IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2021, accepted June 18, 2021, date of publication June 24, 2021, date of current version July 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3092054

Developing an Efficient Feature Engineering and Machine Learning Model for Detecting IoT-Botnet Cyber Attacks

MRUTYUNJAYA PANDA[1], ABD ALLAH A. MOUSA[2], AND ABOUL ELLA HASSANIEN[3,4]

[1]Department of Computer Science and Applications, Utkal University, Odisha 751004, India
[2]Department of Mathematics and Statistics, College of Science, Taif University, Taif 21944, Saudi Arabia
[3]Scientific Research Group in Egypt (SRGE), Giza 12613, Egypt
[4]Faculty of Computers and Information, Cairo University, Giza 12613, Egypt

Corresponding author: Abd Allah A. Mousa (a.mousa@tu.edu.sa)

This work was supported by the Taif University Researchers Supporting Project, Taif University, Taif, under Grant TURSP-2020/48.

ABSTRACT The proliferation of Internet of Things (IoT) systems and smart digital devices, has perceived them targeted by network attacks. Botnets are vectors buttoned up which the attackers grapple the control of IoT systems and comportment venomous activities. To confront this challenge, efficient machine learning and deep learning with suitable feature engineering are suggested to detect and protect the network from such vulnerabilities in the future. For the efficient detection of cyber attacks, the representative dataset shall be well-structured for training the model and then validating the proposed system to develop an optimal security model. In this research, we used the UNSW-NB15, a new IoT-Botnet dataset (a noisy and imbalanced dataset) to classify cyber-attacks. K-Medoid sampling and scatter search-based feature engineering techniques are used to obtain a representative dataset with optimal feature subsets. To validate the proposed methodologies, three most recent machine learning (ML) methods including (i) JChaid*- a recent upgrade version to Chi-square automatic interaction detection (CHAID) decision tree-based, (ii) A2DE (a semi-naive Bayesian averaged two-dependence estimator), & (iii) HGC- a hybrid of Genetic algorithm with K-means clustering and two deep learning (DL) methods such as (i) Deep Multilayer perceptron (DMLP) & (ii) Convolutional neural network (CNN) based classifiers are employed. From the extensive experimental analysis, it is pronounced that scatter search-based DMLP classifier outperforms the other competing models in terms of (i) highest detection rate with100% accuracy, 100% macro-averaged precision, 100% macro-averaged recall & 100% macro-averaged F1-score and (ii) low computational complexity with the least training time of 4.7 seconds & testing time of 0.61 seconds.

INDEX TERMS Network intrusions, IoT, botnet, K-Medoids, scatter search, DMLP, A2DE, JChaid*, CNN, K-means clustering, genetic algorithm, macro precision, macro recall, macro F1-score, UNSW-NB15 dataset.

I. INTRODUCTION

Network attacks or intrusions are collections of events transmitted through network packets that pose a threat to the confidentiality, availability, and integrity of the IoT network, for the incapability of today's firewalls mechanism to detect and block such a current cybersecurity attack scenario. Further, with the extensive usage of smart digital devices in an IoT network environment, secure communications amongst such interconnected devices are the need of the day as the network vulnerabilities are complex and very costly to get removed from such an IoT network system.

Recently it is observed that an efficient network intrusion detection system could not only be able to detect modern security attacks, including zero-day attacks, but also could prevent them from future occurrences. Network intrusion detection systems (NIDS) can be thought of either as misuse detection or anomaly detection. While many business organizations prefer to use misuse detection, anomaly detection is still considered an immature one, attracting many researchers to research anomaly detection in IoT networks to detect new or rare attacks. Some of the anomalies in IoT networks

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

include worms, port, network scan, TCP based and UDP based events, flash crowd, flow bomb, the small or large size of the network packets, etc. The recent literatures outline some security vulnerabilities that are exploited in many IoT network for not having any generalized security standard to be maintained by the IoT devices [1], where it is seen that a technical glitch in smart lightbulbs design process could be used for a ''bricking attack'' that quash the whole city's traffic lights. Similarly, the attack against Honeywell [2] was focused on to go beyond malignant detection to identify new surge of USB attack types, keeping human substantiation part of security, or a set of Z-Wave devices. Z-Wave is a dominion method used to coalesce sensors and actuators over RF and accomplish smart home and office automation services [3]. A Black Hole attack is administering on a real-world Z-Wave network to adorn a conventional routing attack that makes the most of the bare susceptibility and most crucial DDoS attacks [4]–[6]. Investigators have been manifesting easiness in hacking motor cars, the ballot box, and power stations. It is exemplified ransomware attacks against abode humidifiers and uncovers fragilities in engraft mechanical-heart. Recently, several researchers have declared dynamism to create an effective customer brain-computer interface to slash a person's brain as a subsequent safe appearance.

A conventional botnet comprises imperiled workstations or servers, which are termed, Zombies. A group of zombie computers (or bots) is known as botnets. In computing, Zombies are workstations connected through the internet usually get infected with malware that allows the hacker or intruder to take control of them and perform the malicious activities without the knowledge of the owner. In this case, both owner and the intruder control these infected workstations in the botnet and try to transfer information through any non-standard communication channels (or covert channel) which are not allowed by computer security policies.

From IoT perspectives, the botnet is an imperiled IoT device network, including a camcorder, routers, smart sensors, or any smart devices with embedded system design, etc. that are too infected with malware. Using IP protocol for data transmission over the internet poisoned with malicious software authorizes cyber-hackers to control the devices through a Trojan horse virus payload. These bot herders having ill intentions can perform automated and simultaneous malicious activities over interconnected IoT devices resulting in several of the following botnet attacks as DDoS (Distributed Denial of service), credential-stuffing attacks, network application attacks to purloin data, keystroke logging, identity theft, unsolicited mail (spamming) and unlawful ingression to all IoT network-connected device [7].

While comparing IoT botnet with the conventional ones, it is observed that both have the centralized and Peer to Peer (P2P) decentralized architecture but have differences in devices being used. Further, IoT botnet has a difficult detection mechanism and can have a huge impact on the IoT system for its quick and wide spread contrary to its counterpart [8].

IoT botnet malicious software uses either active or passive type's proliferation mechanism. In inactive cases, these botnet malware applications spread by creating their clones. When any smart IoT device gets connected with such an infected device either through Wi-Fi, USB drive, or FTP servers, they become infectious. It makes them part of the botnet, and the propagation of the virus continues spreading further [9], [10].

From several IoT applications including Smart healthcare, smart agriculture, smart homes, smart grid, and intelligent transportation to name a few, it is observed that IoT devices are prone to threats due to their following inherent characteristics and hence, a proper mechanism is sought for their secured implementation.

- IoT devices use lossy wireless links which makes them vulnerable to cyber-attack. Further, the resource limitations and lack of standardization in IoT systems make it quite difficult to develop an effective security mechanism to deal with such threats.
- Due to heterogeneous architecture and diverse applications of IoT systems, it poses real challenges to develop a security scheme that may be applicable for all with different application objectives.
- IoT systems are exposed to new vulnerabilities as old devices are being replaced with new devices due to changes in user's preferences.
- Because of the ubiquitous nature of the IoT devices, physical threats such as man-made or nature-made are inevitable for many applications, a context-aware security scheme shall be adapted to effectively deal with such a situation.
- Further, due to the huge deployment of tiny IoT devices in IoT environments, huge data are been collected and analyzed for effective decision-making to safeguard the interest of the customer and organization at large. This needs a proper security mechanism so that eavesdropping of the devices can be avoided.
- Traditional security mechanisms including data encryption, firewalls, access control with network and application layer security, etc., are found to be ineffective to address the inherent IoT system limitations. Hence, machine learning and deep learning approaches may be a solution to address these issues [11].

In addition, due to the large-scale deployment of smart IoT devices in an IoT system, manufacturers find difficulty in providing a separate key for each device, rather uses the same user name and password for all the same kind of devices. So, as because IoT devices are not being developed with an intrinsic security mechanism, which makes them more endangered while connected to a network. It is also observed that due to the ever-developing nature of the IoT botnets, signature-based network anomaly detection (or the traditional ones) methods failed to detect them efficiently, which demands a separate IoT botnet detection mechanism in contrast to traditional botnet approaches [12].

It is customary to note here that due to IoT cyber-attacks' diverse nature, mere patching of IoT devices against the most common known attacks is not a viable solution. Simultaneously, without going into more details into the IoT security infrastructure and smart device features, it is suggested to perform some counter security vulnerabilities pertinent to the attacks compromised IoT devices against the unseen attacks or intrusions.

To address these emerging security and privacy issues that arise out of the IoT environment and to understand the intelligence embedded thereto, suitable feature engineering techniques and efficient machine learning approaches are recommended to detect network intrusions and IoT botnet attacks [13].

It is also opined that to deal with such IoT security issues, machine learning, and deep learning model is quite efficient, which uses the past behavior of the system to develop a secured intelligent model [11], [12].

From all the above discussions, we are motivated to explore further research in using an efficient feature selection approach with suitable machine learning and deep learning methodologies to build a security-based intelligent model to detect IoT botnet efficiently.

To explore further in this emerging area of research, this paper aims to apply feature engineering and machine learning techniques for the efficient detection of IoT botnet attacks in an IoT anomaly detection system.

Our main contributions are as follows:

- We present an efficient feature subset selection approach using promising metaheuristic methods such as: Scatter search and K-Medoid sampling to create the best features.
- We propose several classification methods such as A2DE Bayesian classifier, JChaid* decision tree, deep learning-based convolutional neural network (CNN) classifier, Deep MLP Classifier, and unsupervised classification using hybrid K-means clustering and genetic algorithm (HGC) on the reduced features for optimal detection of IoT botnet attacks.
- A new IoT botnet attack dataset, UNSW-NB15, is used for extensive simulation to reveal the effectiveness of the proposed IoT based network intrusion detection system
- Several performance evaluation metrics such as micro and macro averages to precision, recall, and F1-score apart from prediction accuracy, training, and testing time, are considered to understand the efficacy of the proposed IoT botnet detection model.

The rest of this paper is assembled as follows. While Section 2 dispenses the related works, Section 3 confers a short chronicle of the dataset. Next, Section 4 talks through the feature selection process; machine learning and deep learning classification methods employed are presented in Section 5. Section 6 accorded to the experimental setup and discussed the results obtained compared to others' similar work available in the literature. Finally, Section 7 brings out the conclusions.

II. RELATED RESEARCH

The real challenge in today's network intrusion detection system (NIDS) lies in spotting the unseen and obscure malicious software as the malware programmer writes different bypassing techniques for information tuck away to avert detection by IDS. With the tremendous growth in internet users, cybersecurity warning such as zero-day attacks has impacted the life miserably in terms of magnitude and potency. Even though machine learning is explored for its effectiveness in detecting network intrusions in DARPA KDDCup 1999 and NSL-KDD datasets in the past several years, it is still fuzzy in deciding the best classifier in detecting network intrusions.

In recent times, with the development of IoT technology, tremendous real-life smart applications have gained attention from several manufacturers to improve human life quality. IoT devices being resource constraints and no specific built-in security mechanism, it becomes easy for the attackers to attack the device and the IoT system as a whole. Hence, the attackers shifted their ambitions to attack the organization rather than individuals. It has led to the urgent develop an efficient IDS to detect new unseen and trailblazing malware, in an IoT system.

Mahmud Hossain *et al.* [14] discussed the undetermined problems in IoT, considering IoT devices are insecure and opined for priority in IoT security research as such devices are easily compromised by the investigators for their various limitations in terms of mobility, workability, and battery life along with their diverse applications ranging from sensors to IBM PC clones.

Pahl *et al.* [15] describe in their paper that even though there is some related work found in IoT, still it is attracting the attention of researchers for its popularity in today's livelihood and designed an anomaly-based detector and firewall for IoT system using K-Means and BIRCH clustering with a predictive accuracy of 96.3%. Brun *et al.* [16] used an expert system based on Dense Random Neural Network to detect IoT security threats that were arisen out of DoS attacks and Denial of sleep attacks.

Diro *et al.* [17] use deep neural networks and their variant for network attack detection using cloud-to-things architecture in an open-source dataset with 98.27% and 96.75% accuracy, respectively. Usmonov *et al.* [18] delineate the challenges faced in protecting data privacy during data transfer by IoT components in an embedded environment and proposed using digital watermarking techniques to counter these.

Anthi *et al.* [19] constituted an IoT-based intrusion detection system in a dataset collected for several days using the Wireshark packet sniffer tool and applied several data mining methods for efficient detection of probing DoS attacks. Ukil *et al.* [20] use smartphone-based anomaly-based attack detection in IoT healthcare using big data analytics and biomedical signal and image processing. Kozik *et al.* [21] engrossed cloud architecture-based extreme learning machines in Netflow arrangement data for IoT network attack classification with an accuracy of 99% for scanning attacks and 95% for infected host attacks.

Hasan *et al.* [22] compared several well-known machine learning algorithms in detecting cyber-attacks in IoT network sites in an open-source synthetic dataset collected from Kaggle and found that Random forest, decision tree, and Artificial neural network all present the same classification accuracy of 99.4%. Koroniotis *et al.* [23] proposed four data mining techniques: C 4.5 Decision tree, Naïve Bayes, Artificial neural network, and Association rule mining to detect IoT botnets in USNW-NB15 dataset. They conclude that C4.5 is the best classifier with 93.23% accuracy and the lowest false positive rate at 6.77% compared to Naïve Bayes with 72.73%, 27.27%; Association rule mining 86.45%, 13.55% and 63.97%, 36.03% respectively for ANN.

Kumar *et al.* [24] proposed a robust, lightweight solution by applying a random forest algorithm to detect IoT botnets at an early stage most accurately (almost 100 %) in less time with very minimal memory consumption. Derhab *et al.* [25] advocated for the use of feature engineering to IoT botnet unbalanced dataset with Synthetic minority over-sampling – Nominal Continuous (SMOTE-NC) and classify the botnet attacks by using temporal deep learning methods and compared the suitability of their proposed approach with two conventional ML methods such as Random forest and logistic regression; and two popular deep learning as Long short-term memory (LSTM) and Convolutional neural network (CNN). They implemented all on IoT dataset from the University of New South Wales and concluded that TCNN predicts the attacks with 99.998% accuracy and takes 424 seconds to train the model, in comparison to 99.9654%, 762 seconds for LSTM; 99.9973%, 419 seconds for CNN; 99.2858%, 709 seconds in Logistic regression and 97.4586%, 191 seconds for Random Forest respectively.

Bagui and Li [26] presented the usefulness of random over-sampling and random under-sampling with SMOTE to deal with highly imbalanced and less imbalanced intrusion detection datasets to improve the classification accuracy of the classifier. They evaluated the model by using artificial neural network (ANN) for attack detection using macro precision; macro recall, and macro F1 score for several sampling techniques and found that SMOTE-Random under-sampling with ANN classifier outperforms all others with 83.61% macro precision; 87.14% macro recall; 82.75% macro F1-score with 342 seconds training time.

Nguyen *et al.* [27] introduced a federated learning-based machine learning approach for detecting backdoor attacks in IoT intrusion detection systems on several datasets. They chose Federated learning with the K-means clustering technique as a defensive approach to classify the model either as an intrusion or Benign. Lin *et al.* [28], on the other hand, proposed to use biologically inspired algorithms (Artificial Fish Swarm optimization-AFSO) for feature selection along with a Support Vector Machine (SVM) classifier for efficient detection of botnet attacks. They reported that AFSO+SVM presents a slight increase in detection accuracy (97.76%, 97.30%) but faster training time (19843 seconds, 22831 seconds) when used with genetic algorithms for botnet detection process.

From the extensive literature survey, it is conspicuous that IoT security is still not matured. Various states of an affair with this new and flourishing field are being come across daily.

III. DATASET USED

The most commonly used dataset in the research study of network induction detection systems are briefly discussed here.

The DARPA 98 (Defense Advanced Research Projects Agency) dataset [29] was initially brought on by Lincoln Laboratory, USA, to judge the intrusion detection system. This data was comprised of 4GB binary data, collected over seven weeks.

The KDDCup 1999 dataset [30], an enhanced version of the earlier DARPA 1998 dataset, is severely imbalanced in intrusive and normal classes but still useful in distinguishing inpouring attacks and normal kinship even today.

NSL-KDD is not a new of its kind, as this is the refined version of the KDD 1998 dataset developed by Ali A. Ghorbani in Network Security Laboratory (NSL) and seems to have addressed some of the ingrained complications, but still have problems as talk about by McHugh [31].

University of Brescia, Italy, explain an IDS dataset, namely UNIBS [32] in which tcpdump was used to collect the network traffic in a cluster of 20 workstations flowing the observed devil to detect the DoS attacks only. The problem lies in the dataset is not having any other features except network packets with no labels to get more information.

The IDS dataset developed by Tezpur University (TUIDS), India [33], [34] created features caused during the flow level process in a physical testbed without including the features that might have occurred during the flow capturing process. The dataset features include attacks concerning DoS, DDoS, and probing or scanning attacks only.

The USW-NB15 is the 100GB synthetic dataset used in this research, coined by Mustafa *et al.* [35] at the University of New South Wales, Canberra, Australia IXIA Perfect Storm tool in the Cyber Range Laboratory. This dataset is generated in PCAP files with a mixture of normal and intrusive traffic and other notable features.

From all these discussions, it is apparent that research concerning generating realistic IoT botnet attack scenarios is not yet explored fully as many available datasets do not come up with new features while other datasets do not include IoT-generated network traffic. Hence, researchers need to explore further to evaluate efficient machine learning mechanisms to address the lacunas found thereto.

Elucidation of the UNSW-NB15 Dataset:

UNSW-NB15 IoT botnet dataset has the following attack categories:

(i) **Fuzzers**: these attacks include protocol fuzzing that may act as proxy altering packets on the fly and echo them. Normally, fuzzers embroil crashes (or Denial of

Service-DoS) allow the attacker to clinch over the IoT devices, notwithstanding the encryption control procedures. Some examples of fuzzers attacks include skewed packets, stack overflows, etc.

(ii) **Backdoors**: A backdoor attack is a malicious software through which hackers can get prohibited access to the website. The malware is installed via weak network entry points such as old plug-ins and input fields. These attacks are stealthy, and hence most of the time, the hackers get undetected.

(iii) **Analysis**: These attacks are also referred to as active surveillance. It is a type of computer attack in which the cybercriminals engross the target to obtain details about the intrusions. In military applications, reconnaissance is used to gather information by entering into the enemy territory. In computer security, reconnaissance is the first step before taking the next step to exploit the target computer system. The examples include port scanning to locate some open and vulnerable ports.

(iv) **Denial of Service (DoS):** This type of attack compromises the available network resources from intended users and makes the IoT network devices perform isolated. DDoS (Distributed DoS) attacks in IoT botnet scenarios are cases where many intruders try to intrude into the extensive network to attack a single IoT device.

(v) **Shellcode**: Shellcode seldom has to do with shell scripting, but it is considered as an IoT botnet attack scenario through which the cybercriminals tries to chronicle the code executed by a compiled program owing to susceptible use and open a remote cell (for example- an exemplar of command-line interpreter). Then, the remote shell is used further to infect the victim's system. With proper inputting to a targeted program, the shellcode attack considers being a very transparent and efficient means of attack.

(vi) **Reconnaissance attacks**: These attacks thought of as a general knowledge congregation attack, can transpire both logically and physically. Traffic analysis, packet sniffers, etc., fall under this category.

(vii) **Worms**: A computer worm is a malicious software that spreads from one IoT device to another and can clone itself without human interplay. More importantly, it does not need to affix itself in the target program for causing harm to the devices. An example of such a worm is a person who always tries to recline and escape.

(viii) **Generic**: The generic attacks are based on ciphers, which is virtually a prang attack on ciphers' secret keys. These attacks usually can be run solo without having implementation details crypto-graphical primordial. Linear and differential cryptanalysis, correlation attacks, algebraic attacks are examples that fall in the generic attack category.

(ix) **Exploits**: These attacks are malware that takes precedence over application software or operating system threats. Such an attack example is known threats in Microsoft Office.

IV. FEATURE ENGINEERING AND CLASSIFICATION METHODS

The feature engineering approach to select the best feature subset from the original set is NP-hard and sometimes computationally complex [36]. However, with proper designing of the feature subset selection method combined with classification algorithms, it will enhance the classifier's performance.

This section aims to describe the characteristics of the proposed Scatter Search and K-Medoid sampling evolutionary metaheuristics for resolving feature engineering problems in classification. The application of scatter search and K-Medoid sampling in feature engineering and that in the Intrusion detection system are not explored by researchers, as we could not find any such from the internet.

Both Scatter search (ScS) established by Glover [37] and Genetic algorithm (GA) launched by Holland [38] are nature-inspired algorithms, population-based, evolutionary optimization methods with some fundamental differences between them. While GA is coined initially to carry out hyperplane sampling, it is also being used for the optimization process. Scatter search is designed to execute as a surrogate constraint relaxation heuristic to find solutions to integer programming problems instead.

Following are some of the steps carried out by nature-inspired evolutionary algorithms to find an optimal solution to hand problems, irrespective of their working procedures.

(i) Establishing, nurturing, and functioning the search techniques with a set of population elements or vectors.

(ii) Generate new vectors by fusing the previously available elements

(iii) Discovering and holding on to the best vectors out of many in step-(i) based on performance quality measures.

A. SCATTER SEARCH

Scatter search [39] is an evolutionary search strategy that tried to solve NP-hard problems successfully. The diverseness and amplification strategies of scatter search make it a promising feature engineering method for optimal feature subset selection [40]. A systematic un-segregated approach is used in Scatter search rather than the volatile process for optimization [41]. The search is complete when the termination condition is above a specified threshold value, or no further performance enhancement is found.

The prime objective of using scatter search is to show the computational experiments result with the best accuracy than others with reduced attributes. While comparing with the Genetic algorithm (GA), it is observed that even though both are evolutionary optimization families, scatter search provides solutions of higher average quality than its GA variants [42].

The pseudo-code for the scatter search optimization is outlined in Figure 1. In Figure 1, the initial population (Pop) shall be a large set incorporating good solutions. The Population size (Pop_size) is calculated as the square of the number of attributes present.

Search Type- Scatter Search

Steps:

1. BEGIN
2. Generate solutions by creating population and its size (pop, pop_size), using diverseness
3. Improve the solution generated in step-2 by using a local search method
4. Obtain the Initial reference set (Ref_set) from the "best m" solutions from Step-3
5. Initialize the current reference set with its size (Ref_set, Ref_set_size)
6. While (new solutions in Refset and objective function assessments is less than the maximum number of assessments)
7. Do{
8. Select _subset(subset, subset_size)
9. Combine solutions(subset, current solution)
10. Improve the solutions obtained in step-10 by using local search until termination criteria are satisfied
11. Update reference set (Refset_updated) by selecting "best m" solutions from the union of current Reference set (Refset) and set of improved solutions from step-11
12. Update the best by choosing the best solution from the updated Refset in step-12 as Current_best
13. If step-13 is true, Best_solution=Current_best
14. new_Refset can be built by combining reduced Refset (the left out after removing worst m/2 solutions from Refset) with m/2 diverse solutions selected from Pop_size improved solutions from steps 2 and 3. This will help to avoid being trapped in local optima.
15. } Until stopping criteria is satisfied

FIGURE 1. Pseudo-code of evolutionary scatter search.

B. K-MEDOID SEARCH

K-Medoids sampling is a supervised similarity-based clustering method used for feature subset selection in the classification process. To eliminate inessential features while comparing with other features, feature engineering via clustering similarity may be embraced for implementation. The most popular prototype-based K-means clustering, which provides globular-shaped clusters, is a choice to achieve this. Still, the strain in defining the initial centroid and number of clusters (K) in K-means clustering goes against it.

The Medoids are used as information points for K-Medoids sampling rather than centroid, as was K-Means clustering. Hence, we propose K-Medoids as a surrogate to K-Means clustering for unsupervised feature subset selection.

The validation of feature subset selection is done by using simplified silhouette (SS) criteria for its computational advantages over original silhouette (Sil), even though both have the value ranging from -1 for wrong clustering to +1 for very well clustering and if around zero, then the observation may fall in either of the two clusters. The original silhouette [43] describes the average distance of a data point to others in the same cluster and all other data points in the neighboring cluster with a computational cost of $O(mn^2)$. Simplified silhouette(SS), on the other hand, measures the distance of a data point to a cluster as the distances between data points and cluster Medoids (or centroids in case of K-means clustering) with a complexity of O(mkn). Here, n represents the number of data points, m is the number of features in each data point, and k is several clusters. For several clusters that are less than the number of data points, a simplified silhouette is used for its computational simplicity compared to the original silhouette, which is the basis for choosing SS in K-Medoid sampling for the feature engineering process. Further, supervised SS filter (S3F) [43], a variation of SS is available, which measures cross-correlations between the independent variables, but it is reported that no substantial improvement in supervised computational complexity is achieved with S3F [44].

The silhouette value enables us to understand the clustering's quality by estimating the average distance between clusters under observations. The silhouette chart exhibit how near data points in one cluster to the points in adjoining clusters. The sampling strategy for K-Medoids clustering works iteratively with k-initial clusters, and then cluster updates

Search Type: K-Medoids Search with Simple Silhouette Filter (SSF)

Steps:

1. Initialization step: Initially select the number of clusters (K) and number of different initial partition (m_p) of features for K-Medoids
2. Cluster Medoids of current partition is calculated, as a part of K-Medoids sampling procedures below:
3. Initially Simplified silhouette value (SS) is assigned with a negative value (or minus infinity)
4. For each K, generate random initial partitions (m_p) for K clusters and then run K-Medoids for each initial partition generated.
5. The individual silhouette value is measured from step-2.2 for each partition and the best of them are chosen as BSV
6. If BSV is larger than the SS value, then BSV is assigned as a new SS value and the K is updated with new K value for the partition for which BSV is obtained
7. Cluster assignment is done to the features with the nearest Medoid
8. If Medoids for two successive iterations are equal then STOP, otherwise go back to step-2 and continue till no further alteration in the cluster allocation.

FIGURE 2. Procedure for K-Medoids sampling search technique.

are made based on some randomized inter-changes [45]. In this paper, the Symmetrical uncertainty distance function is used to obtain the closest representatives for each data point and then improve during the interchange process. This process continues till termination criteria are satisfied. The pseudo-code for K-Medoids is presented in Figure 2.

Here, the limitation of pre-assigning K-value in K-Medoids is nullified using SSF, where the number of clusters can automatically be assigned [43]. Earlier, researchers use the hit and trial method for various K and then select the best K for maximum silhouette value. Still, there was always a chance of being trapped in sub-optimal solutions. In SSF, K's minimum and the maximum value are chosen, and the final value of K is selected within that range to obtain the best silhouette value.

C. CLASSIFICATION METHODS

This unit discusses the potential applications in detecting IoT Botnet attacks. The popular machine learning and deep learning algorithms were considered for envisaging their suitability in IoT network intrusion detection systems such as A2DE (Averaged two-dependence Estimator), JChaid* (Decision tree), MLP classifier (Multilayer Perceptron), and Convolutional neural network (CNN).

D. AnDE (AVERAGED N-DEPENDENCE ESTIMATOR)

AnDE (Averaged n-dependence Estimator) [46] with n being the level of independence is a semi-naïve Bayesian method that softens the conditional probability independence assumption most of its theoretical and computational requirements. AnDE hypothesizes naïve Bayes (NB) as A0DE and A0DE equivalent to A1DE to the next level of dependence. The proposed A2DE, for AnDE, n = 2; while keeping the naïve Bayes method's properties, it is different from other naïve Bayes variants with minimum computational overhead and classification considering the conditional dependence of the contained attribute on any two features excluding class attribute. Further, it is observed that A2DE improves the classification accuracy in a large dataset [46]. The working principle of A2DE is as follows: Initially, during the training phase, A2DE initializes the number of records, constants, probability estimation values, and the joint frequency values. Then, in the prediction phase, the A2DE classifier predicts the class by even outsourcing one dependence estimator to the number of records available in the dataset. Further, the low bias characteristics and single-pass learning through training data of A2DE make it popular for big data analytics with high accuracy [47], linear time complexity but with high variance, and increased space complexity [48].

E. JChaid* DECISION TREE ALGORITHM

JChaid* [49] exploratory model is a recent upgrade to Chi-square automatic interaction detection (CHAID) decision tree induction algorithm where the co-relationships among the predictor class variable and dependent variables are measured in an automated process to build the classification tree. The original CHAID algorithm created by Gordon V. Kass in 1980 is a non-parametric method (also known as XAID) that works on adjusted significance testing (Bonferroni testing) for classification through interaction among the variables in the dataset [50]. For ordinal dependent variables, the JChaid* algorithm uses the F-statistical test to

check any other significant class predictor variable's presence to a dependent variable. In contrast, the chi-square test is used if the dependent variable is categorical.

While comparing with other popular decision tree methods such as Classification and Regression Trees (CART), JChaid* decision tree suits our application for the following reasons:

- JChaid* is capable to deal with a dataset having a dependent variable either categorical type with two or more categories and/ or continuous ones and categorical independent variable type, whereas CART can only deal with a dataset those have binary or continuous dependant and independent variable types.
- Next, while CART performs binary split only by default, JChaid* does multiway split. The depth of the tree is obtained accordingly, to get the knowledge of the bias (underfitting) and variance (overfitting) while selecting the decision tree model.
- Further, JChaid* presents a better feature selection process by preventing the overfitting problem. This is because JChaid* splits a node if a significance criterion is satisfied. In comparison, for the IoT botnet dataset, which is imbalanced and noisy, CART may cause overfitting by creating the unstable tree structure for a small change in the dataset and may also create underfitting for the class imbalance.

F. MULTILAYER PERCEPTRON (MLP) NEURAL NETWORK CLASSIFIER

The multilayer perceptron is a finite directed acyclic graph that is the most common neural network architecture to produce a good generalization. The trained model can provide unerring output for the new, unseen inputs [51]. To enhance the performance of the MLP neural network model's generalization ability by avoiding overtraining, early stopping criteria using a separate validation set and regularization method are proposed by many researchers. Early stopping criteria in MLP classifier give an idea about the number of iterations that can be run before the model begins to overfit. In this evaluation, Levenberg–Marquart (LM) algorithm based on the gradient descent method can provide faster convergence to the model built.

Further, the MLP classifier uses the L2 regularization method to avoid overfitting during training by penalizing weights in sizeable magnitude. When MLP classifier is used for classification, predicting accuracy with low cross-entropy loss function can be achieved by doing feature scaling or standardizing the feature with zero mean and variance = 1. Further, the time complexity of the MLP classifier is O(m.n.z.i.kp), where m training samples, n features, p hidden layers, each containing k neurons, z output neurons, and the number of iterations (i).

G. DEEP LEARNING VIA MULTILAYER PERCEPTRON (DMLP)

DMLP, also known as a deep feedforward neural network, is considered an archetypal deep learning model where backpropagation trains the model. For optimization of the process, MLP uses logistic loss function and quadratic penalty with Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm as limited- main memory (LM) based family of the Quasi-Newton method (QNM), as optimization routine. To train a deep MLP (DMLP) classifier, the following parameters are to be set properly: number of hidden layers (should be chosen intelligently, as the higher value can increase the training time), batch size, activation functions in the hidden and output layers, seed size, loss function, convergence tolerance value (a very small value of tolerance can lead to high predictive accuracy but with more number of iterations), a quadratic penalty by ridge factor and pool size to obtain a parallel calculation of loss function and gradient when multiple CPU cores are present. The performance of the DMLP largely depends upon the optimal hyperparameter setting and large training data size.

H. CNN (CONVOLUTIONAL NEURAL NETWORKS)

Convolutional Neural Networks (CNN) is a well-liked deep learning model, has pulled off many state-of-the-art executions on classification tasks [52]. Initially, CNN found applications in image processing where it differs in having a convolutional filter (or kernels) compared to the fully connected neural network. There are three components in a CNN: convolutional layer, pooling layer, and output or classification layer. While the convolutional layer and pooling layer are used for feature extraction, the output classification layer connected at the end of the network performs intrusion detection. Both forward and backward propagation is used during the training of CNN. During forward propagation, the input sample is fed into the CNN, and the output is observed and compared with the actual. Then the error (if any) is minimized by backpropagation by updating network parameters [53], [54]. Whether simple, deep, and intricate, the architecture of CNN largely depends on the problem at hand regarding design and number of convolutional layers. In CNN, we use a 2×2 max pooling layer to downscale the input samples by 2, picking the maximum value in the pooling layer sub-region, resulting from which convolutional layer is subtle to features by a factor of 2 in comparison to the previous layer. The fixed filter size is used; however, a suitable filter size may be chosen for the best result [55]. During training, hyper-parameter settings, including weights, bias, learning rate, mini-batch size, weight penalty, number of iterations, input, and hidden layer dropout rate, etc., are to be tuned optimally, so that the performance of CNN is maximized. To minimize the error, multithreaded mini-batch stochastic-gradient descent is used during training CNN.

The motivation behind proposing a deep learning classifier in comparison to traditional machine learning ones is its superior performance in dealing with large datasets and its capability to automatic data extraction from complex depictions.

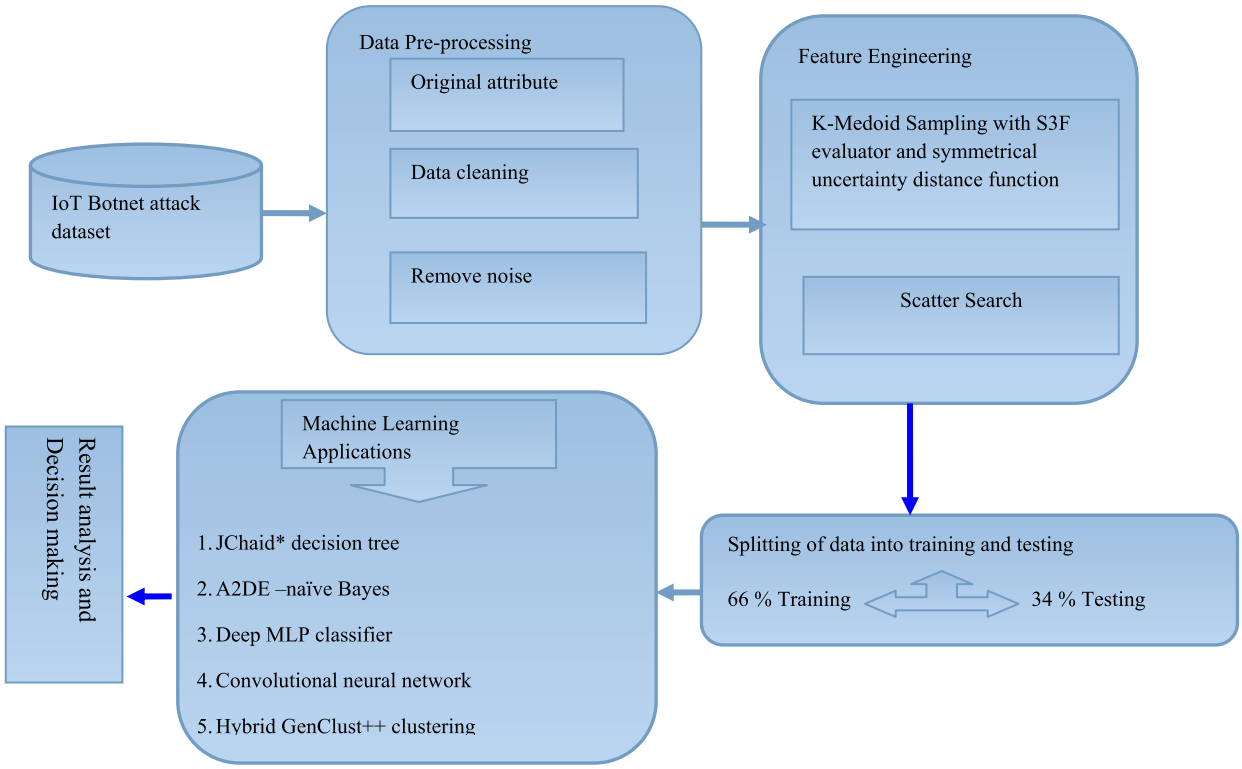

FIGURE 3. Proposed experimental setup.

I. HYBRID GENETIC ALGORITHM WITH K-MEANS CLUSTERING (HGC)

Unsupervised learning or clustering is used for classification using similarity checks among data records. For simplicity in operation, K-means and Fuzzy C-means (FCM) clustering algorithms were found immense popularity in machine learning applications. Still, the clustering quality effectively depends on the centroid's tenacity and an optimal number of clusters. To alleviate such problems, a hybrid GenClust method that combines K-means clustering with a genetic algorithm to enhance the performance of K-means in the classification process via clustering is proposed [56]. In this, clustering quality is improved in each generation so that K-means can attain the optimal global solution, rather than trapped in local optima, as was in the earlier case. Further, the cluster number is determined randomly between 2 to a maximum of the sum of objects in the training dataset.

Similarly, the centroid placement is obtained in chromosome form, where half of the chromosomes are used in the random form and the other in the deterministic form. Rahman *et al.* [57] enhanced the previously developed GenClust and renamed it GenClust++ (HGC) with the mediation of fast hill-climbing cycles of K-MEANS. The genetic operator is not only faster than its predecessor but also generates better clustering quality. Further, the computational complexity of HGC is O(n) in selecting the initial population.

The main motivation behind using HGC is to explore the effectiveness of a clustering algorithm in the detection of attacks in a large IoT botnet dataset.

V. PROPOSED APPROACH

The proposed experimental framework is shown in Figure 3. All the experiments are carried out in an HP pavilion laptop with 1TB HDD, 8GB RAM, 2.67GHz CPU in windows 10 environment using Python and Java.

Initially, the UNSW-NB15 dataset is collected from the UCI machine repository that contains 1, 75, 413 training samples with 45 features [35]. The dataset is extremely noisy, and hence proper care is taken to replace the missing values with the global most common attribute value from each column to fill the NaN (Not a Number) values as a data pre-processing task to make it ready for applications. Further, the feature engineering approach with K-Medoid sampling and meta-heuristic scatter search are used to find the best features. It is needless to state that the best features with appropriate machine learning will enable the administrator to make efficient and effective decision-making.

Here, five different classifiers: JChaid* decision tree, A2DE Bayesian, Deep MLP, CNN, and unsupervised HGC are used to understand the efficacy of the proposed approach.

The parameter settings of all the classifiers are provided below for a better understanding of the proposed approach.

A. PARAMETER SETTINGS OF JChaid* AND A2DE CLASSIFIER

For JChaid* classifier; batch size = 100, a minimum number of instances per leaf = 02, minimum description length correction is used with confidence factor = 0.25(minimum value

ensures more pruning) and significance level = 0.05 are used. For A2DE also, a bath size of 100 is used for experiments.

B. PARAMETER SETTINGS OF DMLP CLASSIFIER

The sigmoid activation function is used in the output layer, whereas soft plus or logistic activation function in the hidden layer as $f(x) = \ln(1 + e^x)$, loss function as Squared error for MLP Classifier: $\text{loss}(a, b) = (a\text{-}b)^2$, pool size, as number of cores in the CPU = 1, ridge penalty factor for the quadratic penalty on the weights = 0.01, tolerance parameter for the delta values = 1E-06, number of hidden units = 2 (more hidden units does not guarantee the best accuracy), batch size = 100 as number of instances to be processed for the batch prediction that is preferred only to improve the runtime for larger datasets. Feature engineering is used for converting Nominal attributes to binary using an unsupervised filter, and missing values are replaced globally.

C. PARAMETER SETTINGS OF CNN

Deep neural network (Convolutional Neural Network) is implemented in this research with dropout regularization and Rectified Linear Units, where training is done with multithreaded mini-batch gradient descent. Here, we use two convolutional layers with patch size 5 × 5 and pool size 2 × 2, each with 20 and 100 feature maps. Mini batch size of 100 is chosen, along with hidden layer dropout rate = 0.5, input layer dropout rate = 0.2, max iterations = 1000 and weight penalty = 1E-08.

D. PARAMETER SETTINGS OF HGC

Initial population size for genetic algorithm = 30, the max iterations for final k-means = 50, max iterations for initial k-means = 60, number of genetic algorithm generations = 60, random number seed used = 10, start-chromosome-selection-generation (the generation after which to start using the chromosome selection operation) = 50.

E. PERFORMANCE METRICS

In this research, we propose to use micro and macro averages to Precision, Recall and F-score apart from classification accuracy to conclude with an intelligent decision making for selecting the best ML/DL model to efficiently detect IoT botnet attacks. From the literature survey, it is observed that to date, no researchers in IoT botnet detection used these metrics to evaluate the classifier performances, which motivated us to consider those measures to evaluate our proposed approach. Further, computational complexity in terms of training and testing times is considered for the effectiveness of the model.

To evaluate the model, a confusion matrix is used to juxtapose the actual target value (intrusion or normal) with ML/DL model predicted values. This way, the confusion matrix presents a comprehensive view of model performance in terms of decision-making with kinds of errors incurred.

From the obtained confusion matrix, the following crucial terms are calculated: true positives (TP)-model predicted value matches with the actual ones; false positives (FP or type-1 error)-actual value was negative but model predicted as positive; True negative (TN)-actual negative value matches with the model predicted negative value and False negative (FN or Type-2 error) - The model predicted a negative value while actually, it is positive.

Now, the several commonly used performance metrics are defined in equation (1) to equation (4).

$$\text{Accuracy} = {}^{(TP+TN)}/_{(TP+FP+TN+FN)} \tag{1}$$

$$\text{Precision} = {}^{TP}/_{(TP+FP)} \tag{2}$$

$$\text{Recall} = {}^{TP}/_{(TP+FN)} \tag{3}$$

$$\text{F-1score} = 2 * \left(\frac{(\text{Pr}\,ecision \times \text{Re}\,call)}{(\text{Pr}\,ecision + \text{Re}\,call)} \right) \tag{4}$$

Now, it is opined by many researchers that accuracy is not the best metric all the time, as it gives an indication about the True Positives and True negatives(with high misclassification cost) but, in intrusion detection model, FN and FP poses the serious consequences. Hence, F1-score is proposed which takes FN and FP into account for efficient classification of intrusions for an imbalanced and noisy dataset. Similarly, precision is preferable when FP is more important than FN, and recall is used in cases where FN outshines FP. 100% precision indicates that 100% of the predicted positive cases are turned out to be actual positive cases. Similarly, 100% recall means 100% of the actual positive cases are correctly predicted by the model. In real-life applications, it is seen that when precision goes up, recall decreases and vice-versa, hence F1-score is coined which amalgamate both and can be calculated as the harmonic mean of the precision and recall. F1-score is also fuzzy in the sense that one is in dilemma to understand whether Recall or precision is maximized by the classification model, but a good F1-score (perfect when F1-score = 1; failure, when F1-score = 0) indicates that the model is correctly identifying intrusions with low false alarms (low FP and low FN).

Since 100% accuracy alone is not a good idea to judge the efficiency of the classifier model for possible chances of overfitting, several other metrics such as precision, recall, and F1-score need to be investigated. So, we propose to evaluate the classifier with other metrics such as micro and macro averaged precision, recall, and F1-score [58] to get more insights.

A macro-average will enumerate the metric independently for each class and then take the mean by treating all classes equally including minority or rare classes, whereas a micro-average metric will bundle the contributions of all classes to compute the average metric by favoring the majority class. These matrices are defined in equation (5) to equation (10).

$$\text{Micro averaged precision} = {}^{(TP1+TP2)}/_{(TP1+TP2+FP1+FP2)} \tag{5}$$

TABLE 1. Experimental results in IoT Botnet UNSW-NB15 dataset with 1, 75,341 samples & 45 attributes.

Metric/Classifier	JChaid*		A2DE		DMLP		CNN		HGC
Feature Engineering*	KM	ScS	KM	ScS	KM	ScS	KM	ScS	ScS
Training time in seconds	6.09	8.2	0.78	1.53	21.32	4.7	118.92	163.24	60.28
Testing time in seconds	0.06	0.38	0.2	10.73	0.53	0.61	1.43	1.19	0.56
Micro Averaged precision (%)	92.465	99.62	92.46	99.61	72.61	**100**	69.71	**100**	**100**
Micro Averaged Recall (%)	92.465	99.62	92.46	99.61	72.61	**100**	69.71	**100**	**100**
Micro Averaged F1-score (%)	92.465	99.62	92.46	99.61	72.61	**100**	69.71	**100**	**100**
Macro Averaged Precision (%)	94.1	99.7	94.1	99.482	48.1	**100**	84.45	**100**	**100**
Macro Averaged Recall (%)	88.7	97.9	88.75	99.61	61.9	**100**	52.5	**100**	**100**
Macro Averaged F1-score (%)	90.8	99.5	90.8	99.55	62.45	**100**	52.05	**100**	**100**

*KM-K-Medoids; ScS- Scatter search

$$\text{Macro averaged precision} = {}^{(P1+P2)}/_{2} \tag{6}$$

$$\text{Micro averaged Recall} = {}^{(TP1+TP2)}/_{(TP1+TP2+FN1+FN2)} \tag{7}$$

$$\text{Macro averaged Recall} = {}^{(R1+R2)}/_{2} \tag{8}$$

$$\text{Micro averaged F-1 score} = 2^{*}\left(\frac{(Micro\_avg\_\text{Pr}ecision \times Micro\_avg\_\text{Re}call)}{(Micro\_avg\_\text{Pr}ecision + Micro\_avg\_\text{Re}call)}\right) \tag{9}$$

$$\text{Macro averaged F1-score} = {}^{(F1+F2)}/_{2} \tag{10}$$

Here, it is to be noted that the micro average is calculated for all the classes (class 1 = normal, class 2 = Attack) and the macro average is the average of precision, recall and F-score for each class.

Further, as described in [58], macro-average to precision, recall, and F1-score are more sensitive to the class-imbalance problem in comparison to their respective micro-averages, they produce less score. It is also suggested that macro averaged metric to be chosen for the sense of the effectiveness of the model while micro average metric shall be used when large classes are available in a test data collection [58], but at the end, the real decision lies in relative less misclassification cost for the classes.

VI. RESULTS AND DISCUSSION

The experimental results obtained are provided in Table 1. The original 45 attributes, including class labels, are reduced to 5 and 3 numbers after applying a feature engineering approach with K-Medoid sampling and scatter search respectively.

It can be observed from Table 1 that micro averaged precision, micro averaged recall, and micro averaged F1-score were the same but at the same time feigned high, ranging from (69.71%, 100%) for CNN; (72.61%, 100%) for DMLP, (92.465%, 99.62%) for JChaid* and (92.46%, 99.61%) for A2DE using K-Medoid and Scatter search feature engineering respectively. The results obtained with the use of scatter search and deep learning methods (DMLP and CNN) are encouraging with 100% micro-averages to precision, recall, and F1-score which makes them promising for most correctly identifying the threats in IoT botnet scenario. Next, as suggested in [58], to understand the effectiveness of the ML/DL model, macro metrics-based evaluations were proposed in this research to gain further insights. Since deep learning-based classifiers (DMLP and CNN) and HGC presents 100% accuracy, they were considered for comparisons with other existing works available to date.

Macro averaged precision increasing from micro averaged precision insinuated that the false positive to true positive ratio is going down. The macro averaged recall decreasing inferred that the ratio of the false-negative to true positive is going up. This suggested that the false positives are going down for these sets of experiments, and the false negatives are going up. Further, the macro averaged F1-score is going down for A2DE and JChaid* classifier but with no change for DMLP, CNN, and HGC.

From the obtained experimental results as shown in Table 1, it is observed that the scatter search (ScS) method combined with all classifiers outperforms the K-Medoid sampling-based counterpart at the cost of more training and testing time. This suggests the ScS method is a better feature selection method in comparison to K-Medoids sampling.

TABLE 2. Comparison with other's work (I).

Metric/Classifier	RU-ADASYN (local machine) [59]	RU-SMOTE (local machine) [59]	RU-ADASYN (AWS with Spark) [59]	RU-SMOTE (AWS with Spark) [59]	Ours ScS + CNN	Ours ScS + DMLP
Macro Average Precision %	39.98	41.11	36.87	36.01	**100**	**100**
Macro Average Recall %	76.42	76.31	73.77	74.69	**100**	**100**
Macro Average F1-score %	44.36	44.9	38.65	39.19	**100**	**100**
Training Time in Seconds	1713.95	1498.34	186	186	**163.24**	**4.7**

Further, from Table 1, while comparing with other machine learning approaches, it is found that the clustering approach (HGC) with ScS presents a 100% score to all micro and macro averaged metrics, but it left out with many unclassified instances, which is a cause of concern while choosing this model for efficient IoT botnet detection.

Next, it is also observed from Table 1 that combining ScS with the deep learning-based methods such as Deep MLP classifier and Convolutional neural network classifier produces 100% micro and macro-averaged precision, recall, and F-score by both, which makes them equally suitable in IoT botnet detection. However, CNN + ScS takes a little more time (163.24 seconds, 1.19 seconds) in comparison to DMLP + ScS (4.7 seconds and 0.61 seconds) for building and classifying the model respectively, makes DMLP + ScS the winner in our proposed research.

While comparing with other's related work, it can be inherited from Table 2 that our proposed scatter search based CNN and DMLP approach outperforms both random under-sampling with Synthetic Minority Oversampling Technique (RU-SMOTE) [59] and Random under-sampling with Adaptive Synthetic Sampling Method (RU-ADASYN) [59] performed on the local machine as well as in AWS spark big data environment with 100% in each macro-averaged precision, recall, and F1-score, whereas RU-SMOTE (local machine), RU-SMOTE (AWS with Spark), RU-ADASYN (local machine) and RU-ADASYN (AWS with Spark) [59] produces (39.98%,76.42%,44.36%), (36.01%,74.69%, 39.19%), (39.98%,76.42%,4 4.36%) and (36.87%, 73.77%, 38.65%) for macro-averaged precision, recall, and F1-score respectively. Also, it is quite evident from table 2 that DMLP + ScS is the fastest among all the methods by taking only 4.7 seconds to build the model followed by CNN with 163.24 seconds, in comparison to 1498.34 seconds by RU-SMOTE (local machine), 186 seconds by RU-SMOTE (AWS with Spark), 1713.95 seconds by RU-ADASYN (local machine) and 186 seconds by RU-ADASYN (AWS with Spark).

From Table 3, It is also verified that the proposed scatter search-based CNN and DMLP approach outperforms all the state-of-the-art classifiers [23] including Naïve Bayes (NB), Decision tree (DT), Association rule mining (ARM), and Artificial neural network (ANN). The proposed DMLP + ScS and CNN + ScS both produce 100% accuracy in IoT botnet detection, whereas the accuracy of 72.73%, 93.23%, 86.45%, and 63.97% are obtained by using NB, DT, ARM, and ANN respectively [23].

To have more validation of our obtained results, it is compared with Random forest (RF) [24], Support vector machine (SVM) [24], Gaussian Naïve Byes (GNB) [24], and Extra trees as a variant of Random forest [60] which is presented in Table 4.

From Table 4, it is inherited that our proposed approach seems to perform the same with random forest [24] with 100% accuracy while outperforming SVM (91%) [24] and GNB (92%) [24]. Further, it is seen that our proposed approach DMLP + ScS and CNN + ScS along with random forest [24] produces 100% to precision, recall, and F1-score followed by extra trees [60] with a score of 99%, 98%, and 98%; NB [24] (91%, 79%, and 88%) and GNB [24] (92%, 88%, and 91%) respectively.

Research Challenges Addressed:

1.1 Since there is scarcity in systematic IoT-botnet datasets, it's challenging to conclude the available ones. Still, NSBW-15 is a recent data that is used in this paper for experimental work.

2.2 Efficient feature engineering models are sought to address the resource consumed by complex models, which is successfully addressed in this research using K-Medoids and Scatter search.

3.3 Lower detection accuracy is addressed using efficient deep neural network-based classifiers (DMLP and CNN) and classification via clustering by combining Genetic algorithms and K-Means clustering (HGC) to avoid the problems that occur due to imbalanced data. Hybrid Genetic algorithm and K-Means clustering

TABLE 3. Comparison with other's work (II).

Classifier	ARM [23]	DT [23]	NB [23]	ANN [23]	Ours ScS+CNN	Ours ScS+DMLP
Accuracy %	86.45	93.23	72.73	63.97	**100**	**100**

TABLE 4. Comparison with other's work (III).

Model	Accuracy	Precision	Recall	F1-score
Random Forest [24]	**1**	**1**	**1**	**1**
SVM [24]	0.91	0.91	0.79	0.88
GNB [24]	0.92	0.92	0.88	0.91
Extra Trees (Random Forest) [60]	--	0.99	0.98	0.98
Ours (ScS+CNN)	**1**	**1**	**1**	**1**
Ours (ScS+DMLP)	**1**	**1**	**1**	**1**
Ours (ScS + HGC)	**1**	**1**	**1**	**1**

could not classify some instances, but no incorrectly classified instances.

4.4 Our proposed approach successfully addresses the lightweight model for detecting IoT-Botnet attacks with low computational power, less training time, low test time, and high detection rate.

5.5 Even though the research work in this field is going on, still low performance in the real world environment is a concern. Hence, more research to be done as a future scope to explore further possibilities.

VII. CONCLUSION AND FUTURE SCOPE

We have proposed a feature engineering-based machine learning/ deep learning solution for the early detection of IoT botnets in-home networks. A comprehensive performance evaluation of our proposed deep learning approaches (DMLP and CNN) and unsupervised HGC combining with efficient scatter search (ScS) based feature engineering method achieves 100% accuracy with zero false alarm rate in detecting network intrusions.

To gain further insights, we used macro averaged precision, recall, and F1-score to evaluate the proposed approach apart from micro-averaged ones and compared it with several existing works in the recent literature. From extensive analysis with existing literature, it is concluded that our proposed DMLP + ScS is the winner followed by CNN + ScS (Both with 100% accuracy, 100% macro and micro-average to precision, recall and F1-score) in comparison to SVM, ARM, GNB, Extra Trees, A2DE, JChaid* tree, HGC, RU-SMOTE (local and AWS with Spark) and RU-ADASYN (local and AWS with Spark). Apart from being most accurate and zero false alarm rate by DMLP + ScS, it is fastest with less training time and testing time (4.7 seconds, 0.61 seconds) makes it a lightweight approach. Looking into the importance of a reliable dataset to build an efficient ML/DL model, this research encourages for further investigation to inspect the adequacy of the proposed approach with unalike datasets. In the future, the authors intend to apply edge computing, cloud computing and blockchain integration with ML/DL approaches in the real-time scenario to find its suitability for efficient detection of IoT-botnet cyber-attacks.

REFERENCES

[1] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT Goes nuclear: Creating a ZigBee chain reaction," *IEEE Secur. Privacy*, vol. 16, no. 1, pp. 54–62, Jan. 2018.

[2] D. Fisher. *Pair of Bugs Open Honeywell Home Controllers Up to Easy Hacks*. Accessed: Apr. 2021. [Online]. Available: https://threatpost.com/pair-of-bugs-open-honeywell-home-controllers-up-to-easy-hacks/113965/

[3] B. Fouladi and S. Ghanoun. (2014). Honey, I'm home!!, hacking Z-wave home automation systems. Black Hat USA. [Online]. Available: https://cybergibbons.com/wp-content/uploads/2014/11/honeyimhome-131001042426-phpapp01.pdf/

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)* Vancouver, BC, Canada: USENIX Association, 2017, pp. 1093–1110.

[5] S. Soltan, P. Mittal, and H. V. Poor, "Black IoT: IoT botnet of high wattage devices can disrupt the power grid," in *Proc. 27th USENIX Secur. Symp. (USENIX Secur.)* Baltimore, MD, USA: USENIX Association, 2018, pp. 15–32.

[6] T. Yeh, D. Chiu, and K. Lu. (2017). Persirai: New Internet of Things (IoT) botnet targets IP cameras. TrendMicro. [Online]. Available: https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/

[7] P. Amini, M. A. Araghizadeh, and R. Azmi, "A survey on botnet: Classification, detection and defense," in *Proc. Int. Electron. Symp. (IES)*, Sep. 2015, pp. 233–238.

[8] S. Dange and M. Chatterjee, "IoT botnet: The largest threat to the IoT network," in *Data Communication and Networks*, vol. 1049, L. C. Jain, Ed. Singapore: Springer, 2020, pp. 137–157.

[9] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 898–924, 2nd Quart., 2014.

[10] N. Negash and X. Che, "An overview of modern botnets," *Inf. Secur. J., Global Perspective*, vol. 24, nos. 4–6, pp. 127–132, Dec. 2015.

[11] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," 2018, *arXiv:1807.11023*. [Online]. Available: http://arxiv.org/abs/1807.11023

[12] W. S. Hamza, H. M. Ibrahim, M. A. Shyaa, and J. Stephan, "IoT botnet detection: Challenges and issues," *Test Eng. Manage.*, vol. 83, pp. 15092–15097, Mar./Apr. 2020.

[13] M. Stevanovic and J. M. Pedersen, "Detecting bots using multi-level traffic analysis," *Int. J. Cyber Situational Awareness*, vol. 1, no. 1, pp. 182–209, Dec. 2016.

[14] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the Internet of Things," in *Proc. IEEE World Congr. Services*, New York, NY, USA, Jun. 2015, pp. 21–28.

[15] M.-O. Pahl and F.-X. Aubet, "All eyes on you: Distributed multi-dimensional IoT microservice anomaly detection," in *Proc. 14th Int. Conf. Netw. Service Manage. (CNSM)*, Rome, Italy, 2018, pp. 72–80.

[16] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," in *Proc. ISCIS Secur. Workshop, Imperial College London Recent Cybersecur. Res. Eur.*, vol. 821, 2018, pp. 79–89.

[17] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[18] B. Usmonov, O. Evsutin, A. Iskhakov, A. Shelupanov, A. Iskhakova, and R. Meshcheryakov, "The cybersecurity in development of IoT embedded technologies," in *Proc. Int. Conf. Inf. Sci. Commun. Technol. (ICISCT)*, Nov. 2017, pp. 1–4.

[19] E. Anthi, L. Williams, and P. Burnap, "Pulse: An adaptive intrusion detection for the Internet of Things," in *Proc. Living Internet Things, Cybersecur. IoT*, London, U.K., 2018, pp. 1–4, doi: 10.1049/cp.2018.0035.

[20] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "IoT healthcare analytics: The importance of anomaly detection," in *Proc. IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2016, pp. 994–997.

[21] R. Kozik, M. Choraś, M. Ficco, and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *J. Parallel Distrib. Comput.*, vol. 119, pp. 18–26, Sep. 2018.

[22] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059.

[23] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques," in *Mobile Networks and Management* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 235, J. Hu, I. Khalil, Z. Tari, and S. Wen, Eds. Cham, Switzerland: Springer, 2017, pp. 30–44, doi: 10.1007/978-3-319-90775-8_3.

[24] A. Kumar, M. Shridhar, S. Swaminathan, and T. J. Lim, "Machine learning-based early detection of IoT botnets using network-edge traffic," Oct. 2020, *arXiv:2010.11453*. Accessed: Feb. 20, 2021. [Online]. Available: http://arxiv.org/abs/2010.11453

[25] A. Derhab, A. Aldweesh, Z. A. Emam, and F. A. Khan, "Intrusion detection system for Internet of Things based on temporal convolution neural network and efficient feature engineering," *Wireless Commun. Mobile Comput.*, vol. 2020, Dec. 2020, Art. no. 6689134, doi: 10.1155/2020/6689134.

[26] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *J. Big Data*, vol. 8, no. 1, pp. 1–41, Dec. 2021, doi: 10.1186/s40537-020-00390-x.

[27] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based IoT intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur. (DISS)*, San Diego, CA, USA, Feb. 2020, pp. 1–7. [Online]. Available: https://tubiblio.ulb.tu-darmstadt.de/id/eprint/119419

[28] K. C. Lin, S. Y. Chen, and J. C. Hung, "Botnet detection using support vector machines with artificial fish swarm algorithm," *J. Appl. Math.*, vol. 2014, Apr. 2014, Art. no. 986428, doi: 10.1155/2014/986428.

[29] S. T. Brugger and J. Chow. *An Assessment of the DARPA IDS Evaluation Dataset Using SNORT*. Accessed: Apr. 2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.674&rep=rep1&type=pdf

[30] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 1–6.

[31] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[32] UNIBS. (2009). *University of Brescia Dataset*. [Online]. Available: http://www.ing.unibs.it/ntw/tools/traces/

[33] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *IJ Netw. Secur*, vol. 17, no. 6, pp. 683–701, 2015.

[34] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *Proc. Int. Conf. Contemp. Comput.* Berlin, Germany: Springer, 2012, pp. 322–334.

[35] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[36] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, Dec. 1997.

[37] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, Jan. 1977.

[38] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.

[39] M. Laguna and R. Martí, *Scatter Search: Methodology and Implementations in C*. Norwell, MA, USA: Kluwer, 2003.

[40] F. G. López, M. G. Torres, B. M. Batista, J. A. M. Pérez, and J. M. Moreno-Vega, "Solving feature subset selection problem by a parallel scatter search," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 477–489, Mar. 2006.

[41] B. González and B. Adenso-Díaz, "A scatter search approach to the optimum disassembly sequence problem," *Comput. Oper. Res.*, vol. 33, no. 6, pp. 1776–1793, Jun. 2006.

[42] R. Marti, M. Laguna, and V. Campos, "Scatter search vs. genetic algorithms," in *Metaheuristic Optimization Via Memory and Evolution* (Operations Research/Computer Science Interfaces Series), vol. 30, R. Sharda, S. Vob, C. Rego, and B. Alidaee, Eds. Boston, MA, USA: Springer, 2005, pp. 263–282.

[43] T. F. Covões and E. R. Hruschka, "Towards improving cluster-based feature selection with a simplified silhouette filter," *Inf. Sci.*, vol. 181, no. 18, pp. 3766–3782, Sep. 2011.

[44] F. Wang, H.-H. Franco-Penya, J. D. Kelleher, J. Pugh, and R. Ross, "An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity," in *Machine Learning and Data Mining in Pattern Recognition* (Lecture Notes in Computer Science), vol. 10358, P. Perner, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-62416-7_21.

[45] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*, C. Aggarwal and C. Zhai, Eds. Boston, MA, USA: Springer, 2012, doi: 10.1007/978-1-4614-3223-4_4.

[46] G. I. Webb, J. R. Boughton, F. Zheng, K. M. Ting, and H. Salem, "Learning by extrapolation from marginal to full-multivariate probability distributions: Decreasingly naive Bayesian classification," *Mach. Learn.*, vol. 86, no. 2, pp. 233–272, Feb. 2012.

[47] M. S. Islam, M. K. Qaraqe, S. B. Belhaouari, and M. A. Abdul-Ghani, "Advanced techniques for predicting the future progression of type 2 diabetes," *IEEE Access*, vol. 8, pp. 120537–120547, 2020.

[48] S. Chen, A. M. Martínez, G. I. Webb, and L. Wang, "Selective AnDE for large data learning: A low-bias memory constrained approach," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 475–503, Feb. 2017.

[49] I. Ibarguren, A. Lasarguren, J. M. Pérez, J. Muguerza, I. Gurrutxaga, and O. Arbelaitz, "BFPART: Best-first PART," *Inf. Sci.*, vols. 367–368, pp. 927–952, Nov. 2016, doi: 10.1016/j.ins.2016.07.023.

[50] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *J. Roy. Stat. Soc. C, Appl. Statist.*, vol. 29, no. 2, pp. 119–127, 1980. [Online]. Available: http://www.jstor.org/stable/2986296

[51] Y. Kutlu, M. Kuntalp, and D. Kuntalp, "Optimizing the performance of an MLP classifier for the automatic detection of epileptic spikes," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7567–7575, May 2009, doi: 10.1016/j.eswa.2008.09.052.

[52] T. Sun, Y. Wang, J. Yang, and X. Hu, "Convolution neural networks with two pathways for image style recognition," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4102–4113, Sep. 2017, doi: 10.1109/TIP.2017.2710631.

[53] Y. Gao, L. Gao, X. Li, and X. Yan, "A semi-supervised convolutional neural network-based method for steel surface defect recognition," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101825.
[54] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with max-pooling convolutional neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–6, doi: 10.1109/IJCNN.2012.6252468.
[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[56] P. A. Zizwan, M. Zarlis, and E. B. Nababan, "Performance analysis of combined methods of genetic algorithm and k-means clustering in determining the value of centroid," *J. Phys., Conf. Ser.*, vol. 930, Dec. 2017, Art. no. 012008, doi: 10.1088/1742-6596/930/1/012008.
[57] M. Z. Islam, V. Estivill-Castro, M. A. Rahman, and T. Bossomaier, "Combining k-means and a genetic algorithm through a novel arrangement of genetic operators for high quality clustering," *Expert Syst. Appl.*, vol. 91, pp. 402–417, Jan. 2018, doi: 10.1016/j.eswa.2017.09.005.
[58] C. D. Manning, H. Schutze, and P. Raghavan, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[59] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *J. Big Data*, vol. 8, no. 1, pp. 1–41, Dec. 2021, doi: 10.1186/s40537-020-00390-x.
[60] T. Mohamed, T. Otsuka, and T. Ito, "Towards machine learning based IoT intrusion detection service," in *Recent Trends and Future Technology in Applied Intelligence* (Lecture Notes in Computer Science), vol. 10868, M. Mouhoub, S. Sadaoui, O. A. Mohamed, and M. Ali, Eds. Cham, Switzerland: Springer, 2018, pp. 580–585, doi: 10.1007/978-3-319-92058-0_56.

MRUTYUNJAYA PANDA received the B.E. degree in electronics and telecommunication engineering from Utkal University, Bhubaneswar, India, the M.E. degree from the University College of Engineering, Burla, the M.B.A. degree from IGNOU, New Delhi, and the Ph.D. degree in computer science from Berhampur University. He is currently working as a Reader with the Department of Computer Science and Application, Utkal University. He is having 20 years of teaching and research experience. He has authored two textbooks on soft computing and data mining, five edited books, and one special issue. He has published more than 100 research articles in various journals and conferences. His area of research interests include applications of data mining, big data analytics, intrusion detection, mobile communication, social networking, sensor networks, image processing, and natural language processing. He is a Life Member of IETE, India, ISTE, India, and CSI, India. He acts as a PC member of various international conferences. He is an active reviewer of many reputed journals and conferences. He serves as an associate editor for two journals and an editorial board member for many journals.

ABD ALLAH A. MOUSA received the M.Sc. and Ph.D. degrees in engineering mathematics from Menoufia University, Egypt, in 2003 and 2006, respectively. He is currently a Full Professor of engineering mathematics with the Department of Basic Engineering Sciences, Faculty of Engineering, Menoufia University, and the Department of Mathematics and Statistics, Faculty of Science, Taif University, Saudi Arabia. He is the author of more than 90 scientific articles and textbooks in refereed journals and international conferences. His research interests include theory of optimization and its application, numerical analysis, mathematical programming, metaheuristics, and computational intelligence. He has served as an editorial board member and a referee for many reputed international journals.

ABOUL ELLA HASSANIEN is currently the Founder and the Head of the Scientific Research Group in Egypt (SRGE) and a Professor of information technology with the Faculty of Computer and Information, Cairo University. His research interests include computational intelligence, medical image analysis, security, animal identification, and multimedia data mining.