International Journal of Machine Learning and Cybernetics (2021) 12:3145–3167
https://doi.org/10.1007/s13042-021-01306-8

ORIGINAL ARTICLE

Recurrent autonomous autoencoder for intelligent DDoS attack mitigation within the ISP domain

Ili Ko[1] · **Desmond Chambers**[1] · **Enda Barrett**[1]

Received: 20 June 2020 / Accepted: 10 March 2021 / Published online: 26 March 2021

Abstract
The continuous advancement of DDoS attack technology and an increasing number of IoT devices connected on 5G networks escalate the level of difficulty for DDoS mitigation. A growing number of researchers have started to utilise Deep Learning algorithms to improve the performance of DDoS mitigation systems. Real DDoS attack data has no labels, and hence, we present an intelligent attack mitigation (IAM) system, which takes an ensemble approach by employing Recurrent Autonomous Autoencoders (RAA) as basic learners with a majority voting scheme. The RAA is a target-driven, distributionenabled, and imbalanced clustering algorithm, which is designed to work with the ISP's blackholing mechanism for DDoS flood attack mitigation. It can dynamically select features, decide a reference target (RT), and determine an optimal threshold to classify network traffic. A novel Comparison-Max Random Walk algorithm is used to determine the RT, which is used as an instrument to direct the model to classify the data so that the predicted positives are close or equal to the RT. We also propose Estimated Evaluation Metrics (EEM) to evaluate the performance of unsupervised models. The IAM system is tested with UDP flood, TCP flood, ICMP flood, multi-vector and a real UDP flood attack data. Additionally, to check the scalability of the IAM system, we tested it on every subdivided data set for distributed computing. The average Recall on all data sets was above 98%.

Keywords Deep learning · Autoencoder · Machine learning · Unsupervised learning · DDoS mitigation · Random walk · Evaluation metrics for unsupervised learning · Cyber security · Network security

1 Introduction

Cyberattackers utilise botnets to launch distributed denial of service (DDoS) attacks to send an enormous amount of junk traffic to flood a victim's server to cause service interruption to legitimate users. DDoS attacks have been in existence for around 20 years, but the continuous development of 5G technology will escalate the magnitude and the frequency of the attacks. The increasing number of poorly secured devices connected on 5G networks, cheap DDoS services for hire and 23 million DDoS attack tools on demand, ensures that the ability to mitigate larger and more advanced DDoS attacks is one of the top 5G security requirements [1,2]. According to the ENISA Threat Landscape Report 2018, the average DDoS attack endured 318.10 mins, while the most prolonged attack persisted over six days [3]. The first terabit attack was 1.35 Tpbs targeting GibHub, and shortly after that, a 1.7 Tbps attack targeted Arbor Networks [4]. Since the coronavirus lockdown, the number of DDoS attacks skyrocketed in 2020, and nearly 90% of attacks were over 100 Gbps. Cloudflare had reported a really large attack, peaking at 754 million packets per second [5]. The average cost of a DDoS attack for businesses in 2017 was $2.5 million in the United States [6]. The increasing number of DDoS attacks raises the cost as well. It is estimated that DDoS attacks could cost the UK more than £1bn in 2019 [7]. DDoS technology has evolved from being a single vector to multi-vector attack [9, 24]. The rise of artificial intelligence (AI) enables the DDoS technology to dynamically change the traffic patterns during the time an attack is active [11], which intensifies the level of difficulty for DDoS mitigation.

The Internet service provider (ISP) is the connector between the Internet and the users, and thus, deploying the mitigation system within the ISP domain can provide an efficient solution. Blackholing is an effective technique to

✉ Ili Ko
i.ko3@nuigalway.ie

[1] National University of Ireland Galway, Galway, Ireland

mitigate DDoS attacks [12, 13], so its usage has increased [12, 14]. Nonetheless, a network traffic classifier is required before employing the blackholing technique to minimise legitimate users' services being interrupted. Machine learning, such as supervised learning and unsupervised learning, have been widely used for DDoS mitigation systems. However, security experts have suggested that supervised learning will have difficulties in dealing with the advanced DDoS attacks because it is impossible to create all types of traffic profiles to train the model. On the contrary, with the ability to learn and adapt to the changes of the attack patterns, unsupervised learning is a superior technique to defend against AI-based DDoS attacks [11]. Deep Learning (DL) overcomes the limitation of the traditional machine learning approaches due to the shallow representation generated by the models [15]. Consequently, an increasing number of useful and exciting applications in the industry and the research community utilise DL [9]. Due to the inherited non-linearity of neural networks [10], DL approaches outperformed other machine learning classification techniques [9]. As a result, DL has started to gain its popularity among researchers for building DDoS detection or mitigation systems [16, 17–21]. Therefore, we propose an unsupervised Deep Learning algorithm, the Recurrent Autonomous Autoencoder (RAA), to construct the intelligent attack mitigation (IAM) system. The IAM system is illustrated in Fig. 1, which contains a Data Processor and an Ensemble-N Module with N number of RAAs to improve the performance of the model [22]. Each RAA has a Feature Selector, a Target Detector, and a NetFlow Identifier, and all of them utilise the Autonomous Autoencoder (AA).

The AA is different from the regular Autoencoder because the output of the AA is a binary classification which is controlled by a class switch that is designed to exploit the imbalanced data set that is generally deemed as a problem for machine learning models. For example, if there are 10 normal IP addresses amongst 1000, when the AA identifies 8, most likely they are 8 normal IP addresses. However, if there are 450 normal and 550 malicious IP addresses, it is difficult to determine whether they are normal or malicious when the AA classifies 485 for a group and 515 for another group. Therefore, the AA works particularly well with imbalanced data. The AA has improved from our previous model that utilised a Complete Autoencoder (CA) [23] because the AA no longer requires an RT calculated from a few time frames before the attack.

The design concept behind the RAA originated from 'Tell me the number and I will identify them'. The 'number' is the reference target (RT), which should be close or equals to the number of normal IP addresses (actual positives) during the attack. Based on our previous research, we discovered that the model could pinpoint the normal IP addresses if the number of them is given [29,30]. Unfortunately, in the real world, the number of normal IP addresses during an attack is unknown. As a result, we utilise the RT to direct the system to classify the data so that the predicted positives are close or equal to the actual positives. Accordingly, we equipped the IAM with the Target Detector that can automatically find the RT via Comparison-Max Random Walk (CMRW)

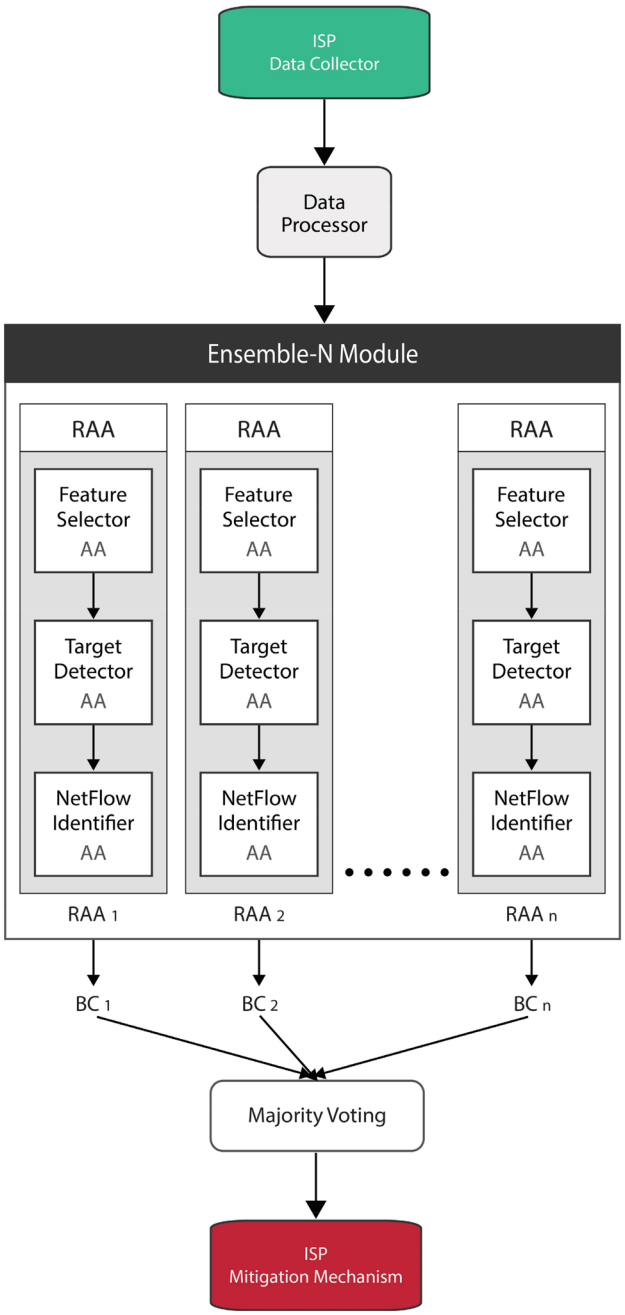


Fig. 1 The IAM contains a Data Processor and an Ensemble-N Module utilising a majority voting scheme to attain the final classification, and the Ensemble-N Module consists of N number of RAAs, which each RAA has a Feature Selector, a Target Detector, and a NetFLow Identifier

algorithm. The intuition of the CMRW came from the game 'Guess number higher or lower' to move the potential RT to a higher or lower direction. More detailed explanations of the CMRW are provided in Section 7.3. With the ability to find the RT, the RAA system is a frame independent, target-driven, and distribution-enabled clustering model. The RAA system does not require time-series data, so it is frame independent. Additionally, the system uses the RT as a guide to classify the network traffic, so it is target-driven. Furthermore, the RAA is distribution-enabled because it can dynamically find an RT for each subdivided data set for distributed computing.

Verisign's Q2 2018 DDoS trends report [24] reveals that 52% of attacks utilised multiple attack types, as shown in Fig. 2, and UDP based, TCP based, and IP fragment attacks were the top three attacks. Consequently, we tested our proposed system with UDP flood, TCP flood, ICMP flood, multi-vector attack and a real UDP flood attack data set. The limitation of the IAM system is that it is designed for mitigating DDoS flood attack.

The contributions of this chapter are as follows:

1. We developed a Comparison-Max Random Walk algorithm to find the RT automatically to guide the system to classify the network traffic.
2. We proposed Estimated Evaluation Metrics (EEM) by offering a systemic way to find the estimated actual positives (EAP) to calculate the estimated actual negatives (EAN), the estimated true positives (ETP), the estimated false negatives (EFN), the estimated true negatives (ETN), and the estimated false positives (EFP) to evaluate unsupervised learning models.

The paper is organised as follows. Section 2 discusses related work and differences of the proposed system. Section 3 gives an overview of the IAM system design. Section 4 elaborates the experimental implementation of our proposed system. Section 5 presents the performance results of the IAM system. Finally, Sect. 6 provides the conclusion and future work.

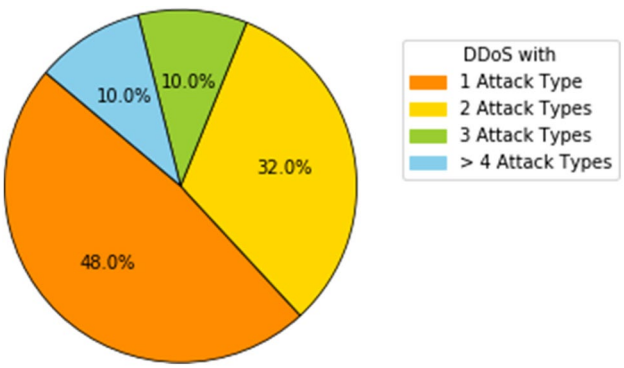


Fig. 2 DDoS attacks utilising different attack types

2 Related work

Deep Learning is a sub-domain of artificial intelligence inspired by the processes of data processing, pattern recognition, and decision making of the brain called Artificial Neural Networks. Deep Learning algorithms utilise a hierarchical learning process to extract complex abstracts for data representation [19].

One of the main reasons for using Deep Learning is due to its ability to analyse and learn from big unlabelled data. The wealth of information hidden in big data provides incredible potential across different domains, which include finance, health care, agriculture, transportation, retail, and customer service [20]. There have been a plethora of applications of Deep Learning in computer vision, speech recognition, marketing, fraud detection, and cybersecurity.

As DDoS attacks remain one of the top security threats, researchers continue to develop new DDoS mitigation systems. To achieve desirable performance, a growing number of researchers are utilising DL models for DDoS attack defence systems. For example, Doriguzzi-Corin et al. [31] presented a LUCID system, which utilised Convolutional Neural Networks (CNNs) to classify network traffic. They validated the performance of the system in a resource-constrained environment. Not only did the performance of the LUCID match with the state-of-the-art DDoS mitigation systems, but also the processing time was reduced by more than 40 times. Another research group which utilised DL is Niyaz's team, where they [9] proposed a multi-vector DDoS detection system that consisted of stacked Sparse Autoencoders and a softmax classifier for feature selection and classification. They tested their model on a data set, which contained regular Internet traffic and different types of DDoS attacks. Their proposed system had high accuracy with a low false-positive rate for attack detection. Additionally, Liu et al. [21] presented a deep reinforcement learning-based system, which can better learn the optimal mitigation policies to mitigate different types of attacks such as TCP SYN, UDP, and ICMP flood in real-time. Their system outperformed a popular state-of-the-art router throttling method. Yuan and his colleagues [15] suggested a Recurrent Deep Neural Network utilising a Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory Neural Network (LSTM) and Gated Recurrent Unit Neural Network (GRU) to learn patterns from sequences of network traffic and attack activities. The experimental results showed that their system outperformed conventional machine learning models. Furthermore, Asad's team [26] proposed a deep neural network-based system that

uses feed-forward back-propagation. The system contained seven hidden layers, and a softmax activation was applied to the output layer to classify network traffic. Their model achieved an accuracy of 98% on the CIC IDS 2017 data set [25]. Salama et al. [27] also suggested a model using a Restricted Boltzmann machine (RBM) to select features for a support vector machine (SVM) to classify network traffic. The accuracy of their model on NSL-KDD data set was 92.84%. Table 1 displays advantages and disadvantages of each related work.

There are a few differences between the proposed model and the systems mentioned above. Firstly, the RAA is a target-driven model that utilises an RT to guide the unsupervised model to classify the network traffic by exploiting the imbalanced characteristic of the attack data set. Secondly, the RAA applies the novel Comparison-Max Random Walk algorithm to find the RT.

Thirdly, the Feature Selector, the Target Detector, and the NetFlow Identifier all employ the Autonomous Autoencoders. Lastly, the RAA can automatically select features, find the RT, and attain the final classification, which offers scalability to deal with big data.

3 System design

The IAM system is designed to work with the ISP's blackholing mechanism to drop malicious traffic for DDoS mitigation. The core of the IAM system is the Recurrent Autonomous Autoencoder. Hitherto we have proposed several target-driven unsupervised models that used the reference target RT to instruct the unsupervised model to attain the final classification [28, 29, 30, 32]. During an attack, there is a huge surge in network traffic and the number of IP addresses. Therefore, we made an assumption that the IP addresses which emerged before the attack are normal. Previously, the RT was equal to the number of IP addresses which existed right before the attack, as shown in Fig. 4; each ball in the time frame before the attack represents one IP address. Even though the RT is more than likely to be different from the number of normal IP addresses during the attack, previously presented systems still performed well on classifying NetFlows. Hence, we continue to utilise the RT as a way to instruct the model to obtain the final classification according to the error which is the absolute value of the difference between the number of normal IP addresses classified by the model (N_f = predicted positives) and the RT. However, there are a couple of issues with previously proposed models because they required the number of normal IP addresses before the attack to be calculated, which is the RT. Firstly, to increase the scalability of the system, the attack data set needs to be subdivided. For example, if 10 proposed models are deployed, the attack data set is divided into 10 sub data sets. Each model requires a sub data set and an RT. The RT for the previously proposed model is calculated as one number from the entire data set before the attack. When the data set is divided into 10 sub data sets, it is very difficult to determine the number of normal IP addresses for each sub data set without knowing how the normal and malicious IP addresses are distributed in each sub data set. Secondly, even after calculating the RT, the real number of normal IP addresses (actual positives) during the attack can be very different from the RT, which can drastically affect the performance of the system. For example, if an attack targeted at an online store occurs immediately after a flash sale started, the number of IP addresses will increase dramatically. Consequently, the number of normal IP addresses during the attack will be much greater than the RT.

To overcome these two problems, the RAA employs a Target Detector to find the RT automatically via Comparison-Max Random Walk. The distance of the walk is guided by a moving range (MR), the direction of the walk is determined by a higher or lower likelihood based on the random steps, and the destination of the walk is decided by the recurrence frequency. More detailed explanations

Table 1 Feature-based comparison of the advantages and disadvantages of each related work

Model	Advantages	Disadvantages
Doriguzzi [31]	Reducing the need for feature engineering	Hyper-parameter values relying on preliminary tuning
Niyaz [9]	Utilising Stacked Sparse Autoencoder to extract additional features	Potential loss of information using bottleneck from the previous Autoencoder as an input for the next Autoencoder
Liu [21]	Learning different attack patterns for selecting optimal mitigation policies	Difficult to learn all attack patterns due to the continuous advancement of DDoS attack technology
Yuan [15]	Improving model performance combining various deep learning models	The complexity of the model increasing the level of difficulty for duplicating the model
Asad [26]	Simplicity of the model making it easy to understand and duplicate	Potential performance reduction on data sets with different attacks exhibiting different traffic patterns
Salama [27]	Applying RBM to select features for a SVM	Requirement an additional machine learning model for classification

are provided in the 'Target Detector' section. Furthermore, to reduce the variance of the system, the IAM applies an ensemble technique by employing RAAs as base learners and utilising a majority voting scheme to attain the final classification. Therefore, the IAM composes of two units, which are a Data Processor and an Ensemble-N Module that uses N number of RAAs, and each RAA includes a Feature Selector, a Target detector, and a NetFlow Identifier as displayed in Fig. 1. To improve the performance of the IAM, the system utilises Top-N, Max-N, minimum error, and Ensemble-N as illustrated in Fig. 3.

The RAA first finds the RT and then, utilises the RT to cluster the number of data points that are close or equal to the RT. The following example explains the intuition of the target-driven unsupervised learning techniques for the RAA. Each ball in Fig. 4 represents an IP address; blue balls are normal IP addresses, and red balls are malicious IP addresses. Assuming that after utilising the CMRW, the RAA determines the RT = 10, and thus, the system utilises threshold-moving to classify the data so that the number of predicted positives N_f is close or equal to 10. The potential thresholds for this example are in the list of [0.7, 0.6, 0.5, 0.4, 0.3]. To better visualise the target-driven process for obtaining the final classification, the distance between the ball represents the output value of the ball. The model determines the optimal threshold by iterating through the threshold list as follows.

Step 1: The model starts with the threshold = 0.7, and it can easily identify six balls, which are ball numbers 2, 3, 4, 7, 9, and 10 because they are further away from other balls. After the first iteration, the $\text{error}_1 = | 6 - 10 | = 4$.

Step 2: The system uses the threshold = 0.6 and ball number 1 is also identified besides ball numbers 2, 3, 4, 7, 9, and 10. The $\text{error}_2 = | 7 - 10 | = 3$.

Step 3: The system classifies with the threshold = 0.5. Even though ball number 5 is close to red ball number 11, the distance between ball number 5 and ball number 11 is larger than the distance amongst red balls. Therefore, the system discovers ball numbers 5 and 6, which makes the $N_f = 9$ and the $\text{error}_3 = 9\ 10 = 1$. Heretofore, the system only found nine balls, which is one ball less than the RT.

Step 4: The system applies the threshold = 0.4. If the system finds additional ball numbers 8, 11, 12, 13, 14, 15, 16 and 17, the $\text{error}_4 = | 17 - 10 | = 7$, which is greater than the error_3. Then, the system stops iterating and the final result is the classification generated by the threshold = 0.5 and the $N_f = 9$.

The Feature Selector, the Target Detector, and the NetFlow Identifier of the RAA all have an Autonomous Autoencoder as illustrated in Fig. 5. The output of an AA is a binomial classification, in which 0 represents malicious data points, and 1 represents normal data points.

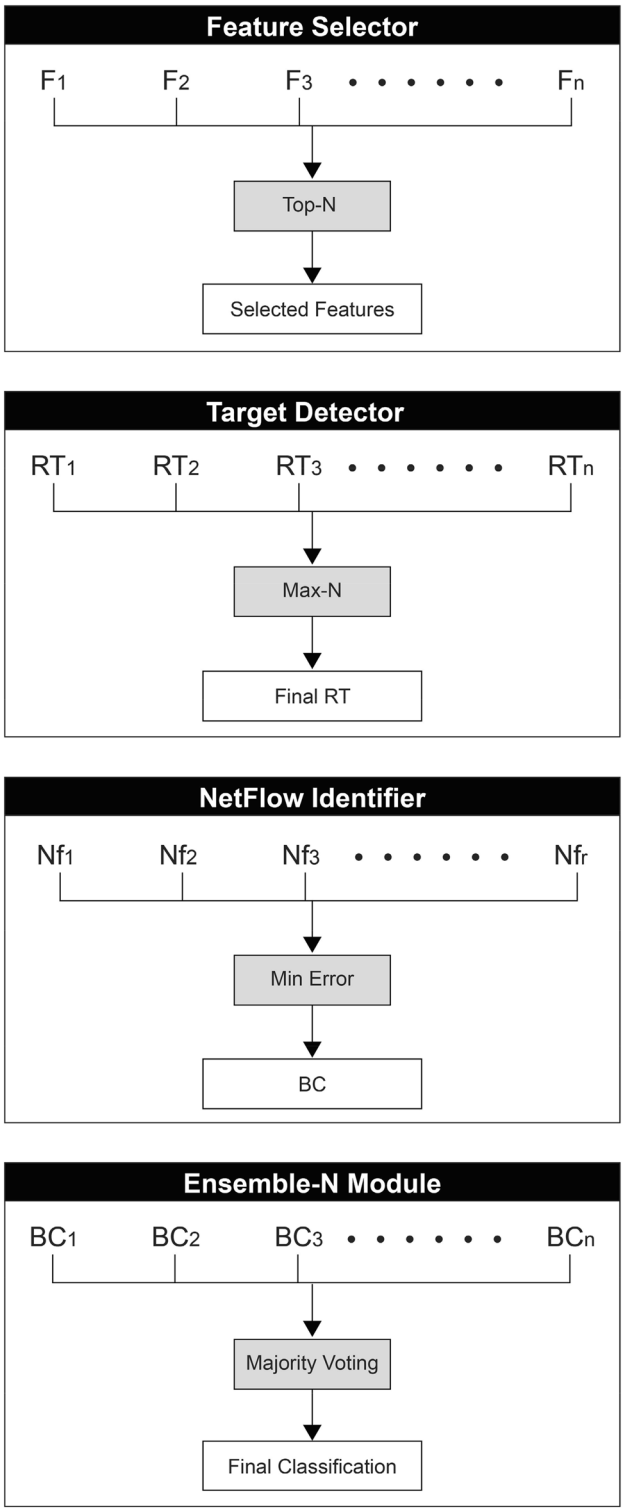


Fig. 3 The process of using Top-N to select features, determining the final RT by utilising Max-N, finding a best classification (BC) with the minimum error, and attaining the final classification by employing several RAAs in the Ensemble-N Module

– The first AA belongs to the Feature Selector. The input X is the transformed data set and the output O_f is a list containing selected features.
– The second AA belongs to the Target Finder. The input X_1 is the data with features selected and the output O_t is the RT.
– The last AA belongs to the NetFlow Identifier. The input is X_1 and the output O_c is a list of malicious IP addresses.

We will start with the design of the Data Processor followed by the Autonomous Autoencoder and each component of the RAA in the remaining part of this section. To aid the understanding of the remaining sections, Table 2 provides descriptions for variables.

3.1 Data processor

The Data Processor has a Horizontal Expansion and Vertical Compression (HEVC) engine [28], that we previously designed to make use of the hierarchical features contained in the data collected by the ISP [29]. The HEVC engine utilises the Apache Spark framework for fast distributed

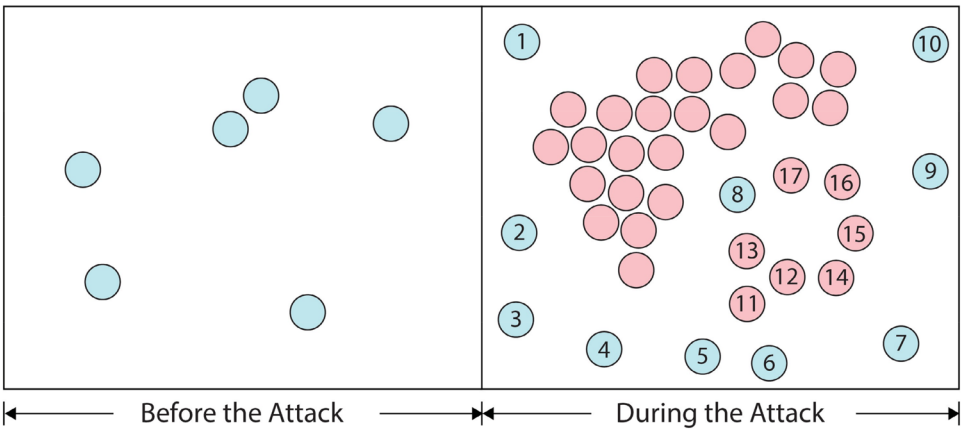


Fig. 4 Each ball represents an IP address as it existed in the time frame right before and during the attack

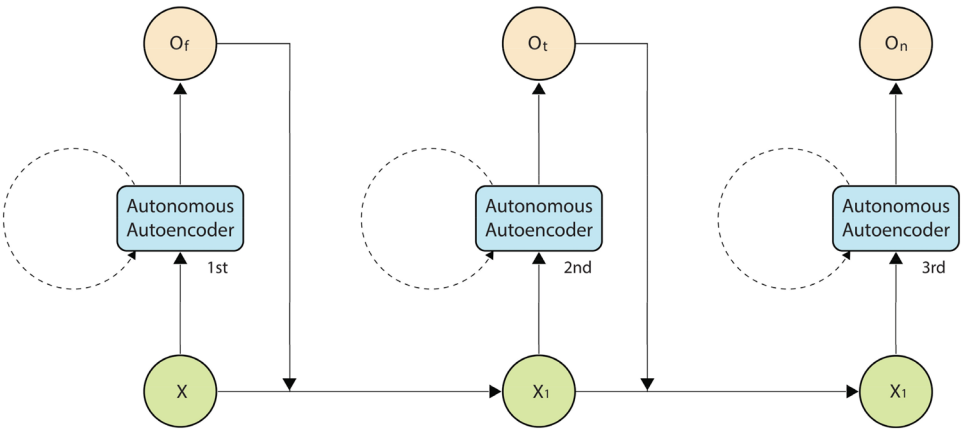


Fig. 5 The network architecture of the Recurrent Autonomous Autoencoder, in which X is the processed data with all features, X_1 is the processed data with selected features, O_f is the output of the Feature Selector, O_t is the output of the Target Detector, and the O_c is the output of the NetFlow Identifier

Table 2 Descriptions of variables or names

Variables or Names	Descriptions
N_f	The number of normal IP addresses predicted by the model (predicted positives)
Nf (0)	The number of final predicted positives that need to be evaluated by Estimated Evaluation Metrics (EEM)
TIPs	The total number of IP addresses in the data
Top-N	Features with the highest N number of N_f average
Max-N	N number of reference targets that the Target Detector needs to generate for finding the final reference target with the maximum mode
Ensemble-N	N number of RAAs in the Ensemble-N Module

computing to enable scalability to deal with an immense amount of network traffic generated by large-scale DDoS attacks. The Horizontal Expansion unit extracts additional features based on statistical analysis and unique values. The Vertical Compression unit groups and aggregates the data according to the unique source IPs. For example, the data contains two features, which are 'srcIP' and 'packets' as shown in Fig. 6. The Horizontal Expansion unit extracts three additional features from the original feature 'packets', which are 'lPSum', 'lPMean', and 'lUP'. The 'lPSum' is the sum of the number of packets for each source IP. The 'lPMean' is the mean of the packets for every source IP. The 'lUP' counts the unique value of packets for each source IP. Next, the original feature 'packets' is removed, and the Vertical Compression unit groups and aggregates the data based on the unique IP addresses. Consequently, each row represents an aggregated NetFlow for a unique IP address. Descriptions of extracted features are presented in the appendix A. It is worth mentioning that the number of rows in the transformed data set equals the number of unique source IP addresses or NetFlows.

3.2 Autonomous autoencoder

The most basic component in the RAA is the Autonomous Autoencoder, as depicted in Fig. 7, which is an extension of the Complete Autoencoder (CA) [23] that we previously designed. The main difference between the AA and CA is that the CA requires an RT that is calculated according to data sets before the attack; however, the AA can find the RT automatically with only the data during the attack. The AA consists of a Deep Autoencoder, which contains two symmetrical Deep-Belief Networks (DBN) as depicted in Fig 7. The encoding DBN is constructed from two hidden layers, which are h1 and h2 with eight neurons and four neurons, respectively. The decoding DBN encompasses two hidden layers, which are h3 and h4 with four neurons and eight neurons, respectively. The bottleneck contains a single neuron which generates a n x1 vector (n = number of rows, IP addresses, or NetFlows). The reason that there is only one neuron at the bottleneck is to create output values that are in the range of (0, 1), so a threshold can be applied to classify the output without deploying another machine learning algorithm. The AA can dynamically find an optimal threshold, and it is equipped with a class switch with the intent to exploit the imbalanced characteristic of the attack data. By default, the controller of the class switch is $\beta = 0.5$, where it instructs the class switch to swap the labels if the predicted positives N_f is more than 50% of the total number of IP addresses (TIPs). The advantage of using the class switch is that it offers a bi-directional comparison of the output value to a threshold without changing the comparison sign of '<' and '>'. This is particularly useful during the feature selecting process since only one feature is fitted to the AA at a time. For example, if a threshold is set at 0.5, the variance of the different number of octets for the normal IP addresses

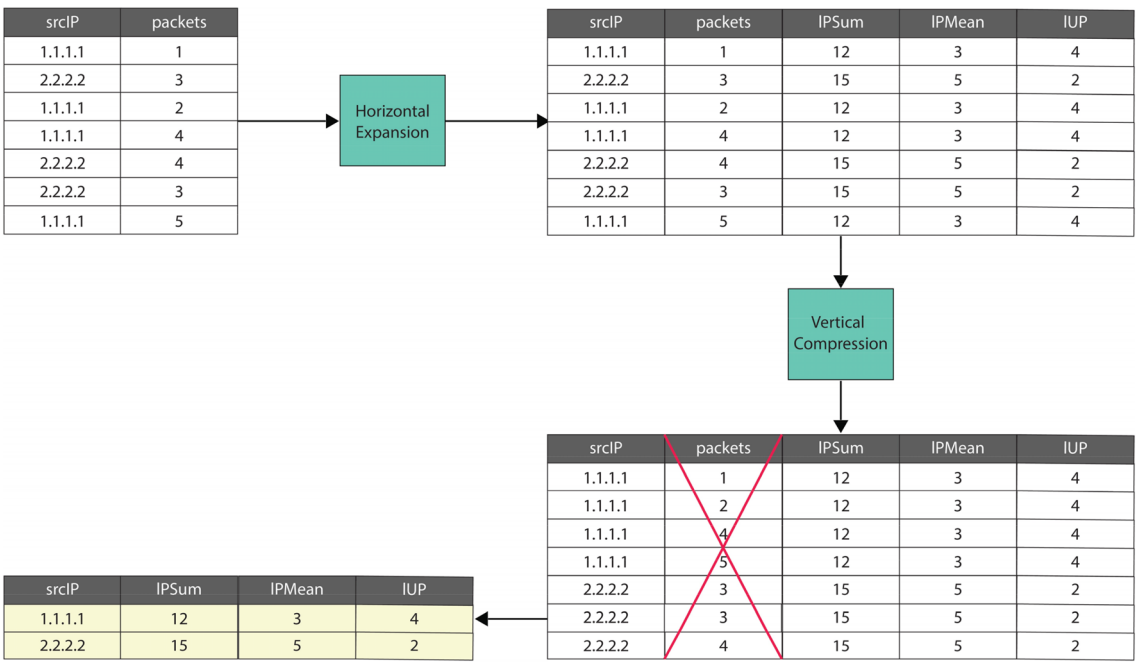


Fig. 6 A simple example of applying the HEVC engine to extract additional features, group and aggregate the data based on unique source IP addresses, and as such, each row represents a NetFlow for a source IP address

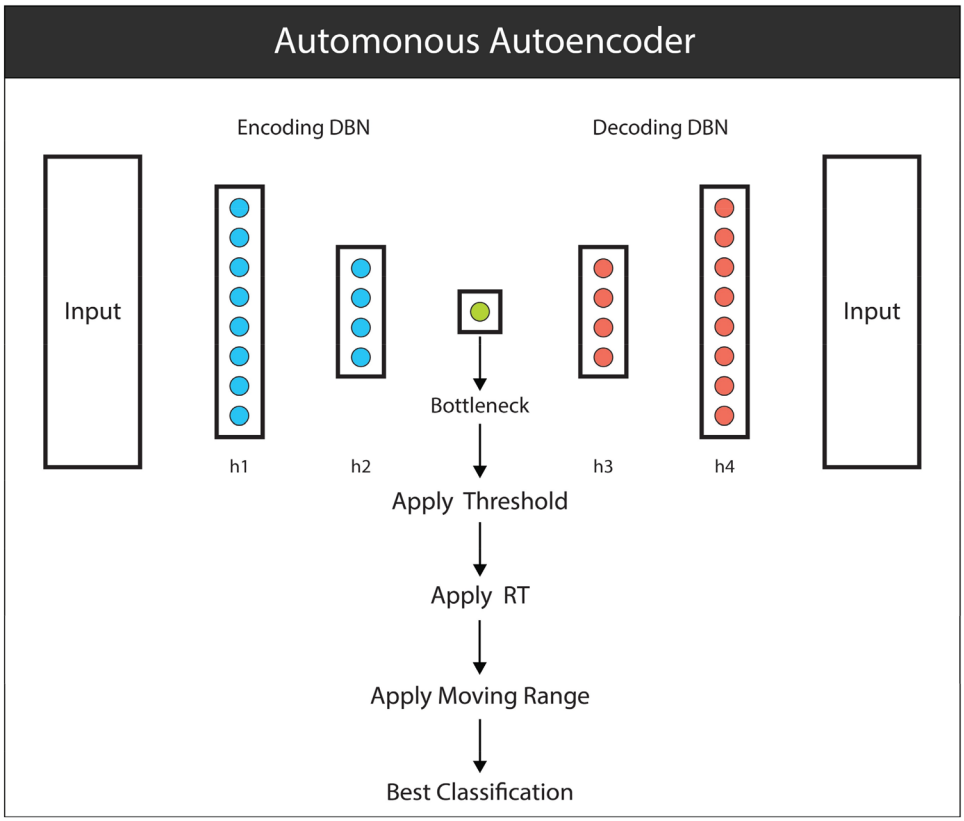


Fig. 7 The AA contains two symmetrical Deep-Belief Networks and it applies the optimal threshold, RT, and the moving range to obtain the best classification

should be larger than the threshold. On the contrary, the number of different source ports should be smaller than the threshold. Therefore, the N_f for each feature belongs to the minority group. The process of the AA obtaining the best classification follows the steps listed below:

Step 1: Transform the data with selected features.

Step 2: Create the Deep Autoencoder.

Step 3: Generate the bottleneck using the Autoencoder from Step 2.

Step 4: Classify values in the bottleneck using a threshold and swap the class if the $N_f > \beta$ the total IP addresses (TIPs).

Step 5: Repeat Step 4 until the end of the threshold list.

Step 6: Find the optimal threshold according to Eq. (1).

Step 7: Find the best classification associated with the optimal threshold. Step 8: Set the RT = N_f if the N_f is not within the moving range of [RT$\times$ $(1 - s)$, RT $\times$ $(1 + s)$].

Step 9: Repeat Step 2 to Step 8 until the N_f is within the moving range and then, return the best classification.

$$error = |N_f - RT| \tag{1}$$

The function of the AA, as shown in Fig. 8a, contains several parameters such as the data, the features, the thresholds. The *thresholds* parameter is a list containing different thresholds with a range of (0.2, 0.8) with a step of 0.05, which enables the AA to automatically find the optimal threshold for the best classification with the N_f that is closest to the RT. The RT has a default value of 100. β is used to control the class switch, and the default value is 0.5. The s provides a moving range (MR) = [RT $\times$ $(1 - s)$, RT $\times$ $(1 + s)$] to constrain the movement of the N_f so that it becomes closer to the RT. Lastly, the da settings contains the epochs, the optimiser, and the batch size.

3.3 Feature selector

The first unit of the RAA is the Feature Selector, and it stops the iteration of the AA when the number of recurrences for each feature is completed. According to our previous research, different types of attacks may require different features to achieve a good performance of classifying the network traffic [30]. Therefore, we enable the Feature Selector to dynamically select features. Based on our previous research [23, 29, 32], removing features with low influence improves the performance of the model. Therefore, we designed the Feature Selector to filter out features that identify none or very few normal IP addresses, so the NetFlow Identifier can better learn the correlations among remaining features yielding higher classification accuracy. Consequently, the Feature Selector is designed to find the Top-N features with the highest average number of normal IP addresses classified from several iterations. To achieve

(a)

```
def AA(data, features, thresholds, da_settings, RT=100, β=0.5, ε=0.2):
    RT = RT
    TIPs = data.shape[0]
    errors = [ ]
    Nfs = [ ]
    data = transform_data(features)
    model = create_model(data, da_settings)
    bottleneck = model.prediction(data)
    for threshold in thresholds:
        classification = [ 1 if b > threshold else 0 for b in bottleneck ]
        Nf = sum(classication)
        NR = Nf / TIPs
        if NR > β:
            classification = class_switch(classification)
            Nfs.append(classification)
            Nf = sum(classification)
            error = | Nf - RT |
            errors.append(error)
    best_classification = find_classification_with_min_error(errors, Nfs)
    Nf = sum(best_classification)
    if RT * (1 - ε) <= Nf <= RT * (1 + ε):
        return best_classification
    else:
        RT = Nf
```

(b)

```
def CMRW(cd, goal):
    RTs = [ ]
    Nfs = [ ]
    reucrrences += 1
    for rs in range(random_steps):
        classification = AA(data, features, thresholds, da_settings, RT, β, ε)
        Nfs.append(sum(classification))
    comparisons = [ 0 if nf < RT else 1 for nf in Nfs ]
    if sum(comparisons) <= cd
        Nfs = [ 0 if rt[0] == 1 else rt[1] for rt in zip(comparisons, Nfs) ]
    frequency = find_highest_frequency(Nfs)
    potential_RT = find_max_mode(Nfs)
    RT = potential RT
    if frequency > goal:
        return RT
    else:
        RTs.append(potential_RT)
        if (recurrance == 2) & (RTs[0] == RTs[1]):
            return RT
        else:
            frequency = find_highest_frequency(RTs)
            potential_RT = find_max_mode(RTs)
            if frequency >= goal:
                return RT
```

(c)

```
def feature_selector(data, Top_N, recurrences, thresholds, da_settings,
                     RT, β=0.5, ε=0.2):
    features = data.columns
    average_Nfs = [ ]
    for feature in features:
        Nf = 0
        for r in range(recurrences):
            Nf += sum(AA(data, feature, thresholds, da_settings, RT, β, ε)
        average_Nfs.append(Nf/recurrences)
    features = select_Top_N_features(average_Nfs, Top_N)
    return features
```

(d)

```
def target_finder(data, Max_N, random_steps, goal, features, thresholds, da_settings, RT, β=0.5,
                  ε=0.2):
    RT = RT
    cd= round(random_steps / 2) + 1 # determine if a conversion is needed
    recurrences = 0
    results = [ ]
    for n in range(Max_N):
        while Ture:
            results.append(CMRW(cd, goal)    )
    final_RT = find_RT_with_Max_N(results)
    return (final_RT, recurrences)
```

(e)

```
def netflow_Identifier(data, recurrences, features, thresholds, da_settings, RT, β=0.5, ε=0.2):
    recurrance = recurrance
    RT = RT
    Nfs = [ ]
    classifications = [ ]
    idx = 0
    for r in range(recurrences):
        classification = AA(data,  features, thresholds, da_settings, RT, β, ε)
        classifications.append(classification)
        Nfs.append(sum(classification))
    frequency = find_highest_frequency(Nfs)
    if frequency == 1:
        best_classification = max(Nfs)
    else:
        errors = [ | Nf - RT | for Nf in Nfs ]
        best_classification = find_classification_with_min_error(errors)
    return best_classification
```

(f)

```
def ensemble_N_module(data, restart, Top_N, Max_N, Ensemble_N, recurrences_f, recurrences_n,
                      random_steps, goal, features, thresholds, da_settings, RT, β=0.5, ε=0.2):
    classifications = [ ]
    results = [ ]
    RT = RT
    for e in range(Ensemble_N):
        features = feature_selector(data, Top_N, recurrences_f, thresholds, da_settings, RT, β, ε)
        while True:
            features = feature_selector(data, Top_N, recurrences, thresholds, da_settings, RT, β, ε)
            RT, recurrences = target_detecotr(data, Max_N, random_steps, goal, features, thresholds,
                                              da_settings, RT, β, ε)
            if recurrences < restart: # select new features and find a new RT
                break
        BC = netflow_Identifier(data, recurrences_n, features, thresholds, da_settings, RT, β, ε)
        classifications.append(BC)
    final_classification = majority_voting(classifications)
    return final_classification
```

Fig. 8 Functions for the AA, the CMRW, the Feature Selector, the Target Detector, the NetFlow Identifier, and the Ensemble-N Module

this, every feature is fitted to the AA individually several times, that is equal to the number of recurrences, as shown in step 3 listed below. The output of the Feature Selector is a list of selected features, and it is sent to the Target Detector. The function of the Feature Selector is presented in Fig. 8c and the Top-N argument chooses features with the highest N number of average N_f. However, if N = 0, any features that have the average $N_f > 0$ are selected. The following steps list the process of selecting features.

Step 1: Fit each feature in the data set to AA, count the number of predicted positives N_f for each recurrence.

Step 2: Repeat Step 1 through a number of recurrences, which is r.

Step 3: Calculate the average of the $Nf = \frac{\sum_{i=1}^{r} Nf(i)}{r}$

Step 4: Repeat Step 2 and Step 3 until the last feature.

Step 5: Select the number of features based on N specified for the top N average N_f; however, if N = 0, features with an average $N_f > 0$ are selected.

3.4 Target detector

The Target Detector utilises the Comparison-Max Random Walk to find the RT. The purpose of the CMRW is to find an RT that is close or equal to the number of actual positives by moving the potential RT towards the number of actual positives. There are a few key ideas involving the process of the CMRW, which are random steps, a moving range, a walking direction, and a goal as described in Table 3. The random steps are the number of classifications generated by the AA, so each random step is a N_f corresponding to a classification. The intuition behind the CMRW is to adjust the potential RT according to the moving range, the walking direction, and eventually, the potential RT converges to a goal that reaches the arrival frequency. The walking direction is determined by a higher or lower likelihood according to the random steps. If over 50% of the random steps are higher than the current potential RT, the walking direction is towards the higher direction, and the random steps remain the same. However, if the RT is walking towards the lower direction, any random step that is greater than the current potential RT is converted to 0. Afterwards, three cases are provided for the CMRW to find the RT.

Case 1: After the first recurrence, which is the completion of the first set of random steps, if the frequency of a maximum mode of the random steps or $N_f \geq$ Goal, which is the frequency required to qualify as an RT among random steps, as shown in Fig. 9, the RT convergences to the mode of N_f.

Case 2: After two recurrences, which is the completion of two sets of random steps, if the potential RT for both recurrences are the same, the potential RT convergences to the N_f.

Case 3: After several recurrences, if the frequency of a maximum potential RT reaches the Goal, the RT convergences to the maximum potential RT.

Take the following steps to find an RT:

Step 1: Generate a classification and record the number of predicted positives N_f generated by the AA in the Nfs list.

Step 2: Repeat Step 1 with the number of times equal to the number of random steps.

Step 3: Compare each N_f with the RT, if the $N_f >$ RT, record 1, else record 0 in the list of comparisons.

Step 4: If more than half of the N_f are smaller than the RT, convert the N_f that is greater than RT to 0 because the RT needs to move towards a smaller direction; otherwise, all the N_f remain the same.

Step 5: If there is a mode of the N_f that has a frequency Goal; the new found RT is the mode; otherwise record the potential RT in the RTs list.

Step 6: If the potential RT does not converge after the first recurrence, select the mode of the Nfs or the maximum of Nfs, if there is no mode, as a potential RT

Step 7: Repeat Step 1 to Step 4 and if the potential RT for both recurrences is the same, the potential RT is the new-found RT.

Step 8: Repeat Step 1 to Step 4 until the frequency of the potential RTs equals to the Goal.

Step 9: Repeat Step 1 to Step 8 for 'N' number of times depending on the 'N' value in Max-N.

Step 10: Determine the final RT based on the maximum mode or value based on the RT found by each Target Detector.

The function of the Target Detector is presented in Fig. 8b and the cd in the function determines if a conversion of a random step is needed to move the RT towards the lower direction. The Goal is the frequency of the maximum mode that needs to qualify an RT as a new found RT for a Target Detector. To aid the understanding of how the CMRW finds the RT, three cases are demonstrated in Fig. 9. Each value in the Random Steps list shown in Fig. 9, in all three cases, is a number of predicted positives. The values in the comparisons list depend on the potential RT, which is the default RT in the first iteration. If the value in the Random Steps list is greater than the potential RT, 1 is recorded, otherwise 0 is recorded in the comparisons list. If the sum of the comparisons list is greater than half of the length of the Random Steps list, it indicates the real RT should be higher than the

Table 3 Descriptions of variables for CMRW

Variables or Names	Descriptions
Random Steps	A collection of N_f generated by the AA for the CMRW to find an RT or a potential RT
S	A value in the range of (0.1, 0.25) that is used to control the Moving Range
Cd	determining if a conversion of a random step is needed to move the RT towards the lower direction
Moving Range	The reference range that the AA is used to determine if the N_f is associated with a qualified classification
Goal	The frequency of the maximum mode of random steps that is used to qualify a random step as an RT or a potential RT
Restart	The number of recurrences needed to rerun the Target Detector

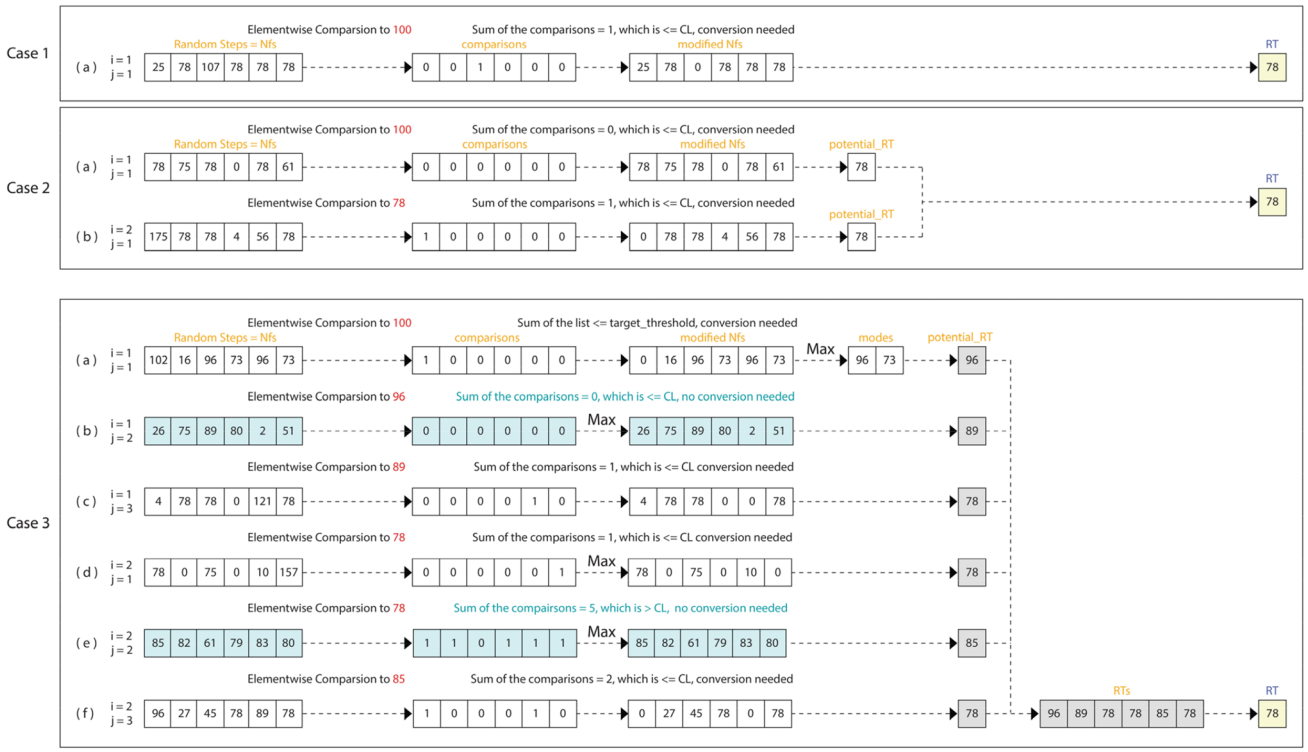


Fig. 9 The process of finding the RT utilising the CMRW algorithm according to three cases

potential RT, and all values in the Random Steps list remain the same. However, if the sum of the comparisons list is smaller than half of the length of the Random Steps list, it suggests that the actual RT is lower than the potential RT. Consequently, any value in the Random Steps list that has the corresponding value of 1 in the comparisons list is converted to 0. The modified list is used to store the unchanged and changed values. To qualify as the RT, the frequency of the mode in the modified list is utilised. Case 1, if the frequency of the mode in the modified list is greater than 3, the mode is the RT. Case 2, if the mode of the modified list is 3 for the first iteration, the potential RT is the mode and the second iteration is needed. If the frequency of the mode is also 3 in the second iteration is the same value as the first iteration, the potential RT is the RT. Case 3 illustrates that if the frequency of the mode of the modified list is less than 3, the largest value is the potential RT and is recorded in the RTs list. Moreover, another integration is required until the frequency of the mode of the RTs list is 3 and then, the mode is the RT.

For example, in case 1, the first potential RT is 100, and 107 is greater than 100, so 1 is recorded in the comparisons list. Since all the other values in the Random Steps list are smaller than 100, so zeros are recorded in the comparisons list. The sum of the comparisons list is 1, so 107 is converted to 0 and stores in the index of 2 position in the modified list. 25 and 78 are placed in the corresponding index positions in the modified list. Therefore, the Random Steps list of [25, 78, 107, 78, 78, 78] becomes [25, 78, 0, 78, 78, 78]. Since the frequency of the mode 78 is 4, which is greater than 3, the RT is 78. In case 2, all values in [78, 75, 78, 0, 78, 61] are smaller than the default RT of 100, so the comparisons list is [0, 0, 0, 0, 0, 0] and the sum of the comparisons list is 0. Therefore, no conversion of the Random Steps list is needed, so the modified list is [78, 75, 78, 0, 78, 61]. Since the frequency of the mode 78 is 3, 78 becomes the potential RT for the second iteration. Only one value in the Random Steps list of [175, 78, 78, 4, 56, 78] is greater than 78, the comparison list is [1, 0, 0, 0, 0, 0] and the sum of the comparison list is 1. Thus, only 175 is converted to 0 and the modified list becomes [0, 78, 78, 4, 56, 78]. The frequency of the mode 78 is 3 and the potential RT from the first iteration is also 78, so the RT is 78. In case 3, the Random Steps list for the first iteration is [102, 16, 96, 73, 96, 73] and the default RT is 100, so only 102 is greater than 100. Therefore, the comparisons list is [1, 0, 0, 0, 0, 0] and the sum of the comparisons list is 1. Consequently, 102 is converted to 0 and the modified list becomes [0, 16, 96, 73, 96, 73]. Since there are two modes 73 and 96, the mode with the largest value 96 is the potential RT. Next, each value in the Random Steps list of [26, 75, 89, 80, 2, 51] is compared to the potential RT of 96 and the comparisons list is [0, 0, 0, 0, 0, 0], so the modified list changes to [26, 75, 89, 80, 2, 51]. Since there is no mode, the largest value 89 is the potential RT.

The potential RT in the third, fourth, fifth, and sixth iteration is 78, 78, 85, and 78, respectively. Therefore, the RTs list is [96, 89, 78, 78, 85, 78], so the RT is 78.

3.5 NetFlow identifier

The last module of the RAA system is the NetFlow Identifier, which also utilises the AA. The NetFlow Identifier uses the selected features and the RT to classify the transformed data and then, find the best classification (BC) for an RAA with the minimum error. The function of the NetFlow Identifier is presented in Fig. 8e and the recurrences argument determines the number of times to run the NetFlow Identifier.

Step 1: Use the final RT found by the Target Detector as the RT for the AA and record the N_f.

Step 2: Repeat Step 1 until the end of recurrences.

Step 3: Select the best classification BC by finding the minimum error based on Eq. (1).

3.6 Ensemble-N module

The Ensemble-N Module employs N RAAs, which is determined by the Ensemble N argument passed into the function in Fig. 8f. If the recurrence of finding the RT is greater than the 'Restart' argument, the Ensemble-N Module reruns the Feature Selector and the Target Finder. After each RAA creates the best classification (BC), the Ensemble-N Module utilises a majority voting scheme to finish labelling the data set. Finally, the list containing malicious IP addresses is provided to the ISP's blackholing mechanism to be dropped. Figure 3 demonstrates the process of obtaining the final classification using Top-N, Max-N, minimum error, and Ensemble-N.

4 Experimental implementation

The testbed of the RAA system utilised the Apache Spark framework running on Ubuntu 16.04 with Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz with 32 GB of RAM. The Apache Spark is a distributed computing engine which facilitates horizontal scalability to cope with big data. We tested the RAA system with 7 labelled data sets, and a real DDoS attack data set. The details of labelled data sets are under the 'Data Information' section below.

Relu activation was used for h1, h2, h3, and h4 to deal with the problem of the vanishing gradient, and sigmoid activation is applied to generate the output values in the bottleneck. The loss function chosen was the root mean squared error, and the batch size was 128. The optimiser chosen was Adam because it is an adaptive learning rate optimisation algorithm [33]. Additionally, the Goal for the Target Detector is 3, and the Restart for the Ensemble-Module is 30. The number of epochs for the Deep Autoencoder were not the same in different modules which are presented in Table 4.

Epochs are the number of times to run the Deep Autoencoder inside the AA. The number of epochs, the threshold and iterations for the Deep Autoencoder varied in different modules which is presented in Table 4. It is important to know that both the iterations and the recurrences control the number of times to run the AA. Additionally, the default settings for the RT = 100, the $\beta = 0.5$ and the $s = 0.2$. Moreover, the N value for the Top-N, Max-N and Ensemble-N is listed in Table 5. Because 'N' for Top-N is 0, any features with the average $N_f > 0$ is selected. Since the 'N' for the

Table 5 The N value for the feature selector, the target detector and the ensemble-N Module

Components	Name	N
Feature Selector	Top-N	0
Target Detector	Max-N	10
Ensemble-N Module	Ensemble-N	5

Table 4 The recurrences of each component in the RAA and the parameter settings for the AA

Components	Recurrences	AA Parameter Settings
3*Feature Selector	3*2	Epochs = 3
		Thresholds = [0.5]
		Iterations = the number of features
3*Target Detector	3*Dynamic	Epochs = 5
		Thresholds = [0.2, 0.8] with a step of 0.05
		Iterations = random steps = 6
3*NetFlow Identifier	3*10	Epochs = 10
		Thresholds [0.2, 0.8] with a step of 0.05
		Iterations = 1

Max-N is 10 and for the Ensemble-N Module is 5, they are called the Max-10 and Ensemble-5, respectively.

4.1 Data information

We selected three real users' as targets from a data set, which includes their network traffic, collected from the border gateway protocol (BGP) router by the ISP. The first, the second, and the third target receive packets from 78, 133, and 281 normal source IP addresses, respectively. These three real users' network traffic behaviours are not identical, as illustrated in Fig. 10. To generate malicious traffic, we utilised BoNeSi [34] to simulate TCP flood, UDP flood, and ICMP SYN flood attacks with 50000 spoofed IP addresses. Then, we combined each target's real network traffic with simulated malicious traffic to create different data sets. All data sets contain 50000 spoofed IP addresses as displayed in Table 6. Figure 11a is a scatter plot of the attack data set containing both the normal and malicious network traffic, which are represented by green and red points, respectively. However, no red points are visible in (a), but the same red points are displayed in Fig. 11b. The reason that malicious data points can not be seen in plot (a) is that they have similar traffic patterns to normal data points. This is often the case for new DDoS attacks, and it reveals the level of difficulty for the system to identify the NetFlows correctly.

4.2 Estimated evaluation metrics for unsupervised learning model

Performance evaluation for unsupervised learning models without labelled data is a problem. To deal with this issue, we provide methods to calculate the estimated actual positives (EAP), estimated actual negatives (EAN), estimated true positives (ETP), the estimated false negatives (EFN), the estimated true negatives (ETN), and the estimated false positives (EFP) that can be used to calculate the estimated Recall (ERecall), the estimated Accuracy (EAccuracy), the estimated Precision, the estimated F1 Score and other Estimated Evaluation Metrics. Notations for estimated evaluation metrics are shown in Table 7. For example, to check the performance of a classification generated by the Ensemble-N Module, the number of classified normal NetFlows $N_{f(0)}$ generated by the Ensemble-N Module is deemed as the estimated true positives. An estimated confusion matrix is in presented in Table 8 for calculating other evaluation metrics. Take the following steps to check the performance of an unsupervised learning model utilising the EEM.

Step 1: Use the $N_{f(0)}$ as the RT to run an Ensemble-10 Module and as such, 10 RAAs are employed.

Step 2: Calculate the N_f for each RAA from Step 1 and record them in the list Nfs = $[N_{f(1)}, N_{f(2)} \ldots N_{f(10)}]$.

Step 3: Find the EAP by identifying the maximum mode of the Nfs or the maximum value of the Nfs, as shown in Eq. (2).

Table 6 The number of actual normal IP addresses and the total number of IP addresses in the processed data sets

Index	Data labels	Actual positives	TIPs
1	Whole UDP 78	78	50,078
2	Whole TCP 133	133	50,133
3	Whole ICMP 281	281	50,281
4	Whole UDP TCP 78	78	50,078
5	Whole UDP ICMP 133	133	50,133
6	Whole TCP ICMP 281	281	50,281
7	Whole UDP TCP ICMP 281	281	50,281

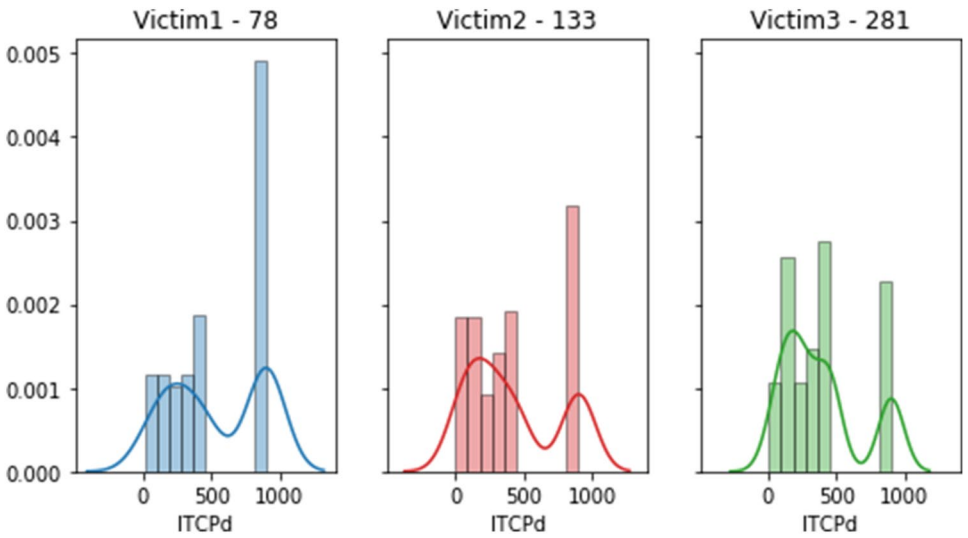
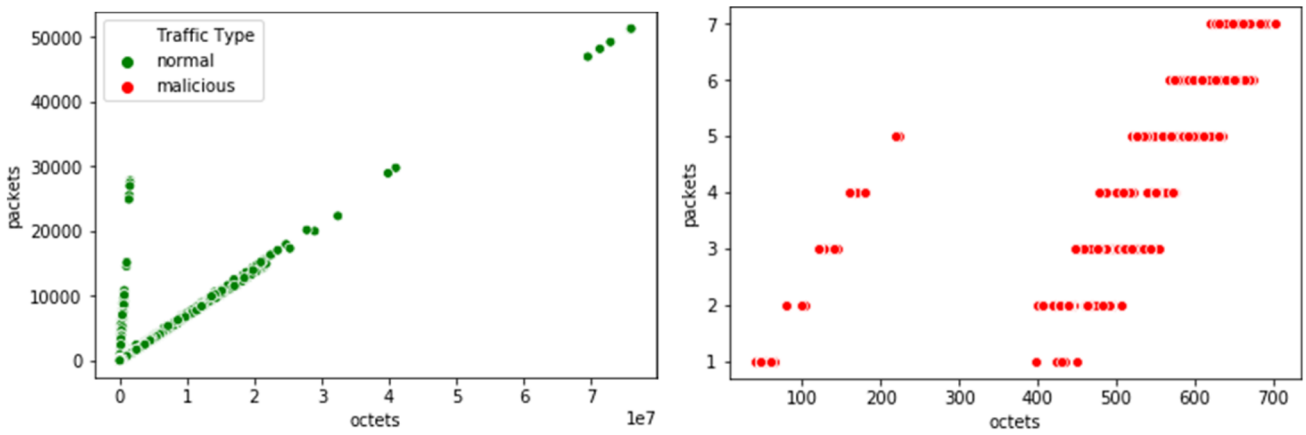


Fig. 10 Three victims' network traffic distribution, in which the first, the second, and the third victim have 78, 131, and 281 normal source IP addresses, respectively



(a) Normal and malicious NetFlows (b) Malicious NetFlows

Fig. 11 Normal and malicious octets vs. packets scatter plots

Table 7 Descriptions of variables for estimated evaluation metrics

Variables or Names	Descriptions
EAP	Estimated actual positives
EAN	Estimated actual negatives
ETP	Estimated true positives
ETN	Estimated true negatives
EFP	Estimated false positives
EFN	Estimated false negatives
ERecall	Estimated recall
EAccuracy	Estimated accuracy

Table 8 Estimated confusion matrix for calculating estimated evaluation metrics

	Estimated Actual Positives (EAP)	Estimated Actual Negatives (EAN)
Predicted positives	ETP = $Nf(0)$	EFP
Predicted negatives	EFN	ETN = TIPs—EAP

$$EAP = \begin{matrix} Modmax(Nfs) \\ max(Nfs) \quad \text{otherwise} \end{matrix} \tag{2}$$

Step 4: Calculate the estimated actual negatives (EAN) by subtracting EAP from the total number of IP addresses in the data (TIPs) as shown in Eq. (3).

$$EAN = TIPs - EAP \tag{3}$$

Step 5: Calculate the estimated false negatives (EFN) according to Eq. (4).

$$EFN = EAP - N_{f0} \tag{4}$$

Step 6: Calculate the estimated false positives (EFP) by Eq. (5).

$$EFP = EAN - ETN \tag{5}$$

Step 7: Calculate the estimated Recall (ERecall) based on Eq. (6).

$$ERecall = \frac{Nf(0)}{EAP} \tag{6}$$

Step 8: Calculate the estimated Accuracy (EAccuracy) as displayed in Eq. (7).

$$EAccuracy = \frac{\text{TIPs} - (\text{EFP} + \text{EFN})}{TIPs} \tag{7}$$

5 Results and findings

To illustrate the importance of feature selection, CMRW, Max-10, and Ensemble5 classification, we compare the performance without and with utilising feature selection, CMRW, Max-10, and Ensemble-5. Additionally, to demonstrate the scalability of the RAA system, we divided the data set whole UDP TCP ICMP 281 into 5, 10, and 20 subsets. Then, we tested the system using every subset for each division, and we checked the local and global performance, which are the results for an individual subset and the entire data set, respectively. Furthermore, we tested the IAM system on a real DDoS UDP flood attack data set. Moreover, we compare the Recall of the IAM with other supervised and unsupervised models.

5.1 Performance comparison with and without utilising comparison-max random walk

We did two experiments to demonstrate the importance of the CMRW. For the first experiment, we ran the Target Detector 100 times on a sample UDP TCP 281 data set with 281 normal IP addresses and 5000 malicious IP addresses, which contains 281 normal IP addresses. Without the CMRW, 0 occurred eighteen times, but 281 did not occur at all. The closest value to 281 was 202, which only occurred once. Consequently, RT = 0 because it is the mode. However, when we ran the Target Detector 100 times with the CMRW, the RT found was 281. In the second experiment, we tested the Target Detector ten times each without and with utilising CMRW. To get the potential target without the CMRW, we ran the Target Detector ten times to generate 10 RTs and each time the recurrences is 30. The reason that the recurrences is 30 is because the Target Detector typically takes less than 30 recurrences for the RT to converge. The results of the second experiment are shown in Table 9.

Table 9 The number of predicted positives or the potential RT without and with utilising CMRW for 10 trials

Trial #	Without CMRW	With CMRW
1	178	281
2	205	281
3	133	281
4	130	281
5	129	280
6	155	279
7	175	280
8	203	280
9	272	281
10	223	281

The number of predicted positives without the CMRW have no duplicated potential RT. However, with the CMRW, the mode of these 10 potential RTs is 281, so 281 is deemed as the RT, and it is equal to the actual positives. These two experiments demonstrate that the Comparison-Max Random Walk helps the potential RT to converge closer to the actual positives.

5.2 RT Comparison without and with Applying Feature Selection and Max-10 for the Target Detector

We divided the whole UDP TCP 78 into ten sub data sets and tested the Target Detector with each subset without and with applying feature selection and Max-10. The result is presented in Fig. 12. The x-axis is the number for the sub data set{1,2.....10} and the y-axis is the RT. Blue points represent correct RT for each distributed data, and as such, every point for the same sub data set should get as close to the blue point as possible. In this figure, 8 orange points are visible, which indicates that these points are incorrect. Furthermore, only three violet points can be seen. This suggests that the Target Detector performs better with feature selection than Max-10. The number of green points and violet points confirms that feature selection has a greater impact on finding the RT. However, the Target Detector has the highest accuracy when it utilises Max-10 and feature selection. Sub data set 6 is the only one that has a difference between the

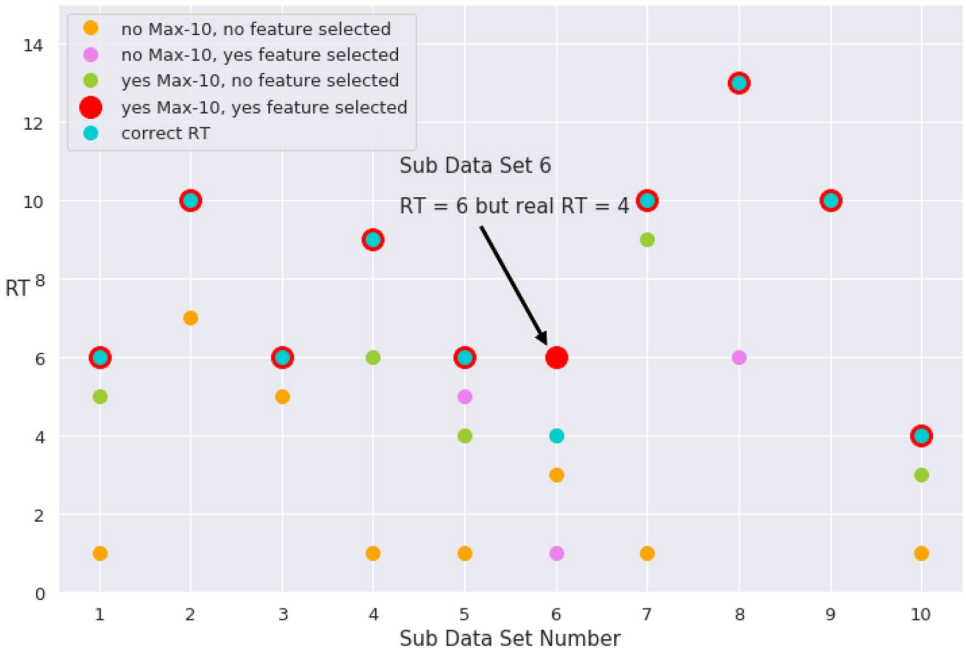


Fig. 12 RT found by the Target Detector with and without the feature selection and Max-10 for 10 sub sets of whole UDP TCP 78 data

red points and the blue points. The RT equals to 6, which is higher than the real RT 4. Nonetheless, from the perspective of the ISP, as long as the false positives are not too many, it is better to have fewer false positives than false negatives, which cause service interruption to the users.

5.3 Performance comparison without and with ensemble-5

We divided the whole UDP TCP ICMP 281 into 20 sub data sets and tested the system without and with utilising 5 RAAs in the Ensemble-N Module.

The results are displayed in Table 10. The predicted positives N_f with only one RAA is shown in the column of 'Without E-5', and with E-5 is shown under 'With E-5'. Without applying E-5, the system miss identified some normal NetFlows in the sub data set Nos. 1, 2, 6, 8, 17 and 18 by 1, 4, 1, 1, 4, and 2 points, respectively. By utilising E-5, the system correctly identified all normal NetFlows in sub data set Nos. 1, 2, 6, and 18. However, the system did not improve on sub data set No. 8, which indicates that both normal and malicious traffic is similar. One thing worth noting is that E-5 performed worse by 1 point on sub data set No. 14. This can be improved by using the maximum mode or value instead of the majority voting. Additionally, the system performed the same on sub data set No. 17 either without or with E-5. Even though the IAM system made mistakes on sub data set No. 8, 14 and 17, it reduced the total number of miss identified normal NetFlows for all 20 sub data sets from 13 to 3. Minimising the number of miss identified normal NetFlows is important because the number of miss identified normal NetFlows represents the number of legitimate users' service being interrupted. The results show that utilising E-5 reduces the miss classified NetFlows by 76.92%.

5.4 Performance comparison without and with distributed data sets for scalability

To check the scalability of the IAM system, we divided the whole UDP TCP ICMP 281 into 5, 10, and 20 subsets, and the evaluation metrics are displayed in Fig. 13.

The Recall and accuracy are very similar for different distribution sizes. However, when the data set is not subdivided, the result has the highest number of false negatives. There are more false negatives than false positives indicating that the system has a higher probability of miss identified normal NetFlows than malicious NetFlows. As such, some legitimate users' services are interrupted. As previously mentioned, this problem can be rectified by changing the aggregating method for attaining the final classification. Additionally, adding more RAAs to the Ensemble-N Module is another solution. However, the number of incorrectly classified NetFlows for distributed data sets is small, which indicates the scalability of the proposed system. During an attack, the majority of the NetFlows are malicious, and hence, we purposely divided the whole UDP TCP ICMP 281 into ten sub data sets, among which 3 data sets contain only malicious traffic. The performance of the system on each sub data set is shown in Table 11. 'Local Recall' is the Recall for the sub data set. Sub data set No. 2, 6, and 10 have only malicious NetFlows, and the IAM system correctly identifies sub data set No. 6 and 10. Even though the system miss identified 5 malicious NetFlows as normal NetFlows in sub data set No. 2, the local Recall is 1, so there is no service interruption. The global Recall for the whole UDP TCP ICMP 281 calculated from aggregating the result from all distributed data sets is 0.9964, and the global accuracy is 0.9999. Though, without distributing the data sets, the Recall and the accuracy are 1. Nonetheless, the time taken for the Target Detector and the NetFlow Identifier to get a best classification for an RAA is reduced by over 60% when data is divided into subsets, as shown in Fig. 14.

5.5 Estimated performance evaluation

The estimated performance for all 20 sub-datasets for the whole_UDP_TCP_ICMP_281 is displayed in Table 10. The results demonstrate that the estimated values are the same as the true values. For example, the actual positives are identical to the estimated actual positives EAP for all 20 subsets. It is the same for the estimated Recall, the estimated false positives and the estimated false negatives. The estimated global accuracy for the entire data set was calculated based on the Eq. (11), and the estimated accuracy is 99.9940%. The results suggest that it is reasonable to use the Estimated Evaluation Metrics to check the performance of an unsupervised learning model.

5.6 Performance on the real attack data set

We tested the proposed system with a real attack data set provided by the ISP. The DDoS attack took place in 2018 during an online game tournament. Initially, we set Top-N as dynamic for feature selection, and the system could not find the RT. Then, we specified Top-N = 5 to select 5 features with the highest average N_f, the system found the RT = 199, and the number of predicted positives was 197. We then use the selected feature and RT to run the NetFlow Identifier with recurrences = 10 to find the estimated actual positives. The result of the N_f for all 10 runs is {199, 193, 212, 179, 170, 194, 188, 172, 177, 199}. Consequently, the estimated actual positives is 199 because it is the mode and the ERecall = 1, EFP = 0, EFN = 2, and the EAccuracy = 0.9987.

Table 10 Performance comparison without and with utilising Ensemble-5, in which the EAP is the estimated actual positives, TRecall is the true Recall, ERecall is the estimated Recall, TFP is the true false positives, EFP is the estimated false positives, TFN is the true false negatives, and the EFN is the estimated false negatives

Sub Data #	Total IPs	Without E-5	With E-5	Actual	Recall	TFP	EFP	TFN	EFN		
1	2514	11	12		1	1	0	0	0	0	
2	2514	9	13		1	1	0	0	0	0	
3	2514	12	12		1	1	0	0	0	0	
4	2514	16	16		1	1	0	0	0	0	
5	2514	15	15		1	1	0	0	0	0	
6	2514	14	15		1	1	0	0	0	0	
7	2514	21	21		1	1	0	0	0	0	
8	2514	12	12		0.9231	0.9231	0	0	1	1	
9	2514	12	12		1	1	0	0	0	0	
10	2514	16	16		1	1	0	0	0	0	
11	2514	11	11		1	1	0	0	0	0	
12	2514	22	22		1	1	0	0	0	0	
13	2514	8	8		1	1	0	0	0	0	
14	2514	17	16		0.9412	0.9412	0	0	1	1	
15	2514	15	15		1	1	0	0	0	0	
16	2514	14	14		1	1	0	0	0	0	
17	2514	11	14		0.9333	0.9333	0	0	1	1	
18	2514	9	11		1	1	0	0	0	0	
19	2514	14	14		1	1	0	0	0	0	
20	2515	9	9		1	1	0	0	0	0	

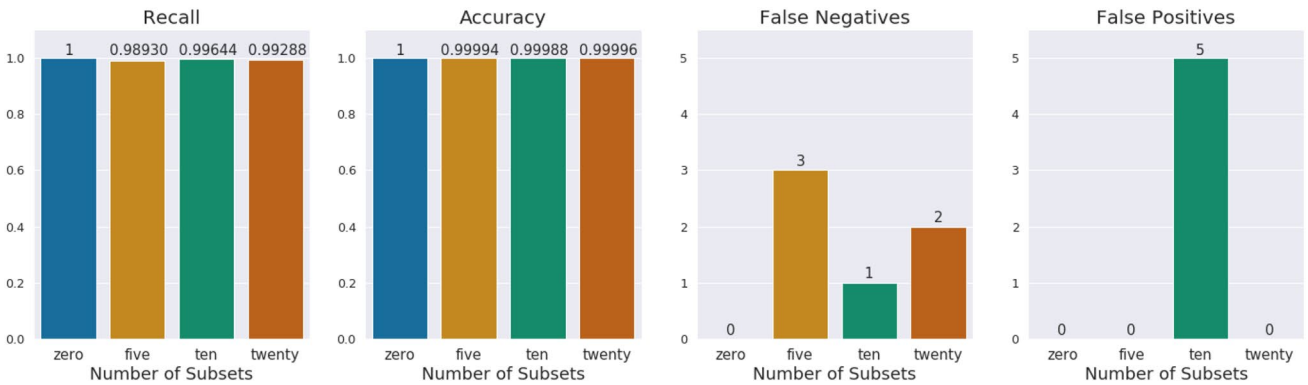


Fig. 13 Recall, false negatives, false positives, and accuracy comparison with different distribution size for the whole UDP TCP ICMP 281 data set, which includes 0, 5, 10 and 20 subsets

Table 11 Performance for 10 distributed data sets of whole UDP TCP ICMP 281 among which sub set No. 2, 6 and 10 contain 0 normal IP addresses

Sub Data Set #	Total IPs	Actual positives	*Nf*	Local Recall	Accuracy	False positives	False negatives
1	5028	53	52	0.9811	0.9998	0	1
2	5028	0	5	1	0.9990	5	0
3	5028	30	30	1	1	0	0
4	5028	34	34	1	1	0	0
5	5028	61	61	1	1	0	0
6	5028	0	0	1	1	0	0
7	5028	25	25	1	1	0	0
8	5028	29	29	1	1	0	0
9	5028	49	49	1	1	0	0
10	5029	0	0	1	1	0	0

5.7 Supervised and unsupervised model comparison

We also compared the Recall of the proposed model with other supervised and unsupervised models on a UDP flood attack data set. To make the comparison between the supervised and the unsupervised model more meaningful, the supervised models were trained with only 5% of the

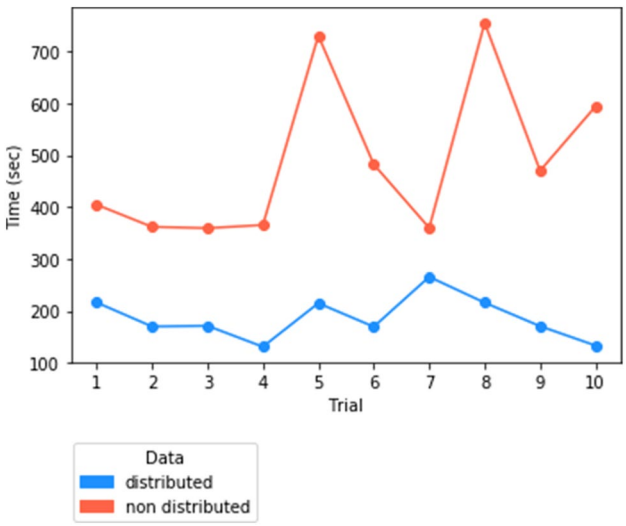


Fig. 14 Time taken for the Target Detector and the NetFlow Identifier to get a best classification for an RAA with and without distributed data sets

labelled data. The result is displayed in Fig. 15. Amongst all models, the KNN has the lowest Recall of 0, and the IAM has the highest Recall of 1. We used a threshold of 0.5 to classify the bottleneck of the Autoencoder, and the Recall is 14.1% lower than the proposed model. The SRDG is a hybrid model that uses two-layers of SOMs to classify the data to train an ensemble module, which includes a Decision Trees, a Random Forests, and a Gradient Boosted Trees (GBT). Even though the Recall of the SRDG is 0.9618, the IAM system outperforms the SRDG by 3.82%. One thing worth mentioning is that the Dual-SOM, the SRDG and the IAM are all target-driven models that utilise dynamic feature selection and threshold moving, and they have higher Recall compared to other models. However, the number of potential thresholds for selecting the optimal threshold for the Dual-SOM and SRDG is much smaller than the IAM. The potential thresholds are [0.6, 0.7] with a step of 0.1 for Dual-SOM and SRDG and [0.2, 0.8] with a step of 0.05 for the IAM. This demonstrates the importance of the number of potential thresholds for finding an optimal threshold. Even though the IAM requires the most computing power amongst all models, the Top-N, Max-N and Ensemble-N are the elements that can be adjusted to deal with the performance and resources trade-off.

5.8 Additional findings

There are five significant findings in this research. Firstly, feature selection affects the speed of finding the RT and the correctness of the final classification. Better feature selection reduces the number of recurrences required for finding the RT as shown in Fig. 16a with three recurrences, which means that the potential RT for all three recurrences are identical. On the contrary, feature selection for Fig. 16g is not as good because it took 20 recurrences to find the RT. Figure 16a and g indicate that this data set for victim 1 has higher separability among features. Therefore, the RT is correctly found even with a feature set that is less than ideal. However, if the features selected were poor, the AA took more than 30 recurrences to find the RT, which was normally very different from the actual positives. For example, when we divided the whole UDP TCP ICMP 281 into 10 sub data sets, the system had trouble to correctly classify sub data set 5 with the first set of features. The first feature set for sub data set 5 was {gPSum, gUO, gUP, gUSPort, gOMean, gPMean, gPStd, lPSum, lOMean, lTCPd, lUO, lUP, gRecF, gResF}. Using this feature set, the RT found was 141, which was very different from the actual positives of 33. Then, the final classification result was 266 normal NetFlows. Even though the Recall was 1, the false positive was 233. Since the recurrence value was over 30, which suggested a poor feature selection. Therefore, the Feature Selector was run again to find a second set of features, which was{gOSum, gPSum, gProtSum, gUDIP, gUP, gUSPort, gTC, gTCPd, glTR, lOSum, lPSum, lOMean, lPMean,

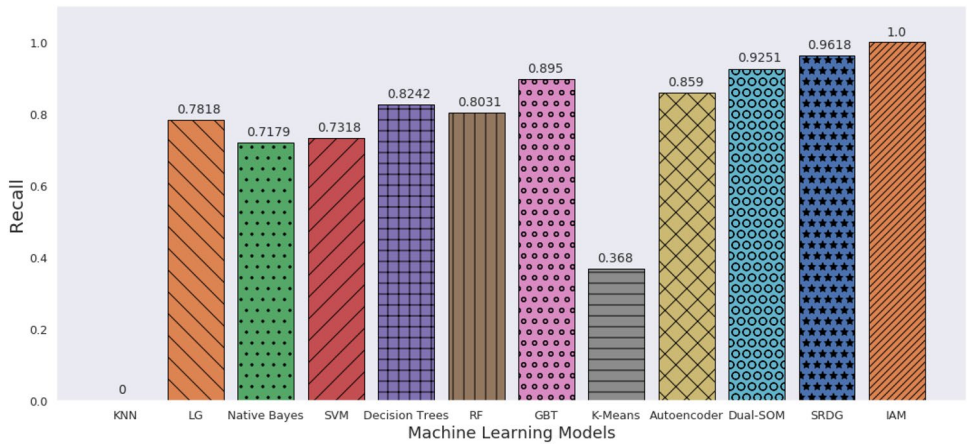
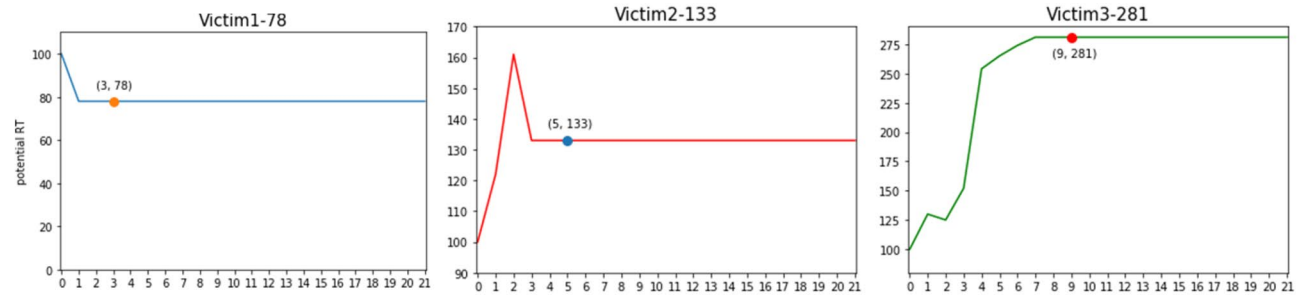
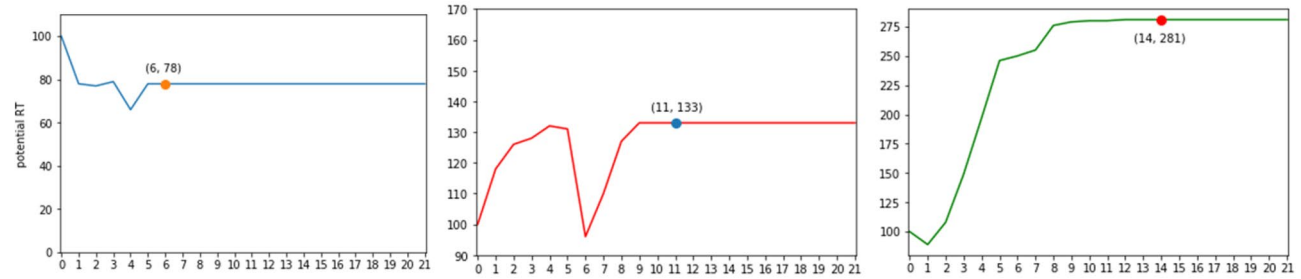


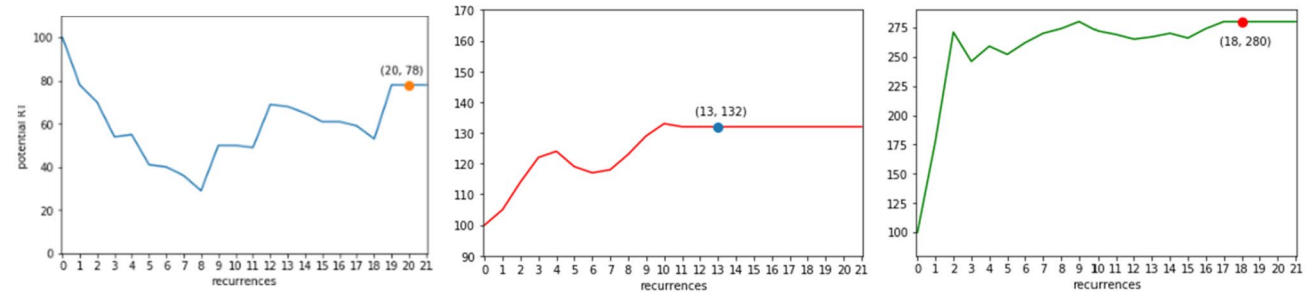
Fig. 15 Recall of supervised and other unsupervised learning models compared to the IAM system on a UDP flood attack data set, in which supervised models were trained with only 5% of labelled data



(a) Convergent at 3rd recurrence (b) Convergent at 5th recurrence (c) Convergent at 9th recurrence



(d) Convergent at 6th recurrence (e) Convergent at 11th recurrence (f) Convergent at 14th recurrence



(g) Convergent at 20th recurrence (h) Convergent at 13th recurrence (i) Convergent at 18th recurrence

Fig. 16 Comparison-Max-Random-Walk achieves convergence of the RT for all three victims

lUP, lUProt, lUSPort, lTCStd, lOMeanSend, lOMeanDiff, lResF, gRecF, gU3, gU5, lU3}. This time, the RT found was 32, which was very close to 33. Then, the system used RT =32 for the Ensemble-5 Module and the final N_f was 33, which is the same as the actual positives for sub data set 5. Each trial in Fig. 16 is a number of predicted positives or a potential RT generated by the Target Finder. The random weight initialisation for the Autoencoder caused the number of predicted positives to be different. Consequently, different features were selected, so the Target Detector generated different values for the potential RT. The Target Detector stops iterating when the potential RT converges. The converging process was stated earlier in Section 7.3.4. The horizontal line after the dot on each plot is to make them look uniform. For example, in Fig. 16a, the Target Detector stopped at the third iteration and (b) stopped at the fifth iteration. Moreover, Fig. 16a, d, g demonstrate that the system can achieve the same performance with different feature sets, which greatly alleviates the problem of trying to figure out which features are important.

Secondly, there is a close relationship between the RT and the performance of the system. We tested the system 100 times with the RT equal to the actual positives. The probability of achieving an accuracy of 1 is 97%. Additionally, we tested the system with the RT that was close to the actual positives 100 times. The probability of having the false negatives = 0 and only a few false positives was about 98%, which indicates that even if the RT is not identical to the actual positives, no users' services will be interrupted. Additionally, the fewer recurrences taken for the Target Detector to find the RT, the closer the RT is to the actual positives as shown in Fig. 16 with Victim2-133 and Victim3-281.

Thirdly, the 10 N_f s generated by every RAA of an Ensemble-10 Module with the RT = $N_{f(0)}$ provides a good measure to find the estimated actual positives to calculate Estimated Evaluation Metrics. When we divided the whole UDP TCP ICMP 281 into ten subsets, the system had trouble with the subset No. 5. Initially, the $N_{f(0)}$ was 141, when we used 141 as the RT to run each RAA on sub data set in the Ensemble-10 Module, the N_f generated each time was different as displayed in Table 12. As such, we could not determine the estimated actual positives, which suggests the performance was poor. However, when we used a new RT of 32, and the mode was 33, as shown in Table 12. The estimated actual positives were 33, which was the same as the actual positives.

Fourthly, the performance of the system depends on the distribution of the data sets, which affects the separability of the data points. The IAM system could easily find the RT that was the same as the actual positives when the standard deviation of the NetFlows was larger. For example, the system had a higher probability of correctly classifying whole UDP TCP 78, whole UDP ICMP 78, whole TCP ICMP 133 and whole UDP TCP ICMP 281 because the standard deviations of these data sets were higher. However, when we divided the whole UDP TCP ICMP 281 into five sub-datasets, the standard deviation for each subset was 9.5071, 9.3881, 9.9379, 9.3847 and 9.4437. Subset No. 2, 4 and 5 had lower standard deviations compared to subset No. 1 and 3. The system correctly identified all NetFlows for subset 3, and only had one false negative for subset 1. Therefore, the system selected new features for subset No. 2, 4 and 5. However, the IAM still could not find an appropriate RT for subset No. 2, 4 and 5 with new features selected. To cope with this problem, we further divided subset No. 2, 4 and 5 into two subsets each to change the distribution of the sub data sets. The distribution of 'gOSum' and 'gUO' for the subset No. 2 and the two subdivided sets of subset No.2, which include set2-subset-1 and set2-subset-2, are shown in Fig. 17.

Table 12 The N_f generated by each RAA of an Ensemble-10 Module using RT = 141 and 32 on a sub data set

Trial #	N_f when RT = 141	N_f when RT = 32
1	266	32
2	326	33
3	231	32
4	288	33
5	239	31
6	454	29
7	238	31
8	542	32
9	489	33
10	482	33

The green distributions were generated from sub data set No. 2. The red plots are generated from sub-data set2-subset-1, and the blue plots are created from sub-data set2-subset-2. Once the system finished classifying each subdivided data sets, the performance for the entire whole UDP TCP ICMP 281 was evaluated. The final global Recall was 0.9964, the global false negatives were 1, and the global false positives was 3. Moreover, after testing all data sets, the result shows that it was easier for the system to efficiently and correctly classify the NetFlows with a single-vector attack than a multi-vector attack. The standard deviation for single-vector attack data sets were all above 2*e*-9, but for multi-vector data sets were smaller by more than 50%.

Fifthly, throughout the experiments, the results show that using the RT to direct the unsupervised model with threshold-moving to classify the data makes the model less sensitive to feature selection. For example, the model was able to generate similar N_f with different features and thresholds. The first feature set includes 7 features, which are in the set of {'gUProt', 'gUDPort', 'lTC', 'lTCpd', 'lUProt', 'lTCMean', 'lTCStd'}, with the threshold = 0.3. The second feature set contains 13 features, which are in the set of {'gOSum', 'gPSum', 'gUDIP', 'gUProt', 'gTC', 'gOMean', 'gPStd', 'gOPPMean', 'glTR', 'lOSum', 'lOMean', 'lTCMean', 'lTCStd'}, with the threshold = 0.5. Both feature sets only have 3 features in common. With the ability to work with different feature sets by tuning the threshold, the target-driven technique greatly reduces the burden of feature engineering.

Finally, even though there is about 5% probability that the proposed system requires to be rerun the Ensemble-N Module, the system automatically detects the need by utilising the number of recurrences used to find an RT. If the system takes more than 30 recurrences to find an RT, it signifies that the RT is not reliable. Nonetheless, there are four ways to improve the performance of the IAM system. The first method is to select a new feature set. The second approach is to divide the data set into smaller data sets to change the traffic distribution. The third technique is to modify the aggregating method for attaining the final classification. The last

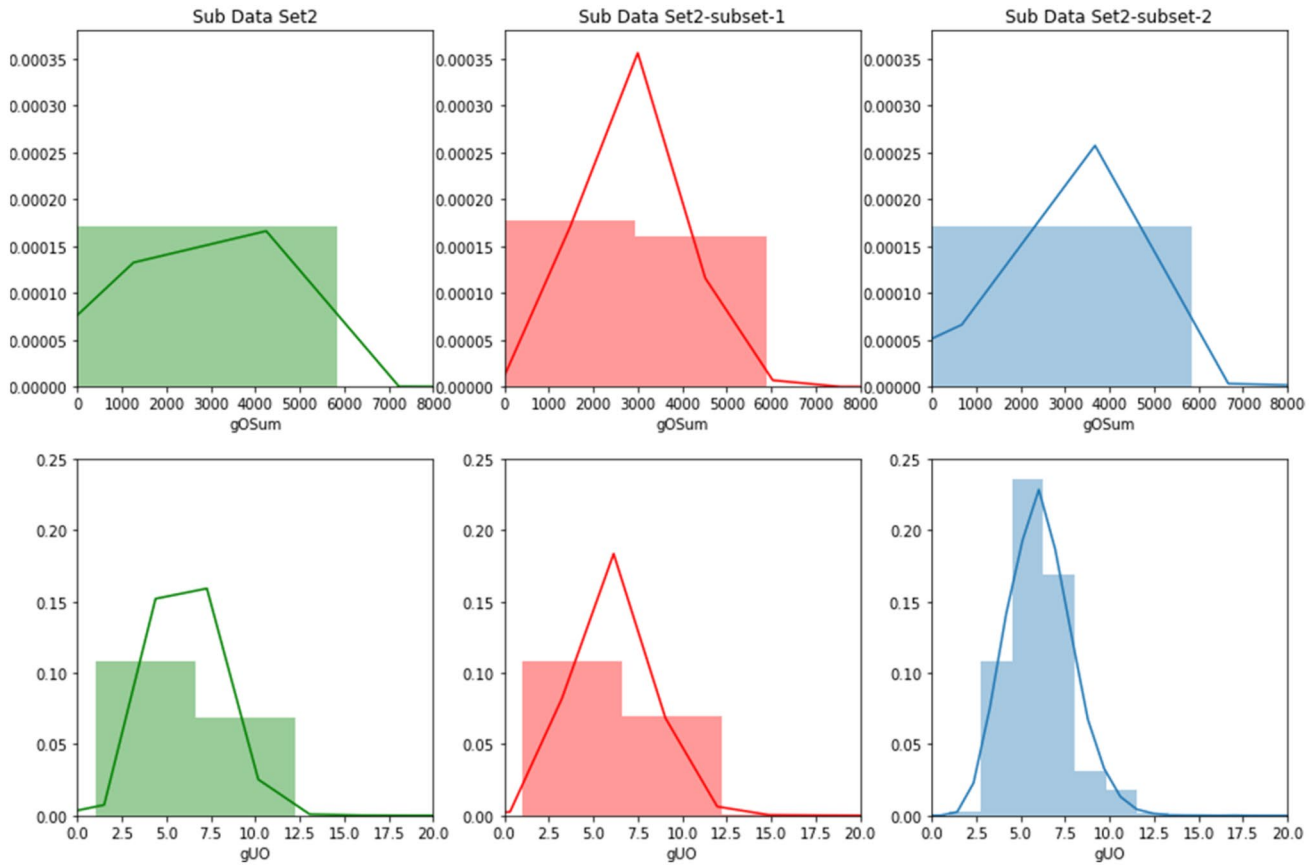


Fig. 17 NetFlow distribution of further subdividing the subset No.2 (Sub Data Set2) in green of whole UDP TCP ICMP 281 into two additional subsets of subset No. 2 into Sub Data-Set2-subset-1 in red and Sub Data Set2-subset-2 in blue

measure is to employ additional RAAs in the Ensemble-N Module.

6 Conclusion and future work

The advancement of DDoS technology escalates the level of difficulty for DDoS traffic identification, which can significantly reduce the effectiveness of the mitigation system. As 5G continues to take shape, a growing number of devices connecting to the network allows the attacker to increase the amount of malicious traffic towards the victim dramatically. The ISP already has blackholing mechanisms. Therefore, to deliver an effective DDoS mitigation system within the ISP domain, an efficient network traffic classifier is necessary. To combat an AI-based attack requires an AI-based approach, and thus, we propose the IAM system, which is a frame independent, ensemble, and distributed-enabled model that utilises Recurrent Autonomous Autoencoder. The proposed system can dynamically transform data, select features, determine the reference target, and identify attack traffic. The autonomous nature of the RAA enables the system to classify distributed data sets, which offers scalability to cope with a large amount of network traffic. Even though there is about 5% possibility that the Ensemble-N Module requires rerun, the need can be detected using the number of recurrences taken to find an RT. Moreover, there are different ways to improve the performance of the IAM system by further dividing the data set, selecting new features, or changing the aggregating method for attaining the final classification. The classification generated by the Ensemble-N Module for each distributed data set is independent. Therefore, malicious NetFlows found by each distributed system can be blocked right away to reduce delay. While the average Recall of the system was over 0.98, the amount of computing power and training time required could be high. Even though adjusting the N value for the Top-N, Max-N, and Ensemble-N can deal with the performance and resources trade-off, future work can focus on making the system more efficient.

Appendix A

See Table 13.

Table 13 Descriptions of extracted features

Features	Descriptions
gOSum	Global octets sum
gPSum	Global packets sum
gProtSum	Global protocol sum
gUSIP	Global unique source ip
gUDIP	Global unique destination ip
gSDIPR	Global source to destination ip ratio
gUO	Global unique octets value
gUP	Global unique packets value
gUProt	Global unique protocol value
gUSPort	Global unique source port value
gUDPort	Global unique destination port value
gTC	Global traffic count
gTCPd	Global traffic count period
gOMean	Global octets mean
gOStd	Global octets standard deviation
gPMean	Global packets mean
gPStd	Global packets standard deviation
gOPPMean	Global octets per packet mean
gOPPStd	Global octets per packet standard deviation
glTR	Global to local traffic count ratio
gRecF	Global receiving factor
gResF	Global response factor
gU3	Global unique sum of guprot, gusport, and gudport
gU5	Global unique sum of guo, gup, guprot, gusport, and sudport
lOSum	Local octets sum
lPSum	Local packets sum
lOMean	Local octets mean
lPMean	Local packets mean
lTC	Local traffic count
lTCPd	Local traffic count period
lUO	Local unique octets value
lUP	Local unique packets value
lUProt	Local unique protocol value
lUSPort	Local unique source port value
lUDPort	Local unique destination port value
lTCMean	Local traffic count mean
lTCStd	Local traffic count standard deviation
lCD	Local communication direction
lOMeanSend	Local octet mean send to other ip addresses
lOMeanDiff	Local octets mean difference between the source ip, and the destination ip
lODiffSign	Local received and sent octet difference
lRecF	Local receiving factor
lResF	Local response facto
lU3	Local unique sum of luprot, lusport, and ludport
lU5	Local unique sum of luo, lup, luprot, lusport, and ludport

Acknowledgements This research is funded by the Irish Research Council.

References

1. Bacon M (2019) DDoS attacks among top 5G security concerns. https://searchsecurity.techtarget.com/feature/DDoS-attacks-among-top-5G-security-concerns. Accessed:13 July 2019
2. How the Mirai botnet changed IoT security and DDoS defense. https://searchsecurity.techtarget.com/essentialguide/How-the-Mirai-botnet-changed-IoT-security-and-DDoS-defense. Accessed: 13 July 2019
3. Marinos L, Louren M (2019) ENISA Threat Landscape Report 2018. https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/enisa-threat-landscape Accessed: 20 Aug 2019
4. Shibu M (2019) Security concerns in a 5G era: are networks ready for massive DDoS attacks?. https://www.scmagazineuk.com/security-concerns-5g-era-networks-ready-massive-ddos-attacks/article/1584554 Accessed: 20 Aug 2019
5. Khalili J (2021) Volume of DDoS attacks continues to surge. https://www.itproportal.com/news/volume-of-ddos-attacks-continue-to-surge/. Accessed:7 Jan 2021
6. Osborne C (2019) The average DDoS attack cost for businesses rises to over $2.5 million. https://www.zdnet.com/article/the-average-ddos-attack-cost-for-businesses-rises-to-over-2-5m/ Accessed: 20 Aug 2019
7. Moore M (2019) DDoS attacks could cost the UK £1bn this year. https://www.techradar.com/news/ddos-attacks-could-cost-the-uk-pound1bn-this-year Accessed: 20 Aug 2019
8. Verisign Q2 2016 DDoS Trends:Layer 7 DDoS Attacks a Grwoing Trend. https://www.centr.org/news/news/com-verisign-q2-2016-ddos-trends-layer-7-ddos-attacks-a-growing-trend.html. Accessed: 21 Aug 2016
9. Niyaz Q, Sun W, Javaid AY (2017) A deep learning based DDoS detection system in software-defined networking (SDN). SESA, EAI, DOI: https://doi.org/10.4108/eai.28-122017.153515
10. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. ISSN 0893–6080. https://doi.org/10.1016/j.neunet.2014.09.003. http://www.sciencedirect.com/science/article/pii/S0893608014002135
11. Conran M (2021) The rise of artificial intelligence DDoS attacks. https://www.networkworld.com/article/3289108/the-rise-of-artificial-intelligence-ddos-attacks.html
12. Dietzel C, Feldmann A, King T (2016) Blackholing at IXPs: on the effectiveness of DDoS Mitigation in the Wild. PAM
13. Miller L, Pelsser C (2019) A taxonomy of attacks using BGP blackholing. In: Sako K, Schneider S, Ryan P (eds) Computer security – ESORICS 2019. ESORICS 2019. Lecture Notes in Computer Science, vol 11735. Springer, Cham
14. Giotsas V, Smaragdakis G, Dietzel C, Richter P, Feldmann A, Berger A (2017) Inferring BGP blackholing activity in the internet. In: Proceedings of the 2017 internet measurement conference (IMC '17). ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3131365.3131379
15. Yuan X, Li C, Li X (2017) DeepDefense: identifying DDoS attack via deep learning. In: IEEE international conference on smart computing (SMARTCOMP), Hong Kong, 2017, pp 1–8. https://doi.org/10.1109/SMARTCOMP.2017.7946998. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7946998&isnum-ber=7946954

16. Yadigar I, Fargana A (2018) Deep Learning Method for Denial of Service Attack Detection Based on Restricted Boltzmann Machine. https://doi.org/10.1089/big.2018.0023
17. Priyadarshini R, Barik RK (2019) A deep learning based intelligent framework to mitigate DDoS attack in fog environment, Journal of King Saud University Computer and Information Sciences, ISSN 1319–1578. https://doi.org/10.1016/j.jksuci.2019.04.010. http://www.sciencedirect.com/science/article/pii/S1319157818310140
18. Liu Y, Dong M, Ota K, Li J, Wu J (2018) Deep Reinforcement Learning based Smart Mitigation of DDoS Flooding in Software-Defined Networks, 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, 2018, pp. 1–6. doi: https://doi.org/10.1109/CAMAD.2018.8514971. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8514971&isnumber=8514932
19. Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E (2015) Deep learning applications and challenges in big data analytics. Journal of Big Data. https://doi.org/10.1186/s40537-014-0007-7
20. Jmj A (2019) 5 Industries that heavily rely on Artificial Intelligence and Machine Learning. https://medium.com/datadriveninvestor/5-industries-that-heavily-rely-on-artificial-intelligence-and-machine-learning-53610b6c1525 Accessed:August 22, 2019
21. Liu Y, Dong M, Ota K, Li J, Wu J (2018) Deep reinforcement learning based smart Mitigation of DDoS flooding in software-defined networks. 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, 2018, pp. 1–6. doi: https://doi.org/10.1109/CAMAD.2018.8514971. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8514971&isnumber=8514932
22. Gao J, Fan W, Han J On the Power of Ensemble: Supervised and Unsupervised Methods Reconciled. http://ews.uiuc.edu/jinggao3/sdm10ensemble.htm
23. Ko I, Chambers D, End B (2020) Adaptable feature-selecting and threshold-moving complete autoencoder for DDoS flood attack mitigation. J Inf Secur Appl 55: https://doi.org/10.1016/j.jisa.2020.102647
24. Bacon M (2019) Q2 2018 DDOS TRENDS REPORT: 52 PERCENT OF ATTACKS EMPLOYED MULTIPLE ATTACK TYPES. https://blog.verisign.com/security/ddosprotection/q2–2018-ddos-trends-report-52-percent-of-attacks-employed-multiple-attack-types/. Accessed: Sep 18, 2019
25. UNB Intrusion Detection Evaluation Dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html. Accessed August, 2016
26. Asad M, Asim M, Javed T, Beg M, Mujtaba H, Abbas S (2019) DeepDetect: Detection of Distributed Denial of Service Attacks Using Deep Learning. The Computer Journal, Comjnl SN 0010–4620, RD 8/22/2019. https://doi.org/10.1093/comjnl/bxz064
27. Salama MA, Eid HF, Ramadan RA, Darwish A, Hassanien AE (2011) Hybrid Intelligent Intrusion Detection Scheme. In: Gaspar-Cunha A, Takahashi R, Schaefer G, Costa L (eds) Soft Computing in Industrial Applications. Advances in Intelligent and Soft Computing, vol 96. Springer, Berlin, Heidelberg
28. Ko I, Desmond C, Enda B (2019) A Lightweight DDoS Attack Mitigation System within the ISP Domain Utilising Self-organizing Map Volume 2. https://doi.org/10.1007/978-3-030-02683714
29. Ko I, Desmond C, Enda B (2019) Unsupervised learning with hierarchical feature selection for DDoS mitigation within the ISP domain. ETRI J. https://doi.org/10.4218/etrij.2019-0109
30. Ko I, Desmond C, Enda B (2019) Feature dynamic deep learning approach for ddos mitigation within the isp domain. Int J Inf Secur. https://doi.org/10.1007/s10207-019-00453-y
31. Doriguzzi-Corin R, Millar S, Scott-Hayward S, Martinez-del-Rincon J, Siracusa D (2020) LUCID: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection. IEEE Transactions on Network and Service Management, pp 1–1. Crossref, Web
32. Ko I, Chambers D, Barrett E (2020) Self-supervised network traffic management for DDoS mitigation within the ISP domain. Future Generation Computer Systems, Volume 112, pp 524–533, ISSN 0167–739X. https://doi.org/10.1016/j.future.2020.06.002
33. Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization, CoRR abs/1412.6980. https://arxiv.org/pdf/1412.6980.pdf
34. Go M (2020) BoNeSi. https://github.com/Markus-Go/bonesi. Accessed:June 10, 2020