Deep Bidirectional Gated Recurrent Unit for Botnet Detection in Smart Homes

Segun I. Popoola, Ruth Ande, Kassim B. Fatai, and Bamidele Adebisi

Abstract Bidirectional gated recurrent unit (BGRU) can learn hierarchical feature representations from both past and future information to perform multi-class classification. However, its classification performance largely depends on the choice of model hyperparameters. In this paper, we propose a methodology to select optimal BGRU hyperparameters for efficient botnet detection in smart homes. A deep BGRU multi-class classifier is developed based on the selected optimal hyperparameters, namely, rectified linear unit (ReLU) activation function, 20 epochs, 4 hidden layers, 200 hidden units, and Adam optimizer. The classifier is trained and validated with a batch size of 512 to achieve the right balance between performance and training time. Deep BGRU outperforms the state-of-the-art methods with true positive rate (TPR), false positive rate (FPR), and Matthews coefficient correlation (MCC) of $99.28 \pm 1.57\%$, $0.00 \pm 0.00\%$, and $99.82 \pm 0.40\%$. The results show that the proposed methodology will help to develop an efficient network intrusion detection system for IoT-enabled smart home networks with high botnet attack detection accuracy as well as a low false alarm rate.

Keywords Botnet detection · Internet of Things · Smart homes · Deep learning · Gated recurrent unit

S. I. Popoola (✉) · R. Ande · B. Adebisi
Department of Engineering, Manchester Metropolitan University, Manchester, UK
e-mail: segun.i.popoola@stu.mmu.ac.uk

R. Ande
e-mail: ruth.e.ande@stu.mmu.ac.uk

B. Adebisi
e-mail: b.adebisi@mmu.ac.uk

S. I. Popoola · R. Ande · K. B. Fatai
Artificial Intelligence for Cybersecurity Research, Cyraatek, Manchester, UK
e-mail: fbk6619@cyraatek.com

H. Chiroma et al. (eds.), *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*, https://doi.org/10.1007/978-3-030-66288-2_2

1 Introduction

Embedded devices are interconnected to each other and further connected to the Internet to form an Internet of Things (IoT) [1, 2]. Smart home paradigm exploits the enormous capabilities of IoT technologies to develop intelligent appliances and applications such as smart televisions, smart fridges, smart lighting, and smart security alarm systems [3, 4]. Primarily, IoT-enabled devices in homes autonomously communicate with residents and other IoT-enabled devices over the Internet. Unfortunately, invalidated assumptions and incompatibility in the integration of multiple IoT technologies, standards, proprietary communication protocols, and heterogeneous platforms have exposed smart homes to critical security vulnerabilities [5]. Most of the IoT devices and applications that are in use today were developed with little or no consideration for cybersecurity [6]. Hence, IoT devices tend to be easier to compromise than traditional computers [7].

Cyber attackers exploit the lack of basic security protocols in IoT devices to gain unauthorized remote access and control over insecure network nodes [8, 9]. Compromised IoT devices (i.e., bots) in smart homes can be connected to a master bot in a remote location. This kind of connection helps hackers to form a coordinated network of bots (botnets) [10, 11]. Botnets launch large-scale distributed denial of service (DDoS) attacks with massive traffic volume [10, 12]. Other botnet scenarios include port scanning, operating system (OS) fingerprinting, information theft, and keylogging [13]. Existing security solutions that are primarily designed for traditional computer networks may not be efficient for IoT botnet detection in smart home scenarios due to the unique characteristics of IoT devices and their systems [13].

Traffic patterns of different botnet attack scenarios can be detected in IoT network traffic data using machine learning (ML) approach. Various shallow learning techniques have been proposed to detect botnet attacks in IoT networks. These include support vector machine (SVM) [13–20], decision trees (DT) [14, 15, 19, 21–25], random forest (RF) [8, 15, 18, 21, 23, 26], bagging [15], *k*-nearest neighbor (*k*-NN) [15, 19, 21, 23, 24, 26], artificial neural network (ANN) [22, 25, 27], Naïve Bayes (NB) [22, 23, 25], isolation forest [16], feedforward neural network (FFNN) [18], *k*-means clustering [28, 29], and association rule mining (ARM) [25]. However, data generation in IoT networks is expected to be *big* in terms of volume, variety, and velocity [30]. Therefore, machine learning techniques with shallow network architecture may not be suitable for botnet detection in big IoT data applications.

Deep learning (DL) offers two main advantages over shallow machine learning, namely, hierarchical feature representation and automatic feature engineering, and improvement in classification performance owing to deeper network architecture. DL techniques have demonstrated good capability for botnet detection in big IoT data applications. State-of-the-art DL techniques for IoT botnet detection include deep neural network (DNN) [14], convolutional neural network (CNN) [8, 31–34], recurrent neural network (RNN) [13, 29], long short-term memory (LSTM) [13, 29, 35, 36], and bidirectional LSTM (BLSTM) [36, 37]. The classification performance of DL methods depends on the choice of optimal model hyperparameters. However, model hyperparameters are often selected by trial and error methods in previous studies.

Gated recurrent unit (GRU) is a variant of RNN, and it is suitable for large-scale sequential data processing [38]. Bidirectional GRU (BGRU) has a unique advantage of accessing both past and present information to make an accurate decision for efficient classification performance with lower computational demands [39]. To the best of our knowledge, previous researches have not investigated the capability of BGRU for botnet detection in a smart home scenario. In this paper, we aim to find the optimal hyperparameters for efficient deep BGRU-based botnet detection in IoT-enabled smart homes. The main contributions of this paper are summarized as follows:

a. A methodology is proposed to determine the most suitable hyperparameters (activation function, epoch, hidden layer, hidden unit, batch size, and optimizer) for optimal BGRU-based multi-class classification.
b. A deep BGRU model is designed based on the selected model hyperparameters to distinguish normal network traffic from IoT botnet attack scenarios.
c. The proposed methodology is implemented, and the deep BGRU model is developed with the bot-IoT dataset.
d. We evaluate the performance of the deep BGRU model based on training loss, validation loss, true positive rate (TPR), false positive rate (FPR), Matthews correlation coefficient (MCC), and training time.

The remaining part of this paper is organized as follows: Sect. 2 describes the proposed methodology for selection of optimal BGRU hyperparameters; the method is employed to develop a deep BGRU model for efficient IoT botnet detection; the results of extensive model simulations are presented in Sect. 3, and Sect. 4 concludes the paper.

2 Deep BGRU Method for Botnet Detection in IoT Networks

In this section, we describe the concept of BGRU, the proposed methodology for optimal BGRU hyperparameters, and the development of efficient deep BGRU classifiers for botnet detection in the context of a smart home. The overview of the framework is shown in Fig. 1.

2.1 Bidirectional Gated Recurrent Unit

GRU is a variant of RNN. This hidden unit achieves performance similar to LSTM with a simplified gated mechanism and lower computation requirements [40]. Unlike

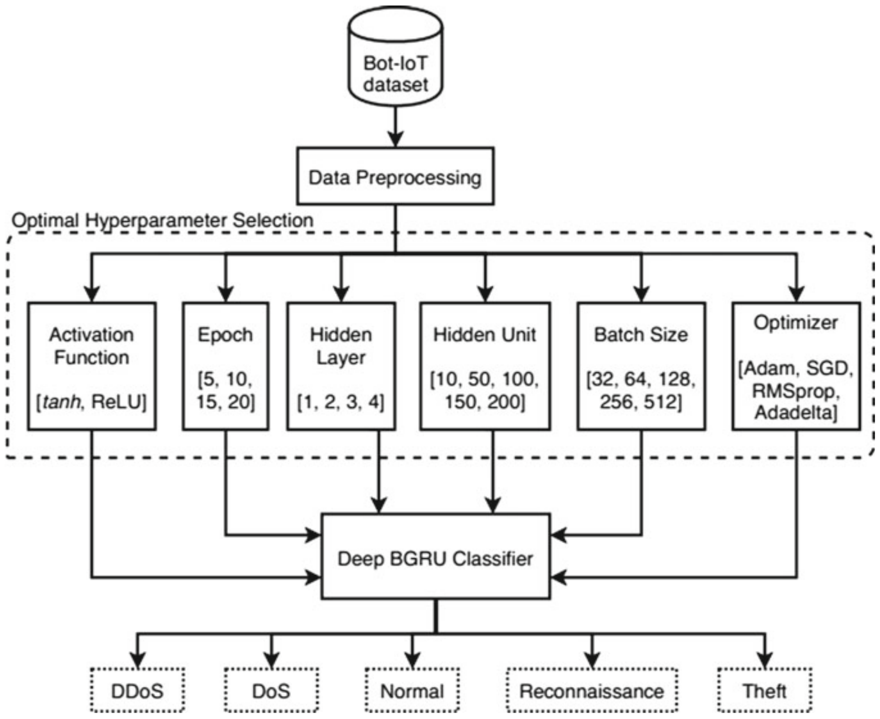
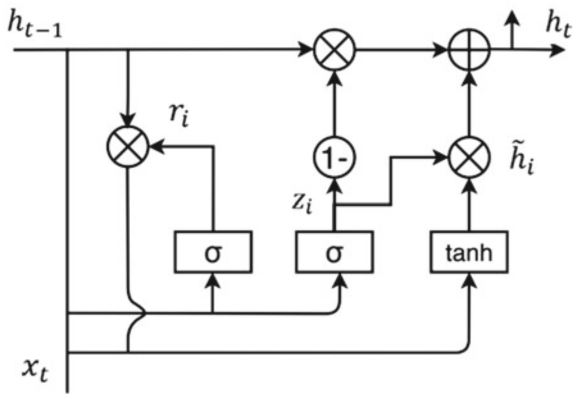

Bot-IoT dataset
Data Preprocessing
Optimal Hyperparameter Selection
Activation Function [*tanh*, ReLU]
Epoch [5, 10, 15, 20]
Hidden Layer [1, 2, 3, 4]
Hidden Unit [10, 50, 100, 150, 200]
Batch Size [32, 64, 128, 256, 512]
Optimizer [Adam, SGD, RMSprop, Adadelta]
Deep BGRU Classifier
DDoS
DoS
Normal
Reconnaissance
Theft

Fig. 1 Optimal hyperparameters of BGRU

LSTM, GRU discards the memory unit and replaces the input and forget gates with an update gate. Figure 2 shows the standard structure of a GRU. A GRU has two gates, namely, the reset gate (r_i) and the update gate (z_i). These gates depend on past hidden state ($\boldsymbol{h}_{t-1}$) and the present input (x_t). The reset gate determines whether the past hidden state should be ignored or not, while the update gate changes the

Fig. 2 The architecture of GRU



past hidden state to a new hidden state ($\tilde{h}_i$). The past hidden state is ignored when $r_i \simeq 0$ such that information that is not relevant to the future is dropped and a more compact representation is obtained. The update gate regulates the amount of data that is transmitted from the past hidden state to the present hidden state.

GRU is a unidirectional RNN, i.e., it employs a single hidden layer, and its recurrent connections are in the backward time direction only. GRU cannot update the present hidden state based on the information in the future hidden state. Interestingly, BGRU updates its current hidden state based on both the past and the future hidden state information [41]. A single BGRU has two hidden layers, which are both connected to the input and output. The first hidden layer establishes recurrent connections between the past hidden states and the present hidden state in the backward time direction. On the other hand, the second hidden layer establishes recurrent connections between the present hidden state and the future hidden states in the forward time direction. The computation of BGRU parameters is obtained by (1)–(9):

$$\overleftarrow{r_i} = \sigma\left(\left[\overleftarrow{\boldsymbol{W}}_r \boldsymbol{x}\right]_i + \left[\overleftarrow{\boldsymbol{U}}_r \overleftarrow{\boldsymbol{h}}_{(t-1)}\right]_i + \overleftarrow{\boldsymbol{b}}_r\right), \tag{1}$$

$$\overleftarrow{z}_i = \sigma\left(\left[\overleftarrow{\boldsymbol{W}}_z \boldsymbol{x}\right]_i + \left[\overleftarrow{\boldsymbol{U}}_z \overleftarrow{\boldsymbol{h}}_{(t-1)}\right]_i + \overleftarrow{\boldsymbol{b}}_z\right), \tag{2}$$

$$\overleftarrow{\tilde{h}_i^{(t)}} = \phi\left(\left[\overleftarrow{\boldsymbol{W}} \boldsymbol{x}\right]_i + \left[\overleftarrow{\boldsymbol{U}}\left(\overleftarrow{\boldsymbol{r}} \odot \overleftarrow{\boldsymbol{h}}_{(t-1)}\right)\right]_i\right), \tag{3}$$

$$\overleftarrow{h_i^{(t)}} = \overleftarrow{z}\overleftarrow{h}_i^{(t-1)} + \left(1 - \overleftarrow{z}_i\right)\overleftarrow{\tilde{h}_i^{(t)}}, \tag{4}$$

$$\overrightarrow{r_i} = \sigma\left(\left[\vec{\boldsymbol{W}}_r \boldsymbol{x}\right]_i + \left[\vec{\boldsymbol{U}}_r \vec{\boldsymbol{h}}_{(t+1)}\right]_i + \vec{\boldsymbol{b}}_r\right), \tag{5}$$

$$\overrightarrow{z_i} = \sigma\left(\left[\vec{\boldsymbol{W}}_z \boldsymbol{x}\right]_i + \left[\vec{\boldsymbol{U}}_z \vec{\boldsymbol{h}}_{(t+1)}\right]_i + \vec{\boldsymbol{b}}_z\right), \tag{6}$$

$$\overrightarrow{\tilde{h}_i^{(t)}} = \phi\left(\left[\vec{\boldsymbol{W}} \boldsymbol{x}\right]_i + \left[\vec{\boldsymbol{U}}\left(\vec{\boldsymbol{r}} \odot \vec{\boldsymbol{h}}_{(t+1)}\right)\right]_i\right), \tag{7}$$

$$\overrightarrow{h_i^{(t)}} = \overrightarrow{z_i}\vec{h}_i^{(t-1)} + \left(1 - \overrightarrow{z_i}\right)\overrightarrow{\tilde{h}_i^{(t)}}, \tag{8}$$

$$\tilde{y}_i = \vartheta\left(\boldsymbol{W}_y \overleftarrow{\boldsymbol{h}_t^{(t)}} + \boldsymbol{U}_y \overrightarrow{\boldsymbol{h}_t^{(t)}} + \boldsymbol{b}_y\right), \tag{9}$$

where $x, r, z, h, \tilde{y}$ and i are the input, reset gate, update gate, hidden state, output, and hidden unit index, respectively; $\overleftarrow{(\cdot)}$ and $\overrightarrow{(\cdot)}$ represent the parameters of the hidden layers in the backward and forward time directions, respectively; $\boldsymbol{W}_{(\cdot)}$ and $\boldsymbol{U}_{(\cdot)}$ are the weight matrices while $\boldsymbol{b}_{(\cdot)}$ is the bias vector; $\sigma(\cdot)$ is a logistic sigmoid activation function; $\phi(\cdot)$ is either hyperbolic tangent (*tanh*) or rectified linear unit (ReLU) activation function; and $\vartheta(\cdot)$ is a softmax activation function.

2.2 The Proposed Method for Selection of Optimal BGRU Hyperparameters

The proposed method for optimal selection of BGRU hyperparameters is presented in Algorithm 1. Network traffic features are considered as sequential data given by (10):

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,\mu} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\delta,1} & x_{\delta,2} & \cdots & x_{\delta,\mu} \end{bmatrix} \tag{10}$$

where μ is the number of network traffic features, and δ is the number of network traffic samples. The network traffic features in the training, validation, and testing sets are represented by $\boldsymbol{X}_{tr}$, $\boldsymbol{X}_{va}$, and $\boldsymbol{X}_{te}$, respectively. The ground truth labels for training, validation, and testing are represented by $\boldsymbol{y}_{tr}$, $\boldsymbol{y}_{va}$, and $\boldsymbol{y}_{te}$, respectively.

The selection of optimal BGRU hyperparameters will lead to efficient detection and classification of IoT botnet attacks. These hyperparameters include activation function (a_f), epoch (e_p), hidden layer (h_l), hidden unit (h_u), batch size (b_s), and optimizer (o_p). The optimal choice is made from a set of commonly used hyperparameters through extensive simulations. The collection of the hyperparameters is given by (11)–(16):

$$\boldsymbol{a_f} = \left[a_{f,1}, a_{f,2}, \ldots, a_{f,n}\right], \tag{11}$$

$$\boldsymbol{e_p} = \left[e_{p,1}, e_{p,2}, \ldots, e_{p,m}\right], \tag{12}$$

$$\boldsymbol{h_l} = \left[h_{l,1}, h_{l,2}, \ldots, h_{l,k}\right], \tag{13}$$

$$\boldsymbol{h_u} = \left[h_{u,1}, h_{u,2}, \ldots, h_{u,q}\right], \tag{14}$$

$$\boldsymbol{b_s} = \left[b_{s,1}, b_{s,2}, \ldots, b_{s,v}\right], \tag{15}$$

$$\boldsymbol{o_p} = \left[o_{p,1}, o_{p,1}, \ldots, o_{p,g}\right], \tag{16}$$

where n is the number of activation functions in $\boldsymbol{a_f}$; m is the number of epochs in $\boldsymbol{e_p}$; k is the number of hidden layers in $\boldsymbol{h_l}$; q is the number of hidden units in $\boldsymbol{h_u}$; v is the number of batch sizes in $\boldsymbol{b_s}$; and g is the number of optimizers in $\boldsymbol{o_p}$. The default hyperparameters are the first elements in each of the sets.

Algorithm 1. Optimal BGRU hyperparameter selection method

```
Input: $\boldsymbol{X}_{tr}, \boldsymbol{y}_{tr}, \boldsymbol{X}_{va}, \boldsymbol{y}_{va}, \boldsymbol{X}_{te}, \boldsymbol{y}_{te}$
Initialization: $c, \alpha, d, j, \beta, \gamma = 1$
Output: $\tilde{a}_f, \tilde{e}_p, \tilde{h}_l, \tilde{h}_u, \tilde{b}_s, \tilde{o}_p$
1.  def $model()$
2.      $N = BGRU(a_{f,c}, h_{l,d}, h_{u,j})$
3.      $l_{tr}, l_{va}, \tilde{\boldsymbol{y}} = BPTT(N, \boldsymbol{X}_{tr}, \boldsymbol{y}_{tr}, \boldsymbol{X}_{va}, \boldsymbol{y}_{va}, e_{p,\alpha}, b_{s,\beta}, o_{p,\gamma})$
4.      $TPR, FPR, MCC = evaluate\_classifier(\tilde{\boldsymbol{y}}, \boldsymbol{y}_{te})$
5.      $parameter = selector(l_{tr}, l_{va}, TPR, FPR, MCC)$
6.      return $parameter$
7.  for $c = 1$ to $n$
8.      $\tilde{a}_f = model()$
9.  for $\alpha = 1$ to $m$
10.     $\tilde{e}_p = model()$
11. for $d = 1$ to $k$
12.     $\tilde{h}_l = model()$
13. for $j = 1$ to $q$
14.     $\tilde{h}_u = model()$
15. for $\beta = 1$ to $v$
16.     $\tilde{b}_s = model()$
17. for $\gamma = 1$ to $g$
18.     $\tilde{o}_p = model()$
```

The development of the BGRU model for efficient IoT botnet detection in smart homes involves two main processes, namely, the choice of suitable deep network architecture and loss minimization through model training and validation. Deep network architecture (N) for BGRU is determined by $a_{f,c}$, $h_{l,d}$, and $h_{u,}j$. The selected BGRU architecture is trained with $\boldsymbol{X}_{tr}$, $\boldsymbol{y}_{tr}$, $\boldsymbol{X}_{va}$, $\boldsymbol{y}_{va}$, $e_{p,\alpha}$, $b_{s,\beta}$, and $o_{p,\gamma}$ using back-propagation through time (BPTT) algorithm [42]. Loss minimization during training and validation is assessed based on the values of training loss (l_{tr}) and validation loss (l_{va}). A categorical cross-entropy loss function was used for loss minimization in a multi-class classification scenario.

The performance of BGRU classifier is based on TPR, FPR, and MCC when evaluated with highly imbalanced testing data ($\boldsymbol{X}_{te}$, $\boldsymbol{y}_{te}$). The definition of these performance metrics is given by (17)–(20) [43]:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{17}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{18}$$

$$\lambda = 2\left[\frac{\mathrm{TP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}}\right] - 1 \tag{19}$$

$$\mathrm{MCC} = \frac{1}{2}\left\{\left[\frac{\mathrm{TPR} + \mathrm{TNR} - 1}{\mathrm{TPR} + (1 - \mathrm{TNR})\left(\frac{1-\lambda}{1+\lambda}\right)}\right] + 1\right\}, \tag{20}$$

where true positive (TP) is the number of attack samples that are correctly classified; false positive (FP) is the number of normal network traffic samples that are misclassified as attacks; true negative (TN) is the number of normal network traffic samples that are correctly classified; false negative (FN) is the number of attack samples that are misclassified as normal network traffic; λ is the class imbalance coefficient. For each of the simulation scenarios, an optimal BGRU hyperparameter is expected to produce the lowest l_{tr}, l_{va}, and FPR as well as the highest TPR and MCC.

2.3 Deep BGRU Classifier for IoT Botnet Detection

Bot-IoT dataset [13] is made up of network traffic samples that were generated from real-life IoT testbed. The testbed is realistic because IoT devices were included. These IoT devices include a weather station, a smart fridge, motion-activated lights, a remote-controlled garage door, and an intelligent thermostat. This network of IoT devices is considered to be a good representation of an IoT-enabled smart home. The Bot-IoT dataset also contains recent and complex IoT botnet attack samples covering five common scenarios, namely, DDoS, DoS, reconnaissance, and information theft. Accurate ground truth labels are given to the IoT botnet attack samples. The samples of DDoS attack, DoS attack, normal traffic, reconnaissance attack, and information theft attack in the Bot-IoT dataset are 1,926,624, 1,650,260, 477, 91,082, and 79, respectively.

Network traffic samples, IoT botnet attack samples, and ground truth labels were pre-processed into appropriate formats suitable for deep learning. First, the complete dataset was randomly divided into a training set (70%), validation set (15%), and testing set (15%) as suggested in the literature [44, 45]. Non-numeric elements in feature matrices ($\boldsymbol{X}_{tr}$, $\boldsymbol{X}_{va}$, $\boldsymbol{X}_{te}$) and label vectors ($\boldsymbol{y}_{tr}$, $\boldsymbol{y}_{va}$, $\boldsymbol{y}_{te}$) were encoded using integer encoding and binary encoding methods, respectively. Furthermore, the elements of feature matrices were transformed using a min–max normalization method such that the value of each element falls between 0 and 1 [46].

The proposed method for the selection of optimal BGRU hyperparameters was implemented. All simulations were performed at a learning rate of 0.0001. The default hyperparameters for the simulations were the ReLU activation function, five epochs, a single hidden layer, 200 hidden units, a batch size of 128, and Adam optimizer. We investigated the suitability of the following: *tanh* and ReLU activation functions; epochs of 5, 10, 15, and 20; hidden layers of 1, 2, 3, and 4; hidden units of 10, 50, 100, 150, and 200; batch sizes of 32, 64, 128, 256, and 512; and finally, the optimizers (Adam, SGD, RMSprop, and Adadelta). Model training, validation, and testing were implemented using Keras library developed for Python programming running on Ubuntu 16.04 LTS workstation with the following specifications: RAM (32 GB), processor (Intel Core i7-9700 K CPU @ 3.60 GHz $\times$ 8), Graphics (GeForce RTX 2080 Ti/PXCIe/SSE2), and 64-bit operating system. The optimal BGRU hyperparameters $\left(\tilde{a}_f, \tilde{e}_p, \tilde{h}_l, \tilde{h}_u, \tilde{b}_s, \tilde{o}_p\right)$ were used to develop a multi-class classifier for efficient IoT botnet detection in smart homes.

3 Results and Discussion

In this section, we evaluate the effectiveness of the method proposed for the selection of optimal BGRU hyperparameters in our attempt to develop an efficient IoT botnet detection system for smart home network security. Specifically, we examine the influence of different activation functions, number of epochs, number of hidden layers, number of hidden units, batch sizes, and optimizers on the performance of BGRU-based multi-class classifier.

3.1 Influence of Activation Functions on Classification Performance

To determine the right activation function for multi-class classification, ReLU and *tanh* activation functions were independently employed in two distinct BGRU neural networks, namely, BGRU-ReLU and BGRU-*tanh*. Apart from the activation function that differs, each of the BGRU neural networks is made up of a single hidden layer with 200 hidden units. BGRU-ReLU and BGRU-*tanh* were separately trained and validated with five epochs, 128 batch size, and Adam optimizer.

Training and validation losses in BGRU-ReLU and BGRU-*tanh* were analyzed to understand the extent of model underfitting and overfitting, respectively. Figure 3 shows that the ReLU activation function is more desirable than *tanh* activation function. Generally, training and validation losses reduced in both BGRU-ReLU and BGRU-*tanh* as the number of epochs increased from 1 to 5. However, training and validation losses were lower in BGRU-ReLU than in BGRU-*tanh* throughout the 5-epoch period. At the end of the experiment, we observed that training loss in

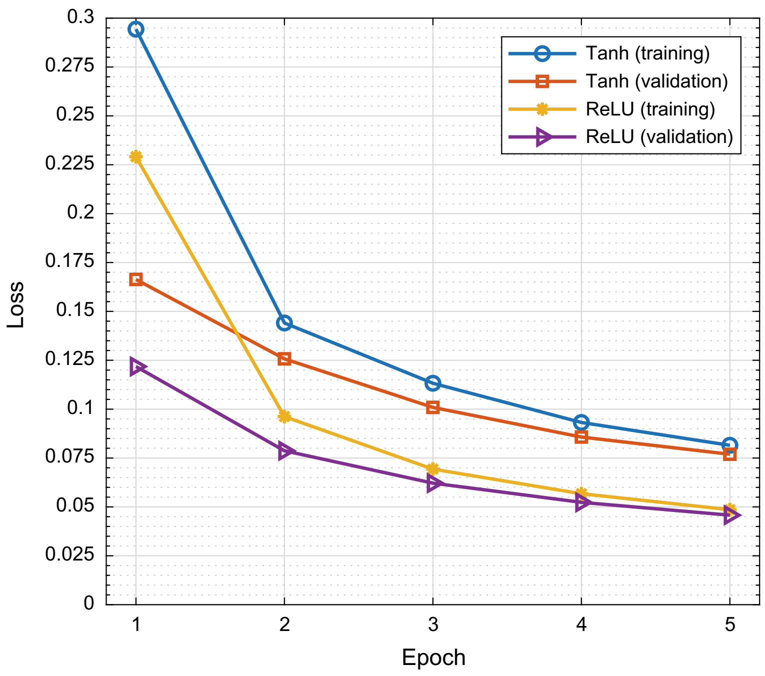


Fig. 3 Training and validation losses of two activation functions

BGRU-ReLU reduced to 0.1000, while validation loss reduced to 0.0721. Also, BGRU-ReLU reduced the average training and validation losses in BGRU-*tanh* by 31.16% and 35.07%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is minimal when the ReLU activation function was used in BGRU. Model underfitting will lead to poor classification accuracy while model overfitting will adversely affect the generalization ability of BGRU classifier when applied to previously unseen network traffic samples. Consequently, the adoption of the ReLU activation function in BGRU will help to achieve high classification accuracy and good generalization ability required for efficient IoT botnet detection in smart homes.

Multi-class classification performance of BGRU-ReLU and BGRU-*tanh* was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. TPR, also known as sensitivity or recall, is the percentage of samples that were correctly classified; FPR, also known as fall-out or false alarm ratio (FAR), is the percentage of samples that were wrongly classified; and MCC is a balanced measure that accounts for the impact of class imbalance on classification performance. Table 1 shows that BGRU-ReLU performed better than BGRU-*tanh*. BGRU-ReLU increased TPR and MCC in BGRU-*tanh* by 24.37% and 25.47%, respectively, while FPR was reduced by 40.86%. The lower the TPR and MCC values, the higher the chances that the BGRU classifier will fail to detect IoT botnet in smart homes. Also, the higher the

Table 1 Performance of BGRU with different activation functions

Metric	AF	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	*tanh*	96.55	98.60	63.53	99.90	0.00	71.72 ± 42.85
	ReLU	98.01	98.86	74.12	99.99	75.00	89.20 ± 13.38
FPR	*tanh*	1.33	3.30	0.00	0.00	0.00	0.93 ± 1.45
	ReLU	1.08	1.90	0.00	0.00	0.00	0.60 ± 0.87
MCC	*tanh*	98.16	97.34	89.85	99.97	0.00	77.06 ± 43.25
	ReLU	98.72	98.38	93.05	99.99	93.30	96.69 ± 3.27

FPR value, the higher the probability that the BGRU classifier will produce a false alarm, i.e., it is more likely that the classifier will wrongly classify incoming network traffic or IoT botnet attack. Therefore, the choice of the ReLU activation function in BGRU will ensure a high detection rate and reduce false alarm in botnet detection system developed for smart homes.

3.2 Influence of the Number of Epochs on Classification Performance

In this subsection, we determine the optimal number of epochs required for efficient BGRU-based IoT botnet detection in smart homes. Four single layers BGRU neural networks were trained and validated with 5, 10, 15, and 20 epochs to produce BGRU-EP5, BGRU-EP10, BGRU-EP15, and BGRU-EP20 classifiers, respectively. Each of these classifiers utilized 200 hidden neurons, the ReLU activation function, a batch size of 128, and Adam optimizer.

Training and validation losses in BGRU-EP5, BGRU-EP10, BGRU-EP15, and BGRU-EP20 were analyzed to understand the extent of model underfitting and overfitting, respectively. Figure 4 shows that the lowest average training and validation losses were realized with 20 epochs. In general, training and validation losses reduced in all the classifiers throughout the epoch period. However, average training and validation losses were lower in BGRU-EP20 than in BGRU-EP5, BGRU-EP10, and BGRU-EP15. At the end of the experiment, we observed that training loss in BGRU-EP20 reduced to 0.0095, while validation loss reduced to 0.0094. BGRU-EP20 reduced the average training losses in BGRU-EP5, BGRU-EP10, BGRU-EP15 by 58.89%, 37.99%, and 17.87%, respectively, while the average validation losses were reduced by 53.80%, 35.22%, and 15.82%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is best minimized when the number of epochs in BGRU was 20. In other words, a sufficiently large number of epochs in BGRU will facilitate high classification accuracy and good generalization ability that are required for efficient IoT botnet detection in smart homes.

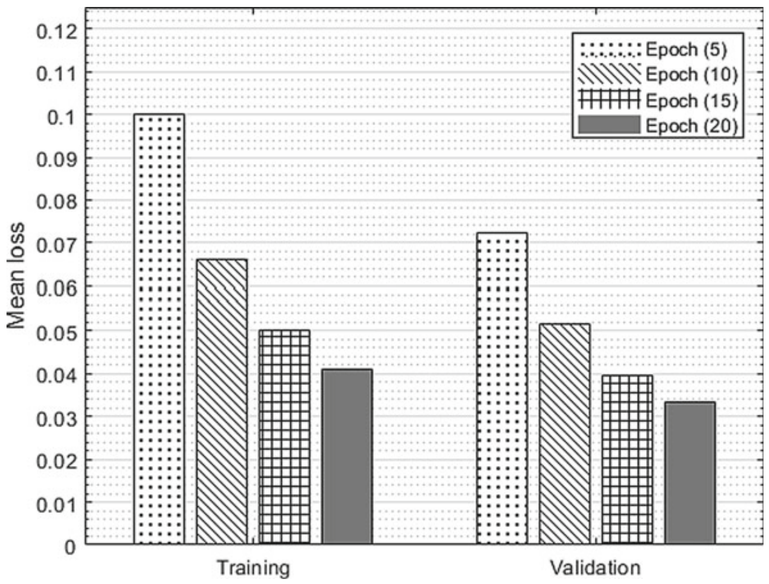


Fig. 4 Mean training and validation loss of different epochs

Multi-class classification performance of BGRU-EP5, BGRU-EP10, BGRU-EP15, and BGRU-EP20 was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. Table 2 shows that BGRU-EP20 performed better than BGRU-EP5, BGRU-EP10, and BGRU-EP15. BGRU-EP20 increased TPR by 7.97%, 3.65%, and 1.05% relative to BGRU-EP5, BGRU-EP10 and BGRU-EP15, respectively; FPR decreased by 87.27%, 73.08%, and 46.15%, respectively; and MCC increased by

Table 2 Performance of BGRU at different number of epochs

Metric	Epoch	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	5	97.97	99.18	76.47	99.98	75.00	89.72 ± 12.80
	10	99.01	99.60	81.18	99.99	87.50	93.46 ± 8.63
	15	99.44	99.90	80.00	99.98	100.00	95.86 ± 8.87
	20	99.72	99.94	84.71	99.98	100.00	96.87 ± 6.80
FPR	5	0.78	1.94	0.00	0.00	0.00	0.55 ± 0.85
	10	0.38	0.95	0.00	0.00	0.00	0.26 ± 0.41
	15	0.10	0.54	0.00	0.00	0.00	0.13 ± 0.23
	20	0.06	0.27	0.00	0.00	0.00	0.07 ± 0.12
MCC	5	98.92	98.43	93.72	99.99	93.30	96.87 ± 3.12
	10	99.48	99.24	95.05	100.00	96.77	98.11 ± 2.11
	15	99.79	99.60	94.72	99.99	100.00	98.82 ± 2.30
	20	99.89	99.80	96.02	99.99	100.00	99.14 ± 1.75

2.34, 1.05, and 0.32%. Therefore, a sufficiently large number of epochs in BGRU will ensure a high detection rate and reduce false alarm in the botnet detection system developed for smart homes.

3.3 Influence of the Number of Hidden Layers on Classification Performance

In this subsection, we determine the optimal number of hidden layers required for efficient BGRU-based IoT botnet detection in smart homes. Four BGRU neural networks with 1, 2, 3, and 4 hidden layers formed BGRU-HL1, BGRU-HL2, BGRU-HL3, and BGRU-HL4 classifiers, respectively, when trained using 200 hidden neurons, ReLU activation function, five epochs, a batch size of 128, and Adam optimizer.

Training and validation losses in BGRU-HL1, BGRU-HL2, BGRU-HL3, and BGRU-HL4 were analyzed to understand the extent of model underfitting and overfitting, respectively. Figures 5 and 6 show that the lowest training and validation losses were realized with four hidden layers. In general, training and validation losses reduced in all the classifiers throughout the five-epoch period. However, training and validation losses were lower in BGRU-HL4 than in BGRU-HL1, BGRU-HL2, and BGRU-HL3. At the end of the experiment, we observed that training loss in

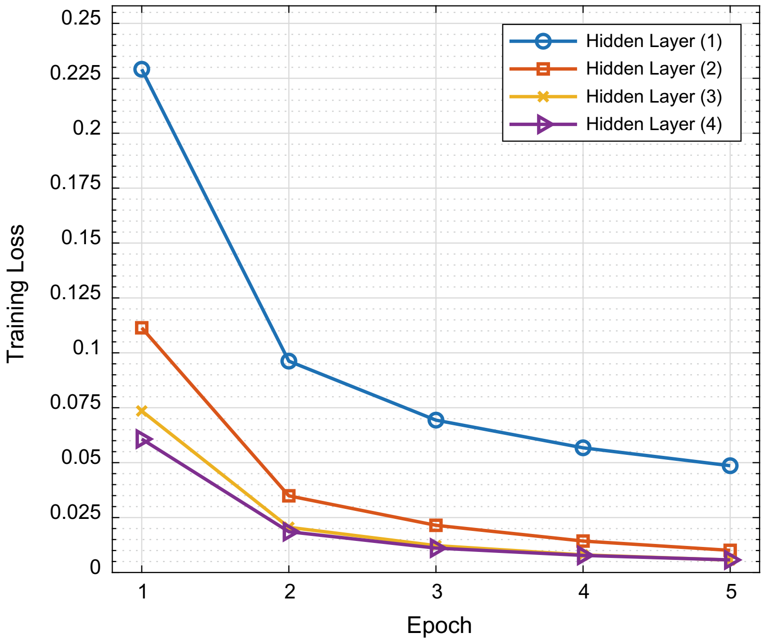


Fig. 5 Training loss of different number of hidden layers

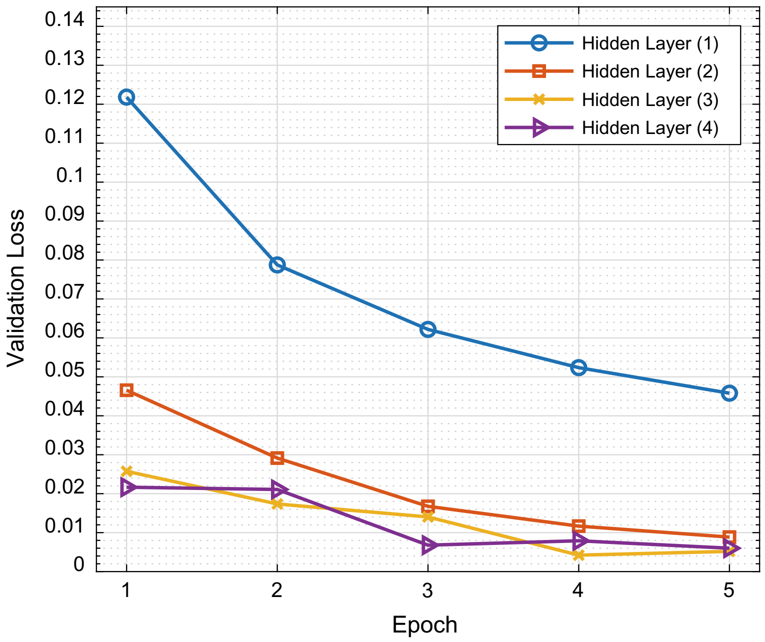


Fig. 6 Validation loss of different number of hidden layers

BGRU-HL4 reduced to 0.0057, while validation loss reduced to 0.0060. BGRU-HL4 reduced the average training losses in BGRU-HL1, BGRU-HL2, and BGRU-HL3 by 79.22%, 45.93%, and 13.43%, respectively, while the average validation losses were reduced by 82.44%, 43.88%, and 4.67%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is best minimized when the number of hidden layers in BGRU was 4. In other words, a sufficiently deep BGRU will facilitate high classification accuracy and good generalization ability that are required for efficient IoT botnet detection in smart homes.

Multi-class classification performance of BGRU-HL1, BGRU-HL2, BGRU-HL3, and BGRU-HL4 was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. Table 3 shows that BGRU-HL4 performed better than BGRU-HL1, BGRU-HL2, and BGRU-HL3. BGRU-HL4 increased TPR by 9.80, 1.71, and 1.69% relative to BGRU-HL1, BGRU-HL2, and BGRU-HL3, respectively; FPR decreased by 87.27%, 12.50%, and 0%, respectively; and MCC increased by 2.79%, 0.45%, and 0.44%. Therefore, a sufficiently deep BGRU will ensure a high detection rate and reduce false alarm in the botnet detection system developed for smart homes.

Table 3 Performance of BGRU at different number of hidden layers

Metric	Layer	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	1	97.97	99.18	76.47	99.98	75.00	89.72 ± 12.80
	2	99.83	99.74	84.71	99.99	100.00	96.85 ± 6.79
	3	99.92	99.72	84.71	99.99	100.00	96.87 ± 6.80
	4	99.67	99.98	92.94	99.99	100.00	98.51 ± 3.12
FPR	1	0.78	1.94	0.00	0.00	0.00	0.55 ± 0.85
	2	0.24	0.16	0.00	0.00	0.00	0.08 ± 0.11
	3	0.27	0.08	0.00	0.00	0.00	0.07 ± 0.12
	4	0.02	0.32	0.00	0.00	0.00	0.07 ± 0.14
MCC	1	98.92	98.43	93.72	99.99	93.30	96.87 ± 3.12
	2	99.78	99.82	96.02	100.00	100.00	99.12 ± 1.74
	3	99.78	99.87	96.02	100.00	100.00	99.13 ± 1.74
	4	99.90	99.77	98.20	100.00	100.00	99.57 ± 0.77

3.4 Influence of Hidden Units on Classification Performance

In this subsection, we determine the optimal number of hidden units required for efficient BGRU-based IoT botnet detection in smart homes. Five single layers BGRU neural networks with 10, 50, 100, 150, and 200 hidden units formed BGRU-HU1, BGRU-HU2, BGRU-HU3, BGRU-HU4, and BGRU-HU5 classifiers respectively when trained using ReLU activation function, five epochs, a batch size of 128 and Adam optimizer.

Training and validation losses in BGRU-HU1, BGRU-HU2, BGRU-HU3, BGRU-HU4, and BGRU-HU5 were analyzed to understand the extent of model underfitting and overfitting, respectively. Figures 7 and 8 show that the lowest training and validation losses were realized with 200 hidden units. In general, training and validation losses reduced in all the classifiers throughout the five-epoch period. However, training and validation losses were lower in BGRU-HU4 than in BGRU-HU1, BGRU-HU2, and BGRU-HU3. At the end of the experiment, we observed that training loss in BGRU-HU4 reduced to 0.0486, while validation loss reduced to 0.0458. BGRU-HU5 reduced the average training losses in BGRU-HU1, BGRU-HU2, BGRU-HU3, and BGRU-HU4 by 53.54%, 31.72%, 21.14%, and 11.82%, respectively, while the average validation losses were reduced by 54.83%, 32.94%, 21.85%, and 12.07%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is best minimized when the number of hidden units in BGRU was 200. In other words, a sufficiently large number of hidden units in BGRU will facilitate high classification accuracy and good generalization ability that are required for efficient IoT botnet detection in smart homes.

Multi-class classification performance of BGRU-HU1, BGRU-HU2, BGRU-HU3, BGRU-HU4, and BGRU-HU5 was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. Table 4 shows that BGRU-HU5 performed

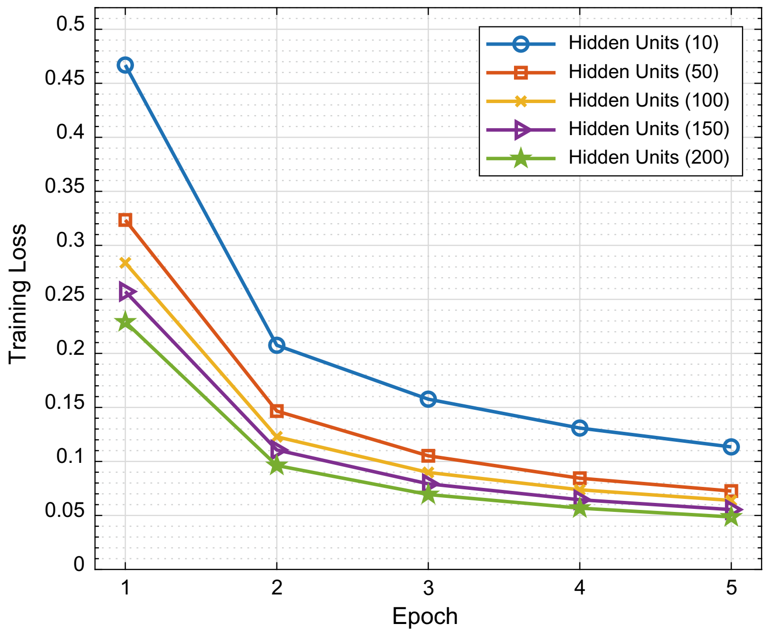


Fig. 7 Training loss of different number of hidden units

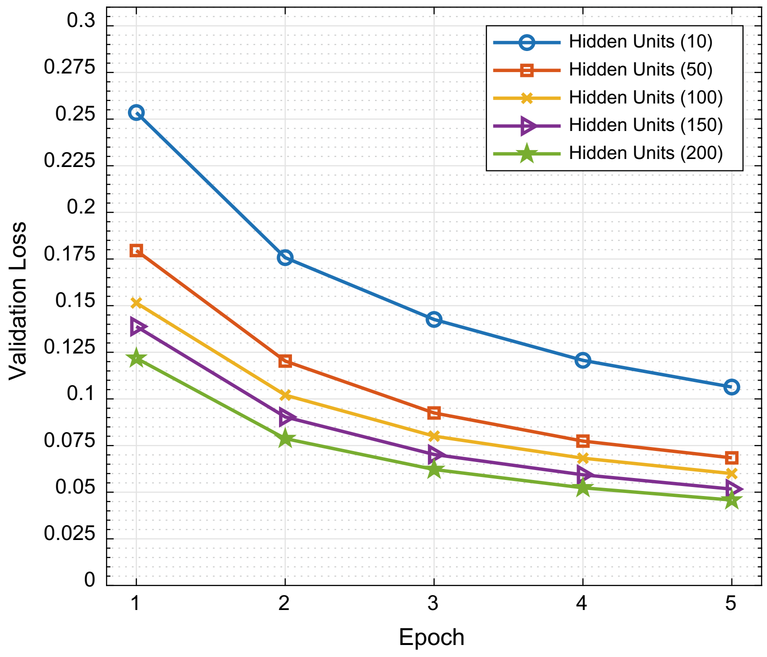


Fig. 8 Validation loss of different number of hidden units

Table 4 Performance of BGRU at different number of hidden units

Metric	Units	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	10	95.41	97.98	0.00	99.78	0.00	58.64 ± 53.55
	50	97.84	98.30	61.18	99.83	0.00	71.43 ± 43.11
	100	97.91	98.57	69.41	99.98	75.00	88.17 ± 14.73
	150	98.01	98.86	74.12	99.99	75.00	89.20 ± 13.38
	200	97.97	99.18	76.47	99.98	75.00	89.72 ± 12.80
FPR	10	1.92	4.40	0.00	0.01	0.00	1.27 ± 1.94
	50	1.62	2.07	0.00	0.00	0.00	0.74 ± 1.02
	100	1.36	2.00	0.00	0.00	0.00	0.67 ± 0.95
	150	1.08	1.90	0.00	0.00	0.00	0.60 ± 0.87
	200	0.78	1.94	0.00	0.00	0.00	0.55 ± 0.85
MCC	10	97.43	96.42	0.00	99.94	0.00	58.76 ± 53.65
	50	98.28	98.12	89.11	99.96	0.00	77.09 ± 43.30
	100	98.49	98.23	91.66	99.99	93.30	96.33 ± 3.63
	150	98.72	98.38	93.05	99.99	93.30	96.69 ± 3.27
	200	98.92	98.43	93.72	99.99	93.30	96.87 ± 3.12

better than BGRU-HU1, BGRU-HU2, BGRU-HU3, and BGRU-HU4. BGRU-HU5 increased TPR by 53, 25.61, 1.75, and 0.58% relative to BGRU-HU1, BGRU-HU2, BGRU-HU3, and BGRU-HU4, respectively; FPR decreased by 56.69%, 25.68%, 17.91%, and 8.33%, respectively; and MCC increased by 64.86, 25.66, 0.56, and 0.19%. Therefore, a sufficiently large number of hidden units in BGRU will ensure a high detection rate and reduce false alarm in the IoT botnet detection system developed for smart homes.

3.5 *Influence of Batch Size on Classification Performance*

In this subsection, we determine the optimal batch size required for efficient BGRU-based IoT botnet detection in smart homes. Five single layers BGRU neural networks with batch sizes of 32, 64, 128, 256, and 512 formed BGRU-B32, BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512 classifiers, respectively, when trained using 200 hidden units, ReLU activation function, five epochs, and Adam optimizer.

Training and validation losses in BGRU-B32, BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512 were analyzed to understand the extent of model underfitting and overfitting, respectively. Figures 9 and 10 show that the lowest training and validation losses were realized with a batch size of 32. In general, training and validation losses reduced in all the classifiers throughout the five-epoch period. However, training and validation losses were lower in BGRU-B32 than in BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512. At the end of the experiment, we observed

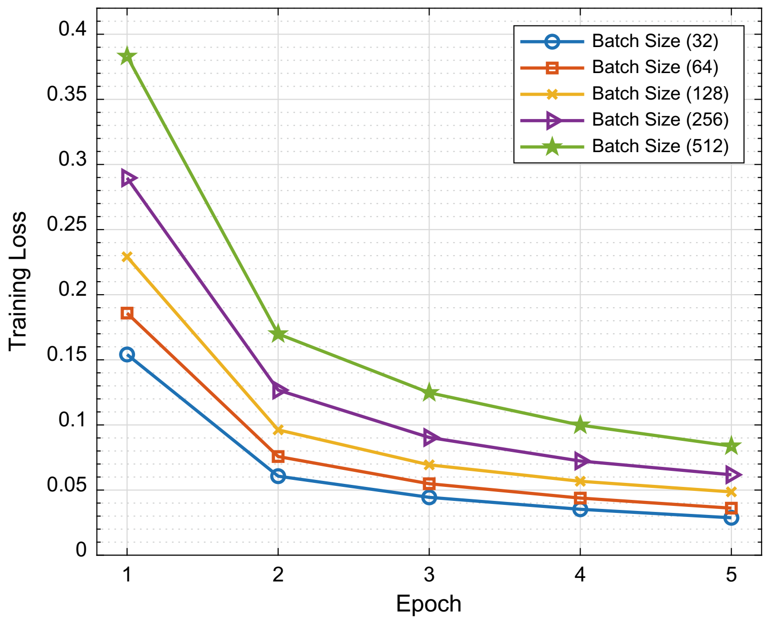


Fig. 9 Training loss of different batch sizes

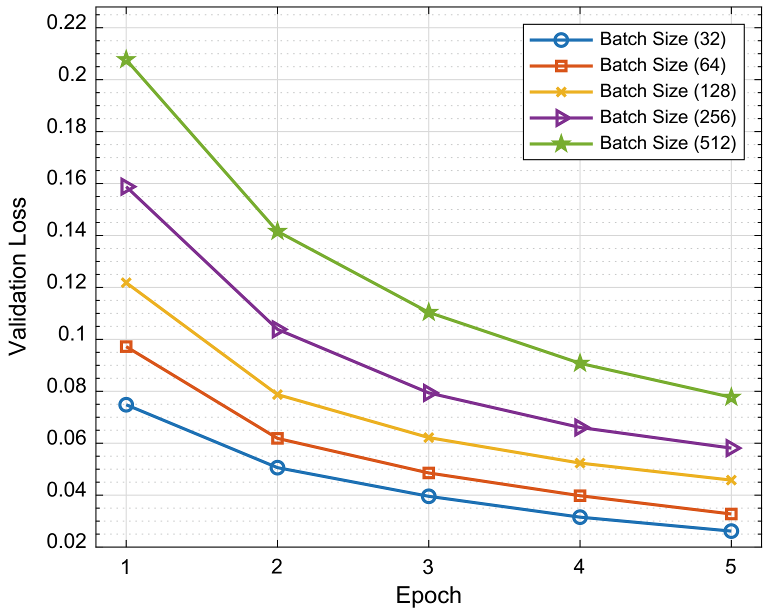


Fig. 10 Validation loss of different batch sizes

that training loss in BGRU-B32 reduced to 0.0287, while validation loss reduced to 0.0262. BGRU-B32 reduced the average training losses in BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512 by 18.49%, 35.38%, 49.58%, and 62.48%, respectively, while the average validation losses were reduced by 20.52%, 38.3%, 52.24%, and 64.55%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is best minimized when the batch size in BGRU was 32. In other words, a sufficiently small batch size in BGRU will facilitate high classification accuracy and good generalization ability that are required for efficient IoT botnet detection in smart homes.

Multi-class classification performance of BGRU-B32, BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512 was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. Table 5 shows that BGRU-B32 performed better than BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512. BGRU-B32 increased TPR by 0.42%, 0.86%, 12.80%, and 26.49% relative to BGRU-B64, BGRU-B128, BGRU-B256, and BGRU-B512, respectively; FPR decreased by 35%, 52.73%, 60%, and 69.05%, respectively; and MCC increased by 0.21%, 0.42%, 3.84%, and 26.24%. Therefore, a sufficiently small batch size in BGRU will ensure a high detection rate and reduce false alarm in the botnet detection system developed for smart homes. Figure 11 shows that training time decreased as the batch size increased. BGRU-B32 took the longest time (101.35 min) to train, while the shortest training time of 6.70 min was achieved in BGRU-B512.

Table 5 Performance of BGRU at different batch sizes

Metric	Units	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	32	99.12	99.51	78.82	99.99	75.00	90.49 ± 12.47
	64	98.60	99.29	77.65	99.99	75.00	90.11 ± 12.63
	128	97.97	99.18	76.47	99.98	75.00	89.72 ± 12.80
	256	97.70	98.93	67.06	99.90	37.50	80.22 ± 27.57
	512	97.60	98.01	62.35	99.76	0.00	71.54 ± 42.95
FPR	32	0.47	0.85	0.00	0.00	0.00	0.26 ± 0.38
	64	0.68	1.34	0.00	0.00	0.00	0.40 ± 0.60
	128	0.78	1.94	0.00	0.00	0.00	0.55 ± 0.85
	256	1.02	2.21	0.00	0.00	0.00	0.65 ± 0.98
	512	1.89	2.31	0.00	0.00	0.00	0.84 ± 1.16
MCC	32	99.44	99.28	94.39	100.00	93.30	97.28 ± 3.18
	64	99.16	98.88	94.06	100.00	93.30	97.08 ± 3.14
	128	98.92	98.43	93.72	99.99	93.30	96.87 ± 3.12
	256	98.68	98.18	90.94	99.97	80.62	93.68 ± 8.11
	512	98.02	97.88	89.48	99.94	0.00	77.06 ± 43.27

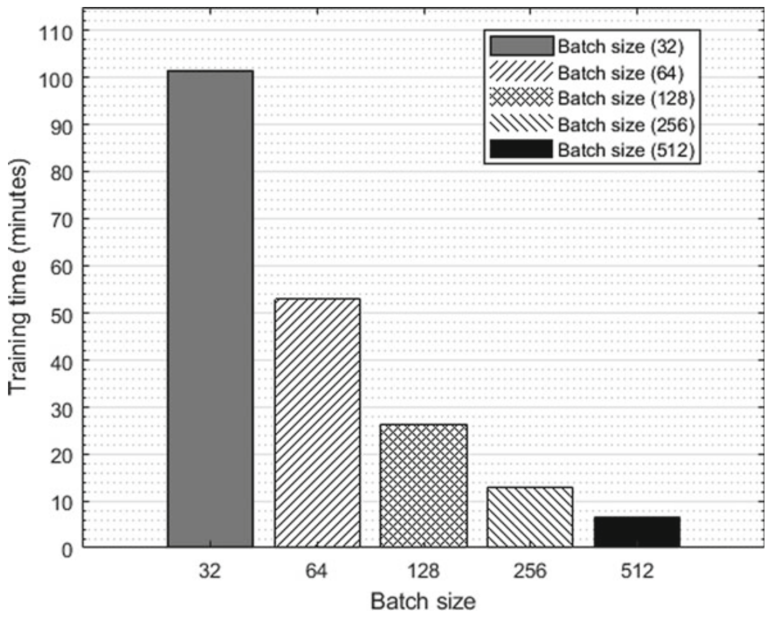


Fig. 11 Training time of different batch sizes

3.6 Influence of Optimizers on Classification Performance

In this subsection, we determine the most suitable optimizer required for efficient BGRU-based IoT botnet detection in smart homes. Four single layers BGRU neural networks with Adam, SGD, RMSprop, and Adadelta optimizers formed BGRU-OP1, BGRU-OP2, BGRU-OP3, and BGRU-OP4 classifiers, respectively, when trained using ReLU activation function, five epochs, and a batch size of 128.

Training and validation losses in BGRU-OP1, BGRU-OP2, BGRU-OP3, and BGRU-OP4 were analyzed to understand the extent of model underfitting and overfitting, respectively. Figures 12 and 13 show that the lowest training and validation losses were realized with Adam optimizer. In general, training and validation losses reduced in all the classifiers throughout the five-epoch period. However, training and validation losses were lower in BGRU-OP1 than in BGRU-OP2, BGRU-OP3, and BGRU-OP4. At the end of the experiment, we observed that training loss in BGRU-OP1 reduced to 0.0486, while validation loss reduced to 0.0458. BGRU-OP1 reduced the average training losses in BGRU-OP2, BGRU-OP3, and BGRU-OP4 by 86.54%, 4.22%, and 90.40%, respectively, while the average validation losses were reduced by 89.70%, 5.49%, and 92.31%, respectively. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is best minimized when Adam optimizer was employed in BGRU. In other words, the use of Adam optimizer in BGRU will facilitate high classification accuracy and good generalization ability that are required for efficient botnet detection in smart homes.

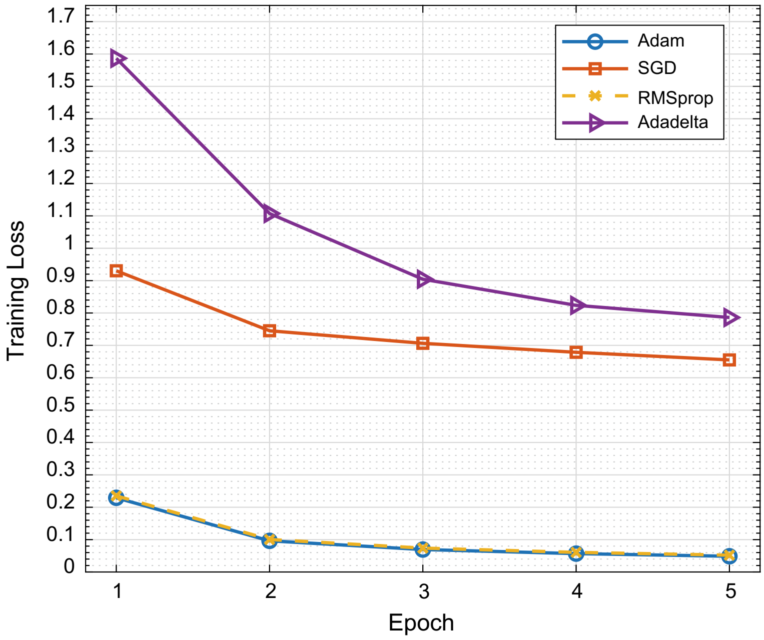


Fig. 12 Training loss of different optimizers

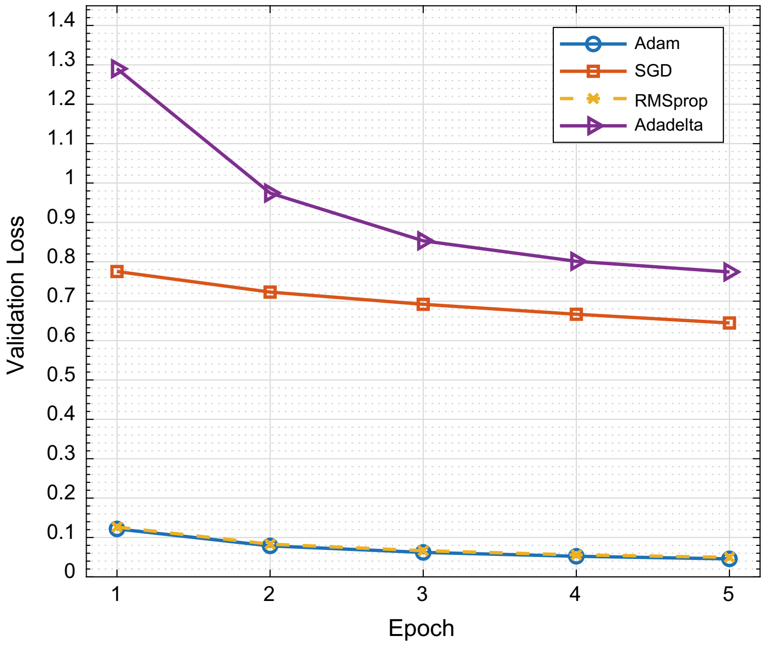


Fig. 13 Validation loss of different optimizers

Table 6 Performance of BGRU for different optimizers

Metric	Optimizer	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
TPR	Adam	97.97	99.18	76.47	99.98	75.00	89.72 ± 12.80
	SGD	82.42	66.80	0.00	0.00	0.00	29.84 ± 41.24
	RMSprop	98.35	98.21	22.35	99.82	0.00	63.75 ± 48.64
	Adadelta	99.89	3.63	0.00	0.00	0.00	20.70 ± 44.29
FPR	Adam	0.78	1.94	0.00	0.00	0.00	0.55 ± 0.85
	SGD	35.06	18.19	0.00	0.00	0.00	10.65 ± 15.75
	RMSprop	1.70	1.59	0.00	0.01	0.00	0.66 ± 0.90
	Adadelta	92.35	3.74	0.00	0.00	0.00	19.22 ± 40.92
MCC	Adam	98.92	98.43	93.72	99.99	93.30	96.87 ± 3.12
	SGD	72.17	76.90	0.00	0.00	0.00	29.81 ± 40.86
	RMSprop	98.35	98.43	73.64	99.95	0.00	74.07 ± 42.83
	Adadelta	52.78	49.79	0.00	0.00	0.00	20.51 ± 28.11

Multi-class classification performance of BGRU-OP1, BGRU-OP2, BGRU-OP3, and BGRU-OP4 was evaluated with respect to the ground truth labels based on TPR, FPR, and MCC. Table 6 shows that BGRU-OP1 performed better than BGRU-OP2, BGRU-OP3, and BGRU-OP4. BGRU-OP1 increased TPR by 200.67%, 40.74%, and 333.43% relative to BGRU-OP2, BGRU-OP3, and BGRU-OP4, respectively; FPR decreased by 94.84%, 16.67%, and 97.14%, respectively; and MCC increased by 224.96%, 30.78%, and 372.31%. Therefore, the adoption of Adam optimizer in BGRU will ensure a high detection rate and reduce false alarm in the botnet detection system developed for smart homes.

3.7 Performance of Deep BGRU-Based Multi-class Classifier

In this subsection, we evaluate the suitability of deep BGRU for IoT botnet detection in smart homes. A deep BGRU multi-class classifier was developed with the optimal hyperparameters in Sects. 3.1–3.6, namely, ReLU activation function, 20 epochs, 4 hidden layers, 200 hidden neurons, a batch size of 512, and Adam optimizer.

Training and validation losses in deep BGRU multi-class classifiers were analyzed to understand the extent of model underfitting and overfitting, respectively. Figure 14 shows that the training and validation losses were shallow when the optimal BGRU hyperparameters were used. Training and validation losses reduced throughout the five-epoch period. At the end of the experiment, we observed that training loss in deep BGRU multi-class classifiers reduced to 0.0018, while validation loss reduced to 0.0006. The reduction in training and validation losses implies that the likelihood of model underfitting and overfitting is minimized when the optimal hyperparameters were employed in BGRU. In other words, the choice of the optimal BGRU

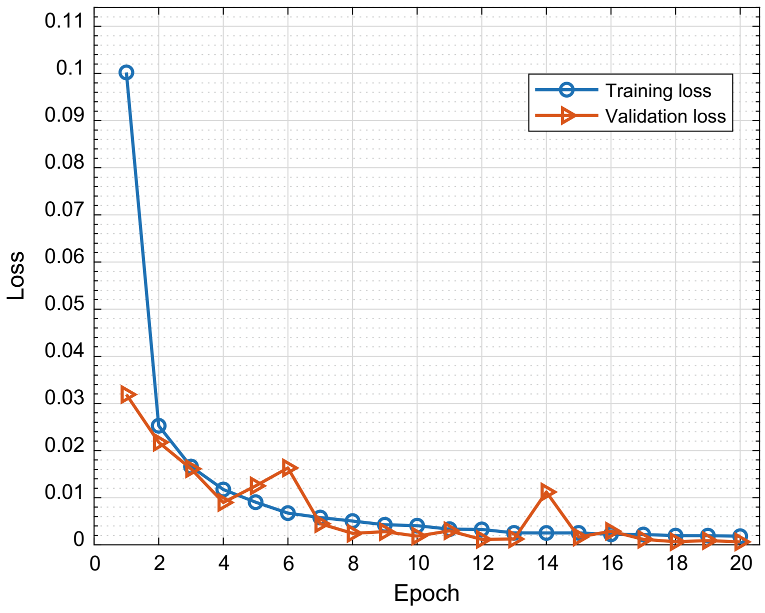


Fig. 14 Training and validation losses of deep BGRU classifier

hyperparameters will facilitate high classification accuracy and good generalization ability that are required for efficient IoT botnet detection in smart homes. The time needed to train the optimal deep BGRU classifier was 33.95 min.

Multi-class classification performance of deep BGRU multi-class classifier was compared with the state-of-the-art methods based on TPR, FPR, and MCC. Tables 7, 8, and 9 show that deep BGRU multi-class classifier outperforms mixture localization-based outliers (MLO) [47], SVM [48], RF [49], artificial immune system (AIS) [50], and feedforward neural network (FFNN) [51]. Deep BGRU multi-class classifier achieved high detection accuracy and low false alarm with true positive rate (TPR), false positive rate (FPR), and Matthews coefficient correlation (MCC) of 99.28 ± 1.57%, 0.00 ± 0.00%, and 99.82 ± 0.40%.

Table 7 TPR of multi-class classifiers for IoT botnet detection in smart homes

Ref	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
MLO [47]	98.24	98.20	0.00	95.67	95.86	77.59 ± 43.39
SVM [48]	98.95	99.87	87.97	99.33	100.00	97.22 ± 5.19
RF [49]	99.40	0.00	0.00	99.37	99.38	59.63 ± 54.43
AIS [50]	100.00	98.53	0.00	98.22	98.90	79.13 ± 44.24
FFNN [51]	99.414	0.00	98.071	98.38	88.92	76.96 ± 43.23
Deep BGRU	**100.00**	**99.99**	**96.47**	**99.96**	**100.00**	**99.28 ± 1.57**

Table 8 FPR of multi-class classifiers for IoT botnet detection in smart homes

Ref	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
SVM [48]	0.00	0.26	0.00	5.96	0.01	1.56 ± 2.36
Deep BGRU	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00 ± 0.00**

Table 9 MCC of multi-class classifiers for IoT botnet detection in smart homes

Ref	DDoS	DoS	Normal	Reconn.	Theft	Mean ± Stdev
SVM [48]	99.74	99.81	96.90	96.82	100.00	98.65 ± 1.64
Deep BGRU	**99.99**	**99.99**	**99.11**	**99.99**	**100.00**	**99.82 ± 0.40**

4 Conclusion

In this paper, an optimal model was developed for efficient botnet detection in IoT-enabled smart homes using deep BGRU. A methodology was proposed to determine the optimal BGRU hyperparameters (activation function, epoch, hidden layer, hidden unit, batch size, and optimizer) for multi-class classification. The proposed methodology was implemented, and the classification performance was jointly assessed based on training loss, validation loss, accuracy, TPR, FPR, MCC, and training time. Extensive simulation results showed that: (a) ReLU performed better than *tanh* activation functions; (b) classification performance improved with an increase in the numbers of epochs, hidden layers, and hidden units; (c) the performance of BGRU improved as the batch size becomes smaller, but this comes with a significant increase in training time; (d) Adam outperformed SGD, RMSprop, and Adadelta optimizers. Finally, the combination of ReLU activation function, 20 epochs, 4 hidden layers, 20 hidden units, a batch size of 512, and Adam optimizer achieved the best multi-class classification performance as follows: low training loss (0.0107 ± 0.0219), validation loss (0.0072 ± 0.0086), FPR (0.00 ± 0.00%); high accuracy (99.99%), TPR (99.28 ± 1.57), and MCC (99.82 ± 0.40).

Acknowledgments The authors would like to thank Cyraatek UK and the Faculty of Science & Engineering, Manchester Metropolitan University, for the research grant provided.

References

1. Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., Markakis, E.K.: A survey on the Internet of Things (IoT) forensics: challenges, approaches and open issues. IEEE Commun. Surv. Tutori. (2020). https://doi.org/10.1109/COMST.2019.2962586
2. Alam, S., Siddiqui, S.T., Ahmad, A., Ahmad, R., Shuaib, M.: Internet of Things (IoT) enabling technologies, requirements, and security challenges. In: Kolhe, M., Tiwari, S., Trivedi, M., Mishra, K. (eds.) Advances in Data and Information Sciences, vol. 94. pp. 119–126. Springer (2020)

3. Zaidan, A., Zaidan, B.: A review on intelligent process for smart home applications based on IoT: coherent taxonomy, motivation, open challenges, and recommendations. Artif. Intell. Rev. **53**(1), 141–165 (2020)
4. Bhattacharyya, R., Das, A., Majumdar, A., Ghosh, P.: Real-time scheduling approach for IoT-based home automation system. In: Data Management, Analytics and Innovation, pp. 103–113. Springer (2020)
5. Mahadewa, K., Wang, K., Bai, G., Shi, L., Liu, Y., Dong, J.S., Liang, Z.: Scrutinizing implementations of smart home integrations. IEEE Trans. Software Eng. (2019). https://doi.org/10.1109/TSE.2019.2960690
6. Singh, J., Pasquier, T., Bacon, J., Ko, H., Eyers, D.: Twenty security considerations for cloud-supported Internet of Things. IEEE Internet Things J. **3**(3), 269–284 (2015)
7. Yin, L., Luo, X., Zhu, C., Wang, L., Xu, Z., Lu, H.: ConnSpoiler: disrupting C&C communication of IoT-based Botnet through fast detection of anomalous domain queries. IEEE Trans. Indus. Inf. **16**(2), 1373–1384 (2020). https://doi.org/10.1109/TII.2019.2940742
8. Pour, M.S., Mangino, A., Friday, K., Rathbun, M., Bou-Harb, E., Iqbal, F., Samtani, S., Crichigno, J., Ghani, N.: On data-driven curation, learning, and analysis for inferring evolving Internet-of-Things (IoT) botnets in the wild. Comput. Secur. **91**, 101707 (2020)
9. Russell, B.: IoT cyber security. In: Intelligent Internet of Things, pp. 473–512. Springer (2020)
10. Alieyan, K., Almomani, A., Abdullah, R., Almutairi, B., Alauthman, M.: Botnet and Internet of Things (IoTs): a definition, taxonomy, challenges, and future directions. In: Security, Privacy, and Forensics Issues in Big Data, pp. 304–316. IGI Global (2020)
11. Al-Duwairi, B., Al-Kahla, W., AlRefai, M.A., Abdelqader, Y., Rawash, A., Fahmawi, R.: SIEM-based detection and mitigation of IoT-botnet DDoS attacks. Int. J. Electr. Comput. Eng. **10**, 2088–8708 (2020)
12. Gupta, B.B., Dahiya, A., Upneja, C., Garg, A., Choudhary, R.: A comprehensive survey on DDoS attacks and recent defense mechanisms. In: Handbook of Research on Intrusion Detection Systems, pp. 186–218. IGI Global (2020)
13. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the Internet of things for network forensic analytics: Bot-iot dataset. Fut. Gen. Comput. Syst. **100**, 779–796 (2019)
14. Asadi, M., Jamali, M.A.J., Parsa, S., Majidnezhad, V.: Detecting Botnet by using particle swarm optimization algorithm based on voting system. Fut. Gen. Comput. Syst. **107**, 95–111 (2020)
15. Nguyen, H.-T., Ngo, Q.-D., Nguyen, D.-H., Le, V.-H.: PSI-rooted Subgraph: A Novel Feature for IoT Botnet Detection Using Classifier Algorithms. ICT Express (2020)
16. Nõmm, S., Bahşi, H.: Unsupervised anomaly based Botnet detection in IoT networks. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) 2018, pp. 1048–1053. IEEE (2018)
17. Al Shorman, A., Faris, H., Aljarah, I.: Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT Botnet detection. J. Amb. Intell. Human. Comput. 1–17 (2019)
18. Yang, Y., Wang, J., Zhai, B., Liu, J.: IoT-based DDoS attack detection and mitigation using the edge of SDN. In: International Symposium on Cyberspace Safety and Security 2019, pp. 3–17. Springer (2019)
19. D'hooge, L., Wauters, T., Volckaert, B., De Turck, F.: In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats. In: Proceedings of the 4th International Conference on Internet of Things, Big Data and Security 2019 (2019)
20. Gurulakshmi, K., Nesarani, A.: Analysis of IoT Bots against DDoS attack using machine learning algorithm. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI) 2018, pp. 1052–1057. IEEE (2018)
21. Nomm, S., Guerra-Manzanares, A., Bahsi, H.: Towards the Integration of a post-Hoc interpretation step into the machine learning workflow for IoT Botnet detection. In: 2019 18th IEEE International Conference on Machine Learning And Applications (ICMLA) 2019, pp. 1162–1169. IEEE (2019)

22. Moustafa, N., Turnbull, B., Choo, K.-K.R.: An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things. IEEE Internet Things J. **6**(3), 4815–4830 (2018)
23. Wildani, I., Yulita, I.: Classifying Botnet attack on Internet of Things device using random forest. In: IOP Conference Series: Earth and Environmental Science 2019, vol. 1, p. 012002. IOP Publishing (2019)
24. Bahşi, H., Nõmm, S., La Torre, F.B.: Dimensionality reduction for machine learning based IoT botnet detection. In: 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV) 2018, pp. 1857–1862. IEEE (2018)
25. Koroniotis, N., Moustafa, N., Sitnikova, E., Slay, J.: Towards developing network forensic mechanism for Botnet activities in the IoT based on machine learning techniques. In: International Conference on Mobile Networks and Management 2017, pp. 30–44. Springer (2017)
26. Guerra-Manzanares, A., Bahsi, H., Nõmm, S.: Hybrid feature selection models for machine learning based Botnet detection in IoT networks. In: 2019 International Conference on Cyberworlds (CW) 2019, pp. 324–327. IEEE (2019)
27. Soe, Y.N., Santosa, P.I., Hartanto, R.: DDoS Attack Detection Based on Simple ANN with SMOTE for IoT Environment. In: 2019 Fourth International Conference on Informatics and Computing (ICIC) 2019, pp. 1–5. IEEE (2019)
28. Haq, S., Singh, Y.: Botnet detection using machine learning. In: 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) 2018, pp. 240–245. IEEE (2018)
29. Bansal, A., Mahapatra, S.: A comparative analysis of machine learning techniques for Botnet detection. In: Proceedings of the 10th International Conference on Security of Information and Networks 2017, pp. 91–98 (2017)
30. Amanullah, M.A., Habeeb, R.A.A., Nasaruddin, F.H., Gani, A., Ahmed, E., Nainar, A.S.M., Akim, N.M., Imran, M.: Deep learning and big data technologies for IoT security. Comput. Commun. (2020)
31. Jung, W., Zhao, H., Sun, M., Zhou, G.: IoT Botnet detection via power consumption modeling. Smart Health **15**, 100103 (2020)
32. Le, H.-V., Ngo, Q.-D., Le, V.-H.: Iot Botnet detection using system call graphs and one-class CNN classification. Int. J. Innov. Technol. Explor. Eng. **8**(10), 937–942
33. Liu, J., Liu, S., Zhang, S.: Detection of IoT Botnet based on deep learning. In: 2019 Chinese Control Conference (CCC) 2019, pp. 8381–8385. IEEE (2019)
34. Nguyen, H.-T., Ngo, Q.-D., Le, V.-H.: IoT Botnet detection approach based on PSI graph and DGCNN classifier. In: 2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP) 2018, pp. 118–122. IEEE (2018)
35. Hwang, R.-H., Peng, M.-C., Nguyen, V.-L., Chang, Y.-L.: An LSTM-based deep learning approach for classifying malicious traffic at the packet level. Appl. Sci. **9**(16), 3414 (2019)
36. McDermott, C.D., Majdani, F., Petrovski, A.V.: Botnet detection in the Internet of things using deep learning approaches. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2018)
37. McDermott, C.D., Petrovski, A.V., Majdani, F.: Towards situational awareness of botnet activity in the Internet of things. In: 2018 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA), Glasgow, UK, pp. 1–8. IEEE (2018)
38. Sachin, S., Tripathi, A., Mahajan, N., Aggarwal, S., Nagrath, P.: Sentiment analysis using gated recurrent neural networks. SN Comput. Sci. **1**(2), 1–13 (2020)
39. Liu, C., Liu, Y., Yan, Y., Wang, J.: An intrusion detection model with hierarchical attention mechanism. IEEE Access (2020)
40. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
41. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)

42. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proc. IEEE **78**(10), 1550–1560 (1990)
43. Luque, A., Carrasco, A., Martín, A., de las Heras, A.: The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recogn. **91**, 216–231 (2019)
44. Baloglu, U.B., Talo, M., Yildirim, O., San Tan, R., Acharya, U.R.: Classification of myocardial infarction with multi-lead ECG signals and deep CNN. Pattern Recogn. Lett. **122**, 23–30 (2019)
45. Hartmann, C., Opritescu, D., Volk, W.: An artificial neural network approach for tool path generation in incremental sheet metal free-forming. J. Intell. Manuf. **30**(2), 757–770 (2019)
46. Patro, S., Sahu, K.K.: Normalization: a pre-processing stage. arXiv preprint arXiv:1503.06462 (2015)
47. AlKadi, O., Moustafa, N., Turnbull, B., Choo, K.-K.R.: Mixture localization-based outliers models for securing data migration in cloud centers. IEEE Access **7**, 114607–114618 (2019)
48. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., Alazab, A.: A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks. Electronics **8**(11), 1210 (2019)
49. Soe, Y.N., Feng, Y., Santosa, P.I., Hartanto, R., Sakurai, K.: Towards a lightweight detection system for cyber attacks in the IoT environment using corresponding features. Electronics **9**(1), 144 (2020)
50. Aldhaheri, S., Alghazzawi, D., Cheng, L., Alzahrani, B., Al-Barakati, A.: DeepDCA: novel network-based detection of IoT attacks using artificial immune system. Appl. Sci. **10**(6), 1909 (2020)
51. Ge, M., Fu, X., Syed, N., Baig, Z., Teo, G., Robles-Kelly, A.: Deep learning-based intrusion detection for IoT networks. In: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC) 2019, pp. 256–25609. IEEE (2019)