Journal of Theoretical and Applied Information Technology
30th November 2022. Vol.100. No 22
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ENHANCED OPTIMIZATION MODEL WITH ENSEMBLE AUTOENCODER FOR ZERO-DAY ATTACK DETECTION

ABHIJIT DAS[1], PRAMOD[2], S. PRAVEEN KUMAR[3]
[1]Research Scholar, VTU, PESITM, Department of CSE, Karnataka, Shimoga-577204, INDIA
[2]Associate Professor, PESITM, Department of ISE, VTU, Shimoga-577204, Karnataka, INDIA
[3]Post-doc Research Assistant, University of Trento, Italy
E-mail: [1]abhijit.tec@gmail.com, [2]pramod741230@gmail.com, [3]pk.sekharamantry@unitn.it

ABSTRACT

Technology and networks have improved significantly in recent decades, and Internet services are now available in almost every business. It has become increasingly important to develop information security technology to identify the most recent attack. Many signature-based intrusion detection techniques promise great accuracy and low false alert rates. However, they struggle when confronted with emerging threats. This work employs the autoencoder, a type of neural network, to find possible zero-day threats by looking for differences in the data. This research is significant because, unlike previous efforts, it does not rely on the use of explicit labels during training. The premature convergence they caused was a problem for the optimization strategy used in the real world to identify zero-day attacks and other novel threats. This work proposed a unified ensemble autoencoder based on the Dynamic weighted Cuckoo Search Algorithm (DWCSA) to choose the most practical features for detecting anomalies in network traffic. As a result, the suggested Dynamic weighted based Cuckoo Search Algorithm (DWCSA) is modified to enhance performance while maintaining its fundamental structure. The model uses an improved DWCSA to determine which dataset features are the most important. Then an ensemble Autoencoder is used to improve further the optimal features that the enhanced DWCSA learned. Accuracy was enhanced by the proposed model, which overcame the constraint problems that had arisen throughout the feature selection process. The study demonstrates how to practically design and deploy a suitable approach and procedure, allowing even non-experts to identify the zero-day attack. The investigation of the results indicates that the proposed zero-day detection methods have more excellent results for the highest overall accuracy of 99.93% on CICIDS18 data sets. Overall, the results for identifying zero-day threats using the proposed technique are promising.

Keywords: *Intrusion Detection System, Autoencoder, Zero-Day Attack, Machine Learning, Cyber-security.*

1. INTRODUCTION

Computer-related crimes are rising in modern society due to the expanding use of digital technology and web applications becoming more widespread and challenging. New dangers from hackers and the number of cybercriminals are increasing [1], which encourages attackers to take over the entire network infrastructure, which might occasionally result in an operational problem. The attackers exploit infiltration as a critical element in integrity, availability, and confidentiality. Henceforth, IDS are being used as a defence mechanism to provide security and protection, and as a result, they carry out anomaly detection and network attack detection. According to recent studies, IDS' finest data mining solutions are defence mechanisms for identifying attack patterns [2].

Under network intrusion detection systems, conventional machine learning algorithms like Bayesian, SVM, Decision Trees, & Logistic Regression are frequently employed. The multiple methods have had positive outputs [3]. However, these algorithms are unsuitable for massive and high-dimensional data because of their sensitivity to outliers and noise. Deep learning approaches have recently become popular in several domains, including image identification and natural language processing. These methods have also produced positive results in intrusion detection by combining low-level information. Convolutional neural networks (CNN), long short-term memory (LSTM), recurrent neural networks (RNN), and deep neural networks (DNN) are the most common forms of neural networks used for intrusion detection. Cyberattacks have risen due to the globalization of computer networks and network applications. According to CNBC, a business news program, the average cost of a cyberattack in 2019 was USD 200,000. The two most common kinds of IDS are signature-based and anomaly-based. An IDS with a signature-based approach can only identify known attacks by looking for patterns in those attacks. In order to stay updated with all known attack signatures, a signature-based IDS must continuously update its database. Anomaly-based IDS identifies variations from typical traffic behaviour. Due to their complexity and speed, zero-day intrusions are extremely difficult to detect. Because of the limitations of the IDS model, modern ML-based IDS can reliably identify known attacks but not zero-day attacks. In recent years, academics have implemented ML and DL for cybersecurity due to their usefulness in a different area.

Algorithms that look for outliers have been used in recent studies to identify previously undetected risks. Current outlier-based IDS studies suffer from a number of drawbacks, the most prominent of which is low accuracy rates caused by high FPR and high FNR [4]. "Alert weariness" or "cybersecurity fatigue" [5] occurs when cybersecurity operation centers become frustrated due to a high false positive rate. According to Cisco [5], just 28% of intrusions that are discovered are actually happening. The difficulty of actually employing the models reduces their utility. In order to carry out their harmful plans, attackers will try any and all methods at their disposal. The survey conducted states that throughout the COVID-19 situation, hackers used a themed attack against both consumers and businesses. During the COVID19 pandemic, various sorts of cyber risks increased, as reported in [6]. Phishing assaults are the most common form of cybercrime. The value of ransomware in 2020 has been projected to be US dollar 20 billion, up from an estimated US$ 11.5 billion in 2019. The rate of growth of ransomware attacks is 350 percent each year, and cyber-security spending is predicted to hit a trillion dollars by 2024, as stated by Cisco [5].

Microsoft has witnessed ZA [7] due to Adobe Type Manager (ATM) library in 2020. Attackers aimed to exploit flaws in ATM that allowed for remote code execution. The vulnerability allows attackers to execute malicious scripts transmitted via spam or unwittingly downloaded remotely. If malicious code were to be executed on the vulnerable ATM, it might lead to a ransomware attack. This type of attack may involve tampering with firewall configurations, gaining unauthorized access to a system, or deploying malware. The number of ZAs that go unrecorded is likely far higher. Different kinds of mysterious cyberattacks are on the rise during the covid age. It motivates authors to come up with a new approach to stopping zero-day cyberattacks. The proposed technique uses an enhanced DWCSA feature selection algorithm with an ensemble autoencoder, which lessens reconstruction error. The model learns to identify risks by observing benign traffic during training.

While signature-based systems are excellent at detecting previously identified risks, they are ineffective against zero-day attacks, which are newly discovered threats. They cannot be adequately generalized and only identify intrusions that match perfectly with known attack signatures. Due to the dynamic characteristics of malicious behavior, these systems can be effective only if their signature databases are regularly updated. Anomaly-based methods can be either monitored or not. Supervised learning has been the basis of the majority of research done before. However, gathering manually classified data can be time-consuming and resource-intensive [8]. Moreover, just like the signature databases, labels must be regularly updated too. The researcher must also account for the issue of imbalance throughout the training set. Unsupervised learning will help with these problems, although at the expense of a more significant rate of false alarms [9].

The use of ML methods in conjunction with detecting network anomalies based on behaviour has become popular in recent cybercrime studies [10]. Even though there is a lot of written material on the subject, deployable ML-based NIDS are still in their early stages [11]. Among the various factors that reduce ML's ability to identify anomalies in networks are: Since 1) data transmission is highly diverse and variable, it is challenging to establish a benchmark of normalcy [12], and 2) monitored Machine learning needs labelled attack data sources, but since traffic labelling is costly and necessitates solid knowledge and skills [13], suspicious network data are currently limited and not updated, this presents a problem. With these issues in mind, we introduce a unified ensemble autoencoder based on the DWCSA to choose the most valuable features for finding abnormal activity in network traffic. Significance and Contribution of this work are:

- Unsupervised representation learning is used to identify malicious activity.
- Rather than categorizing each individual flow or packet, the time dimension is leveraged to identify anomalies.
- Explicit labels are not required. The AE is trained on input that is assured to be benign.
- An ensemble of simple shallow AEs is used, with each member trained on a specific traffic category.
- Threats that were previously known, as well as new emerging threats, can be determined in real-time.

Research Question **1:** An anomaly or a new threat is always unknown. We do not know what we are looking for. How can an IDS identify a new or emerging threat that has never been seen before?

2. RELATED WORKS

According to [14], who compared different classifiers, Random Forest (RF) and Bayesian Network (BN) are the most effective for intrusion detection, even though no single method outperforms the others across all evaluation metrics. The KNN approach showed the lowest False Positives (FP). In contrast, the RF algorithm showed the lowest False Negatives (FN) in another study by [15] using a simulated dataset. Numerous prior research has used the RF method for feature selection or classification, making it the clear frontrunner in intrusion detection. For instance, in [16], RF was utilized for selecting features before being passed off to a Support Vector Machine (SVM) for classification.

Additionally, many different categorization methods and ensembles have been extensively studied in the literature. Discriminative Multinomial Naive Bayes (DMNB) classifier attained a 96.5 % accuracy in two-class classification [17]. The same team also presented the usage of a collection of classifiers to prove its usefulness. Many SVM-based classification models were compared. After examining several tree-based classification techniques, it has been observed that a Random Tree-based method obtained 97% accuracy in categorization. The authors of [18] suggested an IDS that employed many Support Vector Machines (SVMs) for classification and the Genetic Algorithm (GA) for selecting features. In order to build their method, they linearly combined many SVM classifiers that were rated by attack severity. Using the GA-selected features, they trained each classifier to identify a specific form of attack. They used the CICIDS2017 [19] dataset, of which only a subset was available, to test their method. The authors of [20] tested an IDS for identifying DoS attacks using the Fisher score approach for selecting features and the SVM, KNN, and DT algorithms.

Combining Classification and Regression Trees (CART) with a BN classifier, investigated the results and created an ensemble of the two models [21]. Last but not least, the authors presented a hybrid architecture consisting of both the ensemble and the basic classifiers. In [22], researchers investigated the possibility of DDoS detection using a portion of the CICIDS2017 [19] dataset. They used DT and SVM-based models to narrow down the characteristics after first utilizing correlation analysis to facilitate classification. According to the authors, they got it right around 100% of the time. The imbalance in the training data is one of the limits of supervised classification. Concerning this matter, researchers [23] used Synthetic Minority Oversampling Technique (SMOTE) for their AdaBoost-based classifier on the CICIDS2017 [19] dataset to detect DDoS assaults.

The use of autoencoders for feature engineering and learning in cybersecurity has been proposed in [24]. Meng et al. [24] employ autoencoders to extract features, and those features are subsequently fed into a multi-class support vector machine classifier using NSL-KDD datasets. Overall, the model was found to have an accuracy of 86.96% and a precision of 88.64% in its evaluation. Various classes have different levels of accuracy and precision; for example, the DoS class has an accuracy of 97.91 %, the probe class of 88.07%, the R2L class of 12.77 %, and the normal class of 97.46 %. Narayana et al. [25] turn to a sparse autoencoder when extracting features. An SAE-RNN classifier takes input from the autoencoder's bottleneck layer (latent representation). Using the NSL KDD sample, they conducted their evaluations and discovered that an accuracy of 80% is feasible. The overall classification performance of the NSL-KDD sample is 99.71%.

Improved feature engineering for supervised classifiers has been the primary use of AEs. One work that used an AE to create a novel feature representation was [26] using NSL KDD [27] sample. Softmax Regression was employed for the final classification, which used both the new features and the original label vectors. [28] developed daisy chaining of four shallow AEs. Sequential training on the KDD CUP '99 [27] dataset produced shallow AEs. A SoftMax classifier was utilized in the final supervised layer. The overall detection efficacy of this method was 94.71 %.

Anomaly identification using thresholds of reconstruction errors was demonstrated using satellite telemetry data consisting of continuous sensor measurements in a study by [29]. While the work presented here is not directly connected to intrusion detection, the methodology presented applies to detecting anomalies in any area. Researchers used Wednesday's subset of the CICIDS2017 [19] dataset to investigate the feasibility of deploying a deep AE to detect DDoS threats; they found that AE could reach a 95.73 % accuracy rate with 4.32 % FAR [30]. On Wednesday, the data sample was partitioned into training, validation, threshold, and testing. An optimal error threshold, measured by precision and recall (F1), was calculated using the threshold subset. They used a technique that involved learning something twice as fast. Two supplementary AEs were also deployed alongside the main one. The first AE was employed to differentiate between benign flows and FNs on the second level, while the second AE was used to distinguish between attacks and FPs.

3. BACKGROUND

3.1 Introduction to Autoencoders

The autoencoder's primary role is to serve as a zero-day attack indicator by flagging anomalies. Binary classification (normal and zero-day attacks) is what the autoencoder model is utilized for here, not multi-class classification. The first authors to describe autoencoders are Rumelhart et al. [31]. They hope to circumvent backpropagation in an unsupervised setting by focusing on the input. Since the input and output of an autoencoder are identical, this model is considered a self-supervised learner and engages in representation learning. An AE is a self-supervised learner with fewer nodes in the middle layer, which is the bottleneck, but the exact number of nodes in the hidden layer and output layer. An autoencoder can be thought of as a conceptual hourglass. As can be seen in Figure 1, its main components are an encoder and a decoder. The encoder maps the high-dimensional input sample (X) into lower-dimensional space (h) to simplify its representation. It eliminates any redundancies, correlated features, and features that have negligible variance. This compressed representation retains only the most salient features of the data.

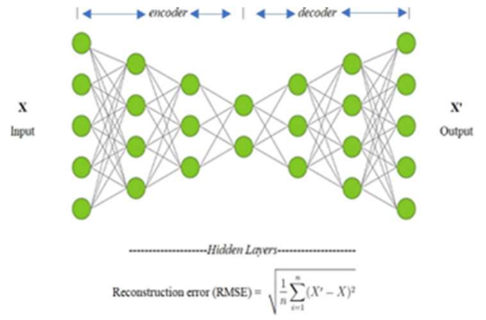
encoder
decoder
X
Input
X'
Output
Hidden Layers
Reconstruction error (RMSE) = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(X'-X)^2}$

Figure 1: Description Autoencoder

The reconstruction error is defined as the deviation between the original input vector (X) and the recreated output vector (X'). This type of error can be quantified using the Root Mean Squared Error loss function (RMSE). The Autoencoder is frequently employed for anomaly identification. Think about all the sensors, processors, and memories that work perfectly. However, when we aggregate their error data, we find a disproportionately large number of "positive" classifications relative to "minority" classes. The process of classifying data for predictive purposes can be laborious and expensive.

Autoencoder is successfully used by the majority of classes (benign data). The objective of the training is to attain the lowest possible reconstruction error. Model weights for the encoder and decoder are updated during training. In this case, we have a downsampling encoder and an upsampling decoder. LSTM, CNN, and ANN can all be used as encoders and decoders.

The Model can learn the practical reconstruction function on normal data, which may then be used to spot outliers. The reconstruction error is small for typical data but large for outliers (minority class).

The following equations can be used to represent the encoder and decoder functions, where (X) is the unprocessed input, (h) is the compressed representation, (X') is the reconstructed output, and (w) and (b) are the weights and biases determined during training.

$$h = f(X, w_e, b_e)$$

$$X' = g(h, w_d, b_d)$$

The RMSE can be calculated from the two matrices, X and X', in which n refers to the total occurrences.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X' - X)^2}$$

The goal is to minimize the dimensionality of the incoming data and then recreate it. The AE approximates it by reducing the reconstruction error.

3.2 Datasets

Our investigation led us to the Canadian Institute of Cybersecurity's most current dataset, CICIDS2017[19], where we discovered that the vast majority of update attack scenarios might be identified. This cutting-edge dataset not only includes recent examples of network intrusions but also satisfies all the characteristics of genuine attacks. The data in this set was collected from Monday through Friday. Each record represents a tagged flow and 84 attributes, and it has been available to academics in PCAP and CSV formats since 2018. On Monday, only normal traffic is permitted. Simulations of attacks were carried out between Tuesday and Friday. Examples of such attacks are DoS, DDoS attacks, the Heartbleed exploit, web attacks, infiltration, and botnets. Table 1 summarizes the CSV files, the number of observations and the orchestrated attacks. For the following reasons, this dataset was chosen to test and validate in this work:

- Newer attacks are included in this massive and comprehensive current IDS standard evaluation dataset.
- Monday's Dataset, which is used to train the models, only has benign traffic.
- It is not necessary to separate daily traffic into the train, validate, and test subsets because the complete set can be used as it occurred naturally.

Table 1: Dataset Description TABLE and Count of each Type

File	Normal Flows	Abnormal Flows
Monday.pcap_ISCX.csv	2359087	0
Tuesday.pcap_ISCX.csv	432074	13835
Wednesday.pcap_ISCX.csv	440031	252672
Thursday_Morning_Web Attacks.pcap_ISCX.csv	168186	2180
Thursday_Afternoon_Infilteration.pcap_ ISCX.csv	288566	36
Friday_Morning.pcap_ISCX.csv	189067	1966
Friday_ Afternoon-PortScan.pcap_ISCX.csv	127537	158930
Friday_Afternoon-DDos.pcap_ISCX.csv	97718	128027

Attacks Types	Count
1.Normal/Benign:	971016
2.FTP-BruteForce:	38703
3.SSH-Bruteforce:	37323
4.DDOS attack HOIC:	137185
5.Bot:	57507
6.DoS attacks GoldenEye:	8377
7.DoS attacks Slowloris:	2234
8.DDOS attack LOIC-UDP:	332
9.Brute Force -Web:	103
10.Brute Force -XSS:	55
11.SQL Injection:	11

4. THE PROPOSED METHODOLOGY

Several unresolved problems have been addressed to detect real-time network-based threats. In order to accomplish this, the framework of the proposed approaches is described below. In Figure 2, we see the overall layout of the proposed methodology. As seen in Figure 2, our proposed method involves preprocessing the data and selecting the most relevant features with the help of the Dynamically Adjusted CSA algorithm that we've recommended for use in the ensemble AE setting. In order to detect malicious traffic, our approach calculates anomaly scores using the reconstruction error for traffic data. The threshold-based detection of the attack by Unified Ensemble AE isolates the attack data from the rest of the network traffic.

4.1 Data Preparation

Table 1 summaries the eight data sets that make up the original data. Each dataset has 84 distinct network traffic characteristics, all of which are extracted from actual network traffic. The following

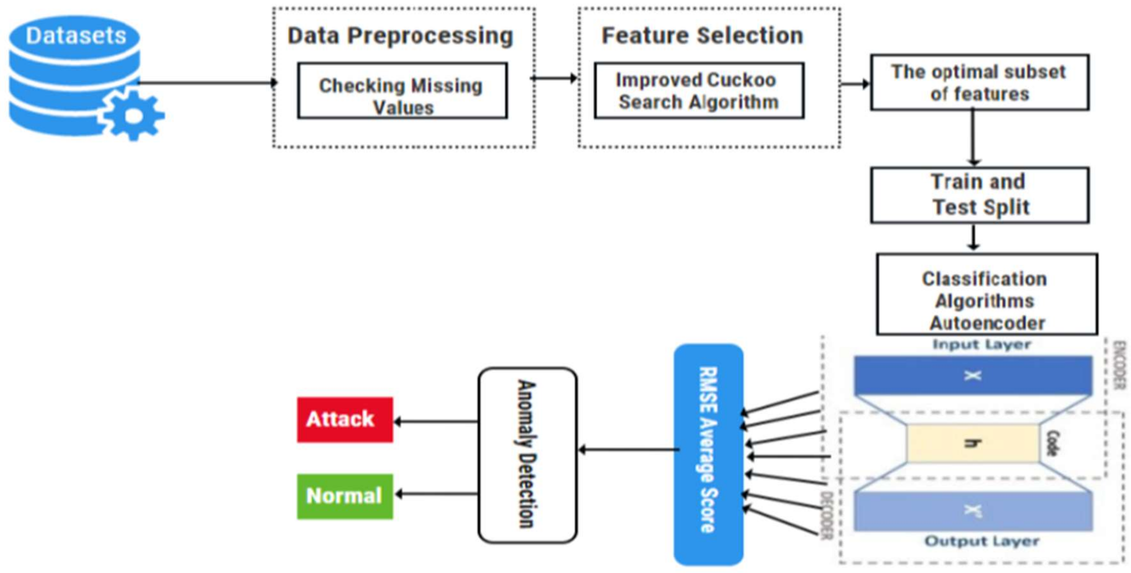

Figure 2: Propose Optimized Ensemble Autoencoder

five options were taken away. The original dataset (since it naturally occurs) has been used for the remaining attributes, allowing the AE to manage the dimensionality reduction, which allowed the model to avoid having to drop features altogether.

- Flow ID - Each flow has its distinct number. It is a sequence of characters made up of the sending and receiving hosts' IP addresses and port numbers.
- IP addresses of the sender and receiver were discarded.
- Date/time stamp - Date/time stamps were not employed in this exploratory study.
- Forward Header Length - This characteristic appears twice with identical values. It was decided to remove one of the duplicates.
- Infinite and not-a-number records have been removed.
- Identical Records have been eliminated for this preliminary examination.
- Classifications are now benign (0) or malicious (1).

4.2 Data Scaling

Min-max scaling [32] will apply a formula to each value in a column and return a new value. The relevant equation is:

$$m = (x - xmin)/(xmax - xmin)$$

The sample from Monday, which only has normal traffic, was used for training. The fit transform() technique was used to resize it. Testing was done with data from Tuesday through Friday. The transform () method is figuring out each feature's average and standard deviation in Monday's dataset. Each feature is given a new mean and variance through the transform technique. The objective was to make it seem like the test results had never been seen before.

4.3 Feature selection

Cuckoo search (CS) is a heuristic strategy [33] that takes its natural inspirations to address optimization issues. It was based on the way nature works and was inspired by it. The algorithm was inspired by the behavior of certain cuckoos, who will lay their eggs within the nests of host birds of a different species to save costs. By scattering their eggs among multiple nests, cuckoos reduce the likelihood of having their eggs stolen by predators. Sometimes the host birds will realize the parasitic eggs they've been incubating. Depending on the number of eggs found, the host bird may either ignore them or abandon the nest altogether before starting a new one. Brood parasites have evolved strategies like rapid egg incubation, rapid nestling growth, and egg color or pattern matching with hosts to increase the likelihood that host birds will ignore the parasites' offspring.

When Eq. (1) carries out a Levy flight, a new solution for cuckoo search is produced [33-35].

$$x_i^{t+1} = x_i^t + \alpha_0(x_i^t - x_{best}) \oplus Levy(s,\lambda) \quad (1)$$

If α_0 is the step size, then the current optimal solution is x best, where α_0>0. Levy flights are selected at random from the Levy distribution given in Eq. (2)

$$Levy(s,\lambda) \sim u = t^{-1}, (1 \leq \lambda \leq 3) \ (2)$$

where

$$Levy(s,\lambda) = \frac{\lambda\Gamma(\lambda)\sin(\frac{\pi\lambda}{2})}{\pi}\frac{1}{s^{1+\lambda}}, \text{ where } (s \gg s_0 > 0)$$

Γ(λ) denotes conventional gamma functions index in this context.

The CS algorithm involves discarding the worst nest with a probability of p_α, then constructing a fresh nest using a stochastic process, as shown in Eq. (3)

$$x_i^{t+1} = x_i^t + r(x_j^t - x_k^t) \quad (3)$$

The preceding equation yields two random solutions, x_j^t and x_k^t are for each iteration t, where r is a random number.

Though effective, the already available models have flaws in the form of excessive time consumption and early convergence compared to a realistic optimization strategy. As a result, the CS is tweaked according to its fundamental structure in order to boost its overall performance.

The effectiveness of metaheuristic algorithms is shown to be very reactive to adjustments made to their control parameters. The algorithm used many parameter optimization procedures. Piecewise, linear, and curved input variables that reduce with reproduction are examples of the adaptive parameter approach. The optimization issue known as a self-adaptive approach for a resource archive system is solved if the control parameters vary with the best fitness. The algorithm's ability to determine both global and local solutions is enhanced by the use of parameters like p_α and α_0. Based on experience, parameters can be modified with a given problem. When a significant value of α_0 is available, the value of p_α is low. There has been a clear trend of worsening algorithmic efficiency as the number of iterations has increased. Value of p_α and α_0 are equal, with more significant values of p_α resulting in larger values of α_0. Faster convergence aids in locating optimal solutions. The following equations (4) and (7) describe the dynamic variation in p_α and α_0 values as a function of reproduction number in Eq. (5)

$$p_\alpha = p_{\alpha i} \cdot 2^\tau,\ \tau = e^{1-\frac{G_m}{G_m+1-G}} \quad (4)$$

$$\alpha_0 = 0.5 \times \exp\left(-\frac{G-1}{G_m}\right) \quad (5)$$

In the preceding formula, G_m is the max step size, G denotes total repetitions, p_α is the previously defined constant. The method of dynamic weighted random walk is constructed using random walk, which has led to a slower rate of convergence and vibration. Since a larger indicates a more thorough examination or exploitation of host nest positions with solutions, it can be used to improve local search. Since a little value during the course improves efficaciously using local search, its linearly reducing relative value reveals a higher value. The user-defined constants ω_{max}, and ω_{min} are used to indicate the maximum and minimum values of the weighted coefficient, respectively. Therefore, the weighted coefficients are represented by the following equations. Considering classification accuracy and incorporating the rate of feature reduction as an adjustment term, as shown in Eq., the suggested evaluation technique is the fitness function that resolves the constraint problems in Eq. (6) and (7) [33-35]

$$x_i^{t+1} = \omega x_i^t + r\left(x_j^t - x_k^t\right) \quad (6)$$

$$\omega = \omega_{max} - \frac{G(\omega_{max}-\omega_{min})}{G_m} \quad (7)$$

To calculate the fitness value, used the Eq. (8) defined as in below.

$$f = \beta \cdot \frac{d-s}{d} + (1-\beta).acc \quad (8)$$

The equations above can be used to calculate the attributes in a data sample, denoted by the symbol d. The attributes chosen by metaheuristic optimization algorithms have the s-value. The typical value of accuracy weight is 1.

Dynamically Adjusted CSA algorithm [33-35]:

```
Begin
  x= (x 1, x 2..., x d); basis functions f(x).
  Set the parameters $p_{ai}, \omega_{max}, \omega_{min}$
  Create an xi population of n host nests, $x_i(i = 1,2,..,n)$;
  When $G = 1: G_m$
      Produce $p_\alpha, \alpha_{0,}, \omega$
      Obtain a cuckoo (m) at random by Levy flights, Levy $(s,\lambda) \sim u = t^{-\lambda}$;
      Produce new nests, $x_i^{t+1} = x_i^t + \alpha_0(x_i^t - x_{best}) \oplus Levy(s,\lambda)$;
      Assess its quality or fitness $f$;
      Pick a nest at random from $n(n)$;
        if $(F_m > F_n)$
          j should be replaced with a new solution.
        end
        if$(r > p_\alpha)$
          Remove a portion of the worst nests.
          And create new ones at the locations using $x_i^{t+1} = \omega x_i^t + r(x_j^t - x_k^t)$
        end
        Retain the finest solutions (or nests that contain optimal solutions).
        Rank the finest solutions, then determine the most recent excellent top feature.
  End
```

4.4 Choosing a Neural Network Architecture

79 characteristics and 1 target were present in our final dataset. The target was dropped because we are using unsupervised learning. So, input and output layers were made of 79 neurons. How many intermediate layers to have and how many neurons in each layer were key questions. Indirectly, the external problem does not aid in determining the total number of hidden units. The question of how many neurons should make up each hidden layer is still up to study [36]. A single hidden layer ANN has the potential for universal approximation. In previous studies, it has been seen that other studies using the same dataset employed different ANN designs as they provided the best results. Experiments were conducted to determine the effect of changing neurons number and hidden layers. We determined a solution to be "It does not matter.", which is exciting and relevant. In the initial step, a middle layer containing 10 neurons was selected arbitrarily. As a result, the Autoencoder (AE) was configured as follows:

- AE Architecture: Input (79): Hidden (10): Out(79)
- Activation Function: ReLu
- Libraries: TensorFlow (Version 2.3.0)
- Batch size: 1024 (10% of the training data)
- Optimizer: Adam Optimizer with default parameters
- Number of epochs: 100
- Loss: RMSEs sqrt() (Mean Square Error)

4.5 Training and Validation

The projected train set was shown in scatter plots using Principal Component Analysis. The initial train set was split in half. Every data point was assigned to one of two categories depending on its location in the scatter plot. Each group was further divided into a validation (25%) and a train (50%). A threshold (25%) was also created. Two models were independently trained and validated using their respective validation and training data. A checkpoint was used to save the weights on a local hard disc.

An ANN is used to build the autoencoder. The ANN hyperparameter optimization is done using a random search. The ANN's structure, nodes, and learning rate were set based on the random search outcomes. When looking for near-optimal values, a random walk can be much quicker than a grid search. Arbitrary search is more effective than grid search if only a few criteria need to be considered. It also limits the potential for overfitted parameters.

The model training begins once the hyperparameters have been determined. The overall training process is described in Algorithm 1. The first is to verify 75% of the cases as normal and 25% as abnormal. The model is set up with the best possible ANN model. Then, n epochs are used to train the model. Assuring convergence requires examining the autoencoder's accuracy and loss curves. Once convergence is confirmed, the model is evaluated with Algorithm 2, as depicted in Figure 3. An attacker event is categorized as zero-day when reconstruction error between decoding and source example (X_j) exceeds greater than a predetermined threshold. The cut-off is calculated from the value obtained from hyperparameter optimization using a random search. Different cut-off points will be tested, including 0.05, 0.15, and 0.2. It is vital to determine the threshold in order to determine the value at which an instance will be regarded as a zero-day assault. The MSE is within acceptable ranges if it is less than or equal to this.

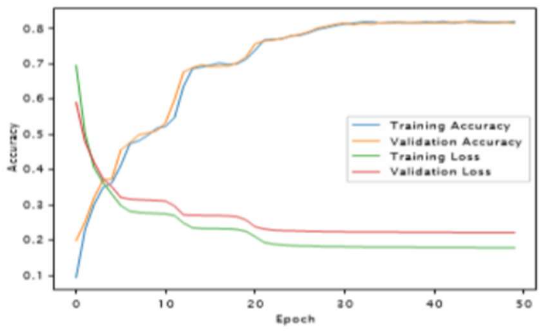

Figure 3: Autoencoder Convergence Curve

```
Algorithm 1: AE Training
In: benign data, ANN with different Input, Hidden and Output layers, epochs numbers, Regularization
Out: Autoencoder with Training
Step 1: train = 75% of benign data
Step 2: test = benign data − train data
Step 3: AE - Construct an AE (ANN Architecture, Regularization value)
Step 4: batch size ← 10% of the training data
Step 5: AE. train (batch size, epochs, train, test)
Step 6: return AE
```

```
Algorithm 2: AE Evaluate
In: AE with Training, attack, thresholds
Out: Identifying accuracy (accu)
Step 1: Identifying_accu ← {}
Step 2: predictions ← model_prediction(attack)
Step 3: for thr ∈ thresholds do
Step 4: accu ← (mse (predictions, attack) > thr)/length(attack)
Step 5: Identifying_accu.add (threshold, accu)
Step 6: end for
Step 7: return Identifying_accuracy
```

5. RESULTS AND DISCUSSION

The effectiveness of the classifiers is measured in this research by their accuracy (Accu), recall, F-Score, sensitivity, and specificity (SPEC) [37]. All statistical measures can be written as the equation (Eq) (9-13).

$$Accu = \frac{TP+TN}{FN+FP+TP+T} \quad (9)$$

$$Recall = \frac{TP}{TP+} \quad (10)$$

$$Specificity = \frac{TN}{TN+FP} \quad (11)$$

$$F-score = \frac{2TP}{2TP+FP+FN} \quad (12)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (13)$$

Cases in which the type of network intrusion is accurately distinguished are called true positives (TP), cases in which regular traffic in the network is correctly identified as normal are called true negatives (TN), cases when a threat is incorrectly identified as normal traffic, are called false negatives (FN), and cases in which normal traffic is incorrectly identified as a threat are called false positives (FP). For instance, recall measures how many different sorts of attacks were accurately identified across all cases labeled as threats, whereas specificity measures how often normal traffic was labeled as such. Better performance is indicated by increased accuracy and recall and a decrease in ER.

The accuracy of autoencoder models for all CICIDS2017 domains is summarised in Table 2. Note that the definition of "accuracy" varies between "harmless" and "attack" classes. The success of the model in thwarting attacks is reflected in its precision. The reconstruction error must be more than the specified threshold for this to work. The accuracy for the benign class is the proportion of cases that are not identified as zero-day attacks, in contrast to the attack class. Table 2 shows that the accuracy of identifying benign classes with different thresholds (0.2, 0.15, 0.1, and 0.05) ranges from 96.56% to 95.19% to 90.47% to 81.13%. Figure 4 shows the graphical representation of the accuracy of identifying benign classes with a different threshold. Additionally, there are three distinct levels of attack detection precision. We'll start with cyber-attack classes that can be easily distinguished from harmless ones. As an illustration, both DoS (Hulk) and DDoS have a high detection accuracy [83% - 99%] independent of the threshold. Second, subtypes of cyber-attacks that are distinguishable from normal ones (for example, SSH Brute-force and Port Scan). In this example, the accuracy of detection appears to be value-dependent, with lower thresholds resulting in higher detection accuracies. This emphasizes the significance of the threshold value selection in determining the precision of the detection. Finally, there are classes of cyber-attacks that cannot be distinguished from normal traffic and therefore are identified only with a degree of precision. Such attacks' behaviors are nearly identical to those of everyday web browsing.

The effectiveness of an autoencoder method at identifying previously undetectable cyberattacks is

Table 2: Zero-Day threat Detection using Dynamically Adjusted CSA algorithm with ensemble autoencoder

	Detection Accu			
	Threshold .2	Threshold .15	Threshold .1	Threshold .05
Benign	97.01%	96.18%	91.17%	80.93%
FTP Brute-force	6.08%	6.11%	7.01%	83.02%
SSH Brute-force	8.1%	9.18%	77.95%	81.11%
DoS (Slowloris)	66.13%	72.03%	79.10%	81.05%
DoS (GoldenEye)	67.01%	86.05%	88.01%	89.98%
DoS (Hulk)	97.99%	97.93%	97.99%	97.98%
DoS (SlowHTTPTest)	23.12%	25.01%	29.02%	40.05%
DDoS	84.27%	93.13%	98.18%	99.57%
Heartbleed	29.21%	29.119%	40.10%	44.14%
Web BF	10.1%	10.85%	83.01%	86.11%
Web XSS	12.24%	12.98%	97.18%	99.36%
Web SQL	17.18%	17.37%	23.02%	28.18%
Infiltration - Dropbox 1	48.01%	53.14%	95.22%	95.02%
Infiltration - Dropbox 2	86.11%	86.21%	99.98%	99.99%
Infiltration - Dropbox 3	17.2%	24.9%	90.05%	99.14%
Infiltration - Cooldisk	49.01%	52.32%	87.14%	93.08%
Botnet	18.56%	18.87%	38.05%	67.28%
PortScan	17.12%	29.34%	76.23%	99.07%

assessed. Findings presented in Table 2 are visualised in Figure 5, which displays ROC curves for every attack type in the CICIDS2017 data.

During both the training and testing phases, we made a variety of observations to deduce the consequences for performance. For each data point in the CICIDS2017 test dataset, our model's anomaly score in terms of reconstruction error is displayed in Figure 6. In this case, we can plainly see that a threshold leads to the incorrect classification of some data points in both normal and abnormal traffic. We define an outlier as a data point with a reconstruction error greater than the mean + 2 standard deviations of the reconstruction loss for the normal training samples. In order to fix the problem of threshold-based misclassification, we have used the Dynamic Weighted CSA algorithm to pick the best features, which includes excluding outlying data from the normal and abnormal sets.

Our suggested approach worked well on the CICIDS2017 data, as seen in Table 3, with a 97.74% accuracy and a 93.16% recall. We compared our method's performance metrics to those of other approaches in the literature using the table below. More significant improvements in performance metrics indicate improved model performance.

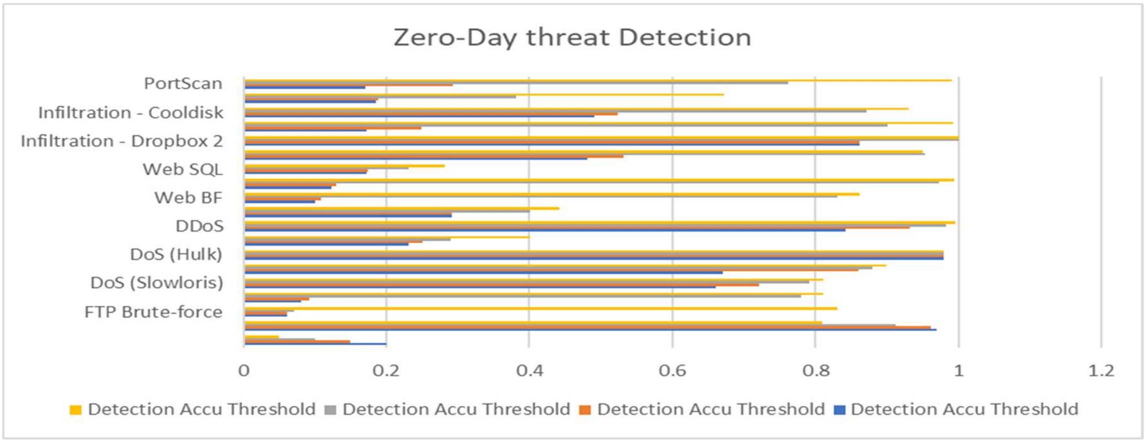

Figure 4: Zero-Day threat Detection with enhanced ensemble autoencoder for CICIDS2017 datasets

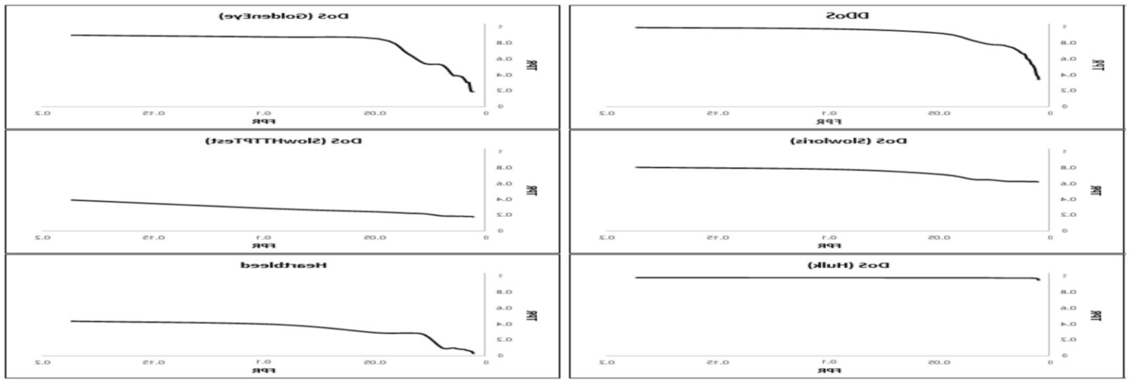

Figure 5: Proposed ensemble Autoencoder Classification ROC Curves

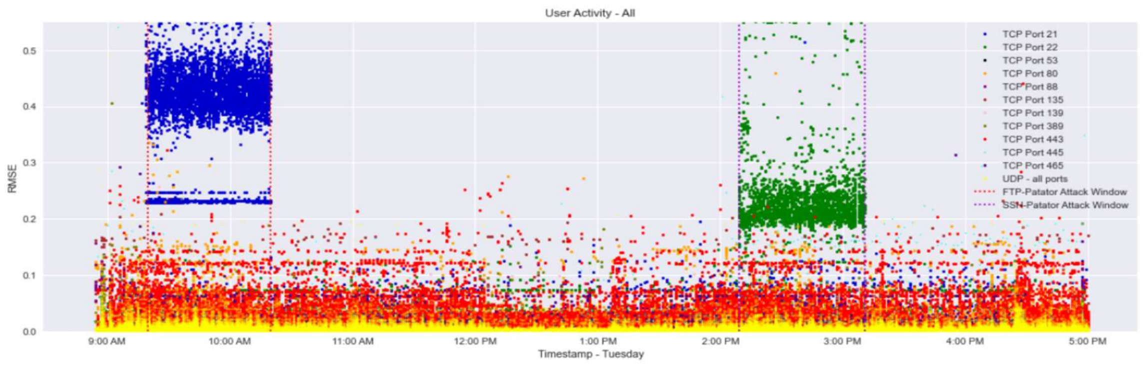

Figure 6: Anomaly score based on reconstruction error in CICIDS2017 on Tuesday RMSEs w.r.t time

Table 3: Proposed model comparison with others model

Paper	Dataset	Techniques	Accu	Precision	Recall	F1 score
Mbona et al. [38]	CICIDS2017	one-class support vector machines	74%	Not Reported	Not Reported	85%
Wenfeng et al. [39]	CICIDS2017	LogAE-XGBoost	99.92	99.71	99.86	99.79
	CICIDS2017	XGBoost	99.43	99.22	99.30	99.25
Roshan et al. [40]	CICIDS2017	OPT_AE	99.29	Not Reported	Not Reported	Not Reported
Verkerken et al. [41]	CICIDS2017	Autoencoder	94.26	94.59	97.78	96.16
Neuschmied et al. [42]	CICIDS2017	AE	77.6	41.5	29.7	34.6
		AEC	82.7	67.4	25.6	37.1
		AE-CNN	81.7	59.7	25.4	35.7
		VAE	77.4	40.3	28.0	33.1
		VAE-Prob	74.2	37.1	42.0	39.4
		AE+VAE-Prob	79.7	49.0	41.9	45.2
		OCSVM	78.6	43.6	26.1	32.6
Proposed Method	**CICIDS2017**	**Dynamically Adjusted CSA algorithm with ensemble autoencoder**	**99.93**	**98.91**	**98.81**	**98.46**

6. CONCLUSION

Our proposed methods were meant to help find a good and feasible way to find new and emerging threats, also called "zero-day attacks." When we looked over articles on intrusion detection that had been published in research journals, we came across studies that had looked into this issue. The implementation of representation learning, with an ensemble AE serving as its central component, was one particular approach that piqued our interest as a potential solution. The ensemble AE based on the Dynamic weighted Cuckoo Search Algorithm was used to model benign behavior and then measure the deviation of future activity from this learned model. The model uses an improved DWCSA to determine which dataset features are the most important. Then an ensemble Autoencoder is used to improve further the optimal features that the enhanced DWCSA learned. This model was able to detect previously unseen attacks. Experiments indicated a high accuracy for detecting zero-day threats. Accuracy, precision, recall, and F1-score for zero-day detection at CICIDS2017 are 99.93%, 98.91%, 98.81%, and 98.46%, respectively. The ensemble autoencoder outperforms previous models in detection accuracy.

The assumption is that there is no absolute answer, and data determine the parameters for fine-tuning. With the use of improved statistical approaches, it will be possible to foresee even more desirable actions in the future. In addition, a model can be re-trained as needed to account for data drift brought on by variations in network characteristics over time.

Limitations and Future Work of the Proposed work: The suggested work's shortcoming is that we could not appropriately identify SQL Injection and botnet attacks. We intend to strengthen our method in the future so that it can identify zero-day attacks with behaviours that are distinct from those of previously discovered threats. We'll perform to reduce the length of time it takes for the training algorithm to identify the intrusive pattern. In the future, we can add different IDS datasets to this study. In addition, we can investigate the same research with other ML approaches like LSTM and CNN for zero-day attack detection. Last but not least, if the models discussed in this study were to be tested in a realistic environment, we might learn even more. In contrast to evaluating benchmark datasets, this would show the demands and needs in various operational contexts.

REFERENCEs:

[1] Yang, L.; Quan, Y. Dynamic Enabling Cyberspace Defense; People's Posts and Telecommunications Press: Beijing, China, 2018.

[2] Yu, N. A Novel Selection Method of Network Intrusion Optimal Route Detection Based on Naive Bayesian. Int. J. Appl. Decis. Sci. 2018, 11, 1–17.

[3] Ren, X.K.; Jiao, W.B.; Zhou, D. Intrusion Detection Model of Weighted Navie Bayes Based on Particle Swarm Optimization Algorithm. Comput. Eng. Appl. 2016, 52, 122–126.

[4] Khraisat, A., Gondal, I., Vamplew, P. et al. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur 2, 20 (2019). https://doi.org/10.1186/s42400-019-0038-7

[5] Cisco Annual Internet Report (2018–2023) White Paper 2020. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[6] Santha Subramoni Global Head, Cyber Security Services, Prashant Deo Security Consultant, Geetali Raj Presales Lead, TCS, "How Covid-19 is Dramatically Changing Cybersecurity", 2020. https://www.tcs.com/perspectives/articles/how-covid-19-is-dramatically-changing-cybersecurity

[7] Microsoft Advisory: https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/adv200006

[8] Corey Nachreiner, Chief Security Officer, WatchGuard Technologies, "New Research Reveals Network Attacks at Highest Point Over the Last Three Years", SEPTEMBER 15, 2022, Available online: https://www.cyberdefensemagazine.com/new-research-reveals/

[9] "Naked Security" https://nakedsecurity.sophos.com/2022/09/13/serious-security-browser-in-the-browser-attacks-watch-out-for-windows-that-arent/ Accessed: 2020-04-15

[10] Tuan A Tang, Lotf Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. 2016. Deep learning approach for network intrusion detection in software defned networking. In 2016 International Conference

on Wireless Networks and Mobile Communications (WINCOM). IEEE, 258–263.

[11] Anna Giannakou, Daniel Gunter, and Sean Peisert. 2018. Flowzilla: A methodology for detecting data transfer anomalies in research networks. IEEE.

[12] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE symposium on security and privacy. IEEE, 305–316.

[13] Sebastian Abt and Harald Baier. 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS). IEEE, 40–55.

[14] Almseidin, M., Alzubi, M., Kovács, S., & Alkasassbeh, M. (2018). Evaluation of Machine Learning Algorithms for Intrusion Detection System. CoRR, abs/1801.02330. http://arxiv.org/abs/1801.02330

[15] Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. K. A. A. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. Procedia Computer Science, 171, 1251–1260. https://doi.org/https://doi.org/10.1016/j.procs.2020.04.133

[16] Y. Chang, W. Li, and Z. Yang, “Network intrusion detection based on random forest and support vector machine,” in 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1, 2017, pp. 635–638.

[17] M. Panda, A. Abraham, and M. R. Patra, “Discriminative multinomial naive bayes for network intrusion detection,” in 2010 Sixth International Conference on Information Assurance and Security, IEEE, 2010, pp. 5–10.

[18] R. Vijayanand, D. Devaraj, and B. Kannapiran, “Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with geneticalgorithm-based feature selection,” Computers & Security, vol. 77, pp. 304–314, 2018.

[19] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorba-ni, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Confer-ence on Information Systems Security and Privacy (ICISSP), Purtogal, January 2018

[20] D. Aksu, S. U¨ stebay, M. A. Aydin, and T. Atmaca, “Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm,” in International Symposium on Computer and Information Sciences, Springer, 2018, pp. 141–149.

[21] S. Chebrolu, A. Abraham, and J. P. Thomas, “Feature deduction and ensemble design of intrusion detection systems,” Computers And security, vol. 24, no. 4, pp. 295–307, 2005.

[22] S. Sarraf et al., “Analysis and detection of ddos attacks using machine learning techniques,” American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), vol. 66, no. 1, pp. 95–104, 2020.

[23] A. Yulianto, P. Sukarno, and N. A. Suwastika, “Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset,” in Journal of Physics: Conference Series, IOP Publishing, vol. 1192, 2019, p. 012 018.

[24] Q. Meng D. Catchpoole D. Skillicorn and P. J. Kennedy "Relational Autoencoder for Feature Extraction" arXiv:1802.03145 [cs stat] pp. 364-371 May 2017.

[25] Narayana Rao, K., Venkata Rao, K., & P.V.G.D., P. R. (2021). A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network. Computer Communications, 180, 77–88. https://doi.org/https://doi.org/10.1016/j.comcom.2021.08.026

[26] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, “A deep learning approach for network intrusion detection system,” in Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 2016, pp. 21–26.

[27] Choudhary, S., & Kesswani, N. (2020). Analysis of KDD-Cup’99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. Procedia Computer Science, 167, 1561–1573. https://doi.org/https://doi.org/10.1016/j.procs.2020.03.367

[28] F. Farahnakian and J. Heikkonen, “A deep auto-encoder based approach for intrusion detection system,” in 2018 20th International Conference on Advanced Communication Technology (ICACT), IEEE, 2018, pp. 178–183.

[29] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, 2014, pp. 4–11.

[30] M. Catillo, M. Rak, and U. Villano, "Discovery of dos attacks by the zed-ids anomaly detector," Journal of High Speed Networks, vol. 25, no. 4, pp. 349–365, 2019.

[31] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[32] Pires IM, Hussain F, M. Garcia N, Lameski P, Zdravevski E. Homogeneous Data Normalization and Deep Learning: A Case Study in Human Activity Classification. Future Internet. 2020; 12(11):194. https://doi.org/10.3390/fi12110194

[33] I. Publications (Ed.), Cuckoo Search via Levy flights, 2009.

[34] X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, International Journal of Mathematical Modelling and Numerical Optimisation 1 (4) (2010) 330–343.

[35] X.-S. Yang, S. Deb, Multiobjective cuckoo search for design optimization, Computers and Operations Research (0). URL http://dx.doi.org/10.1016/j.cor.2011.09.026

[36] M. T. Hagan, H. B. Demuth, and M. Beale, Neural network design. Martin T. Hagan and Howard B. Demuth., 2014.

[37] Liu, Yangguang & Zhou, Yangming & Wen, Shiting & Tang, Chaogang. (2014). A Strategy on Selecting Performance Metrics for Classifier Evaluation. International Journal of Mobile Computing and Multimedia Communications. 6. 20-35. 10.4018/IJMCMC.2014100102.

[38] Mbona, Innocent & Eloff, Jan. (2022). Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches. IEEE Electron Device Letters. 10. 69822-69838. 10.1109/ACCESS.2022.3187116.

[39] Wenfeng Xu, Yongxian Fan, "Intrusion Detection Systems Based on Logarithmic Autoencoder and XGBoost", Security and Communication Networks, vol. 2022, Article ID 9068724, 8 pages, 2022. https://doi.org/10.1155/2022/9068724

[40] K. Roshan and A. Zafar, "An Optimized Auto-Encoder based Approach for Detecting Zero-Day Cyber-Attacks in Computer Network," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), 2021, pp. 1-6, doi: 10.1109/ISCON52037.2021.9702437.

[41] Verkerken, M., D'hooge, L., Wauters, T. et al. Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques. J Netw Syst Manage 30, 12 (2022). https://doi.org/10.1007/s10922-021-09615-7

[42] Neuschmied, H.; Winter, M.; Stojanović, B.; Hofer-Schmitz, K.; Božić, J.; Kleb, U. APT-Attack Detection Based on Multi-Stage Autoencoders. Appl. Sci. 2022, 12, 6816. https://doi.org/10.3390/app12136816