we explain the RNN and SRNN architectures proposed for botnet detection in SHN networks. Extensive experiments are performed in Section 4 to validate the effectiveness of RNN and SRNN models. Experimentation results are presented and discussed in Section 5. Finally, we conclude the paper in Section 6.

2. Review of Related Works

In this section, we review the state-of-the art ML and DL methods that were proposed for IoT botnet detection in the literature. This review of related works is limited to ML and DL models that were evaluated with network traffic data generated in a typical SHN environment. Table 1 shows the degree of class imbalance in the training data used for ML/DL model development.

In the literature, different ML and DL methods have been proposed for botnet detection in SHN. Koroniotis *et al* [6] investigated the effectiveness of Support Vector Machine (SVM), RNN and Long Short Term Memory (LSTM). Ferrag and Maglaras [13] applied RNN. Ibitoye *et al* [21] and Ge *et al* [22] employed Feed-Forward Neural Network (FFNN). Khraisat *et al* [18] combined C5 decision tree with One-Class Support Vector Machine (OC-SVM). Ferrag *et al* [14] recommended a discriminative DL method and a generative DL method namely Convolutional Neural Network (CNN) and Deep Autoencoder (DAE) respectively. Alkadi *et al* [23] combined Gaussian mixture method with a local outlier factor function to form Mixture Localization-based Outliers (MLO) method.

Asadi *et al* [24] combined Particle Swarm Optimization (PSO) algorithm with a voting system, which comprised of Deep Neural Network (DNN), SVM and C4.5 decision tree. The hybrid model, DNN-SVM-C4.5, is referred to as DSC in this paper. Shafiq *et al* [25] selected Naive Bayes (NB) as most effective ML method using bijective soft set algorithm. Ferrag *et al* [19] developed a method by combining REP Tree with JRip algorithm and Forest PA. Soe *et al* [26] combined CST-GR feature selection method with J48 classifier. Aldhaheri *et al* [27] adopted Dendritic Cell Algorithm (DCA) and Self-normalizing Neural Network (SNN) to form a new framework. Koroniotis *et al* [16] proposed a new network forensics framework which employed PSO algorithm and DNN classifier. Das *et al* [28] recommended Random Tree (RT) as an effective ML method. Shafiq *et al* [29] identified C4.5 decision tree as an effective classifier.

Bhuvaneswari and Selvakumar [17] developed Vector Convolutional Deep Learning (VCDL) model with 43 network traffic features and 3037933 samples for 2-class and 5-class classification. Min-max transformation method was used to normalise network traffic features in the range of

[1] Imbalance Ratio (IR) of a given class is the number of samples in the class divided by the total number of samples in the remaining classes

Table 1: IR of the training data used for ML/DL model development

Scenario	Class	Abbrev.	[6]	[16]	[17]	[18]	[13]	[14]	[19]	[20]	Ours
Binary	Attack	Attack	0	0	0	-	-	-	-	-	0
	Normal	Norm	7690	7931	2983	-	-	-	-	-	7900
5-class	DDoS	DDoS	-	-	0	6	-	-	-	-	1
	DoS	DoS	-	-	2	2	-	-	-	-	1
	Normal	Norm	-	-	2983	1	-	-	-	-	7900
	Reconnaissance	Reconn	-	-	25	61	-	-	-	-	39
	Data theft	Theft	-	-	25528	252	-	-	-	-	45049
11-class	DDoS-HTTP	DD-H	-	-	-	-	3162	3714	3714	-	3710
	DDoS-TCP	DD-T	-	-	-	-	3	3	3	-	3
	DDoS-UDP	DD-U	-	-	-	-	3	3	3	-	3
	DoS-HTTP	D-H	-	-	-	-	1993	2473	2473	-	2454
	DoS-TCP	D-T	-	-	-	-	5	5	5	-	5
	DoS-UDP	D-U	-	-	-	-	3	3	3	-	3
	Normal	Norm	-	-	-	-	469	769	769	-	7900
	OS Fingerprinting	OSF	-	-	-	-	208	204	204	-	204
	Service Scanning	SS	-	-	-	-	50	49	49	-	49
	Data ex-filtration	DE	-	-	-	-	18416	62527	62527	-	641967
	Key-logging	KL	-	-	-	-	17721	5001	5001	-	48449

$[0, 1]$. Also, VCDL-10 model was developed with the best 10 features in [6]. The VCDL and VCDL-10 models employed a vector convolutional network for feature extraction and a fully-connected network for classification task. The convolutional network had two convolutional layers and two pooling layers while the fully-connected network had two hidden layers with 9 and 7 neurons, respectively.

Ferrag and Maglaras [13] developed RNN model with 43 network traffic features and 1878561 samples in the training set. Ferrag *et al* [14] developed discriminative ML models (DNN, RNN and CNN) and generative ML models namely Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), Deep Boltzmann machine (DBM) and Deep Autoencoder (DAE) with 43 features and 5877647 samples in the training set. Ferrag *et al* [19] developed a hierarchical botnet detection system with 43 features and 5877647 samples in the training set.

Alkadi *et al* [20] developed Bidirectional LSTM (BLSTM) model with 60% of samples in the Bot-IoT dataset. The values of network traffic features were scaled into a range of $[0, 1]$. BLSTM model employed 2 hidden layers with 60 neurons each, binary cross-entropy loss function, Adam optimizer, 200 epochs and a batch size of 100. Hyperbolic tangent and softmax activation functions were used at the hidden and output layers.

In the literature, ML and DL models were evaluated with Bot-IoT dataset [6] in binary, 5-class and 11-class classification scenarios. Koroniotis *et al* [6], Shafiq *et al* [25], Asadi *et al* [24] and Koroniotis *et al* [16] focused on binary classification scenario. Ibitoye *et al* [21], Khraisat *et al* [18], Ge *et al* [22], Alkadi *et al* [23], Aldhaheri *et al* [27] and Das *et al* [28] focused on 5-class classification scenario. Ferrag et al focused on 11-class classification in [13], [14] and [19]. DDoS attacks were not included in [29]. Likewise, DoS attacks were not included in [26]. In addition, it is computationally expensive to develop separate ML/DL models for the detection of each botnet attack type as done in [6] and [22].

In summary, we present the advantages and shortcomings of the related works. Also, we explain how the method proposed in this paper differs from the previous methods.

1. ML/DL methods in related works considered the effect of class imbalance on classification performance. However, the degree of class imbalance was limited to IR of 25528 and 62527 in 5-class and 11-class classification scenarios, respectively. In this paper, the SRNN model handles high-class imbalance in 5-class and 11-class classification scenarios with IR of 45049 and 641967.
2. ML/DL methods in related works were evaluated with network traffic data of a typical SHN. However, the effectiveness of these methods was not assessed in all the three classification scenarios (i.e. binary, 5-class, and 11-class). In this paper, we evaluate the RNN and SRNN models' performance in all three classification scenarios.
3. The empirical investigation results in [30] showed that MCC is the most reliable metric for evaluating the classification performance of ML/DL in class imbalance cases. The classification performance of some ML/DL methods in related works was evaluated in terms of MCC. However, these values are $< 37\%$ (for Norm class with IR of 7931) and $< 91\%$ (for Norm class with IR of 7931) in binary and 5-class classification scenarios, respectively. The effec-

tiveness of 11-class models in related works was not evaluated in terms of MCC. In this paper, SRNN model achieved higher MCC (> 95%) in all classification scenarios with high IR up to 7900, 45049, and 641967 in binary, 5-class, and 11-class classification scenarios, respectively.

4. The proposed ML/DL methods for binary classification in related works required long training time. The fastest binary model (SVM-10 [6]) was trained for 1270.48 seconds. The authors in [17, 18] did not report the time required to train the 5-class models. For 11-class classification, the fastest model training in [14] took 991.60 seconds. On the other hand, RNN and SRNN methods require relatively lower training time in all classification scenarios.

3. Proposed Model for Botnet Detection in SHN

In this section, we present RNN and SRNN algorithms proposed for botnet detection in SHN as shown in Algorithms 1 and 2. Traffic flow information in SHN is presented as sequential data; and botnet attack detection is formulated as a sequence classification task. Boldface uppercase alphabets and boldface lower case alphabets represent matrices and row vectors respectively.

Traffic flow data in a SHN is represented by Eq. (1):

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n], \tag{1}$$

where n is the number of samples. Features of a network traffic sample is represented by Eq. (2):

$$\mathbf{x} = [x_1, x_2, \ldots, x_j], \tag{2}$$

where j is the number of features in a network traffic sample. Ground truth labels of the network traffic data are represented by Eq. (3):

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]. \tag{3}$$

3.1. Recurrent Neural Network

RNN is used to learn the feature representations of highly imbalanced network traffic data for discriminative classification. The learning is done $\mathbf{X}$ in batches as three-dimensional tensors such that $\mathbf{X} \in \mathbb{R}^{b \times t \times j}$, where b is the batch size and t is the timestep. Information about previously seen network traffic data is stored in hidden state, $\mathbf{h}$. A list of tensors is used to produce initial hidden state, $\mathbf{h}_{init}$. RNN and SRNN processes present network traffic features, $\mathbf{x}$, jointly with initial hidden state, $\mathbf{h}_{init}$, to produce a new hidden state, $\mathbf{h}_{1k}$ given by Eq. (4):

$$\mathbf{h}_{1k} = \sigma_h \left(\mathbf{W}_{xh}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_{init} + \mathbf{b}_h \right), \tag{4}$$

where $\mathbf{W}_{xh}$ is the kernel weight matrix used for linear transformation of the input vector, $\mathbf{x}_k$; $\mathbf{W}_{hh}$ is the recurrent kernel weight matrix used for linear transformation of the recurrent state, $\mathbf{h}_{init}$; $\mathbf{b}_h$ is the bias vector; and σ_h is a Rectified Linear Unit (ReLU).

Algorithm 1: RNN Algorithm

Input: $\mathbf{X}$
Target: $\mathbf{y}$
Output: $\tilde{\mathbf{y}}$

for $e = 1$ *to* u **do**
  for $k = 1$ *to* n **do**
    $\mathbf{h}_0 = \mathbf{h}_{init}$
    $\mathbf{h}_{1k} = \sigma_h \left(\mathbf{W}_{xh}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_0 + \mathbf{b}_h \right)$
    $\tilde{\mathbf{y}}_k = \sigma_y \left(\mathbf{W}_{hv}\mathbf{h}_{1k} + \mathbf{b}_v \right)$
    $L_k = \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
  end
  $L = \sum_{k=1}^{n} \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
  $\mathbf{W}'_{(\cdot)}, \mathbf{b}'_{(\cdot)} = \psi \left(\mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)} \right)$
end

Algorithm 2: SRNN Algorithm

Input: $\mathbf{X}$
Target: $\mathbf{y}$
Output: $\tilde{\mathbf{y}}$

for $e = 1$ *to* u **do**
  for $k = 1$ *to* n **do**
    $\mathbf{h}_0 = \mathbf{h}_{init}$
    $\mathbf{h}_{1k} = \sigma_h \left(\mathbf{W}_{xh}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_0 + \mathbf{b}_h \right)$
    for $m = 2$ *to* d **do**
      $\mathbf{h}_{mk} = \sigma_h \left(\mathbf{W}_{xm}\mathbf{h}_{(m-1)k} + \mathbf{W}_{hm}\mathbf{h}_0 + \mathbf{b}_m \right)$
    end
    $\tilde{\mathbf{y}}_k = \sigma_y \left(\mathbf{W}_{hv}\mathbf{h}_{mk} + \mathbf{b}_v \right)$
    $L_k = \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
  end
  $L = \sum_{k=1}^{n} \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
  $\mathbf{W}'_{(\cdot)}, \mathbf{b}'_{(\cdot)} = \psi \left(\mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)} \right)$
end

A fully-connected dense output layer was used to classify the output of the RNN layer in RNN based on Eq. (5):

$$\tilde{\mathbf{y}}_k = \sigma_y \left(\mathbf{W}_{hv}\mathbf{h}_{1k} + \mathbf{b}_v \right), \tag{5}$$

where $\tilde{\mathbf{y}}_k$ is the predicted label vector of SRNN; σ_y is either a sigmoid or a softmax activation function for binary or multi-class classification respectively; $\mathbf{W}_{hv}$ is the kernel weight matrix used for linear transformation of $\mathbf{h}_{1k}$; and $\mathbf{b}_v$ is the bias vector of the output layer.

Finally, RNN is trained to perform binary and multi-class classification tasks based on Backpropagation Through Time (BPTT) algorithm [31, 32]. Training loss (L) in RNN is minimized using cross-entropy loss function (θ). Binary and multi-class classification performance of RNN is optimized using Adam optimization method [33] given by

Eq. (6):

$$\mathbf{W}^{'}_{(\cdot)}, \mathbf{b}^{'}_{(\cdot)} = \psi\left(\mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)}\right), \tag{6}$$

where ψ is the optimization function; $\mathbf{W}_{(\cdot)}$ and $\mathbf{W}^{'}_{(\cdot)}$ are the old and new Weight matrices respectively; $\mathbf{b}_{(\cdot)}$ and $\mathbf{b}^{'}_{(\cdot)}$ are the old and new bias vectors respectively. The optimization process is performed for u epochs as presented in Algorithm 1.

3.2. Stacked Recurrent Neural Network

Unlike the simple RNN with a single hidden layer, additional $(d-1)$ RNN layers are stacked with the first RNN layer to perform hierarchical feature learning and improve classification performance of highly imbalanced network traffic data in SHN. This neural network, which has two or more RNN layers, was named SRNN.

In SRNN Algorithm presented in Algorithm 2, $\mathbf{W}_{xm}$ is the kernel weight matrix used for linear transformation of $\mathbf{h}_{(m-1)k}$; $\mathbf{W}_{hm}$ is the recurrent kernel weight matrix used for linear transformation of the recurrent state, $\mathbf{h}_0$; $\mathbf{b}_m$ is the bias vector of the additional RNN layers; and m is the number of RNN layers in SRNN. Weight matrices in SRNN are initialized using *He* uniform initialization method [34].

A fully-connected dense output layer was used to classify the output of m^{th} RNN layer in SRNN based on Eq. (7):

$$\tilde{\mathbf{y}}_k = \sigma_y\left(\mathbf{W}_{hv}\mathbf{h}_{mk} + \mathbf{b}_v\right), \tag{7}$$

where $\mathbf{W}_{hv}$ is the kernel weight matrix used for linear transformation of $\mathbf{h}_{mk}$.

Finally, SRNN is trained to perform binary and multi-class classification tasks based on Backpropagation Through Time (BPTT) algorithm. Training loss (L) in SRNN is minimized using cross-entropy loss function (θ). Binary and multi-class classification performance of SRNN is optimized using Adam optimization method [33] given by Eq. (6). The optimization process is performed for u epochs as presented in Algorithm 2.

4. Model Development and Experimentation

In this section, we implement SRNN-based botnet detection in a typical SHN as shown in Fig. 1. RNN and SRNN models are developed with highly imbalanced network traffic data in Bot-IoT dataset [6] to detect and distinguish normal network traffic from botnet attack traffic in binary, 5-class and 11-class classification scenarios.

4.1. Bot-IoT dataset

Bot-IoT dataset [6] was considered the most relevant, publicly available dataset to verify the effectiveness of SRNN method for botnet detection in smart homes. It contains well-labeled features of normal IoT network traffic and IoT botnet attack traffic generated by IoT devices in a typical SHN. These IoT devices include a weather station, a smart fridge, motion-activated lights, a remotely activated garage door and a smart thermostat. The smart devices generated normal IoT network traffic using heterogeneous communication protocols namely: 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). In addition, recent and complex IoT botnet attack scenarios were included in Bot-IoT dataset. These include DoS, DDoS, reconnaissance and information theft. Specifically, there are 10 classes of IoT botnet attacks in Bot-IoT dataset namely: DDoS-HTTP, DDoS-TCP, DDoS-UDP, DoS-HTTP, DoS-TCP, DoS-UDP, Operating System (OS) fingerprinting, service scanning, data ex-filtration and key-logging. Data distribution and IR of normal IoT network traffic and botnet attack traffic are presented in Table 2. More information about Bot-IoT dataset can be found in [6].

4.2. Data pre-processing

As shown in Fig. 1, the model development process and experimentation involves data pre-processing and deep learning. First, we observed that some of the values of source port (*sport*) and destination port (*dport*) are in hexadecimal. To eliminate value error in data processing, these values were converted to decimal. The full list and description of all the features can be found in [6]. Initially, the Bot-IoT dataset contains forty-three (43) features, but there are six redundant features in the dataset: *pkSeqID*, *saddr*, *daddr*, *proto*, *state*, and *flgs*. *pkSeqID* is just a row identifier, and it provides no information about the network traffic. Also, *saddr* and *daddr* are device-specific. Furthermore, *proto*, *state*, and *flgs* contain the same information as *proto_number*, *state_number*, and *flgs_number*. Therefore, thirty-seven (37) features of network traffic in Bot-IoT dataset were extracted to form input feature matrix, $\mathbf{X}$. Similarly, corresponding ground truth labels of $\mathbf{X}$ were also obtained from Bot-IoT dataset [6] to form target vector, $\mathbf{y}$. In order to facilitate feature learning in RNN and SRNN, elements of $\mathbf{X}$ were normalized to a range of $[0, 1]$ using min-max transformation method given by Eq. (8) [35]:

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

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. On the other hand, the elements of binary, 5-class and 11-class label vectors were encoded with integers to ensure ease of computation during model training.

4.3. Deep learning model development

Table 2 shows that highly imbalanced network traffic data in Bot-IoT was randomly divided into training set (70%), validation set (15%) and testing set (15%) to avoid model over-fitting and ensure model generalisation. In order to evaluate and compare the effectiveness of RNN and SRNN models in binary, 5-class and 11-class classification

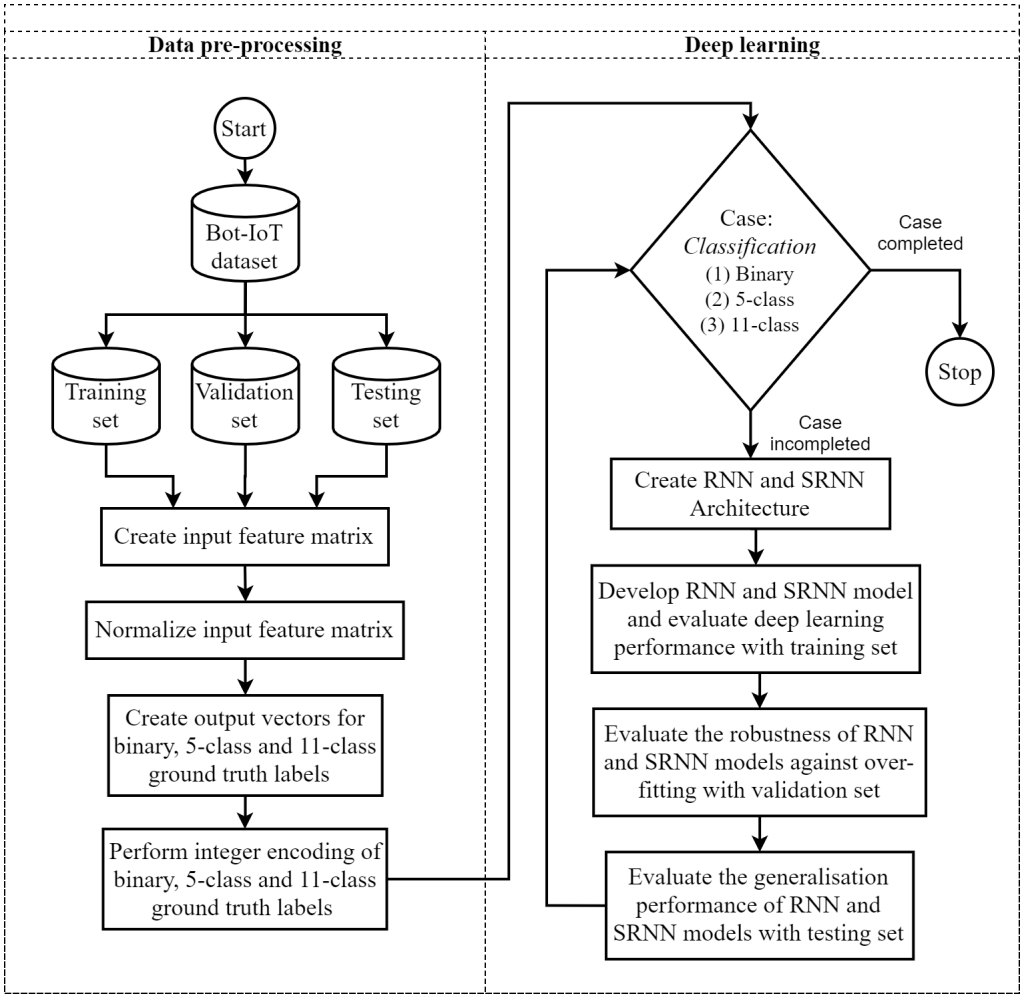

Figure 1: Model development framework for botnet detection in a typical smart home

scenarios, these models were trained, validated and tested with network traffic features and ground truth labels in the training, validation and testing sets respectively.

The deep learning models were trained using the following hyperparameters: 200 units per RNN layer; learning rate of 0.0001; and batch size of 512. The complete information about the hyperparameters of RNN and SRNN models are presented in Tables 3 and 4, respectively. All the experiments performed in this paper leveraged Numpy, Pandas, Scikit-learn and Keras libraries that were developed using Python programming language. Python codes are 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).

5. Results and Discussion

In this section, we present and discuss the results of deep learning model development and experimentation. First, we evaluate the fitness of RNN and SRNN models and their robustness against over-fitting in binary, 5-class and 11-class classification scenarios. Then, we evaluate the generalisation ability of the models in the three classification scenarios.

5.1. Training and validation performance of deep learning models

In this subsection, we assess the performance of RNN and SRNN models when they are trained and validated with network traffic samples in the training and validation sets, respectively. Training losses in binary, 5-class and 11-class classification scenarios are computed and analysed to evaluate the effectiveness of learning the discriminating features of highly imbalanced network traffic data with RNN and SRNN model architectures. Also, we compute and analyse the validation losses in binary, 5-class and 11-class classification scenarios to verify the resilience of RNN and SRNN to model over-fitting.

The training and validation losses of RNN and SRNN models in binary, 5-class and 11-class classification scenarios are shown in Figs. 2 - 8. We observed that SRNN

Table 2: Data distribution for DL model training, validation and testing

Scenario	Class	Abbreviation	Training	Validation	Testing	IR
Binary	Attack	Attack	2,567,548	550,303	550,194	0
	Normal	Norm	325	67	85	7900
5-class	DDoS	DDoS	1,348,654	288,809	289,161	1
	DoS	DoS	1,155,031	247,680	247,549	1
	Normal	Norm	325	67	85	7900
	Reconnaissance	Reconn	63,806	13,800	13,476	39
	Data theft	Theft	57	14	8	45049
11-class	DDoS-HTTP	DD-H	692	140	157	3710
	DDoS-TCP	DD-T	684,011	146,924	146,445	3
	DDoS-UDP	DD-U	663,951	141,745	142,559	3
	DoS-HTTP	D-H	1,046	229	210	2454
	DoS-TCP	D-T	430,892	92,675	92,233	5
	DoS-UDP	D-U	723,093	14,776	155,106	3
	Normal	Norm	325	67	85	7900
	OS Fingerprinting	OSF	12,535	2,708	2,671	204
	Service Scanning	SS	51,271	11,092	10,805	49
	Data ex-filtration	DE	4	1	1	641967
	Key-logging	KL	53	13	7	48449

Table 3: Model hyperparameters for RNN model in binary, 5-class, and 11-class classification scenarios

Hyperparameter	Binary class	5-class	11-class
Input layer neurons	37	37	37
Hidden neurons (layer 1)	200	200	200
Hidden activation function	ReLU	ReLU	ReLU
Output layer neurons	1	5	11
Learning rate	0.0001	0.0001	0.0001
Batch size	512	512	512
Epochs	5	5	5, 10, 15, 20, and 25
Output activation function	Binary cross-entropy	Categorical cross-entropy	Categorical cross-entropy

Table 4: Model hyperparameters for SRNN model in binary, 5-class, and 11-class classification scenarios

Hyperparameter	Binary class	5-class	11-class
Input layer neurons	37	37	37
Hidden neurons (layer 1)	200	200	200
Hidden neurons (layer 2)	200	200	200
Hidden neurons (layer 3)	200	200	200
Hidden neurons (layer 4)	200	200	200
Hidden activation function	ReLU	ReLU	ReLU
Output layer neurons	1	5	11
Learning rate	0.0001	0.0001	0.0001
Batch size	512	512	512
Epochs	5	5	5, 10, 15, 20, and 25
Output activation function	Binary cross-entropy	Categorical cross-entropy	Categorical cross-entropy

model had lower training and validation losses than RNN model in all classification scenarios. For binary classification in Fig. 2, SRNN reduced initial training loss by 99.83% while initial validation loss reduced by 80.46%. At the end of the 5-epoch training, SRNN reduced final training loss by 88.95% while final validation loss reduced by 68.85%. For 5-class classification in Fig. 3, SRNN reduced initial training loss by 76.23% while initial validation loss reduced by 85.74%. At the end of the 5-epoch training, SRNN reduced final training loss by 91.33% while final validation loss reduced by 94.34%. For 11-class classification in Fig. 4, SRNN reduced initial training loss by 75.82%

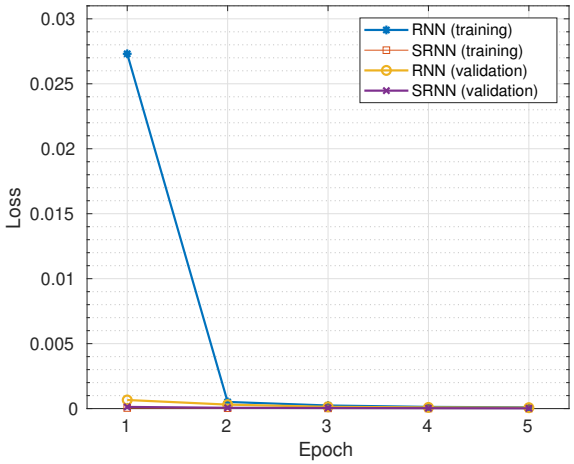

Figure 2: Training and validation losses in binary classification scenario

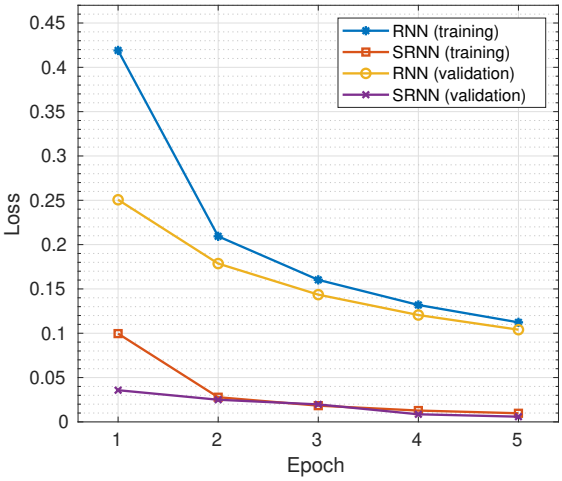

Figure 3: Training and validation losses in 5-class classification scenario

while initial validation loss reduced by 84.18%. At the end of the 5-epoch training, SRNN reduced final training loss by 89.42% while final validation loss reduced by 89.55%. To further reduce the training loss and the validation loss in 11-class classification scenario, the number of epochs was increased from 5 to 10, 15, 20, and 25, as shown in Figs. 5 - 8, respectively. In Figs. 4 and 8, we observed that SRNN reduced final training loss from 0.0124 in 5-epoch training to 0.002198 in 25-epoch training while final validation loss reduced from 0.01122 in 5-epoch validation to 0.001673 in 25-epoch validation.

A lower training loss in SRNN model implies that, compared to RNN model, the probability distribution of labels predicted by SRNN model was closer to that of the ground-truth labels in the training set. Hence, SRNN model learned the discriminating features of highly imbalanced network traffic samples in the training set with

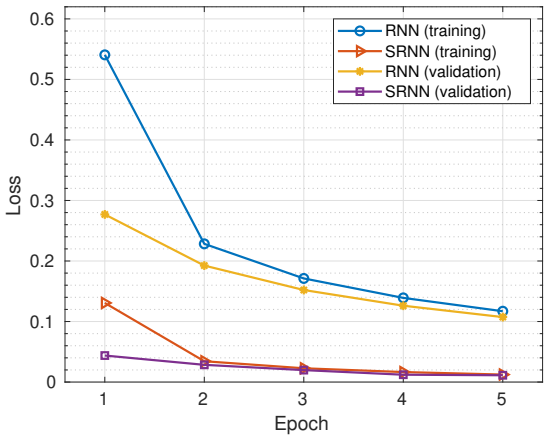

Figure 4: Training and validation losses in 11-class classification scenario (5 epochs)

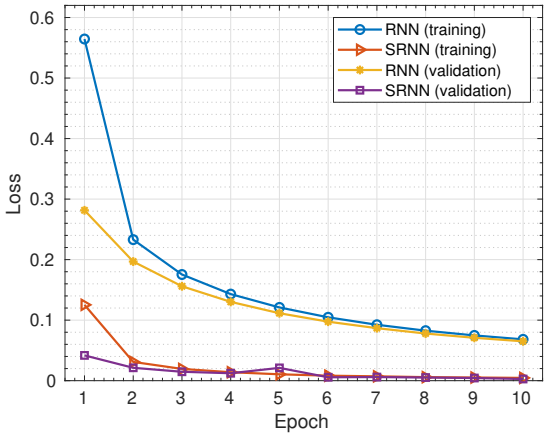

Figure 5: Training and validation losses in 11-class classification scenario (10 epochs)

better representations than RNN model. On the other hand, a lower validation loss in SRNN model means that, compared to RNN model, the probability distribution of labels predicted by SRNN model was closer to that of the ground-truth labels in the validation set. Hence, SRNN model is more robust and it demonstrated better capability to effectively handle over-fitting problem than RNN model.

Table 5: Number of trainable parameters in RNN model

Layer	Binary	5-class	11-class
Hidden layer	47600	47600	47600
Output	201	1005	2211
Total	**47801**	**48605**	**49811**

The number of trainable parameters in RNN and SRNN models are presented in Tables 5 and 6 respectively. We observed that the stacking three RNN layers in SRNN model introduced 240600 additional trainable parameters

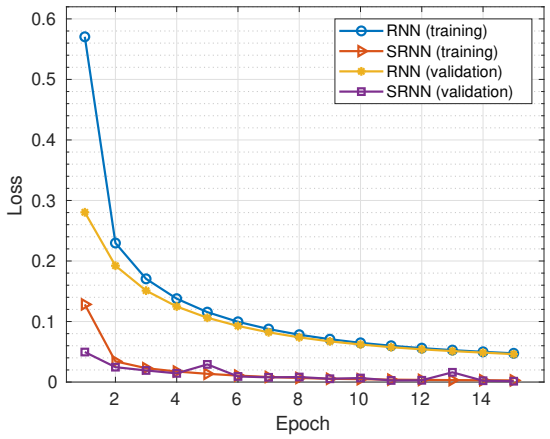

Figure 6: Training and validation losses in 11-class classification scenario (15 epochs)

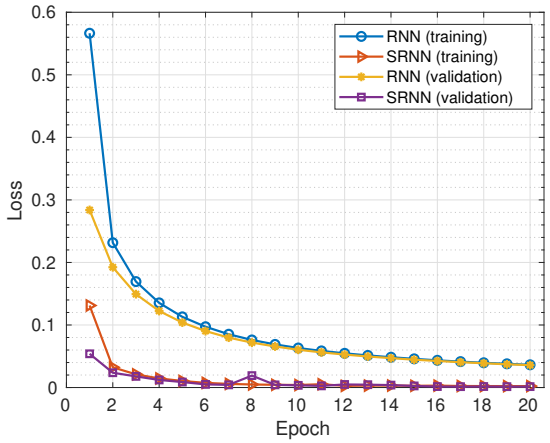

Figure 7: Training and validation losses in 11-class classification scenario (20 epochs)

Table 6: Number of trainable parameters in SRNN model

Layer	Binary	5-class	11-class
Hidden layer 1	47600	47600	47600
Hidden layer 2	80200	80200	80200
Hidden layer 3	80200	80200	80200
Hidden layer 4	80200	80200	80200
Output	201	1005	2211
Total	**288401**	**289205**	**290411**

to learn hierarchical representations of highly imbalanced network traffic data compared to RNN model. By so doing, SRNN model employed different levels of abstraction to learn the most important discriminating features and it discarded irrelevant variations.

The time taken to train RNN and SRNN models in binary, 5-class and 11-class classification scenarios are presented in Table 7. Compared to RNN, we observed that the architecture of SRNN prolonged the training time in

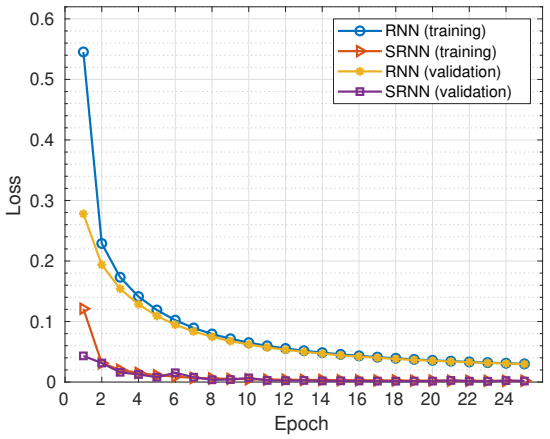

Figure 8: Training and validation losses in 11-class classification scenario (25 epochs)

Table 7: Training time of ML/DL models

Scenario	Model	Training time (s)
Binary	SVM [6]	6636.98
	SVM-10 [6]	1270.48
	RNN [6]	6888.08
	RNN-10 [6]	8035.00
	LSTM [6]	14073.63
	LSTM-10 [6]	10482.19
	MLP [16]	40020.00
	RNN	**43.23**
	SRNN	134.15
5-class	RNN	**53.05**
	SRNN	131.82
11-class	DNN [14]	991.60
	RNN [14]	1400.60
	CNN [14]	1367.20
	RBM [14]	2111.90
	DBN [14]	2921.70
	DBM [14]	2800.10
	DAE [14]	2816.20
	BLSTM [20]	**149.60**
	RNN	178.10
	SRNN	514.30

all the classification scenarios. The computation of the weights of additional trainable parameters in SRNN model extended the time taken to train the model. However, Table 7 shows that the training of SRNN model was faster than that of the state-of-the-art models in all classification scenarios.

5.2. Generalisation performance of deep learning models

In this subsection, we evaluate the generalisation performance of RNN and SRNN models, which were used to predict the labels of highly imbalanced network traffic samples in the testing set. Then, we compute and analyse the accuracy, precision, recall, F1 score, FPR, NPV,

Table 8: Model performance (in %) for each class in binary classification scenario

Class	Model	Accuracy	Precision	Recall	F1	FPR	NPV	AUC	GM	MCC
Attack	SVM [6]	99.99	99.99	**100.00**	99.99	86.58	**100.00**	56.71	99.99	36.63
	SVM-10 [6]	88.37	**100.00**	88.37	93.83	**0.00**	0.11	94.19	3.34	3.14
	RNN [6]	97.91	**100.00**	97.91	98.94	20.55	0.49	88.68	7.01	6.15
	RNN-10 [6]	99.74	99.99	99.75	99.87	73.38	1.37	63.19	11.69	5.98
	LSTM [6]	98.06	**100.00**	98.06	99.02	9.85	0.60	94.10	7.75	7.27
	LSTM-10 [6]	99.74	99.99	99.75	99.87	68.76	1.60	65.49	12.67	7.03
	VCDL [17]	99.75	**100.00**	99.75	99.87	12.16	4.39	93.80	20.95	19.60
	VCDL-10 [17]	99.76	**100.00**	99.76	99.88	9.43	4.62	95.16	21.49	20.42
	MLP [16]	99.90	**100.00**	99.90	99.90	**0.00**	-	100.00	-	-
	DSC [24]	99.91	-	99.76	99.72	0.616	-	-	-	-
	RNN	**100.00**	**100.00**	**100.00**	**100.00**	23.53	**100.00**	88.24	**100.00**	87.45
	SRNN	**100.00**	**100.00**	**100.00**	**100.00**	**0.00**	**96.59**	**100.00**	**98.28**	**98.28**
Normal	RNN	**100.00**	**100.00**	76.47	86.67	**0.00**	**100.00**	88.24	**100.00**	87.45
	SRNN	**100.00**	**96.59**	**100.00**	**98.27**	**0.00**	**100.00**	**100.00**	**98.28**	**98.28**

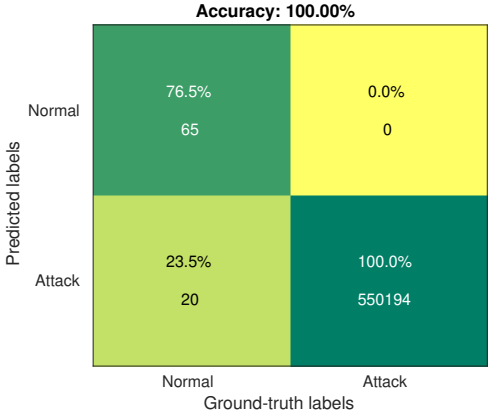

Figure 9: Confusion matrix of RNN in binary classification scenario

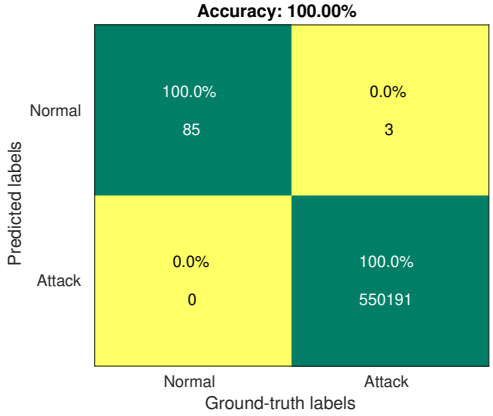

Figure 10: Confusion matrix of SRNN in binary classification scenario

AUC, GM and MCC of RNN and SRNN models in binary, 5-class and 11-class classification scenarios based on Eqs. (9) – (17):

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{9}$$

$$Precision = \frac{TP}{TP + FP}, \tag{10}$$

$$Recall = \frac{TP}{TP + FN}, \tag{11}$$

$$F1 = \frac{2 \times TP}{(2 \times TP) + FP + FN}, \tag{12}$$

$$FPR = \frac{FP}{FP + TN}, \tag{13}$$

$$NPV = \frac{TN}{TN + FN}, \tag{14}$$

$$AUC = \frac{1}{2}\left[\frac{TP}{(TP + FN} + \frac{TN}{(TN + FP}\right], \tag{15}$$

$$GM = \sqrt{\frac{TP}{TP + FP} \times \frac{TN}{TN + FN}}, \tag{16}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \tag{17}$$

where TP is the number of network traffic samples in the positive class that are correctly classified as positive; FP is the number of network traffic samples in the negative class that are misclassified as positive; TN is the number of network traffic samples in the negative class that are correctly classified as negative; and FN is the number of network traffic samples in the negative class that are misclassified as positive.

Accuracy in Eq. (9) is the ratio of the number of samples correctly classified as either positive or negative to the total number of samples in all classes. Precision in Eq. (10) is the ratio of the number of samples in the positive class that are correctly classified as positive to the total number of samples that are either correctly classified or

Table 9: Model performance (in %) for each class in 5-class classification scenario

Class	Model	Accuracy	Precision	Recall	F1	FPR	NPV	AUC	GM	MCC
DDoS	VCDL [17]	99.76	99.68	99.79	99.74	0.26	**99.83**	99.76	99.75	99.52
	VCDL-10 [17]	99.76	99.83	99.71	99.77	0.19	99.68	99.76	99.76	99.52
	OC-SVM [18]	99.84	**100.00**	98.95	99.47	**0.00**	99.82	99.48	**99.91**	99.38
	RNN	96.96	98.12	96.06	97.08	2.04	95.73	97.01	96.92	93.93
	SRNN	**99.87**	99.94	**99.82**	**99.88**	0.07	99.80	**99.87**	99.87	**99.74**
DoS	VCDL [17]	99.76	99.68	99.79	99.74	0.26	99.83	99.76	99.75	99.52
	VCDL-10 [17]	99.76	99.68	99.80	99.74	0.26	99.84	99.77	99.76	99.53
	OC-SVM [18]	99.97	99.52	99.87	99.70	0.26	99.93	99.81	99.73	99.53
	RNN	96.96	95.49	97.85	96.66	3.78	98.21	97.04	96.84	93.89
	SRNN	**99.87**	**99.78**	**99.93**	**99.86**	**0.18**	**99.94**	**99.88**	**99.86**	**99.74**
Norm	VCDL [17]	**100.00**	87.84	87.84	87.84	0.00	**100.00**	93.92	93.72	87.84
	VCDL-10 [17]	**100.00**	90.57	90.57	90.57	0.00	**100.00**	95.28	95.17	90.56
	OC-SVM [18]	94.18	**100.00**	87.97	93.60	0.00	89.85	93.99	94.79	88.91
	RNN	99.99	**100.00**	62.35	76.81	0.00	99.99	81.18	**100.00**	78.96
	SRNN	**100.00**	98.77	**94.12**	**96.39**	0.00	**100.00**	**97.06**	99.38	**96.41**
Reconn	VCDL [17]	99.97	99.07	99.89	99.48	0.02	**100.00**	99.93	99.53	99.47
	VCDL-10 [17]	99.99	99.62	99.92	99.77	0.01	**100.00**	99.95	99.81	99.77
	OC-SVM [18]	94.13	21.46	99.33	35.30	5.96	99.99	96.69	46.33	44.76
	RNN	99.99	99.84	99.73	99.79	**0.00**	99.99	99.86	99.92	99.78
	SRNN	**100.00**	**99.96**	**99.98**	**99.97**	**0.00**	**100.00**	**99.99**	**99.98**	**99.97**
Theft	VCDL [17]	**100.00**	91.80	70.89	80.00	**0.00**	100.00	85.44	95.81	80.67
	VCDL-10 [17]	**100.00**	98.39	77.22	86.52	**0.00**	100.00	88.61	99.19	87.16
	OC-SVM [18]	99.99	98.65	**100.00**	99.32	0.01	100.00	**100.00**	99.32	99.32
	RNN	**100.00**	-	0.00	0.00	**0.00**	100.00	50.00	-	-
	SRNN	**100.00**	**100.00**	**100.00**	**100.00**	**0.00**	100.00	**100.00**	**100.00**	**100.00**

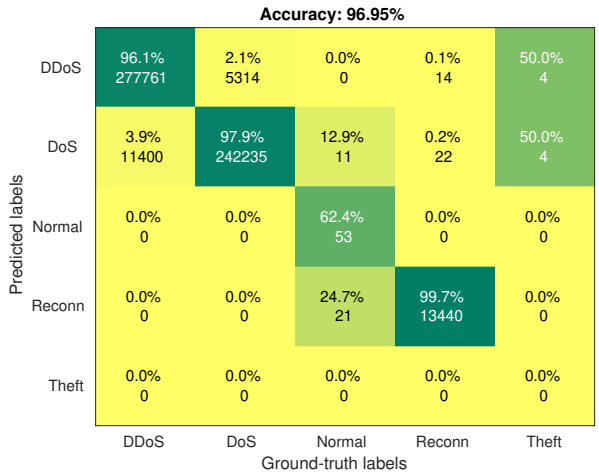

Figure 11: Confusion matrix of RNN in 5-class classification scenario

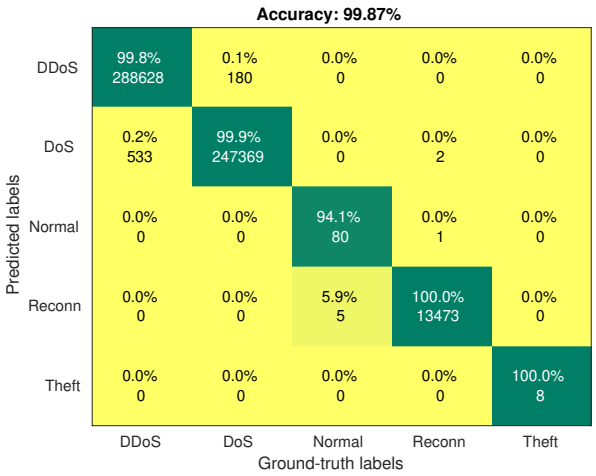

Figure 12: Confusion matrix of SRNN in 5-class classification scenario

misclassified as positive. Recall in Eq. (11) is the ratio of the number of samples in the positive class that are correctly classified as positive to the total number of samples in the positive class. F1 score in Eq. (12) refers to the harmonic mean of precision and recall. It is also known as F-measure. FPR in Eq. (13) is the ratio of the number of samples in the negative class that are misclassified as positive to the total number of samples in the negative class.

NPV in Eq. (14) is the ratio of the number of samples in the negative class that are correctly classified as negative to the total number of samples that are either correctly classified or misclassified as negative. AUC in Eq. (15) is the arithmetic mean of recall and True Negative Rate (TNR), where TNR is the ratio of the number of samples in the negative class that are correctly classified as nega-

Table 10: Model performance (in %) for each class in 11-class classification scenario

Class	Model	Accuracy	Precision	Recall	F1	FPR	NPV	AUC	GM	MCC
DD-H	RNN	99.99	80.14	74.52	77.23	0.01	99.99	87.26	89.52	77.27
	SRNN	**100.00**	**95.60**	**96.82**	**96.20**	**0.00**	**100.00**	**98.41**	**97.77**	**96.20**
DD-T	RNN	99.05	98.79	97.63	98.21	0.43	99.14	98.60	98.97	97.56
	SRNN	**99.99**	**99.99**	**99.98**	**99.99**	**0.00**	**99.99**	**99.99**	**99.99**	**99.98**
DD-U	RNN	100.00	100.00	99.99	100.00	0.00	100.00	100.00	100.00	99.99
	SRNN	100.00	100.00	99.99	100.00	0.00	100.00	100.00	100.00	**100.00**
D-H	RNN	99.98	79.60	76.19	77.86	0.01	99.99	88.09	89.22	77.87
	SRNN	**100.00**	**99.01**	**95.24**	**97.09**	**0.00**	**100.00**	**97.62**	**99.50**	**97.10**
D-T	RNN	99.05	96.30	98.12	97.20	0.76	99.62	98.68	97.94	96.63
	SRNN	**99.99**	**99.97**	**99.99**	**99.98**	**0.01**	**100.00**	**99.99**	**99.98**	**99.98**
D-U	RNN	100.00	99.99	100.00	99.99	0.00	100.00	100.00	99.99	99.99
	SRNN	100.00	99.99	100.00	**100.00**	0.00	100.00	100.00	**100.00**	99.99
Norm	RNN	100.00	90.24	87.06	88.62	0.00	100.00	93.53	95.00	88.64
	SRNN	100.00	**96.43**	**95.29**	**95.86**	0.00	100.00	**97.65**	**98.20**	**95.86**
OSF	RNN	100.00	100.00	99.29	99.64	0.00	100.00	99.64	100.00	99.64
	SRNN	100.00	100.00	**100.00**	**100.00**	0.00	100.00	**100.00**	100.00	**100.00**
SS	RNN	100.00	99.89	99.95	99.92	0.00	100.00	99.98	99.94	99.92
	SRNN	100.00	**99.96**	**99.97**	**99.97**	0.00	100.00	**99.99**	**99.98**	**99.97**
DE	RNN	100.00	0.00	0.00	0.00	0.00	100.00	50.00	0.00	0.00
	SRNN	100.00	**100.00**	**100.00**	**100.00**	0.00	100.00	**100.00**	**100.00**	**100.00**
KL	RNN	100.00	62.50	71.43	66.67	0.00	100.00	85.71	79.06	66.81
	SRNN	100.00	**100.00**	**100.00**	**100.00**	0.00	100.00	**100.00**	**100.00**	**100.00**

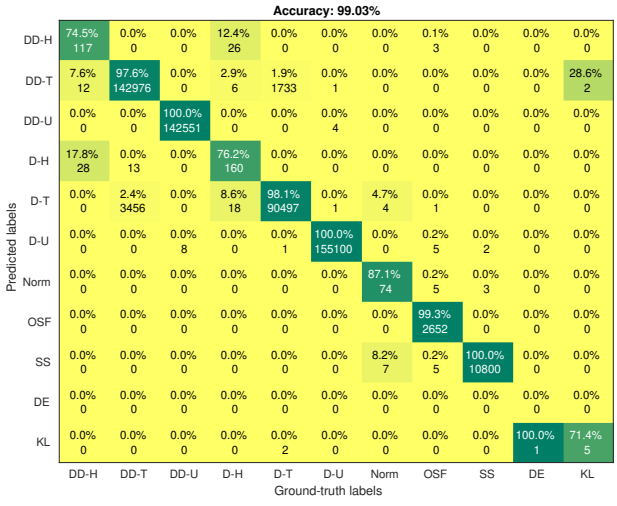

Figure 13: Confusion matrix of RNN in 11-class classification scenario

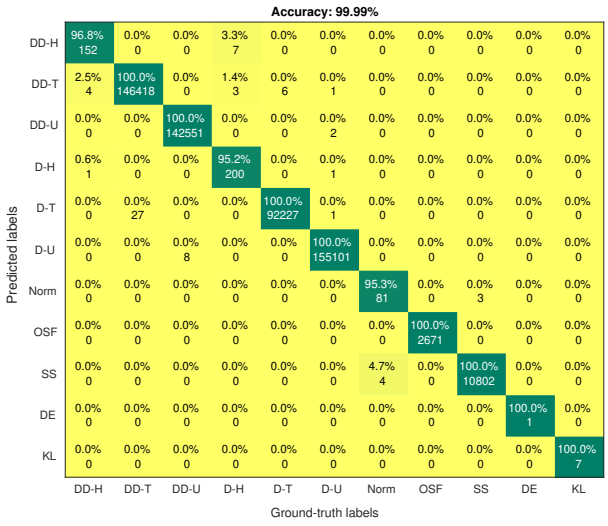

Figure 14: Confusion matrix of SRNN in 11-class classification scenario

tive to the total number of samples in the negative class. GM in Eq. (16) is the geometric mean of precision and NPV. MCC in Eq. (17) is a balanced ratio of TP, FP, FP, and FN.

The performance of state-of-the-art ML and DL methods was evaluated largely based on accuracy, precision, recall, FPR and F1 score. However, a recent systematic study has proven that class imbalance adversely affects the value and meaning of these classification performance metrics, except GM and MCC [30]. Accuracy, precision, recall, FPR and F1 score were found to be biased in favour of the majority class. The results of the study revealed that GM and MCC are the best null-biased classification performance metrics. GM focuses on classification success without considering classification error while MCC accounts

Table 11: Comparison of recall (in %) for each class in 11-class classification scenario

Model	DD-H	DD-T	DD-U	D-H	D-T	D-U	Norm	OSF	SS	DE	KL
RNN [13]	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	-	92.22	87.91	99.75	77.91
SVM [13]	62.24	89.56	98.14	70.14	71.26	**100.00**	-	70.14	72.82	89.67	65.12
RF [13]	82.26	88.28	55.26	82.20	81.77	82.99	-	82.20	69.82	86.55	81.56
NB [13]	50.78	78.67	78.50	68.68	65.56	**100.00**	-	68.68	65.21	66.55	65.62
DNN [14]	96.62	96.22	96.12	96.70	96.63	96.53	-	96.14	96.43	**100.00**	96.76
RNN [14]	96.56	96.65	96.67	96.87	96.77	96.76	-	96.76	96.87	**100.00**	97.00
CNN [14]	97.01	97.00	97.01	97.51	97.11	97.11	-	97.00	97.10	**100.00**	98.10
RBM [14]	96.54	96.51	96.52	96.80	96.57	96.56	-	96.30	96.30	**100.00**	97.11
DBN [14]	96.72	96.60	96.62	96.91	96.72	96.83	-	96.61	96.60	**100.00**	97.66
DBM [14]	96.21	96.08	96.11	96.99	96.33	96.65	-	96.08	96.07	**100.00**	98.22
DAE [14]	97.99	97.71	97.99	98.41	98.00	98.03	-	97.72	97.71	**100.00**	98.33
RDTIDS [19]	93.17	95.84	98.66	77.47	**100.00**	**100.00**	-	98.16	99.47	**100.00**	**100.00**
BLSTM [20]	99.25	99.10	99.45	99.75	99.65	99.79	-	92.77	92.20	96.50	89.90
RNN	74.52	97.63	99.99	76.19	98.12	**100.00**	87.06	99.29	99.95	0.00	71.43
SRNN	96.82	99.98	99.99	95.24	99.99	**100.00**	**95.29**	**100.00**	**99.97**	**100.00**	**100.00**

Table 12: Testing time

Scenario	Model	Testing time (s)
Binary	RNN	**0.42**
	SRNN	1.29
5-class	RNN	**0.43**
	SRNN	1.26
11-class	RNN	**0.42**
	SRNN	1.22

for both classification success and classification error. In this paper, we evaluate and compare the learning efficiency of RNN and SRNN architectures based on training cross-entropy loss while the robustness of the deep learning models against over-fitting was assessed based validation cross-entropy loss. In addition, the generalisation performance of RNN and SRNN models was evaluated based on accuracy, precision, recall, F1 score, FPR, NPV, AUC, GM and MCC.

Table 8 shows that SRNN model achieved better generalisation performance than RNN model in binary classification scenario. For the normal class whose IR was 7900, SRNN model increased recall, F1 score, AUC and MCC by 23.53%, 11.60%, 11.76% and 10.83% respectively. Although SRNN model reduced precision and GM, the reduction in each of the two metrics was below 3.5%. The confusion matrices of RNN and SRNN models in binary classification scenario are shown in Figs. 9 and 10 respectively. We observed that, despite the fact that the normal class has a very high IR, SRNN model correctly classified more network samples in this minority class compared to RNN model. Also, Table 8 shows that SRNN model outperformed state-of-the-art ML/DL models in binary classification scenario.

Table 9 shows that SRNN model achieved better classification performance than RNN model in 5-class classification scenario. For the normal class whose IR was 7900, SRNN model increased recall, F1 score, AUC and MCC by 31.77%, 19.58%, 15.88% and 17.45% respectively. SRNN model reduced precision and GM but the reduction in each of the two metrics was less than 1.3%. For the theft class whose IR was 45049, SRNN model increased precision, recall, F1 score, NPV, GM and MCC by 100% each and it increased AUC by 50%. The confusion matrices of RNN and SRNN models in 5-class classification scenario are shown in Figs. 11 and 12 respectively. We observed that, despite the fact that the normal and theft classes have very high IR, SRNN correctly classified more network samples in these minority classes compared to RNN model. Also, Table 9 shows that SRNN model outperformed state-of-the-art ML/DL models in 5-class classification scenario.

Table 10 shows that SRNN model achieved better classification performance than RNN in 11-class classification scenario. RNN and SRNN models could not correctly classify the sample in DE class when the number of epochs was 5, 10, 15, and 20. However, the SRNN model correctly classified the DE class sample when the number of epochs was increased to 25. For DD-H class whose IR was 3710, SRNN model increased precision, recall, F1, AUC, GM, and MCC by 15.46%, 22.29%, 18.97%, 11.15%, 8.26% and 18.93% respectively. For D-H class whose IR was 2454, SRNN model increased precision, recall, F1, AUC, GM, and MCC by 19.41%, 19.05%, 19.23%, 9.53%, 10.29%, and 19.24% respectively. For Norm class whose IR was 7900, SRNN model increased precision, recall, F1, AUC, GM, and MCC by 6.18%, 8.24%, 7.24%, 4.12%, 3.20%, and 7.22% respectively. For DE class whose IR was 641967, SRNN model increased precision, recall, F1, AUC, GM, and MCC by 100%, 100%, 100%, 50%, 100%, and 100% respectively. For KL class whose IR was 48449, SRNN model increased precision, recall, F1, AUC, GM

and MCC by 37.50%, 28.57%, 33.33%, 14.29%, 20.94% and 33.19% respectively. The confusion matrices of RNN and SRNN models in 11-class classification scenario are shown in Figs. 13 and 14 respectively. We observed that, even though DD-H, D-H, Norm, DE and KL classes have extremely high IR, the SRNN model correctly classified more network samples in these minority classes than the RNN model. Also, Table 11 shows that the SRNN model outperformed state-of-the-art ML/DL models in 11-class classification scenarios, especially in minority classes with extremely high IR such as Norm, DE, and KL. Table 12 shows that SRNN architecture prolonged model response time in binary, 5-class and 11-class classification scenarios but the difference is insignificant, considering the volume of network traffic samples in the testing set.

6. Conclusion

In this study, we proposed SRNN for botnet detection in highly imbalanced network traffic data generated within SHN. Multiple layers of RNN were stacked to learn the hierarchical representations of the highly imbalanced network traffic data using different levels of abstraction. The performance of SRNN in 2-class, 5-class and 11-class classification scenarios was evaluated with Bot-IoT dataset.

Experiment results showed that SRNN outperformed RNN in all classification scenarios. Specifically, SRNN model learned the discriminating features of highly imbalanced network traffic samples in the training set with better representations than RNN model. Also, SRNN model is more robust and it demonstrated better capability to effectively handle over-fitting problem than RNN model. Furthermore, SRNN model achieved better generalisation ability in detecting network traffic samples of the minority classes whose IR were very high (up to 45049 and 641967 in 5-class and 11-class classification scenarios, respectively). In this work, better classification performance was achieved at the cost of longer training time and response time. However, the SRNN model training was faster than that of the state-of-the-art ML/DL models in all classification scenarios. The additional time costs were also insignificant, considering a large number of network traffic samples in the training and testing sets. Therefore, SRNN model can be deployed to secure real-life SHN against complex botnet attacks effectively.

Declaration of Competing Interest

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

CRediT author statement

Segun I. Popoola: conceptualisation, methodology, software, validation, formal analysis, investigation, data curation, writing - original draft preparation, writing - review & editing, and visualisation; **Bamidele Adebisi**: conceptualisation, methodology, validation, formal analysis, investigation, writing - review & editing, visualisation, supervision, project administration, and funding acquisition; **Mohammad Hammoudeh**: conceptualisation, methodology, validation, formal analysis, investigation, writing - review & editing, visualisation, supervision, and project administration; **Haris Gacanin**: validation, formal analysis, investigation, writing - review & editing, and visualisation; **Guan Gui**: validation, formal analysis, investigation, writing - review & editing, and visualisation.

Acknowledgment

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).

References

[1] G. Vormayr, T. Zseby, J. Fabini, Botnet communication patterns, IEEE Communications Surveys & Tutorials 19 (2017) 2768–2796.

[2] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, Ddos in the iot: Mirai and other botnets, Computer 50 (2017) 80–84.

[3] S. Soltan, P. Mittal, H. V. Poor, Blackiot: Iot botnet of high wattage devices can disrupt the power grid, in: 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 15–32.

[4] S. Soltan, P. Mittal, V. Poor, Protecting the grid against mad attacks, IEEE Transactions on Network Science and Engineering (2019).

[5] N. Koroniotis, N. Moustafa, E. Sitnikova, Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions, IEEE Access 7 (2019) 61764–61785.

[6] N. Koroniotis, N. Moustafa, E. Sitnikova, 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 100 (2019) 779–796.

[7] E. Bertino, N. Islam, Botnets and internet of things security, Computer 50 (2017) 76–79.

[8] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, S. A. Khayam, A taxonomy of botnet behavior, detection, and defense, IEEE communications surveys & tutorials 16 (2013) 898–924.

[9] D. Yan, Y. Yang, B. Li, An improved fuzzy classifier for imbalanced data, Journal of Intelligent & Fuzzy Systems 32 (2017) 2315–2325.

[10] C. Gui, Analysis of imbalanced data set problem: The case of churn prediction for telecommunication., Artif. Intell. Research 6 (2017) 93.

[11] A. Fernández, S. García, M. J. del Jesus, F. Herrera, A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets, Fuzzy Sets and Systems 159 (2008) 2378–2398.

[12] E. Lin, Q. Chen, X. Qi, Deep reinforcement learning for imbalanced classification, Applied Intelligence (2020) 1–15.

[13] M. A. Ferrag, L. Maglaras, Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids, IEEE Transactions on Engineering Management (2019).
[14] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, Journal of Information Security and Applications 50 (2020) 102419.
[15] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, H. Gacanin, Hybrid deep learning for botnet attack detection in the internet of things networks, IEEE Internet of Things Journal (2020).
[16] N. Koroniotis, N. Moustafa, E. Sitnikova, A new network forensic framework based on deep learning for internet of things networks: A particle deep framework, Future Generation Computer Systems (2020).
[17] B. A. NG, S. Selvakumar, Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment, Future Generation Computer Systems (2020).
[18] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, A. Alazab, A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks, Electronics 8 (2019) 1210.
[19] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, H. Janicke, Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks, Future Internet 12 (2020) 44.
[20] O. Alkadi, N. Moustafa, B. Turnbull, K.-K. R. Choo, A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks, IEEE Internet of Things Journal (2020).
[21] O. Ibitoye, O. Shafiq, A. Matrawy, Analyzing adversarial attacks against deep learning for intrusion detection in iot networks, arXiv preprint arXiv:1905.05137 (2019).
[22] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, Deep learning-based intrusion detection for iot networks, in: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), IEEE, 2019, pp. 256–25609.
[23] O. AlKadi, N. Moustafa, B. Turnbull, K.-K. R. Choo, Mixture localization-based outliers models for securing data migration in cloud centers, IEEE Access 7 (2019) 114607–114618.
[24] M. Asadi, M. A. J. Jamali, S. Parsa, V. Majidnezhad, Detecting botnet by using particle swarm optimization algorithm based on voting system, Future Generation Computer Systems 107 (2020) 95–111.
[25] M. Shafiq, Z. Tian, Y. Sun, X. Du, M. Guizani, Selection of effective machine learning algorithm and bot-iot attacks traffic identification for internet of things in smart city, Future Generation Computer Systems 107 (2020) 433–442.
[26] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, K. Sakurai, Towards a lightweight detection system for cyber attacks in the iot environment using corresponding features, Electronics 9 (2020) 144.
[27] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani, A. Al-Barakati, Deepdca: Novel network-based detection of iot attacks using artificial immune system, Applied Sciences 10 (2020) 1909.
[28] A. Das, S. A. Ajila, C.-H. Lung, A comprehensive analysis of accuracies of machine learning algorithms for network intrusion detection, in: International Conference on Machine Learning for Networking, Springer, 2019, pp. 40–57.
[29] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, M. Guizani, Iot malicious traffic identification using wrapper-based feature selection mechanisms, Computers & Security (2020) 101863.
[30] A. Luque, A. Carrasco, A. Martín, A. de las Heras, The impact of class imbalance in classification performance metrics based on the binary confusion matrix, Pattern Recognition 91 (2019) 216–231.
[31] P. J. Werbos, et al., Backpropagation through time: what it does and how to do it, Proceedings of the IEEE 78 (1990) 1550–1560.
[32] M. C. Mozer, A focused backpropagation algorithm for temporal, Backpropagation: Theory, architectures, and applications 137 (1995).
[33] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[34] S. K. Kumar, On weight initialization in deep neural networks, arXiv preprint arXiv:1704.08863 (2017).
[35] L. Friedman, O. V. Komogortsev, Assessment of the effectiveness of seven biometric feature normalization techniques, IEEE Transactions on Information Forensics and Security 14 (2019) 2528–2536.

Segun I. Popoola is currently pursuing a Ph.D. degree in the School of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, U.K. He is a Registered Engineer with the Council for the Regulation of Engineering in Nigeria (COREN). His research interests include wireless communications, machine/deep learning, cybersecurity, and Internet of Things.

Bamidele Adebisi is currently a Professor in electrical and electronic engineering. He is particularly interested in the research and development of communication technologies for electrical energy management, critical infrastructures protection, and cyber physical systems. He has several publications and a patent in 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 Professor of computer networks and security in the School of Computing, Math and Digital Technology, Manchester Metropolitan University, UK. He is a certified cybersecurity professional. His research interests include highly decentralized algorithms, communication, and cross-layered solutions to Internet of Things, and wireless sensor networks.

Haris Gacanin is 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. He is a Senior Member of the Institute of Electronics, Information, and Communication Engineering (IEICE).

Guan Gui is currently a Professor in 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. He has authored more than 200 IEEE Journal/Conference papers and won several best paper awards. He is on the editorial boards of several journals.