Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.039207
Article

Securing Cloud Computing from Flash Crowd Attack Using Ensemble Intrusion Detection System

Turke Althobaiti[1,2], **Yousef Sanjalawe**[3,*] **and Naeem Ramzan**[4]

[1]Department of Computer Science, Faculty of Science, Northern Border University (NBU), Arar, 73222, Saudi Arabia
[2]Remote Sensing Unit, Northern Border University (NBU), Arar, 73222, Saudi Arabia
[3]Deparment of Cybersecurity, American University of Madaba (AUM), Amman, 11821, Jordan
[4]School of Engineering and Computing, University of West of Scotland, Paisley, PA1 2BE, UK
*Corresponding Author: Yousef Sanjalawe. Email: y.sanjalawe@aum.edu.jo
Received: 14 January 2023; Accepted: 10 March 2023; Published: 26 May 2023

Abstract: Flash Crowd attacks are a form of Distributed Denial of Service (DDoS) attack that is becoming increasingly difficult to detect due to its ability to imitate normal user behavior in Cloud Computing (CC). Botnets are often used by attackers to perform a wide range of DDoS attacks. With advancements in technology, bots are now able to simulate DDoS attacks as flash crowd events, making them difficult to detect. When it comes to application layer DDoS attacks, the Flash Crowd attack that occurs during a Flash Event is viewed as the most intricate issue. This is mainly because it can imitate typical user behavior, leading to a substantial influx of requests that can overwhelm the server by consuming either its network bandwidth or resources. Therefore, identifying these types of attacks on web servers has become crucial, particularly in the CC. In this article, an efficient intrusion detection method is proposed based on White Shark Optimizer and ensemble classifier (Convolutional Neural Network (CNN) and LighGBM). Experiments were conducted using a CICIDS 2017 dataset to evaluate the performance of the proposed method in real-life situations. The proposed IDS achieved superior results, with 95.84% accuracy, 96.15% precision, 95.54% recall, and 95.84% F1 measure. Flash crowd attacks are challenging to detect, but the proposed IDS has proven its effectiveness in identifying such attacks in CC and holds potential for future improvement.

Keywords: Cloud computing; CNN; flash crowd attack; intrusion detection system; LightGBM; White Shark Optimizer

1 Introduction

Cloud Computing (CC) has enhanced the computational approaches that involve the use of virtualization. The literature presented various definitions of CC. Specifically, the National Institute of Standards and Technology (NIST) [1] described CC as "a virtualized pay-as-you-go computing model to assist the prevalent, efficient, and desired network access to a shared pool of customizable computing resources that could be distributed at a high rate with the lowest management action or service provider interaction”. Examples of these resources include storage, networks, servers, services, and applications. Although adversaries are aiming at numerous CC attributes, such as high demand and flexibility, the cloud environment is vulnerable to a diverse range of security threats [2]. Provided that the cloud offers on-demand usage to its provided services [3], attackers attempt to begin an organized Distributed Denial of Service (DDoS) attack on the CC servers that resemble an authorized flash crowd event, which could affect the availability of cloud services. This action enables the attacker to succeed in initiating the attack without being caught. The absence of services has a significant impact on the revenue and business of the service providers, considering the high possibility of transitioning to other service providers due to dissatisfaction regarding the Quality of Service (QoS) [4].

A flash crowd refers to a significant number of individuals who gather in one location for the same purpose in a short period. In computer networks, flash crowd denotes the increased website traffic within a relatively brief time. This situation occurs as a result of unique phenomena, which include breaking news and the delivery of a popular product. In some cases, a flash event happens when a well-known site is connected to a smaller site, which leads to a substantial rise in traffic identified as a flash-dot effect [5].

Flash events and flash-dot impact the server operation of websites and network infrastructure, considering that overcrowding at the network layer could prevent several user requests from reaching the server. The requests may arrive at the server after a significant delay due to requests for resending and packet loss. Certain web server configurations and descriptions are not capable of managing the number of flash event demands [6]. As a result, the users who attempt to access the website in a flash event would be dissatisfied due to the long wait or inability of the event to achieve the target. The severity of the phenomenon increases upon an attacker’s attempt to avoid the defense mechanism by imitating the traffic pattern of authorized users in a flash event [7]. As illustrated in Fig. 1, these categories of attacks occurring during flash events are described as flash crowd attacks.

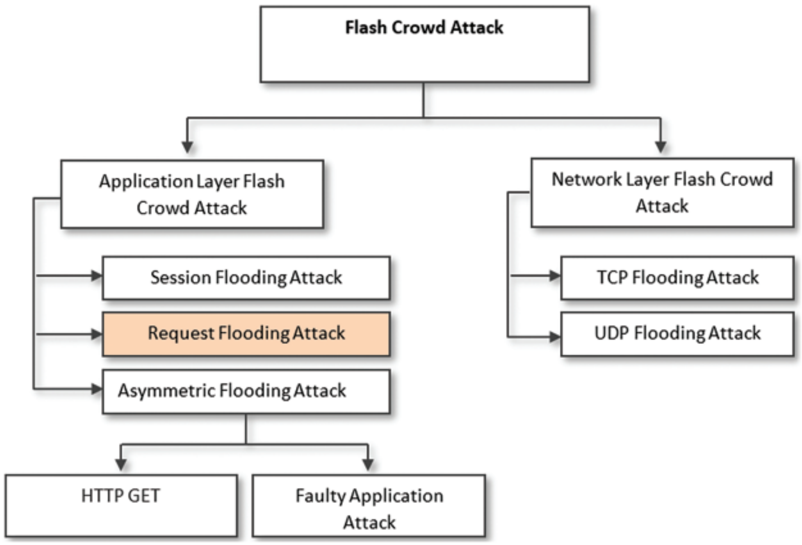

Flash Crowd Attack
Application Layer Flash Crowd Attack
Network Layer Flash Crowd Attack
Session Flooding Attack
Request Flooding Attack
Asymmetric Flooding Attack
TCP Flooding Attack
UDP Flooding Attack
HTTP GET
Faulty Application Attack

Figure 1: Classes of flash crowd attack

Detecting flash crowd attacks during flash events in CC is a primary challenge for web servers, as they need to differentiate between legitimate user requests for the event and malicious demands.

This differentiation is difficult to achieve using existing methods, which can result in delayed feedback to authorized users or even the entire web server crashing. To overcome this challenge, this article proposes a new approach to detect flash crowd attacks with greater accuracy. The contributions of this approach are:

- Enhancing the current body of literature concerning the detection of flash crowd in CC.
- Proposing an adapted version of the White Shark Optimizer (WSO) for selecting the most significant feature subset.
- Adaptation version of Principal Component Analysis (PCA) for reducing the dimensionality of selected features.
- Adaptation of LightGBM and Convolutional Neural Network (CNN) for the detection of flash crowd attacks with superior performance.

The other sections of this article begin with Section 2, which analyses the existing related works and highlights the research gaps. Section 3 discusses the approach proposed in this research in detail. Section 4 presents the findings and their discussion. This article ends with a conclusion in Section 5.

2 Related Works

CC assists the auto-scaling feature in scaling the resources on-demand in a dynamic manner. However, this attribute causes critical financial losses to customers when attacks take place on their purchased instances. Among the most critical and most employed attacks on the cloud are the TCP SYN EDOS attacks [8]. Notable initiatives have been performed in the previous decade as a defense against these attacks. Several simulation platforms were suggested for the measurement and analysis of the effect of EDOS attacks on CCE. To identify the traffic anomaly and prevent the emerging categories of DDoS attacks, various techniques are employed, including the artificial intelligence-based approach [9–11], statistical anomaly identification [12–14], machine learning-based approach [15,16], data mining approach [17–21], classifiers-based [22,23], hybrid anomaly detection [24,25], and signature-based detection [26,27]. Based on the comparative summary of all the methods presented in Table 1, machine learning anomaly identification is applied in this article.

Table 1: Comparison between ML-based anomaly detection

Approach	Efficiency	Adaptability	Overhead	Scalability	Overfitting
Statistical based	X				
AI-based	X	X		X	X
Data mining-based	X				
ML-based	X	X	X		
Classifier-based	X	X	X		
Signature-based	X		X	X	
Hybrid-based	X	X	X		

Anomaly detection using statistical methods is a useful technique for identifying unusual traffic patterns, especially in terms of resource and computation efficiency. This method involves comparing incoming network traffic statistics to normal network traffic patterns to identify any anomalies. Once an anomaly is detected, statistical inference tests are utilized to assess the reliability of the patterns [28]. However, this method is not considered "adaptive," as shown in Table 1. There have been efforts by researchers to develop methods that combine the efficiency of statistical anomaly detection with the adaptive nature of Software-Defined Networking (SDN). In the context of defending against DDoS attacks, a popular approach is the use of TCP SYN cookies [29], which can effectively block TCP SYN flooding attacks on a server. This technique can be implemented on a cloud instance to prevent such attacks while also reducing the financial cost of the instance due to resource usage. When a TCP SYN attack with a payload is used, the inbound traffic accepted by the instance may require a large amount of bandwidth. After accepting a large number of TCP SYN requests, the instance processes these packets to identify the attack, which may also consume instance resources and result in charges for resource usage for the client.

Gaurav et al. [30] proposed a method called EDOS-Shield to mitigate E-DoS attacks, which uses a virtual firewall that maintains lists of client IP addresses as either "whitelisted" or "blacklisted" based on their classification using a Graphic Turing Test (GTT). Clients who pass the GTT are included in the whitelist, while those who fail are added to the blacklist. However, this approach has the disadvantage of creating overhead, causing delays for legitimate users trying to access the CC. Shawahna et al. [31] proposed a reactive method called EDOS-Attack Defense Shell (ADS), which is designed to block NAT-based attackers by using the port number and IP address of the attacker device and blocking requests from that port number. The authors used a trust factor calculation based on the GTT to determine whether a request was an attack or not. EDOS-ADS can identify clients using their port number and IP address, and it also effectively handles IP spoofing to allow legitimate users to access services. However, there are several issues with this approach. One issue is that when the attacker starts a new request, the NAT router assigns a different port number to the attacker, allowing them to continue the attack from a different port. Another issue is that the GTT involves a different channel assignment for every request, causing the server to generate numerous puzzles for a high number of requests, which could accumulate the attack if the puzzle is not solved in time. The GTT feedback duration is 13.06 s, allowing illegal users to generate massive volume of requests from one source and use massive number of channels for the GTT. Additionally, URL redirection adds an overhead of 0.63 s.

Bawa et al. [32] introduced an IDS, called EDOSEMM, to detect EDOS attacks in a CC. The model consists of three main modules, one of which is the data preparation module, which processes and organizes the flows of incoming packets, which can create overhead. This module deals with both UDP and HTTP attack traffic as well as legitimate traffic. The model uses Hellinger distance and entropy approaches to accurately detect anomalies. A mitigation approach for SYN flooding was suggested by Mendonça et al. [33], which is based on SDN and is executed on a controller. This approach involves utilizing a threshold value to identify and prevent TCP SYN attackers. Once the controller detects that the number of SYN requests from a specific host has exceeded the threshold value, it automatically blacklists and blocks the host. However, this method is solely dependent on the threshold value, which may lead to the blocking of legitimate users due to network disturbances or other related factors.

In [34], a proposal was made to enhance security measures for the Industrial Internet of Things (IIoT) because of its decentralized architecture. The authors suggested a prediction model that utilizes Deep Learning (DL) and is based on sparse evolutionary training (SET) to forecast various types of cybersecurity attacks, including intrusion detection, data type probing, and DoS. The SET-based model that was proposed achieved high performance within a short timeframe (i.e., 2.29 ms). Furthermore, in a real scenario of IIoT security, the performance in terms of detection rate was enhanced by an average of 6.25% in comparison with state-of-the-art models. In addition, [35] explores the application of SDN in enhancing intelligent machine learning methodologies for IDS. The authors propose a new IDS called HFS-LGBM IDS for SDN that utilizes a hybrid Feature Selection (FS) algorithm to obtain the optimal subset of network traffic features and a LightGBM algorithm to detect attacks, aiming to address security concerns associated with SDN. Based on the experimental outcomes from the NSL-KDD benchmark dataset, the proposed system surpasses current methods in terms of accuracy, precision, recall, and F-measure. The authors emphasize the importance of having accurate, high-performing, and real-time systems to tackle the risks linked to SDN.

The paper [36] puts forward a DL-based IDS that can detect diverse kinds of attacks on IoT devices. According to the proposed IDS, it has demonstrated excellent effectiveness in identifying various types of attacks, with a detection accuracy of 93.74% for both simulated and real intrusions. The overall detection rate of this IDS is 93.21%, which is deemed satisfactory in terms of enhancing the security of IoT networks. On the other hand, [37] introduces a new FS technique that improves the performance of Deep Neural Network-based IDS. This approach prunes features based on their importance, which is derived from a fusion of statistical importance. The performance of this approach has been evaluated on various datasets and through statistical tests, providing evidence of its effectiveness. The proposed approach provided important contributions to the field of securing IoT and a novel technique to enhance performance and improve security against vulnerabilities and threats.

3 Proposed IDS

After a long while since the first "Full-Scale High-Rate Flooding" (HRF) DDoS attacks against the Internet, several types of attacks continue to comprise a malicious threat to different Internet-based environments. Further, the detection of attacks efficiently remains a significant challenging issue. In this section, an attempt to propose an efficient solution to detect the flash crowd attack in flash events in a cloud computing environment is discussed, and the detailed design of the corresponding IDS, as depicted in Fig. 2, is comprehensively presented. The proposed IDS contains six main stages, namely: (i) data preprocessing, (ii) feature selection, (iii) dimensionality reduction, (iv) hybrid classifier, (v) flash crowd detection, and (vi) performance evaluation.

3.1 Preprocessing

To develop a precise IDS model, several actions should be conducted before the data is included to train the model. Notably, pre-processing is crucial for developing an effective IDS method and reducing the computationally intensive processes. In this study, the following actions were conducted for data preparation:

3.1.1 Data Normalization

To compare the attributes, which had different ranges, the data was standardized using Z-score normalization (as shown in Eq. (1)) to transform it onto a different scale. This resulted in the standardized data having a standard deviation of 1 and a mean value of 0 [38].

$$n(x) = M(n(x))/\sigma(n(x)) \tag{1}$$

where M denotes the mean, and σ is the standard deviation of given values.

Normalization is a process applied to dataset samples in IDS to standardize them, making them more consistent and easier to analyze. This includes techniques such as scaling, centering, transforming, and removing outliers and missing data. This article presents both binary and multiclass categorizations. In the binary experiment, normal strings were given a binary value of 0, while all malicious packet was given a value of 1. Each attack in the multiple class categorization was assigned a distinct digit value.

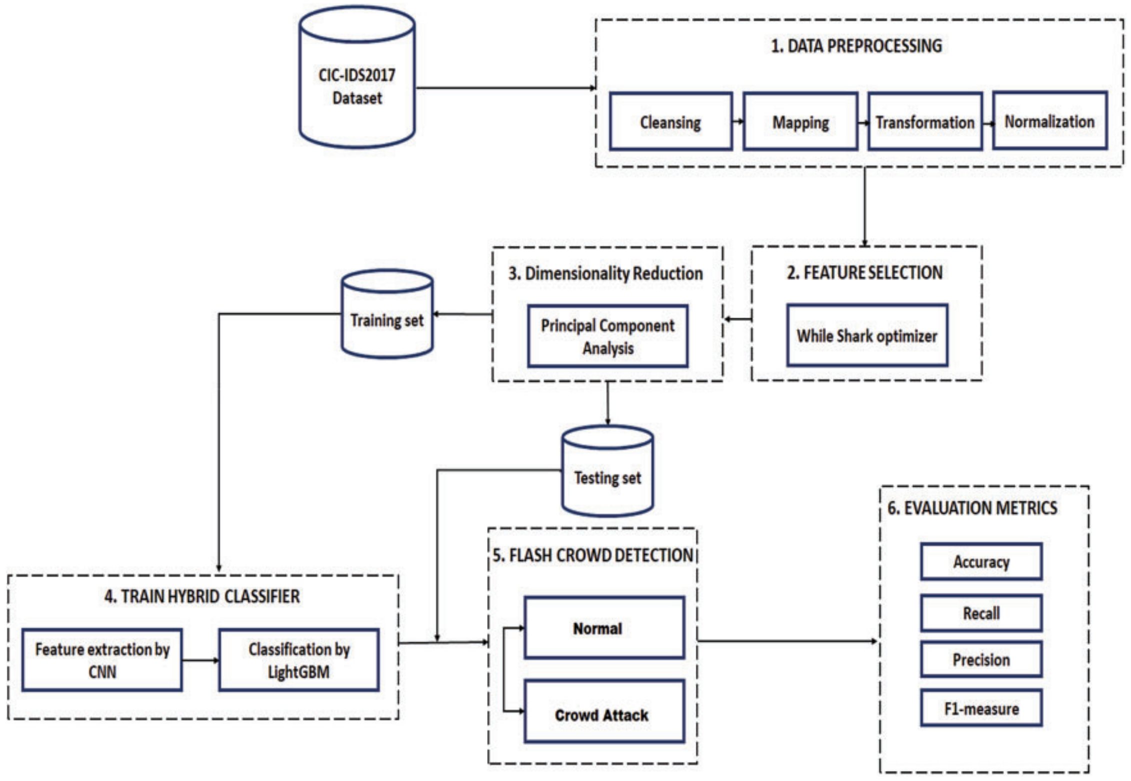


Figure 2: Architecture of proposed IDS

3.1.2 Data Reshaping

CNN requires input in the form of an image with 3-D: width, channel, and height. However, network traffic is in the form of 1-D dimension, which is not compliance with the architecture of CNN. Therefore, a transformation is necessary to convert the shape of the input packet to the resolution dimensions required by a CNN. For the subset of 48 attributes, the 48-D vector was converted into 8×6 images, while the 9-dimensional vector input was converted into 3×3 images. Since this article only uses grayscale images with a single channel, the channel number was set to 1.

3.2 White Shark Optimizer

The WSO is an algorithm that uses mathematical models based on the characteristics of great white sharks to solve optimization problems within a fixed search space [39]. It is a meta-heuristic algorithm that aims to balance the exploration process and exploitation process of the search space, using the search agents to find the best results. The process and pseudocode of WSO are illustrated in Figs. 3 and 4 respectively. The key concepts and foundations of WSO are inspired by the hunting behaviors of great white sharks, such as their highly developed senses of smell and hearing, which they use to locate and pursue their prey.

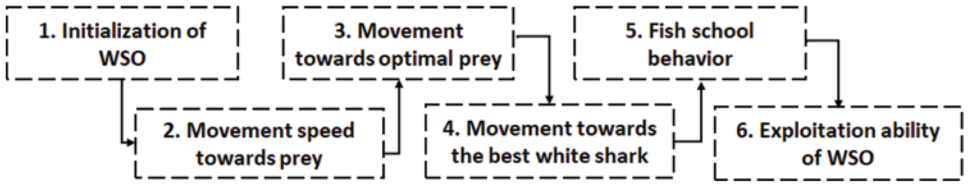

1. Initialization of WSO
2. Movement speed towards prey
3. Movement towards optimal prey
4. Movement towards the best white shark
5. Fish school behavior
6. Exploitation ability of WSO

Figure 3: Stages of SWO

Algorithm 1 A pseudo code summarizing the iterative optimization process of WSO.

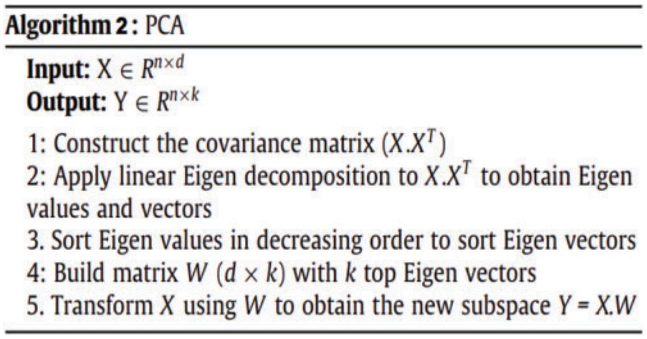
```
Initialize the parameters of the problem
Initialize the parameters of WSO
Randomly generate the initial positions of WSO
Initialize the velocity of the initial population
Evaluate the position of the initial population
while ($k < K$) do
  Update the parameters $\nu$, $p_1$, $p_2$, $\mu$, $a$, $b$, $w_o$, $f$, $mv$ and $s_s$
  for i=1 to $n$ do
    $v^i_{k+1} = \mu\left[v^i_k + p_1(w_{gbest_k} - w^i_k) \times c_1 + p_2(w^{v^i_k}_{best} - w^i_k) \times c_2\right]$
  end for
  for i=1 to $n$ do
    if $rand < mv$ then
      $w^i_{k+1} = w^i_k \cdot \neg \oplus w_o + u \cdot a + l \cdot b$
    else
      $w^i_{k+1} = w^i_k + v^i_k / f$
    end if
  end for
  for i=1 to $n$ do
    if $rand \leq s_s$ then
      $\vec{D}_w = \left|rand \times (w_{gbest_k} - w^i_k)\right|$
      if $i == 1$ then
        $w^i_{k+1} = w_{gbest_k} + r_1 \vec{D}_w \mathrm{sgn}(r_2 - 0.5)$
      else
        $\acute{w}^i_{k+1} = w_{gbest_k} + r_1 \vec{D}_w \mathrm{sgn}(r_2 - 0.5)$
        $w^i_{k+1} = \frac{w^i_k + \acute{w}^i_{k+1}}{2 \times rand}$
      end if
    end if
  end for
  Adjust the position of the white sharks that proceed beyond the boundary
  Evaluate and update the new positions
  $k = k + 1$
end while
Return the optimal solution obtained so far
```

Figure 4: Pseudocode of WSO [39]

Three characteristics of white sharks were adapted to locate their prey (e.g., the optimal food source). These characteristics are: (1) the movement towards prey based on the wave hesitation that occurs after the prey moves, which involves the white shark using its senses of smell and hearing to make an undulating movement towards the prey; (2) scavenging for prey in deep ocean areas, where the white shark navigates to the prey's location and gets close to the optimal prey; and (3) detecting the prey once it is within close-proximity, using fish school behavior to move towards the best white shark in the vicinity of the optimal prey. If the prey is not found, the location of each white shark would determine the optimal solution.

3.3 Principal Component Analysis

PCA is a technique used to decrease the number of dimensions in a dataset by identifying Principal Components (PCs), which are the directions that explain the highest variance in the data. It is a linear, unsupervised transformation technique that creates new features in a new subspace using orthogonal axes [40]. Fig. 5 demonstrates that the 1st PC has the largest variance, followed by lower variances for the subsequent PCs. The purpose of PCA is to retain as much information from the initial data as feasible while reducing its dimensionality.



```
Algorithm 2: PCA
Input: X ∈ $R^{n\times d}$
Output: Y ∈ $R^{n\times k}$
1: Construct the covariance matrix $(X.X^T)$
2: Apply linear Eigen decomposition to $X.X^T$ to obtain Eigen
values and vectors
3. Sort Eigen values in decreasing order to sort Eigen vectors
4: Build matrix $W$ $(d \times k)$ with $k$ top Eigen vectors
5. Transform $X$ using $W$ to obtain the new subspace $Y = X.W$
```

Figure 5: Pseudocode of PCA

PCA is used to minimize the dimensionality of a given benchmark dataset by transforming the initial d-dimensional dataset X into a new k-dimensional space Y (where k $\leq$ d) using a transformation matrix W [41]. The method used to obtain this transformation matrix is the linear Eigen-decomposition technique, which involves calculating the Eigenvalues and Eigenvectors (PCs) of the covariance matrix (X.X T). The Eigenvectors represent the directions of the data, and the Eigenvalues represent the magnitude of the data. To obtain the columns in the matrix W, each Eigenvector is assigned to a column, with the Eigenvalues being used to determine their order [42]. The Eigen-decomposition method is defined by breaking down the covariance matrix into three other matrices:

$$x^\wedge(T).X \rightarrow \mathrm{B.D.B^\wedge T} \tag{2}$$

In the definition of Eigen-decomposition, B is a square matrix (d $\times$ d) consisting of the Eigenvectors, and D is a diagonal matrix (d $\times$ d) with all elements except for those on the core diagonal set to zero. These elements represent the specific Eigenvalues, and BT is the transpose of matrix B.

3.4 Ensemble Classifier

3.4.1 CNN

CNN is a neural network architecture used for computer vision tasks, which utilizes a technique called convolution to efficiently process visual data. As shown in Fig. 6, the CNN architecture has three core layers: the first is pooling layer, the second is convolution, and the last is fully connected layers [43]. In the convolution layer, a filter is applied to the input data by multiplying it with a set of weights, creating a new two-dimensional array called a feature map. The filter is moved over the input using a step size called the "stride", which determines the size of the output feature map. This process is repeated, creating multiple feature maps, which are then processed by the next layers of the CNN, as illustrated by Eq. (3).

$$s = \theta(x \otimes \mathrm{w} + \mathrm{b}) \tag{3}$$

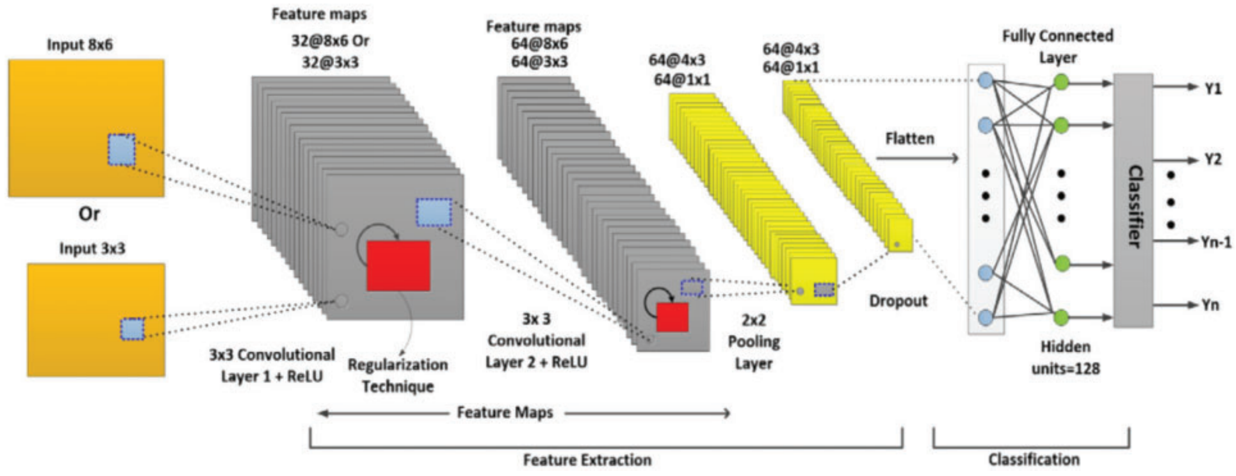


Figure 6: Architecture of CNN

The convolution equation involves various elements, such as θ, representing the non-linear activation function, x, representing the input data, b, which is the bias term, s, which denotes the feature map, and w, indicating the weight of the kernel function. In CNN, the *Relu* function is commonly used to set all negative values in the feature map to zero, thereby increasing the level of non-linearity in the convolutional layers. A CNN typically includes multiple convolutional layers, with the initial layer designed to detect basic features like edges or corners, while the later layers capture more advanced features. However, multiple convolutional layers may cause the output dimension to become smaller than the input, resulting in loss of information after a certain number of iterations. To address this issue, the padding technique can be employed by adding a border around the image. Two types of padding exist: "same" and "valid." The "same" padding method involves adding a border around the image to ensure that the input and output images are the same size. The padding size should satisfy the following equation to be valid:

$$\rho = \frac{(f-1)}{2} \tag{4}$$

where f represents the filter size while denotes the padding size.

The valid convolution technique involves the utilization of the original image without incorporating any zero-pixel padding surrounding the input matrix. In CNN, the pooling layer is responsible for reducing the spatial size of convolved features. This can be achieved through two methods: max pooling or average pooling. Max pooling involves selecting the maximum value from a portion of the input image that corresponds to the kernel filter, whereas average pooling takes the average value instead. Max pooling is typically preferred because it can effectively reduce dimensionality and remove noise from the image. Following this step, the output is subjected to a fully connected layer for classification. Several architectures have been developed to enhance the performance of CNN, such as AlexNet [44], LeNet [45], GoogLeNet [46], VGGNet [47], ZFNet [48], and ResNet [49].

3.4.2 LightGBM

LightGBM is a machine learning model developed by Microsoft in 2017, based on Gradient Boosting Decision Trees (GBDT). GBDT involves combining weak learners to create strong learners, using only regression trees for Decision Trees (DT). Each DT makes predictions and retains residuals from all previous trees. The training process for LightGBM is depicted in Fig. 7, where the residuals of the target value become the target for the next learning and each tree is trained to predict the residuals. The final predicted output is a combination of multiple DTs' outputs. Although GBDT has shown success in many machine learning tasks, it can experience decreased precision and efficiency with increasing data volume. To tackle this problem, Microsoft introduced the LightGBM algorithm, which maintains prediction precision, significantly improves prediction speed, and reduces memory usage [50].

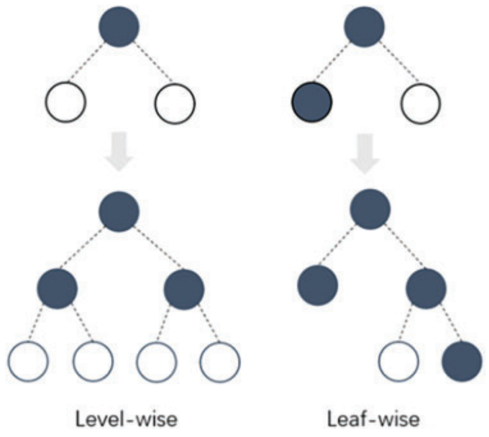


Figure 7: Generation strategy of LightGBM

The traditional Gradient Boosting Decision Trees (GBDT) algorithms can be sluggish and require a lot of memory since they sort feature values and enumerate all possible feature points to find the optimal segmentation point. However, the LightGBM algorithm solves this problem by using a histogram algorithm. This algorithm divides constant eigenvalues into k intervals and chooses the division points among these k values. As a result, the LightGBM algorithm trains faster and is more space efficient than GBDT. Furthermore, the decision trees generated by the histogram algorithm have regularization effects, which can prevent overfitting.

3.4.3 Ensembling using CNN and LightGBM

A model that combines the LightGBM algorithm and CNN is proposed. The ensemble process is shown in Fig. 3. The features from the dataset are extracted and refined by passing through the convolutional layer of CNN. Then, the output of the flattening layer is fed to the LightGBM model for classification and additional analysis. By combining these two methods, the proposed model achieves better prediction performance.

The ensemble classifier follows this procedure for categorization:

- The dataset resulting from preprocessing, feature selection, and dimensionality reduction is split into training and testing sets.
- The training data is fed into a developed CNN model for pre-training and to obtain the convolutional layer parameters and fully connected layer.
- In CNN, the hyperparameters of the convolutional layers are then frozen and the data resulting from the flattening layer is used as an input for LightGBM for extra training.
- The test dataset is then classified, a confusion matrix is generated, and performance metrics are computed.

4 Experiments and Discussion

4.1 Dataset

CICIDS 2017 [51] is utilized to determine the performance of the proposed IDS, which comprises favorable and the most updated regular attacks that have similarities to the true real-world data (PCAPs). It also presents the outcomes of the network traffic analysis with the use of CICFlowMeter and labeled flows in line with the source, protocols and attack, timestamp, source and destination ports, and destination IPs.

4.2 Experimental Environment

The experiment was carried out using Python and the Keras library with Tensorflow. An experimental setup used to assess the model parameters is shown in Table 2.

Table 2: Specifications of the experimental environment

Item	Description
Programming language	Python
OS	Windows 11
RAM	16 GB
CPU	11th Gen Core™ i7-1195G7
HDD	1 TB

4.3 Evaluation Metrics

As depicted in Table 3, the confusion matrix is used to calculate the evaluation metrics for the proposed model to ensure its effectiveness. Five common evaluation metrics, including accuracy (AC), false-positive rates (FP), and false-negative rates (FN), are utilized to determine the effectiveness of the model. These performance measurements are calculated using the confusion matrix of a 2-class classifier.

Table 3: Confusion matrix

		Predicted	
Actual		Positive	Negative
	True	Attack presents	No attack
	False	No attack	Attack presents

The equations below are used to demonstrate the performance of the introduced IDS:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

$$P = \frac{TP}{TP + FP} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (9)$$

where the number of true positives is represented as TP, false negatives as FN, true negatives as TN, and false positives as FP [38,52,53].

4.4 Analysis and Findings

In this section, a thorough analysis of the findings acquired through the models suggested in this study is presented. An evaluation was conducted regarding the effectiveness of the suggested models on a CICIDS 2017 benchmark dataset through a sequence of various experiments. In this section, the possibility for the suggested models was illustrated to identify the Flash Crowd attacks. Furthermore, several experiments had been performed, which involved the testing of the suggested learning models with a new unlabeled category of attack. Meanwhile, the remaining attacks are employed during the training. In every experiment, a comparison is made between the effectiveness of the proposed model against state-of-the-art models. The IDSs used in this evaluation comprised: DT-IDS [6], bGWbPS-IDs [38], RF-IDS [54], RT-AMD IDS [55], and LSTM-IDS [56]. The default setting from Skit-learn and TensorFlow libraries were employed to implement these IDDs. The performance comparison was also conducted between the proposed IDS and these state-of-the-art models in terms of accuracy, precision, and recall F1 measure using the CICIDS 2017 benchmark. The results of this comparison are presented in Table 4.

Table 4: Comparison results

	IDS	Proposed	LSTM-IDS	RT-AMD IDS	RF-IDS	bGWbPS-IDS	DT-IDS
%	Accuracy	95.84	91.05	84.34	90.41	85.62	83.38
	Precision	96.15	90.62	84.81	93.24	86.66	84
	Recall	95.54	91.77	84.27	87.34	83.87	81.81
	F1 measure	95.84	91.19	84.54	90.19	85.24	82.89

According to the results presented in Table 4, the performance of the proposed IDS was superior to that of the majority of state-of-the-art IDSs, and the hybrid models demonstrated even better performance. The use of a reduced set of features resulted in improved detection model performance. To ensure fairness, the comparison was conducted on both the training and testing datasets, using the CICIDS 2017 benchmark dataset. Furthermore, the proposed IDS had lower time consumption than other IDSs, which was attributed to factors such as the number of features used for training and testing, the number of hidden layers, and the number of neurons per layer. However, using a CNN could reduce time consumption by addressing the issue of parameter explosion through shared parameters across layers. While training a DL model can be challenging and costly, using a Graphics Processing Unit (GPU) accelerator can significantly improve computational speed, which has increased by more than 10 times in recent years and is expected to continue improving with advancements in GPU architectures and specialized training chips. The convolutional layer of the CNN is also used for feature extraction and filtering, and the LightGBM model is applied to the flattening layer output for classification and information. These steps improve the model's prediction accuracy.

In comparison to other state-of-the-art IDSs, the CNN-LightGBM required a shorter training duration and testing duration. However, the long training duration does not have a significant effect on the model function. Determining the ideal set of hyper-parameters may be among the processes of developing a machine-learning model that requires the longest duration. This condition is in line with DL. When the correct values are discovered to function properly in the training data and exhibit high quality in the test data, the manual scheduling of the hyperparameters due to real-time identification would not be required. Meanwhile, the short detection period denotes the strength of the hybrid approaches for anomaly identification, particularly in virtual environments including CC. These environments are vulnerable to setbacks in their specific structure. Therefore, rapid and efficient identification approaches should be applied to manage the attacks at a fast rate before the brain of the environment is targeted and critical consequences take place in the entire network.

4.5 Discussion and Limitations

The purpose of this article is to propose an IDS that uses the WSO and ensemble classifier to address security issues in a CC environment, particularly Flash crowd events. One of the challenges faced by deep and machine learning models during the training phase is overfitting, which occurs when the model learns from noisy and biased samples that do not accurately represent the patterns of interest. Regularization techniques can help to mitigate this issue and improve prediction, but they do so by focusing on individual weight values rather than the relationships between matrix entries, which can make small changes in an attribute more significant in the forecast. To overcome these limitations, the authors used feature engineering which involves selecting and extracting relevant attributes to improve DL performance for attack identification. The experimental results showed that the introduced IDS outperformed previous approaches and achieved superior precision for both binary and multiclass detection. The IDS consists of four phases: preprocessing, feature engineering, a hybrid ML and DL classifier, and a detection stage.

CNN was applied for the reduction in the number of training parameters, which led to the development of a new model that can identify network disruptions intrusions without high computational cost. Moreover, CNN is capable of reducing the dimension of the input attributes with the use of the pooling layer. Several DL and ML algorithms were employed for categorization situations to gain accurate DL outcomes. The suggested IDS was developed to create binary and multiclass categorizations to differentiate between the types of attacks. However, the following limitations were present in the suggested model:

- Despite the importance of the precision and assessment metrics for the evaluation of the model performance, these metrics are not adequate without the actual application of the created model in the CC environment.
- The assessment of the network performance by considering the resource use, account throughput, and time delay specifications is highly crucial to perform an intensive test on the capability of the suggested intrusion identification.
- Henceforward, the suggested IDS would be applied in a real CC environment, followed by a test on the potential for an intrusion identification to manage the identified attacks in an actual event.

In sum, using WSO in combination with an ensemble classifier in attack detection offers several advantages, including:

- Optimal Model Selection: WSO can be utilized to identify the most effective individual classifiers for inclusion in the ensemble, based on their performance on the training data, resulting in a stronger overall ensemble.
- Rapid Convergence: WSO can facilitate a quicker convergence of the ensemble classifier to the optimal solution compared to other optimization algorithms, thereby reducing computational time for attack detection.
- Improved Robustness: The combination of WSO and an ensemble classifier provides enhanced robustness to inconsistent or noisy data, as the ensemble can counterbalance the impact of outliers while WSO can help the model rapidly converge to the optimal solution.
- Better Handling of Imbalanced Data: In situations where one class is underrepresented, WSO can assist in selecting the best individual classifiers to handle such imbalanced data, further boosting the performance of the ensemble classifier in attack detection.
- Elevated Model Performance: By optimizing the parameters of the individual classifiers and ensemble using WSO, the overall performance of the model can be enhanced, resulting in more accurate and dependable attack detection.

5 Conclusion and Future Work

The volume of services and information available on the Internet is extensive, which contributes to high exchange traffic. This excessive scalability disrupts the network. Flash Crowd attacks, specifically the DDoS-based attacks that involve authorized HTTP requests to overwhelm the victim resources, are regarded as the primary disturbing attacks on the providers and users of online services. The victim is surrounded by service requests created by the attacker through DDoS tools. Furthermore, the challenge in detecting the Flash Crowd attacks would increase with the presence of the Flash Event. Following the similarity of the two anomalies (the legitimate Flash Crowd and Flash Crowd attack), the attack can travel under the identification system. However, the suggested detection method has proven its effectiveness in identifying Flash Crowd attacks in CC and showing higher performance than other state-of-the-art methods. This method has also created opportunities for future studies in the field of application layer attack detection. Some of the future directions of work would include the hybridization of LightGBM with a deep learning model, the use of ensemble feature selection, and the utilization of automatic data augmentation and transfer learning to enhance detection performance.

Acknowledgement: I express my gratitude to Northern Border University, Saudi Arabia, for administrative and technical support.

Funding Statement: The authors gratefully acknowledge the approval and the support of this research study by grant no. SCIA-2022-11-1551 from the Deanship of Scientific Research at Northern Border University, Arar, K.S.A.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Kaushik, P. Bhardwaj and K. Lohani, “Game of definitions—Do the NIST definitions of cloud service models need an update? A remark,” in *Futuristic Trends in Networks and Computing Technologies: Select Proc. of Fourth Int. Conf. on FTNCT 2021*, Singapore, Springer Nature Singapore, pp. 653–666, 2022.

[2] V. Maheshwari, S. Sahana, S. Das, I. Das and A. Ghosh, “Factors influencing security issues in cloud computing,” in *Advanced Communication and Intelligent Systems: First Int. Conf., ICACIS 2022, Virtual Event*, Cham: Switzerland, Springer Nature Switzerland, pp. 348–358, 2023.

[3] Y. Sanjalawe, M. Anbar, S. Al-E’mari, R. Abdullah, I. Hasbullah *et al.,* “Cloud data center selection using a modified differential evolution,” *CMC-Computers, Materials & Continua*, vol. 69, no. 3, pp. 3179–3204, 2021.

[4] M. Sharma, A. Gupta and J. Singh, “Resource discovery in inter-cloud environment: A review,” *International Journal of Advanced Intelligence Paradigms*, vol. 23, no. 1, pp. 129–145, 2022.

[5] V. Pai, P. Druschel and W. Zwaenepoel, “Flash: An efficient and portable web server,” in *USENIX Annual Technical Conf., General Track*, Berkeley, CA, United State, vol. 1, pp. 199–212, 1999.

[6] C. Tinubu, A. Sodiya, O. Ojesanmi, E. Adeleke and A. Adebowale, “DT-model: A classification model for distributed denial of service attacks and flash events,” *International Journal of Information Technology*, vol. 1, no. 1, pp. 1–11, 2022.

[7] A. Da Silva, L. Silva, E. Bezerra, A. Guelfi, C. De Armas *et al.,* “A proposal to distinguish DDoS traffic in flash crowd environments,” *International Journal of Information Security and Privacy (IJISP)*, vol. 16, no. 1, pp. 1–16, 2022.

[8] E. Oleg Kupreev and A. Gutnikov, “*DoS Attacks in Q4 2019 Report, Kaspersky Lab*,” 2020. [Online]. Available: https://securelist.com/ddos-report-q4-2019/96154/

[9] M. Aladaileh, M. Anbar, A. Hintaw, A. Hasbullah, A. Bahashwan *et al.,* “Renyi joint entropy-based dynamic threshold approach to detect DDoS attacks against SDN controller with various traffic rates,” *Applied Sciences*, vol. 12, no. 12, pp. 61–79, 2022.

[10] I. Aziz, I. Abdulqadder and T. Jawad, “Distributed denial of service attacks on cloud computing environment,” *Cihan University-Erbil Scientific Journal*, vol. 6, no. 1, pp. 47–52, 2022.

[11] S. Shah, F. Khan and M. Ahmad, “The impact and mitigation of ICMP based economic denial of sustainability attack in cloud computing environment using software defined network,” *Computer Networks*, vol. 187, no. 1, pp. 107– 119, 2021.

[12] A. Shawahna, M. Abu-Amara, A. Mahmoud and Y. Osais, “EDoS-ADS: An enhanced mitigation technique against economic denial of sustainability (EDoS) attacks,” *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 790–804, 2018.

[13] Y. Alemami, A. Al-Ghonmein, K. Al-Moghrabi and M. Mohamed, “Cloud data security and various cryptographic algorithms,” *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 2–23, 2023.

[14] M. Nguyen and S. Debroy, “Moving target defense-based denial-of-service mitigation in cloud environments: A survey,” *Security and Communication Networks*, vol. 1, no. 1, pp. 1–24, 2022.

[15] N. Kathirkamanathan, B. Thevarasa, G. Mahadevan, G. Skandhakumar and N. Kuruwitaarachchi, “Prevention of DDoS attacks targeting financial services using supervised machine learning and stacked LSTM,” in *2022 IEEE 7th Int. Conf. for Convergence in Technology (I2CT)*, Mumbai, India, IEEE, pp. 1–5, 2022.

[16] A. Aljuhani, “Machine learning approaches for combating distributed denial of service attacks in modern networking environments,” *IEEE Access*, vol. 9, no. 1, pp. 42236–42264, 2021.

[17] M. Mayuranathan, M. Murugan and V. Dhanakoti, “Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3609–3619, 2021.

[18] J. Gutierrez and K. Lee, “High-rate denial-of-service attack detection system for cloud environment using Flume and Spark,” *Journal of Information Processing Systems*, vol. 17, no. 4, pp. 675–689, 2021.

[19] I. Divyasree and K. Selvamani, “DAD: Domain adversarial defense system against ddos attacks in cloud,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 554–568, 2021.

[20] G. Alam and M. Raj, "An efficient SVM based DEHO classifier to detect DDoS attack in cloud computing environment," *Computer Networks*, vol. 215, no. 1, pp. 109–138, 2022.

[21] A. Nagaraja, U. Boregowda and R. Vangipuram, "Study of detection of DDoS attacks in cloud environment using regression analysis," in *Int. Conf. on Data Science, E-Learning and Information Systems 2021*, Dubai, UAE, pp. 166–172, 2021.

[22] P. Verma, S. Tapaswi and W. Godfrey, "An adaptive threshold-based attribute selection to classify requests under DDoS attack in cloud-based systems," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2813–2834, 2020.

[23] M. Mayuranathan, M. Murugan and V. Dhanakoti, "Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3609–3619, 2021.

[24] A. Nagaraja, U. Boregowda and R. Vangipuram, "Study of detection of DDoS attacks in cloud environment using regression analysis," in *Int. Conf. on Data Science, E-Learning and Information Systems 2021*, Petra, Jordan, pp. 166–172, 2021.

[25] G. Kushwah and V. Ranga, "Detecting DDoS attacks in cloud computing using extreme learning machine and adaptive differential evolution," *Wireless Personal Communications*, vol. 1, no. 1, pp. 1–24, 2022.

[26] G. Kushwah and V. Ranga, "Voting extreme learning machine based distributed denial of service attack detection in cloud computing," *Journal of Information Security and Applications*, vol. 53, no. 1, pp. 102–113, 2020.

[27] M. Dimolianis, A. Pavlidis and V. Maglaris, "Signature-based traffic classification and mitigation for DDoS attacks using programmable network data planes," *IEEE Access*, vol. 9, no. 1, pp. 113061–113076, 2021.

[28] M. Raj and S. Pani, "A meta-analytic review of intelligent intrusion detection techniques in cloud computing environment," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 57–72, 2021.

[29] S. Shah, F. Khan and M. Ahmad, "Mitigating TCP SYN flooding based EDOS attack in cloud computing environment using binomial distribution in SDN," *Computer Communications*, vol. 182, no. 1, pp. 198–211, 2022.

[30] A. Gaurav, B. Gupta, C. Hsu, D. Peraković and F. Peñalvo, "Filtering of distributed denial of services (DDoS) attacks in cloud computing environment," in *2021 IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, NJ, US, IEEE, pp. 1–6, 2021.

[31] A. Shawahna, M. Abu-Amara, A. Mahmoud and Y. Osais, "EDoS-ADS: An enhanced mitigation technique against economic denial of sustainability (EDoS) attacks," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 790–804, 2018.

[32] P. Bawa, S. Rehman and S. Manickam, "Enhanced mechanism to detect and mitigate economic denial of sustainability (EDoS) attack in cloud computing environments," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 9, pp. 97–107, 2017.

[33] R. Mendonça, J. Silva, R. Rosa, M. Saadi, D. Rodriguez *et al.,* "A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms," *Expert Systems*, vol. 39, no. 5, pp. 79–94, 2022.

[34] G. Logeswari, S. Bose and T. Anitha, "An intrusion detection system for SDN using machine learning," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 121–143, 2023.

[35] A. Awajan, "A novel deep learning-based intrusion detection system for IoT networks," *Computers*, vol. 12, no. 2, pp. 34–45, 2023.

[36] A. Thakkar and L. Ritika, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, no. 1, pp. 353–363, 2023.

[37] V. Dang, T. Huong, N. Thanh, P. Nam, N. Thanh *et al.,* "SDN-based SYN proxy—A solution to enhance performance of attack mitigation under TCP SYN flood," *The Computer Journal*, vol. 62, no. 4, pp. 518–534, 2019.

[38] Q. Alzubi, M. Anbar, Y. Sanjalawe, M. Al-Betar and R. Abdullah, "Intrusion detection system based on hybridizing a modified binary grey wolf optimization and particle swarm optimization," *Expert Systems with Applications*, vol. 204, no. 1, pp. 117–135, 2022.

[39] M. Braik, A. Hammouri, J. Atwan, M. Al-Betar and M. Awadallah, "White Shark OptImizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems," *Knowledge-Based Systems*, vol. 243, no. 1, pp. 108457, 2022.

[40] B. Ghojogh, M. Samad, S. Mashhadi, T. Kapoor, W. Ali *et al., Feature selection and Feature Extraction in Pattern Analysis: A Literature Review*, 2019. [Online]. Available: https://arxiv.org/abs/1905.02845v1

[41] S. Karamizadeh, S. Abdullah, A. Manaf, M. Zamani and A. Hooman, "An overview of principal component analysis," *Journal of Signal and Information Processing*, vol. 4, no. 1, pp. 111–129, 2020.

[42] H. Abdi, "The eigen-decomposition: Eigenvalues and eigenvectors," *Encyclopedia of Measurement and Statistics*, vol. 1, no. 1, pp. 304–308, 2007.

[43] R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[44] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally *et al., SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5 MB Model Size*. 2016. [Online]. Available: https://arxiv.org/abs/1602.07360

[45] Y. LeCun, "LeNet-5, convolutional neural networks," 2015. [Online]. Available: http://yann.lecun

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed *et al.,* "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, pp. 1–9, 2015.

[47] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[48] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional network," in *European Conf. on Computer Vision*, Tel Aviv, Israel, Springer, pp. 818–833, 2014.

[49] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp. 1492–1500, 2017.

[50] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen *et al.,* "LightGBM: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, no. 1, pp. 3149–3157, 2017.

[51] I. Sharafaldin, A. Lashkari and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of the 4th Int. Conf. on Information Systems Security and Privacy (ICISSP 2018)*, Funchal, Madeira, Portugal, pp. 108–116, 2017.

[52] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, no. 1, pp. 353–363, 2023.

[53] S. Al-E'mari, M. Anbar, Y. Sanjalawe, S. Manickam and I. Hasbullah, "Intrusion detection systems using blockchain technology: A review, issues and challenges," *Computer Systems Science and Engineering*, vol. 40, no. 1, pp. 87–112, 2022.

[54] M. Alduailij, Q. Khan, M. Tahir, M. Sardaraz, M. Alduailij *et al.,* "Machine-learning-based DDOS attack detection using mutual information and random forest feature importance method," *Symmetry*, vol. 14, no. 6, pp. 109–125, 2022.

[55] O. Bamasag, A. Alsaeedi, A. Munshi, D. Alghazzawi, S. Alshehri *et al.,* "Real-time DDoS flood attack monitoring and detection (RT-AMD) model for cloud computing," *PeerJ Computer Science*, vol. 7, no. 1, pp. 114–129, 2022.

[56] H. Aydın, Z. Orman and M. Aydın, "A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment," *Computers & Security*, vol. 118, no. 1, pp. 102–125, 2022.