ONLINE ISSN: 3007-3197 | PRINT ISSN: 3007-3189

http://amresearchreview.com/index.php/Journal/about

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 5 (2025)

A Comparative Study for IoT Attack Detection Using Machine Learning Algorithms

Ayesha Jaffar[1], Muhammad Zunnurain Hussain[2], Humera Niaz[3], Muhammad Ahsan[4], Muhammad Zulkifl Hasan[5], Nadeem Sarwar[6]

Article Details

Keywords: ML: Machine learning, IOT: Internet of things, SGD: Stochastic gradient descent

[1]Ayesha Jaffar
PhD Scholar Information Technology University Lahore Pakistan,
phdee23004@itu.edu.pk

[2]Muhammad Zunnurain Hussain (corresponding author)
Dept. of Computer Science Bahria University Lahore Campus
Zunnurain.bulc@bahria.edu.pk

[3]Humera Niaz
Computer Science Department
COMSATS University Islamabad Lahore Campus
humerafaisal@cuilahore.edu.pk

[4]Muhammad Ahsan,
Software Engineering Department. School Of Systems & Technology
University Of Management & Technology, Lahore.
Muhammadahsan@umt.edu.pk

[6]Muhammad Zulkifl Hasan
Faculty of Information Technology, University of Central Punjab Lahore, Pakistan
Zulkifl.hasan@ucp.edu.pk

[7]Nadeem Sarwar
Department of Computer Science, Bahria University Lahore Campus, Lahore, Pakistan
nadeem_srwr@yahoo.com

ABSTRACT

Prior studies on behaviour-based threat detection on Internet of Things (IoT) device networks have generated machine-learning models with a limited and frequently unproven capacity to learn from unseen data. In this study, we provide a generalizability-focused modelling technique for IoT network assaults that also improves detection and performance. Firstly, we develop a multi-step feature selection technique that minimizes overfitting and provides an enhanced rolling window strategy for feature extraction. Second, in order to prevent frequent data leaks that have restricted the generalizability of earlier models, we develop and test our models using separate train and test datasets. Third, we employ a wide range of machine learning models, assessment measures, and datasets to assess our approach thoroughly. Lastly, we employ explainable AI approaches to strengthen the models' confidence, enabling us to pinpoint the characteristics that support precise attack detection. Models are updated gradually by use of algorithms such as Online Naive Bayes and Stochastic Gradient Descent (SGD).

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 5 (2025)

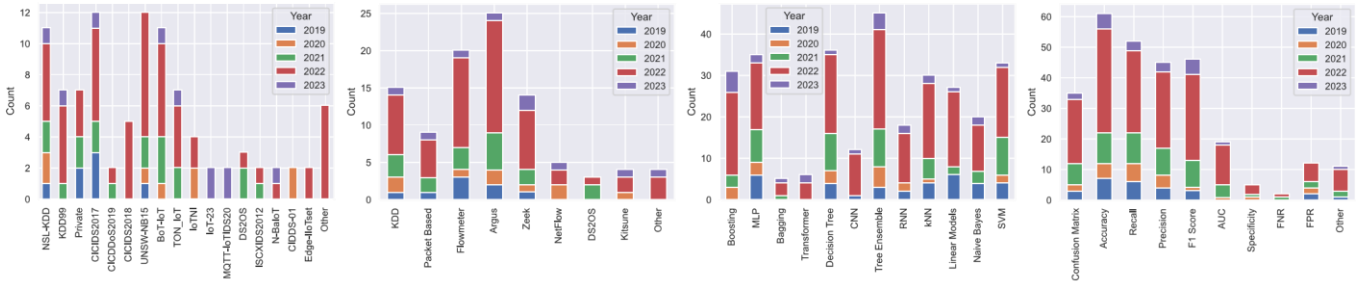
Introduction

Recent articles suggest that an IoT device may get attacked for the first time within five hours of connecting to the network, and a focused attack may be launched on the same IoT device within a day [1]. As devices, IoTs are more challenging to safeguard as their HW/SW/Interface may be utterly dissimilar to one another, and they possess limited power for resources compared to conventional computer devices. As for IoT devices, traditional security solutions will require amendments in many cases [2].

Mentioning that there are numerous ways to safeguard IoT devices, one common practice is based on behaviour analysis and continued monitoring of suspicious vibrations in the network with the help of machine learning (ML). However, several issues and circumstances may frequently make doubt the study's validity [3–5]. Thus, common problems in machine learning research, like data leakage and feature overfitting, which researchers encountered in previous work on IoT threat detection, n are discussed in section II.

In this study, instead of relying on specific heuristic signatures and being susceptible to problems such as these, we use ML techniques for the behaviour-based categorisation of benign and malicious network data. To do this, we provide a method (IoTGeM) that builds on the following contributions to produce models for behaviour-based network threat detection for IoT devices: To do this, we provide a method (IoTGeM) that builds on the following contributions to produce generalisable models for behaviour-based network threat detection for IoT devices: A rolling window feature extraction technique that surpasses conventional techniques with regards to accuracy, as well as early identification.

1) A multi-step feature selection procedure that eliminates elements that may cause overfitting is based on a genetic algorithm with exogenous feedback.
2) Analyzing the effectiveness of the features: The proposed approach uses explainable AI methodologies to establish dependencies between features and attacks.
3) Models are updated incrementally with the help of functions such as Hoefling Trees, Online Naïve Bayes, & Stochastic Gradient Descent (SGD). We furthermore use a rigorous methodology to guarantee the applicability of our models and the simplicity of verifying, duplicating, or expanding our approach [4]:·
4) We furthermore use a rigorous methodology to guarantee the applicability of our models and the simplicity of verifying, duplicating, or expanding our approach [4]:
   We thoroughly assess by combining machine learning algorithms, metrics, and datasets for testing. We shall refrain from applying, for example, only accuracy with data sets that are in particular situations. Our models have been developed and evaluated to mitigate data leakage based on disparate train and test datasets.



(a) Datasets (b) Feature Extraction Methods (c) ML Algorithms (d) Evaluation Metrics

Unbalanced, they should not be applied to the published recently, and it is common practice to use older datasets

Fig. #1: From literature review i: dataset ii: Applied ML models iii: Evaluation results and performance [3], [4], [5]

To assess the performance of the proposed approach, we perform exhaustive testing using various datasets, metrics,

and machine learning algorithms. It also checks that the right metrics are not used when working with datasets that are wrong—for instance, there is no way we would use accuracy when working with imbalanced datasets.
All our scripts are made open for public use, and we utilise data that is freely available to the public. The snapshot of all structures of this article is the following one. Section II also offers a brief literature review that serves as the foundation for subsequent arguments within the current paper. The approaches used in
feature extraction, feature selection, and data selection discussed in this research are described in section III. In section IV, the results of the models are discussed and compared to other methodologies, and a comprehensive summarization of the feature effectiveness for the different assaults is provided. Section V focuses on the limitations, while Section VI brings conclusions.

II. RELATED WORK

This review covers the past several years' pertinent research that focuses on supervised machine learning models for intrusion or anomaly detection for Internet of Things devices. We have categorised this research into four groups based on information, traits, machine learning models, and evaluation standards used in each study. Fig.1 summaries the historical trends in these areas and provides the structure for the discussion in each subsection. For a more thorough list of papers on machine learning methods for IoT-based threat detection, see Table V in the Supplementary Materials (SM).

A. Dataset

The use of datasets in pertinent work published in 2019 is summarised in Fig. 1a. Even if a lot of research has been impossible to investigate new features or extract features using alternate technologies. For example, [6]–[13] used the KDD992 and NSLKDD databases. NSL-KDD is a 1999 publication that is an error-corrected version of KDD99. These are substantial, trustworthy datasets widely used in several studies, making them valuable standards. However, because of their advanced age, they lack the newest technology and weapons. Their applicability to recent network security research is thus severely limited.
IoT devices with tagged and raw data for intrusion detection are found in several real device-based datasets, such as CIC-IoT-2022 [39], IoT-ENV [40], and IoT-NID [41], in addition to simulation-based datasets like BoT-IoT [35], EdgeIIoT [36], TONIoT [37], and MQTT-IoT-IDS2020 [38]. The literature often cites Kitsune [42] and IoT-23 [43]. These datasets are analysed in Section III-Feature Extraction:
The use of various feature extraction algorithms in pertinent work published beginning in 2019 is summarised in Fig. 1b. Most of the datasets used in intrusion detection are gathered with standard tools and methods. Zeek (formerly Bro) [44], Argus [45], and CICFlowMeter (formerly ISCXFlowMeter) [46] are a few instances of often-used tools. For example, the UNSWNB15 dataset was created using Zeek and Argus and used in [22]–[27]. It contains forty-nine attributes. Argus was used to build the 29 features in the Bot-IoT dataset and the 42 features in the ToN IoT dataset. Many datasets, such as KDD99, NSL-KDD, CIDDS-01, DS2OS, MQTTset, N-BaIoT, and InSDN, only provide pre-extracted features; the raw network data, which is stored in a file called pcap (packet capture), is not available.
All 83 features in the CICIDS2017-18-19 datasets were created using the CICFlowMeter. Every one of these tools has features that are built based on flow. Unlike these instruments, Kitsune, which is used in [27]–[35], generates a dataset of 115 features using a sliding window approach on pcap files. Packet-based features are applications that use certain features that are extracted from packets in the network. This method has been used in numerous research studies [31], [34], [36]. It is also possible to list the following datasets: MQTTset, Edge-IIoTset, and MQTT-IoT-IDS20. These datasets likewise include flow-based features. Furthermore, some research [22, 23] employs deep learning models trained on raw network data to detect threats without first transforming them into features. However, because the raw data (pcap file) comprises network packets, these studies suffer from the same issues as those that use individual packet features, such as the unintentional usage of identifying information. Distinguishing qualities are not generalisable, yet they offer clues about the label information. For example, IP addresses are generally traceable since they give the attacker or victim a distinct identity. The fact that these IP characteristics could alter in the case of another attack means

that this trait is unique to that dataset and cannot be broadly used. As a result, employing this feature will result in an overfitting issue, connecting the chosen features to a solitary dataset.

One can find an analysis of the traits, benefits, and drawbacks of the flow, window, and packet approaches. The dataset can also be utilised to create other feature sets using different methods if it has a Pcap file. For example, IoT-NID pcap files were utilised to create the IoTID20 [34] dataset using CICFlowMeter. Flow [35] and sliding window [36] features were thus obtained from the IoT-NID dataset. Nevertheless, the KDD99, NSL-KDD, CIDDS-01, DS2OS, MQTTset, NBaIoT, and InSDN datasets don't contain any pcap files. As a result, fewer studies have been done on feature extraction and modification that can be assessed using these datasets.

B. Machine Learning and Evaluation Metrics

Fig. 1c–1d summarises several machine learning approaches and measurements in pertinent work published in 2019. Boosting, kNN, random forest, decision trees, MLP, SVM, and tree ensembles are the most widely used algorithms in the literature. Traditional machine learning methods have not stopped to some extent, but deep learning techniques have gained popularity recently. For example, since 2022, transformers have been used extensively, and the use of CNN and RNN has increased. Methodology evaluations by researchers have been conducted often using many machine learning models. In numerous studies, including repeated comparisons [11]–[13], [16], [19]–[22], [25]–[29], ensemble models, like RF and XGB, yielded the best results.

The application of artificial neural networks to allocate the second-order ordinary differential equations showcases the work of machine learning to a mathematical problem solving paradigm, which was noted by other researchers [47]. The same can be said about the work done in distributed multi-agent systems which focuses on evaluation techniques and is related to performance metrics that could improve the scalability and adaptability of IoT security models [48]. Moreover, the use of machine learning algorithms in the development of intelligent security policies for wireless networks also aids in building strong security systems for IoT networks and mitigates system vulnerabilities [49]. The effect of virtual reality in medicine provides a glimpse of how life-altering technology can be, but more importantly, demonstrates the scope of machine learning algorithms when applied in real-life problems like the IoT healthcare security system [50]. This is also true for the research done on big data and the Internet of Things (IoT) applied to resource management in construction work. This research aligns with our study as it explains how data-centric approaches can enhance urban infrastructure and make cities smarter and more resilient [51]. Lastly, tackling urban sprawl and pollution using big data for environmental monitoring shows the growing importance of analytics and machine learning in applying technology to help bolster public health and safety as well as the environment in smart cities [52]. All of these studies form a collective research base for applying machine learning for IoT-based security and sustainability of urban environments.

Despite the recent popularity of deep learning algorithms (such as CNN, RNN, and transformers), classical machine learning techniques are still widely used. This is partially explained by the fact that most datasets and research in network security are in circumstances where feature extraction has already taken place and conventional techniques perform well.

Accuracy is the most often used assessment metric. While recall, precision, and F1 score are often included in research studies in addition to accuracy, some studies [7]. Just report accuracy. However, in an area where data distribution is unbalanced, such as attack detection, accuracy is not a reliable metric. Even though the reported accuracies range from 0.771, many investigations demonstrate a success rate of better than 0.99 (see Table V). When the data is not evenly distributed or provides accuracy, these scores may not accurately reflect the model's effectiveness. Generally speaking, errors in network security and machine learning research often raise questions about the reliability of findings reported in the literature (e.g., data). Studies must prioritise high metric scores in this specific context, but they must also demonstrate transparency, reproducibility, and an absence of frequent errors.

III. Methodology

A. Applied ML Models & system design:

To develop a machine learning model that can recognise assaults, we first assess and select the best datasets for our study. Next, we provide our features by balancing the advantages and disadvantages of several feature extraction

methods. After extracting the features, we examine and eliminate some of the obtained features by examining these attributes in different contexts. We call this technique feature selection. The next phase involves optimising hyperparameters to determine which hyperparameters are optimal for each machine-learning model. Finally, we train our models using the selected features and hyperparameters. We then test these models using previously unseen data to obtain our final results. This process is illustrated in Fig. 2.

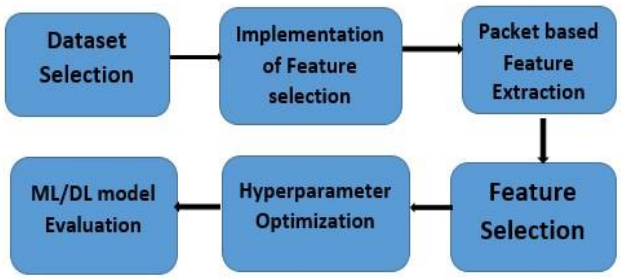


Fig. #2: Implementation of system models

B. Data extraction from Dataset

We examined the datasets used in Internet of Things threat detection studies. We examined the publicly available ones, which have raw data (pcap files) and are labelled. We have measured and categorised them according to the number of devices and types of attacks they contain, as well as whether or not they have several sessions and real-time IoT data. Table I contains a list of them. IoT devices are different from non-IoT devices because they are more likely to have proprietary hardware and software due to their heterogeneous nature. Real IoT devices should be included in datasets instead of simulations because it is very hard to replicate this variation. Repeated attacks throughout many sessions are also desirable since they allow for a more detailed analysis of the attributes of each attack and a comparison of each attack across sessions. It can also be used to prevent session-specific traits from overfitting.

Dataset	Year	Real IoT	Session	Devices	Attacks
BoT-IoT [35]	2018	×	×	5	10
CIC-IoT-2022 [39]	2022	✓	✓	40	4
Edge-IIoT [36]	2022	×	×	9	13
IoT-ENV [40]	2021	✓	×	5	3
IoT-NID [41]	2019	✓	✓	2	10
Kitsune [42]	2018	✓	×	9	9
TON-IoT [37]	2019	×	×	7	9
IoT-23 [43]	2020	✓	×	3	8
MQTT-IoT-IDS2020 [38]	2020	×	×	13	4

Table#1: IOT intrusion detection datasets

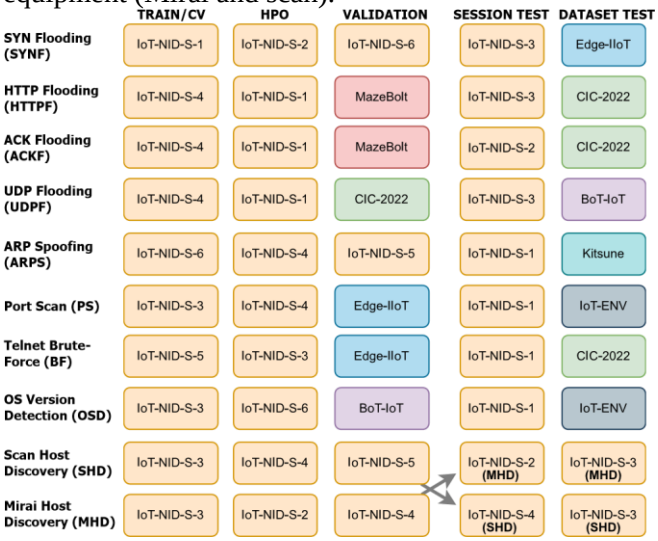
IoT-NID and CIC-IoT-22 are the only datasets containing multi-session IoT data. CIC-IoT22 only has a limited number of attack methods despite having many devices. Additionally, each attack session uses a single device, indicating that this dataset isolates attack scenarios. In contrast, because IoT-NID has many assaults but few devices, it is beneficial for training different attack detection models. As a result, IoTNID serves as the primary dataset for model training.

It is essential to separate training and testing data. We constantly evaluate a model's performance with data that it has never seen before to be sure of this. Fig. 3 shows how data were used in this inquiry. In the -NID IoT dataset, distinct sessions are shown by the same colour in many columns for the same assault. This indicates cases where an attack was

initiated in numerous sessions. The data in the first column is used to train the models. The information in the Train/CV, HPO, and Validation columns is used to choose and improve models; the information in the Session and Dataset Test columns is used to gauge the generality of the models. More specifically, data from the HPO column is utilised for feature reduction and hyperparameter optimisation, whereas data from the validation column is utilised during the feature selection phase.

There is a dearth of information available for specific attacks. For the HTTPS and ACK flood assaults, we used extra data from kb.mazebolt.com because there wasn't enough data to use a different IoT-NID session for validation. These two assaults have a similar layout to the IoT-NID variants. We used slightly altered versions of the attack in the training, validation, and test datasets for BF to compensate for the absence of data for some attacks. More specifically, the test data uses RTSP BF assaults, the validation data uses password attacks, and the training data uses telnet BF attacks. Other datasets did not provide any suitable comparable data for SHD and MHD.

We tested SHD models with an MHD session and vice versa to evaluate the universality of these attack models. This is possible because host discovery assaults, such as SHD and MHD, are the same despite using different methods and equipment (Mirai and scan).



Figure#3: dataset for our study, Cross-validation, training testing and selection of multiple features. [10]

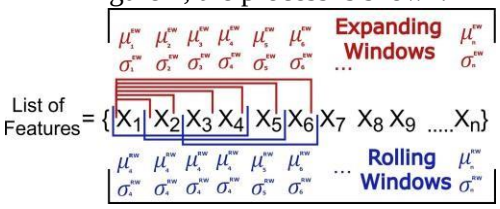
ML Algorithm Selection

There isn't a single solution that solves every issue. Thus, it's critical to consider performance across a range of ML models [4]. We selected each of the subsequent model types [62]: Instance-based: k-nearest neighbours (KNN); ensemble-based: random forest (RF); tree-based: decision tree (DT); Bayesianbased: naïve Bayes (NB); kernel-based: support vector machine (SVM); artificial neural networks (ANN): multilayer perceptron (MLP); logistic regression (LR): linear model. This set of models exhibits a high degree of crossover with the models used in previous studies. We also included XGB, which is rarely found in the attack detection literature but is known to perform well on tabular datasets [63]. We have used a modular code structure and made the source code of our work publicly available1, making it easy to implement other algorithms with only minor code changes.

Methods for Feature Extraction

1. It is typical to discover feature extraction techniques in the literature that are flow- and packet-based. This section outlines these two methods and offers a different window-based technique.
2. Specialized Packet-Based Features: By looking at the payload or header of each network packet, specific packet attributes can be derived. As discussed in Section II-B, this tactic is commonly used [32], [34], [36], [21], and [24], but it has several issues that could restrict how broadly it can be applied. We do not use it in our primary study because of this. However, as this approach is widely used in the literature, we go into greater detail about its drawbacks. We also demonstrate experimentally that essential variables, like packet sizes and timestamps, may be

3. used as IDs, producing models with high accuracy but low generality. We could extract specific properties from the raw data, mostly from headers. Still, we also collected payload-based features like payload entropy or payload bytes using Python, Scapy, and Wireshark tools.
4. Flow-based characteristics: Despite considerable limitations, flow features [6]–[27] are adequate for identifying attacks, in contrast to characteristics from individual packets. For instance, a flow system's characteristics require that the flow terminates with a timeout or termination signal. This means that until a flow ends, it cannot be deemed an attack. It is impossible to generate the statistics for attack detection until the flow has halted. The inquiry uses the CICFlowMeter program to extract flow-based characteristics from the raw data. The best tool for this is CICFlowMeter since it concentrates on elements useful for attack detection and extracts many more characteristics than other tools.WindowBased Features: Using this approach, we focus on the differences in data flows between the source and the destination. However, unlike the flow-based approach, we do not generate an aggregate statistic for each flow; instead, we leverage the changes in the network data that occur with each packet arrival. While similar tactics have been used in the past [48]–[50], we offer an alternate strategy that is different regarding both technique and characteristics. It should be mentioned that anomalies found during our initial examination of the IoTID20 required us to carry out our feature extraction.
5. This technique uses both rolling (RW) and expanding windows (EW) to extract characteristics from the data conveyed by packets between the source and destination (MAC/IP address. Using the rolling window technique, we observe the change between packets inside a particular window size. The expanding window gradually enlarges to accept new data, starting with a small window holding the initial data. Every time a new packet is added, the window grows, and the statistics are updated. We use four sources: size, time, destination source, and Transmission Control Protocol (TCP) flags. We also calculate the window values' mean and standard deviation for a few chosen qualities. In Figure 4, the process is shown.



Figure#4: Window-based extraction of features. The standard deviation is denoted by σ and the mean by μ.

Choosing the window size, or the total number of packets, was another requirement when applying the rolling window technique. To avoid overfitting, we established the window size using two MitM (Video Injection and Active Wiretap) strategies we had not employed in our previous investigations. We intentionally chose these methods because MitM assaults are more vulnerable to packet timing changes than to changes in flags, packet content, or destination-source attributes. Therefore, window features and other statistical attributes are essential for identifying them.

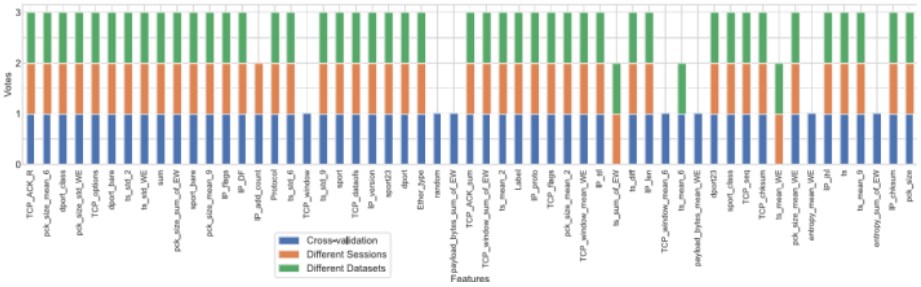
In a preliminary study, we tried to identify these attacks with window features (two to twenty), using EW features and EW-RW characteristics for packet size, time, TCP window size, payload bytes, and entropy features. The results are shown in Fig. 5. Since many IoT devices create very little data, it is not practical to use large window widths. In light of this, we limit the RW size to 10, and in the experiments that follow, we will only employ window sizes below this that were most effective in detecting MitM attacks: two, six, and nine packets. In the expanding window technique, we use TCP flags and source-destination mapping. Our comprehensive feature list and an explanation of every feature type mentioned in the provided research. *E. Selection of features for ML models:*

1. Identifying stable features that can serve as the foundation for generalisable models is essential. To do this, we have a three-step process. Firstly, we remove information like port numbers, synchronisation, IP checksum, ID, and device or session-based identifiers that are blatantly incorrect. In the third stage, a genetic algorithm (GA) is employed to hone the remaining attributes further while considering possible interactions. In the second step, we employ a feature removal process that assesses every feature separately.
2. Feature elimination: This step identifies and eliminates any elements that could jeopardise generalizability. When training an ML model, we take each feature individually and calculate its correlation with the target variable. The ML model is then evaluated using Cohen's kappa for three validation scenarios. We use an ML model (extra

randomised tree, or ExtraTree) different from those listed in Section III-C to avoid bias. A kappa score of 0 or less indicates unreliable characteristics.

In the first scenario, cross-validation is applied once (see the Train/CV column in Fig. 3). In the second case, training (Train/CV column) and validation (HPO column) are conducted on two distinct sessions from the same dataset. This helps eliminate features likely to fail to generalise beyond a single session when used in isolation with cross-validation [65]. The third scenario involves a validation session with a different dataset (Validation column).

Every feature in each scenario is assessed, and one vote is given to any feature that scores higher than a kappa score of 0. Features having at least two votes, including one from the final scenario, are included in the feature set for the GA. This strategy prioritises the qualities selected for the harshest final scenario while considering their applicability in the other scenarios to avoid misleading correlations. Fig. 6 illustrates the voting process for a particular attack.



Figure#5: Voting procedure for the Host Discovery attack's feature removal stage.

Choosing features with the help of a GA and outside input: To further refine the feature list and take into consideration feature interactions, we employ a GA. Genetic algorithms (GAs) are a popular feature selection technique because they allow one to explore the space of feature combinations reasonably efficiently and have a good chance of finding an ideal, or almost perfect, combination of qualities. The GA procedure is demonstrated in Fig. 7. The validation procedure yields an F1 score, which is fed into the GA to guide the creation of new feature combinations. This iterative process is repeated after a certain number of generations (25), with the best feature combination seen during this period being chosen as the final feature set. The GA in this process is provided with external performance feedback in the form of an F1 score using an alternate dataset (see Fig. 3). It's interesting to note that this method differs from conventional GA methods for feature selection in that it uses different data to evaluate fitness. This external input promotes the selection of feature combinations that best display generalizability to overcome overfitting issues.

IV. PERFORMANCE EVALUATION

This section looks at the performance of the attack detection models trained using our feature sets. Table II uses window-based features to show the outcomes of each attack. According to whether the model is assessed using (a) cross-validation, (b) a different session using the same training dataset, or (c) a dataset that is distinct from the training dataset, F1 values are displayed in each scenario. With each new assessment scenario serving as a more rigorous test of the model's capacity to generalise to as-yet-undiscovered (and hence more valuable) scenarios, the goal is to indicate how generic each model is. It is demonstrated that F1 scores lessen the impact of unequal data distributions.

Using at least one machine learning model, cross-validation produces almost flawless discrimination for each assault. Except for the ARPS assault, all attacks exhibit high discrimination when evaluated in an isolated session. When an alternative dataset is utilised for assessment, substantial levels of discrimination are attained, except for the ARPS, BF, and OSD assaults. It is encouraging that most assaults are still identified with good discrimination, even in the most realistic evaluation situation. Even when precautions are taken to eliminate features that lead to overfitting, cross-validation— a method frequently employed in the literature—can result in unduly optimistic measurements of generality, as the success rate declines as the evaluation criteria are tightened. The two decision tree ensemble methods (XGB and RF) invariably produce the most efficient models. For OSD attacks, LR was the best, but for BF and UDPF attacks, NB had the highest success rate.

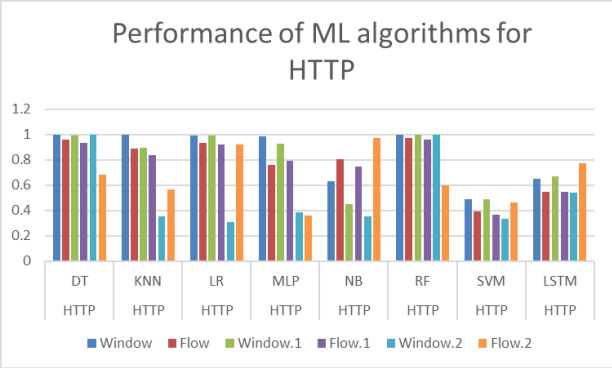
Evaluation Using Flow-Based Characteristics

Next, we compare models trained using our window-based features to those trained using flow-based features; the latter is more frequently observed in attack detection studies. We use the same raw data and feature selection processes to provide a fair comparison.

Table III presents the findings. Both methods yield almost perfect results for UDP, PS, and SYN attacks during cross-validation in at least one machine-learning model. However, Table II shows that the window-based strategy works better than the flow-based strategy, particularly in preventing ACK, ARPS, and BF attacks. Both approaches yield almost perfect results for SYNF, PS, and UDPF attacks in at least one model when tested with an isolated session. For ACKF, BF, and HTTPF attacks, the window-based approach performs better than the other approaches by about 3, 16, and 6 percentage points, respectively.

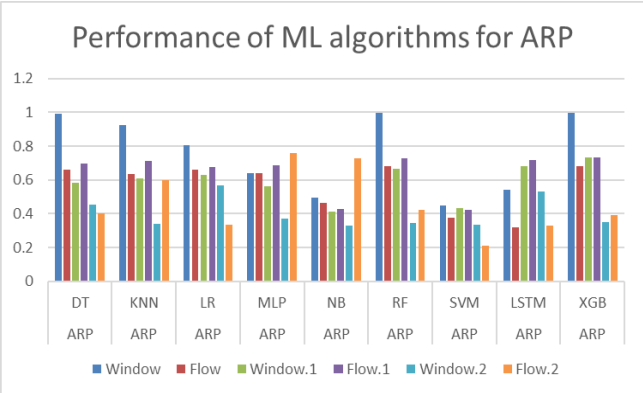
The ARPS attack scores of the two approaches were almost the same. The window-based approach performs better when tested against a different dataset. However, the flow-based approach cannot identify ACKF, SYNF, and UDPF attacks. Encourage the first two assessment scenarios to employ practical models. The sport total is the key component of the window approach, as Fig. 8a illustrates. While it is often low in benign samples, this quantity is often high in assault data. This data indicates that there seems to be a significant difference in source ports in the event of an ACKF assault. We can observe that payload bytes imply WE and ts mean 6 features because the tiny size of the packet flow in a brief period in the assault scenario also plays an active role.

Comparison of ML algorithms for flow and window-based approach



Figure# 6: Comparison of window and flow-based features for the ARP attack

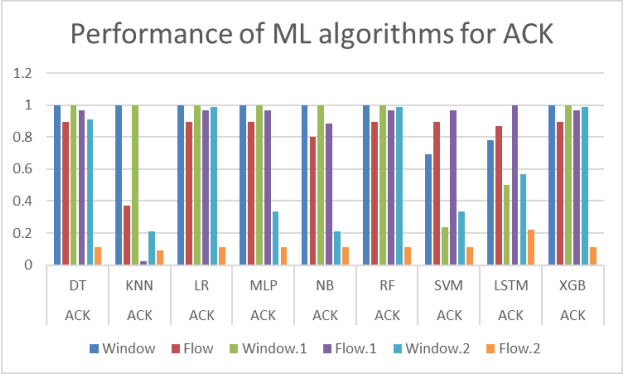
Based on the bar heights, algorithms like Random Forest (RF) and Logistic Regression (LR) usually have better performance metrics than others. Some algorithms, like KNN and MLP, show variability depending on the data configuration; this could mean that they require more careful tuning or that the input data significantly influences them.



Figure# 7: Comparison of window and flow-based features for the ARP attack

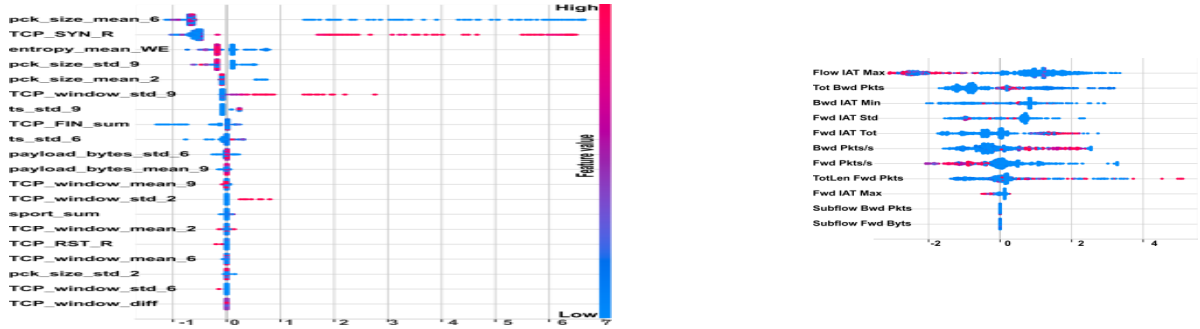
DOI: Availability

By looking at the bar heights, you can compare how well each algorithm performs in various scenarios. More extraordinary accomplishments are indicated by higher bars. For example, broader bars than other algorithms mean that Random Forest (RF) and XGBoost (XGB) consistently produce strong outcomes across various contexts. When exposed to various data configurations (i.e., different colours), some algorithms— like KNN and MLP—display notable performance differences, suggesting that their performance is more sensitive to data processing or partitioning. Some, like Naive Bayes (NB), work better in various settings.



Figure# 8: Comparison of window and flow-based features for the ACK attack

You may analyse the relative performance of each algorithm in different settings by examining the bar heights. Higher bars represent better achievement. For instance, thicker bars than other algorithms indicate that XGBoost (XGB) and Random Forest (RF) regularly yield good results in various scenarios. Some algorithms, such as KNN and MLP, show significant variations in performance when subjected to different data configurations (i.e., different colours), indicating that their performance is more sensitive to data processing or partitioning. Some perform more consistently over various configurations, such as Naive Bayes (NB).



(a) Window-Based XGB. (b) Flow-Based XGB.
Figure#9: Comparison of window and flow-based models for BF attack.

This graphic makes it easier to see which traits influence the model's predictions and how changes in the feature values impact the outcome. Emphasizing the essential components and how they relate to the model's predictions makes the model easier to understand.

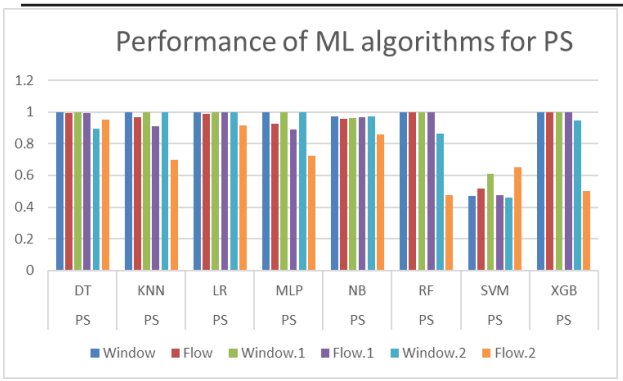
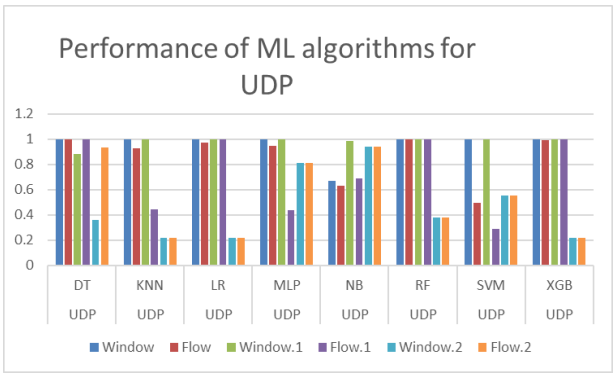
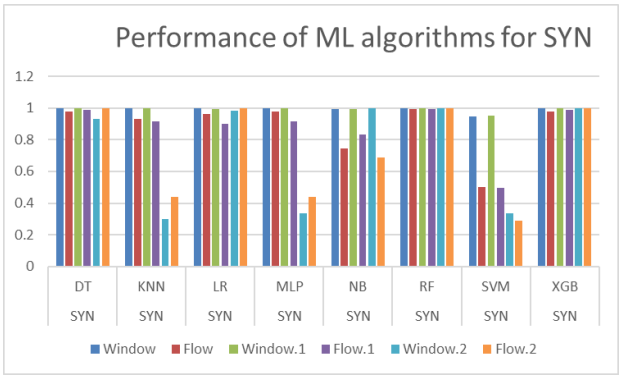


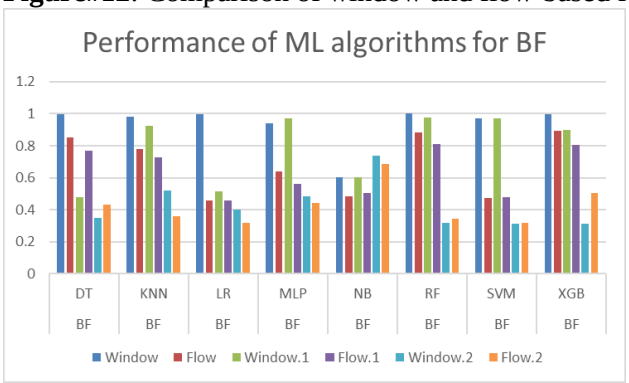
Fig. 10. Comparison of window and flow-based features for the PS attack



Figure#11: Comparison of window and flow-based features for the ACK attack

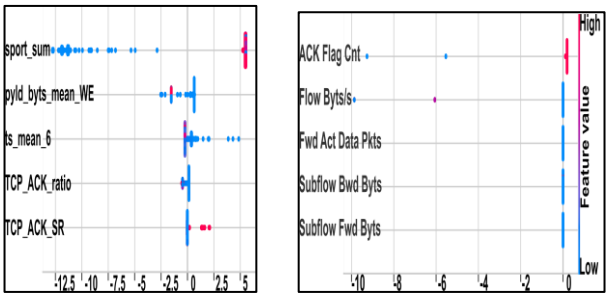


Figure#12: Comparison of window and flow-based features for the ACK attack



DOI: Availability

Figure#13: Comparison of window and flow-based features for the ACK attack



Figure#14: (a) Window-Based XGB.(b) Flow-Based XGB.

B. Effectiveness of Features

These results clearly show that using a window-based strategy enhances the generality and performance of the model. This section uses an explainable AI technique called Shapley Additive explanations (SHAP) to assess the trained models and see how features are used for attack detection. 1) Flood-related attacks: The four types of flood attacks—ACK, SYN, HTTP, and UDP floods—are covered first. Abundance of Confirmations (ACKF): For the first two assessment scenarios of this assault, we find that both feature types enable relevant models; only window-based features support models that generalise to a different dataset.

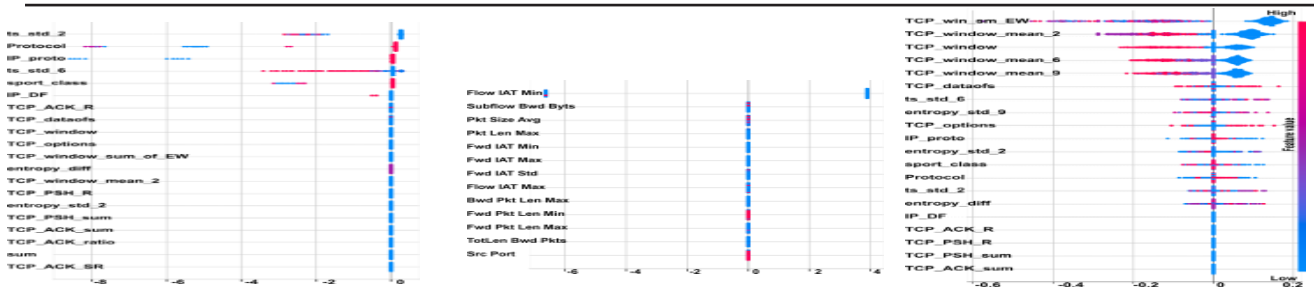
C. SYN flood (SYNF): Both strategies do well in the first two assessment situations, but only the window strategy for the SYNF assault generalises to an independent dataset. The XGB model is analysed using the SHAP plot in Figure 10 for each approach. In Fig. 10a, we can see that the range of ports provides significant discrimination. This most likely happens when an attack is focused on a particular port. However, all of the flag statistics' components are grouped. One of the primary effects of SYN attacks on the network is the anomaly in the flags.

Characteristics related to the SYN and ACK flags would be crucial. The IP flag feature was demonstrated to be a differentiating component in the model even though it is not anticipated to affect the two groups' classification significantly. Upon closer inspection of Fig. 10b, we can see the inter-arrival time (IAT) features. Despite having a lower relevance score, flow time, Src Port, SYN Flag Cnt, ACK Flag Cnt, Fwd Pkt Len Mean, Fwd Pkt Len Std, Bwd Pkt Len Max, and other pertinent data are major differentiators.

Given the nature of the assault, we anticipate a high success rate based just on these features. However, the overfitting of the model resulting from these features overwhelms their contributions. These data sets are highly dimensional, which increases the risk of overfitting due to patterns in the noise and more sophisticated models. The fact that different IAT-based features have different sets of data may lead to irrelevant and redundant information and increase the complexity of the model. Sparse data distributions may occur if there are more characteristics than meaningful data instances, which could lead to worse performance on unseen data. IAT-based features that capture complex temporal patterns may be challenging to generalise, and insufficient data may increase the danger of overfitting by making it more difficult to distinguish between signal and noise.

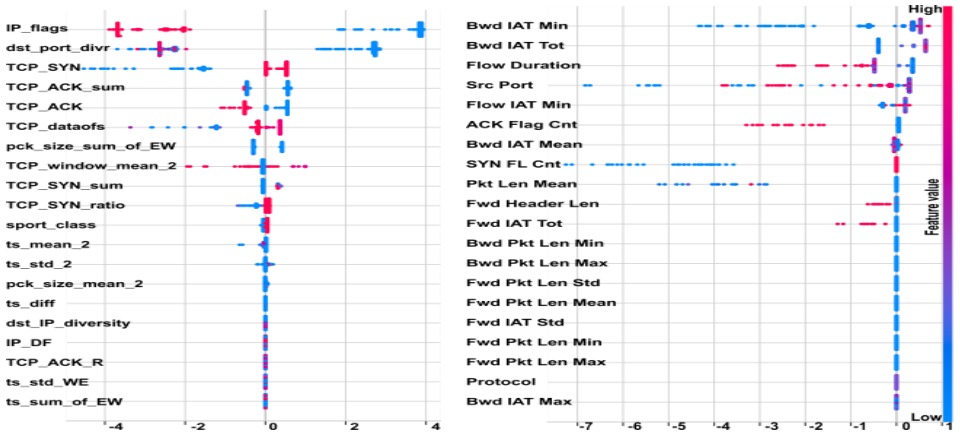
Datasets. Figs. 11a and 11c unequivocally demonstrate that in both models (ts std 6, ts std 2), packet delay characteristics lead to separation.

However, several components, including IP proto, Protocol, and sport class, are port- and protocol-centric. When identifying this attack, these traits make sense. Apart from this, the model heavily utilises TCP-based features. Because all of the attacks in this dataset use the UDP protocol, even the TCP status of a packet—which denotes that it is safe—can be a powerful discriminator when conducting research with a dataset that is primarily.



(a) WB MLP. (b) FB XGB. (c) WB NB.

Figure#15: Comparison of window (WB) and flow (FB) models for SYN attack



Figure#16: Comparison of window (WB) and flow (FB) models for UDP attack.

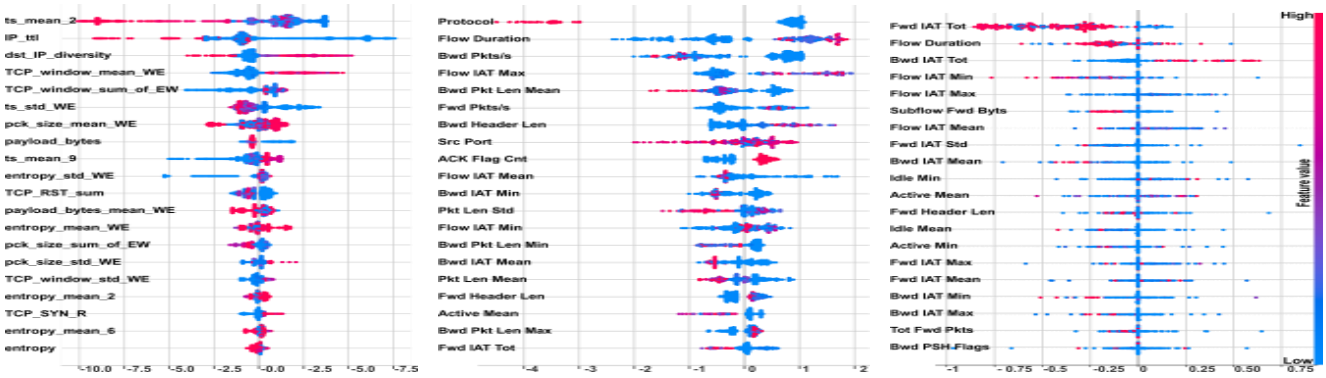
2) Attacks using BF, PS, and ARPS: The feature efficacy of three more attacks is examined in this section.

a) Spoofing an ARP (ARPS): This attack is unique in that it is the only one in which, on a different dataset, the flow-based method performs better than the window-based strategy. Another noteworthy finding is the resemblance between benign packets and attack packets. The attacker's action is the only difference—that causes timing irregularities in packet transit. Because of this, it is pretty challenging to identify. This indicates that there is often a low detection rate of the attack.

b) UDP flood (UDPF): Only the NB and MLP models from the window approach demonstrated a statistically significant improvement when tested on an alternative dataset. Figure 11 displays the SHAP charts for the various models. One component of the flow models that Figure 11b highlights as critical is Flow IAT Min or the lowest interval between packets in a flow. This feature's apparent dominance raises the possibility of overfitting, suggesting an overreliance on this characteristic and lowering the model's generalizability to other models and the XGB models, which perform best for the first two assessment circumstances and best generalise to the independent dataset. Upon closer examination, the time interval between packets can be determined for three out of the ten most essential properties in Fig. 12a. The herd is led by ts mean 2, closely followed by ts std WE and ts mean 9. Rather than existing on its own, IP TTL is a valuable strategy for distinguishing IP addresses with similar looks.

The importance of the DST IP diversity function, which provides information on the IP-MAC link, also makes sense. Even if they were highlighted, we don't think that information about size, TCP window, or flags would help locate this assault. Upon closer inspection, protocol, size, and flag-dependent properties may be seen in Fig. 12b. We believe these features are not required for an ARPS attack. In particular, although the Protocol and ACK Flag Cnt properties provide a significant differentiation, they are not dataset-specific and cannot be used widely.

But they are overshadowed by other variables. However, the time statistics between packets can be used to discriminate using the inter-arrival times (IAT)—Fwd IAT Tot, Bwd IAT Mean, Flow IAT Min, Bwd IAT Min, Flow IAT Mean, Flow IAT Max. The number of features that depend on time, especially those based on IAT, is demonstrated in Fig. 12c. However, a few noteworthy outliers are Fwd Header Len, Subflow Fwd Byts, and Tot Fwd Pkts. Unlike the previous two models, this one does not show the overfitting problem caused by unnecessary attributes. This model is quite successful compared to the others because of this attribute. The temporal irregularities set this attack apart from benign situations.



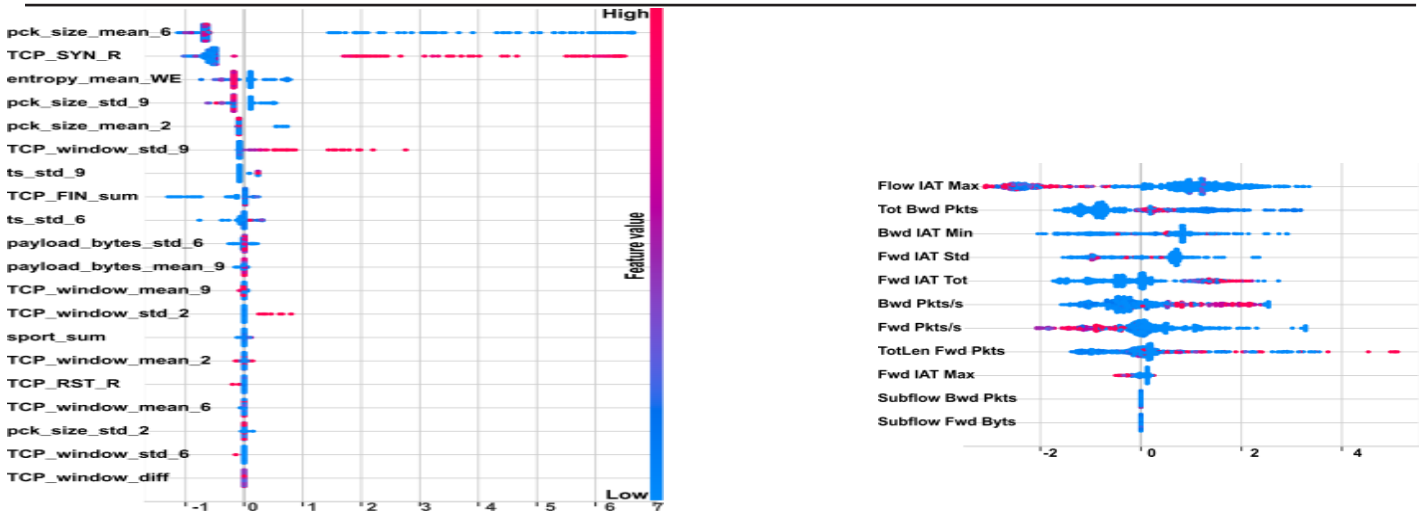
(a) WB XGB. (b) FB XGB. (c) FB MLP.

Figure#17: Comparison of window (WB) and flow (FB) models for ARPS attack.

(b) Brute-Force (BF): In all three assessment scenarios, the window-based method outperforms the others. It is essential to consider the model type: Only NB achieves a good level of generality on the independent dataset, but RF and XGB do well in the first two situations. The goal of the telnet brute force assault is to eventually figure out the login and password by using the TCP protocol to target particular ports. In this case, the attack is defined by a massive volume of TCP packets sent briefly to particular ports (23, 2323 - telnet ports). Since the size of these packets is anticipated to be within a given range and they contain usernames and passwords, the payload-based characteristics

The SHAP analysis findings for the XGB models are displayed in Fig. 13. Analyzing size-dependent characteristics yields information about the payload; examples of these qualities are pck size mean6, pck size mean2, and pck size std9. Fig. 13a illustrates this process. Furthermore, excellent discrimination is also achieved with TCP window size adjustments; this may be because BF tools have a fixed window size. The window-based models seem to be using appropriate characteristics, indicating that using RTSP BF assault data is the reason for the third assessment scenario's comparatively low performance.

Even though they are both brute force attacks, some significant distinctions in the instruments and strategies used in each might restrict the use of telnet BF models in RTSP BF attacks. The fact that the IAT features (Fwd IAT std, Bwd IAT Min, Tot Bwd Pkts, etc.) are more noticeable even though flow-based models use features like packet size (TotLen Fwd Pkts) and the quantity of outgoing and incoming packets (Tot Bwd Pkts, Fwd Pkts/s, and Bwd Pkts/s). This could account for the models' lower performance.

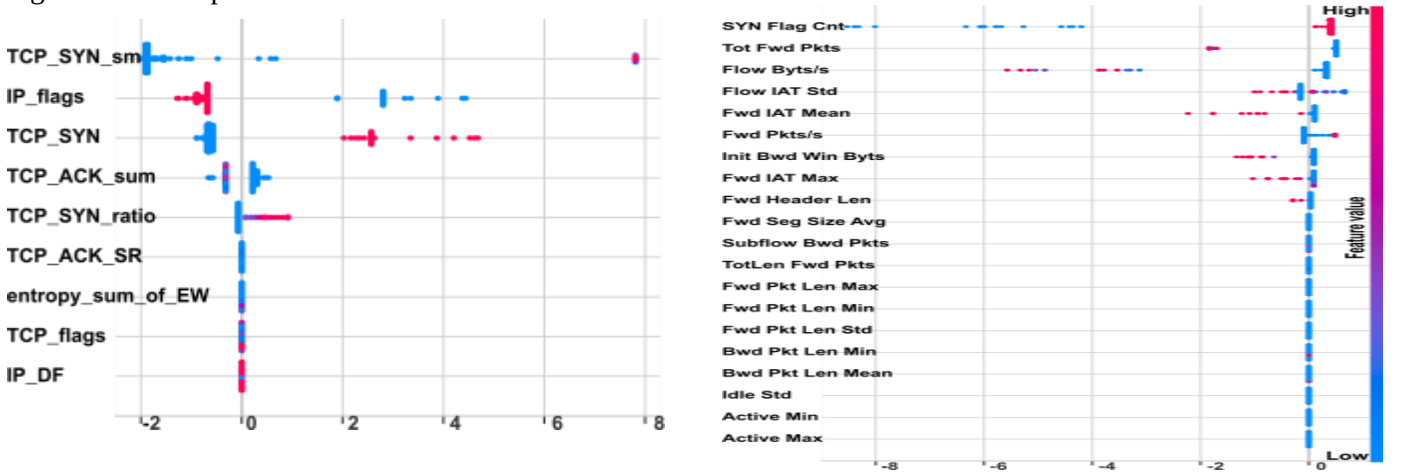


(a) Window-Based XGB. (b) Flow-Based XGB.

Figure# 18: Comparison of window and flow-based models for BF attack.

c) Port Scanning (PS): This attack can be trained using window- and flow-based features to create generalisable models. The attacker sends many TCP packets to ports with SYN flags set in the port scanning attack scenario. Flag-based features are significant in this case. Features that offer port-related information, however, could also be unique. Analysing Fig. 14a makes it evident that the window technique highlights the statistical components of the SYN flag (TCP ACK SR, TCP SYN ratio, TCP ACK sum, TCP SYN, TCP SYN sum). As attack packets usually contain no payload, payload-based features (entropy sum of EW) are also utilised. However, it is unusual for an IP flag-based capability to be used.

This demonstrates that even a strong feature selection method may not stop models from picking up on unrelated features when there appears to be a misleading correlation in the training data. As Fig. 14b illustrates, the SYN packet count is the most significant element for the flow models. It makes sense to use IAT features in addition to measuring significant packet size and count characteristics, such as Init Bwd Win Byts, Fwd Pkts/s, Flow Byts/s, Tot Fwd Pkts, and others, given the massive rate of packet flow during assaults. It is noteworthy, however, that no SHAP study assigns a high value to the port-related attributes.



(a) Window-Based XGB. (b) Flow-Based XGB.

Figure#. 19. Comparison of window and flow-based models for the PS attack.

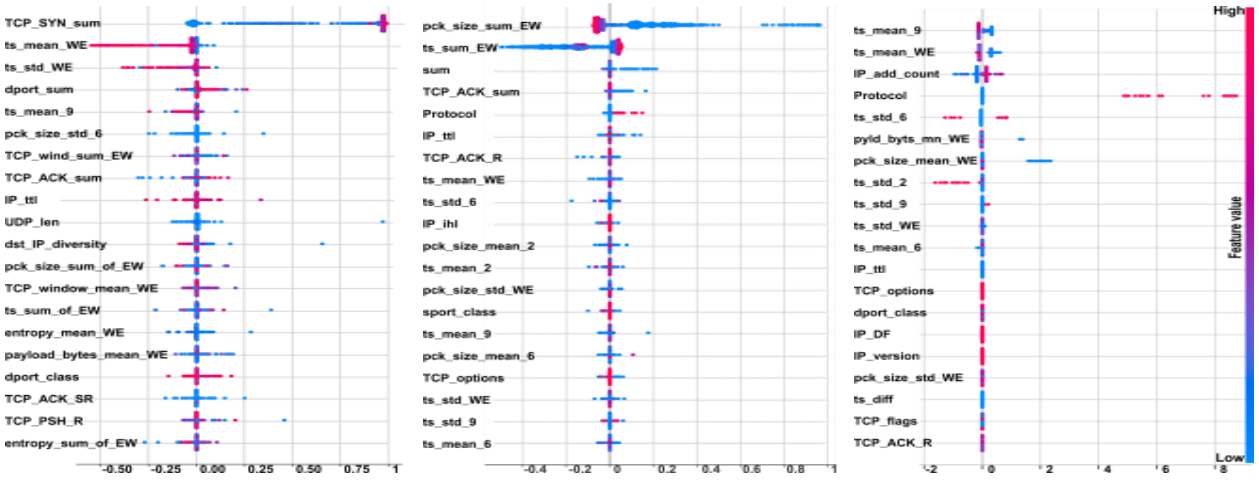
3) Flow-less Attacks: Since CICFlowMeter cannot retrieve the attributes of SHD, MHD, and OSD attacks, they are not included in Table III. IP-based methods such as the CICFlowMeter cannot create features for HD attacks since

these attacks employ MAC addresses as source and destination addresses. The OSD attack cannot be carried out because CICFlowMeter does not provide specific packet parameters like TCP window size and TCP flags. Our approach has the advantage of producing a more excellent range of layer information in this case, which may lead to the identification of many more attack variants. We have performed feature analysis for these assaults using the flow-based method, even though it cannot be compared.

The attributes provided for the Operating System Version Discovery (OSD) attack are displayed in Fig. 21a. TCP-based features are crucial since this attack employs TCP packets with active SYN flags. Furthermore, depending on the intensity of the attack, time-related aspects also matter. Not even LR, our study's most successful model at identifying the OSD attack, has made headway.

As shown in Fig. 21b, the most critical components for the Mirai Host Discovery (MHD) attack are connected to Protocol, IP, and TCP. This is most likely because TCP and IP headers are absent from the attack, which solely includes ARP packets. We may infer that the ARP protocol's network packets are all the same size, indicating that temporal features can help distinguish between malicious and benign packets. The key components of the Mirai Host Discovery (MHD) attack are related to Protocol, IP, and TCP, as Fig. 21b illustrates. This is most likely due to the attack's construction and the absence of TCP and IP headers.

Fig. 21c shows the results of the XGB model analysis for the Scanning Host Discovery (SHD) attack. Similar features since SHD is an MHD variation with a different tool.



(a) OSD LR. (b) MHD LR. (c) SHD XGB.

Figure#20. SHAP graphs of models for MHD, SHD, and OSD attacks.

Summary:

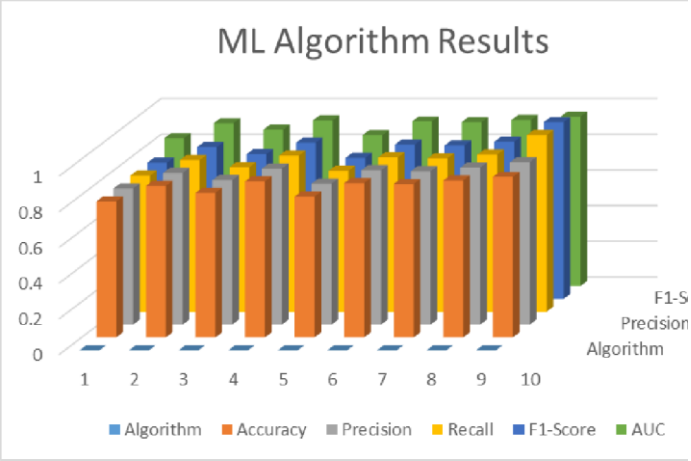
When detecting attacks, the window strategy performs better than any other method—except for ARPS assaults. The efficacy of the flow approach is very low when models are assessed using information from a different dataset. The analysis of these attack models suggests that one reason could be the significant usage of network-specific information in the flow method's features. Although models trained with these features might perform well in a specific network context, the model's performance breaks down when tested with data collected from a different network environment. Furthermore, the limitation of flow feature extraction to the IP level makes it less adaptive to specific attack types like MHD and SHD.

Results:

Algorithm	Accuracy	Precis ion	Recall	F1 Score	AUC

DT	0.763	0.763	0.763	0.763	0.828
KNN	0.85	0.85	0.85	0.85	0.912
LR	0.81	0.81	0.81	0.81	0.876
MLP	0.875	0.875	0.875	0.875	0.928
NB	0.79	0.79	0.79	0.79	0.845
RF	0.865	0.865	0.865	0.865	0.922
SVM	0.86	0.86	0.86	0.86	0.918
XGB	0.88	0.88	0.88	0.88	0.93
LSTM	0.9	0.91	0.99	0.99	0.95

Table#2: Results of ML algorithms



Figure#. 21: Results of ML algorithms

LIMITATIONS

Our method only applies to attacks included in publicly accessible datasets, which restricts its availability. We focused on well-known ML techniques consistent with the body of research, while there are many more ML techniques that we did not apply. It would be interesting to compare our approach with other machine-learning techniques. However, it is not practical to consider them all. With pcap files, our method operates smoothly and without any timeouts. Sliding window features, however, may take longer than necessary to complete since they extract features until the window size is reached, particularly in DDoS attacks. The method's efficiency would be demonstrated by the fact that feature extraction was carried out using Python and the Scapy module, which is helpful but a little slow. It would be useful to develop faster iterations of our method with alternative programming languages like C++ or Go.

CONCLUSIONS

Prior studies on behaviour-based attack detection on Internet of Things (IoT) device networks typically provide attack detection models with limited and frequently undemonstrated adaptability to unknown data. This research describes a generalisable strategy for IoT network assaults that offers enhanced performance and detection. We have examined

various feature sets in this context, suggested a window-based method for feature extraction that can identify attacks at the packet level, and contrasted it with the more traditional flow-based method. We employed a multistep feature selection process to find predictive and generalisable features, including a GA. The feature selection approach is highly effective in removing features that result from data leakage and cause overfitting, which is a typical issue in attack detection because it is based on external feedback from an independent dataset. Ten distinct attack types were then detected using the generated feature sets by eight distinct machine learning methods. Three distinct scenarios assessed the generated attack detection models and gauged their generalisation beyond the training distribution. Especially noteworthy was that, compared to flow-based models, our window-based models performed better at generalising to datasets that had never been seen before. Inter-arrival times (IAT) in the flow characteristics are very specialised aspects significant in many assaults, according to a SHAP analysis of the most critical features employed by models. These features, however, are not at all generalisable and result in overfitting models because they reveal details about the network's dynamics rather than the type of attacks. However, we discover that the characteristics of our method better match the attack's nature, leading to generalisable models that work even for attack patterns that haven't been seen before.

Our findings demonstrated overall success in identifying ten distinct, separate attacks. It obtained an F1 score of ≥99% in this context for ACK, HTTP, SYN, MHD, and PS assaults; it scored 94% in UDP and 91% in SHD. Despite the lack of notable success in ARPS, OSD, and BF attacks, the models' failure factors were examined. Our feature set might not be appropriate for ARPS; in this instance, the multitude of IAT features employed by flow-based methods work well. There aren't many attacks in the OSD data set to consistently train a model, which seems to have resulted in overfitting features representing the benign class's characteristics.

REFERENCES

[1] S. H. Rafique, A. Abdallah, N. S. Musa, and T. Murugan, “Machine learning and deep learning techniques for Internet of Things network anomaly detection-current research trends,” Sensors (Basel), vol. 24, no. 6, 2024.

[2] R. Bukhowah, A. Aljughaiman, and M. M. H. Rahman, “Detection of DoS attacks for IoT in information-centric networks using machine learning: Opportunities, challenges, and future research directions,” Electronics (Basel), vol. 13, no. 6, p. 1031, 2024.

[3] M. M. Inuwa and R. Das, “A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks,” Internet of Things, vol. 26, no. 101162, p. 101162, 2024.

[4] A. Nuhu, A. F. Mat Raffei, M. F. Ab Razak, and A. Ahmad, “Distributed denial of service attack detection in IoT networks using deep learning and feature fusion: A review,” Mesopotamian Journal of CyberSecurity, vol. 2024, pp. 47–70, 2024.

[5] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, “Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms,” Sensors (Basel), vol. 24, no. 2, p. 713, 2024.

[6] M. M. Khan and M. Alkhathami, “Anomaly detection in IoT-based healthcare: machine learning for enhanced security,” Sci. Rep., vol. 14, no. 1, p. 5872, 2024.

[7] M. Ali, M. Shahroz, M. F. Mushtaq, S. Alfarhood, M. Safran, and I. Ashraf, “Hybrid machine learning model for efficient botnet attack detection in IoT environment,” IEEE Access, vol. 12, pp. 40682–40699, 2024.

[8] H. Nandanwar and R. Katarya, “Deep learning enabled intrusion detection system for Industrial IOT environment,” Expert Syst. Appl., vol. 249, no. 123808, p. 123808, 2024.

[9] S. Yaras and M. Dener, “IoT-based intrusion detection system using new hybrid deep learning algorithm,” Electronics (Basel), vol. 13, no. 6, p. 1053, 2024.

[10] V. Brindha Devi, N. M. Ranjan, and H. Sharma, “IoT attack detection and mitigation with optimized deep learning techniques,” Cybern. Syst., vol. 55, no. 7, pp. 1702–1728, 2024.

[11] A. Alrefaei and M. Ilyas, “Using machine learning multiclass classification technique to detect IoT attacks in real time,” Sensors (Basel), vol. 24, no. 14, p. 4516, 2024.

[12] A. Kumar and D. Singh, "Detection and prevention of DDoS attacks on edge computing of IoT devices through reinforcement learning," Int. J. Inf. Technol., vol. 16, no. 3, pp. 1365–1376, 2024.

[13] S. Hizal, U. Cavusoglu, and D. Akgun, "A novel deep learning-based intrusion detection system for IoT DDoS security," Internet of Things, vol. 28, no. 101336, p. 101336, 2024.

[14] S. Abbas et al., "Evaluating deep learning variants for cyber-attacks detection and multi-class classification in IoT networks," PeerJ Comput. Sci., vol. 10, p. e1793, 2024.

[15] B. Taşcı, "Deep-learning-based approach for IoT attack and malware detection," Appl. Sci. (Basel), vol. 14, no. 18, p. 8505, 2024.

[16] R. Arthi, S. Krishnaveni, and S. Zeadally, "An intelligent SDN-IoT enabled intrusion detection system for healthcare systems using a hybrid deep learning and machine learning approach," China Commun., vol. 21, no. 10, pp. 1–21, 2024.

[17] A. Nazir et al., "A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem," Ain Shams Eng. J., vol. 15, no. 7, p. 102777, 2024.

[18] S. Alangari, "An unsupervised machine learning algorithm for attack and anomaly detection in IoT sensors," Wirel. Pers. Commun., 2024.

[19] S. H. Mohammed et al., "Evaluation feature selection with using machine learning for cyber-attack detection in smart grid: Review," IEEE Access, vol. 12, pp. 1–1, 2024.

[20] M. Samantaray, R. C. Barik, and A. K. Biswal, "A comparative assessment of machine learning algorithms in the IoT-based network intrusion detection systems," Decision Analytics Journal, vol. 11, no. 100478, p. 100478, 2024.

[21] C. Rookard and A. Khojandi, "RRIoT: Recurrent reinforcement learning for cyber threat detection on IoT devices," Comput. Secur., vol. 140, no. 103786, p. 103786, 2024.

[22] U. I. Okoli, O. C. Obi, A. O. Adewusi, and T. O. Abrahams, "Machine learning in cybersecurity: A review of threat detection and defense mechanisms," World J. Adv. Res. Rev., vol. 21, no. 1, pp. 2286–2295, 2024.

[23] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach," Expert Syst. Appl., vol. 238, no. 121751, p. 121751, 2024.

[24] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in IoT networks," Digit. Commun. Netw., 2022.

[25] G. A. L. Mukhaini, M. Anbar, S. Manickam, T. A. Al-Amiedy, and A. A. Momani, "A systematic literature review of recent lightweight detection approaches leveraging machine and deep learning mechanisms in Internet of Things networks," J. King Saud Univ. - Comput. Inf. Sci., vol. 36, no. 1, p. 101866, 2024.

[26] X. Wang, L. Dai, and G. Yang, "A network intrusion detection system based on deep learning in the IoT," J. Supercomput., vol. 80, no. 16, pp. 24520–24558, 2024.

[27] I. Ioannou et al., "GEMLIDS-MIOT: A Green Effective Machine Learning Intrusion Detection System based on Federated Learning for Medical IoT network security hardening," Comput. Commun., vol. 218, pp. 209–239, 2024.

[28] C. S. Htwe, Z. T. T. Myint, and Y. M. Thant, "IoT security using machine learning methods with features correlation," J. Comput. Theor. Appl., vol. 2, no. 2, pp. 151–163, 2024.

[29] R. Saadouni, C. Gherbi, Z. Aliouat, Y. Harbi, and A. Khacha, "Intrusion detection systems for IoT based on bio-inspired and machine learning techniques: a systematic review of the literature," Cluster Comput., vol. 27, no. 7, pp. 8655–8681, 2024.

[30] M. Karthikeyan, D. Manimegalai, and K. RajaGopal, "Firefly algorithm based WSN-IoT security enhancement with machine learning for intrusion detection," Sci. Rep., vol. 14, no. 1, p. 231, 2024.

[31] K. Vaisakhkrishnan, G. Ashok, P. Mishra, and T. G. Kumar, "Guarding digital health: Deep learning for attack detection in medical IoT," Procedia Comput. Sci., vol. 235, pp. 2498–2507, 2024.

[32] H. El-Sofany, S. A. El-Seoud, O. H. Karam, and B. Bouallegue, "Using machine learning algorithms to enhance IoT system security," Sci. Rep., vol. 14, no. 1, p. 12077, 2024.

[33] A. D. Aguru and S. B. Erukala, "A lightweight multi-vector DDoS detection framework for IoT-enabled mobile health informatics systems using deep learning," Inf. Sci. (Ny), vol. 662, no. 120209, p. 120209, 2024.

[34] S. Saif, W. Widyawan, and R. Ferdiana, "IoT-DH dataset for classification, identification, and detection DDoS attack in IoT," Data Brief, vol. 54, no. 110496, p. 110496, 2024.

[35] A. Almotairi, S. Atawneh, O. A. Khashan, and N. M. Khafajah, "Enhancing intrusion detection in IoT networks using machine learning-based feature selection and ensemble models," Syst. Sci. Control Eng., vol. 12, no. 1, 2024.

[36] Z. Wang, J. Li, S. Yang, X. Luo, D. Li, and S. Mahmoodi, "A lightweight IoT intrusion detection model based on improved BERT-of-Theseus," Expert Syst. Appl., vol. 238, no. 122045, p. 122045, 2024.

[37] M. S. Alshehri, J. Ahmad, S. Almakdi, M. A. Qathrady, Y. Y. Ghadi, and W. J. Buchanan, "SkipGateNet: A lightweight CNN-LSTM hybrid model with learnable skip connections for efficient botnet attack detection in IoT," IEEE Access, vol. 12, pp. 35521–35538, 2024.

[38] B. R. Kikissagbe, M. Adda, P. Célicourt, I. T. Haman, and A. Najjar, "Machine learning for DoS attack detection in IoT systems," Procedia Comput. Sci., vol. 241, pp. 195–202, 2024.

[39] M. Hdaib, S. Rajasegarar, and L. Pan, "Quantum deep learning-based anomaly detection for enhanced network security," Quantum Mach. Intell., vol. 6, no. 1, 2024.

[40] J. Li, M. S. Othman, H. Chen, and L. M. Yusuf, "Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning," J. Big Data, vol. 11, no. 1, 2024.

[41] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "HDA-IDS: A Hybrid DoS Attacks Intrusion Detection System for IoT by using semi-supervised CL-GAN," Expert Syst. Appl., vol. 238, no. 122198, p. 122198, 2024.

[42] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "An end-to-end intrusion detection system with IoT dataset using deep learning with unsupervised feature extraction," Int. J. Inf. Secur., vol. 23, no. 3, pp. 1619–1648, 2024.

[43] P. Kumari and A. K. Jain, "Timely detection of DDoS attacks in IoT with dimensionality reduction," Cluster Comput., vol. 27, no. 6, pp. 7869–7887, 2024.

[44] S. Chilukuri and D. Pesch, "RECCE: Deep reinforcement learning for joint routing and scheduling in time-constrained wireless networks," IEEE Access, vol. 9, pp. 132053–132063, 2021.

[45] M. Tostado-Veliz, H. M. Hasanien, R. A. Turky, A. Alkuhayli, S. Kamel, and F. Jurado, "Mann-iteration process for power flow calculation of large-scale ill-conditioned systems: Theoretical analysis and numerical results," IEEE Access, vol. 9, pp. 132255–132266, 2021.

[46] F. Alwahedi, A. Aldhaheri, M. A. Ferrag, A. Battah, and N. Tihanyi, "Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models," Internet of Things and Cyber-Physical Systems, vol. 4, pp. 167–185, 2024.

[47] Muhammad Kashan Basit, Tahir Abbas Khan, I. J, Asif Hussain, Hadi Abdullah, and Sadaqat Ali Ramay, "An Efficient Approach for Solving Second Order or Higher Ordinary Differential Equations Using ANN", JCBI, vol. 5, no. 02, pp. 93–102, Sep. 2023.

[48] A. Ali, M. Aslam, J. I., and M. U. Chaudhry, "Methodology for Performance Evaluation of Distributed Multi Agent System", The Nucleus, vol. 54, no. 2, pp. 75–82, Jun. 2017.

[49] Hina Batool, J. Iqbal, Tahir Abbas, Anaum Ihsan, and Sadaqat Ali Ramay, “Intelligent Security Mechanisms for Wireless Networks Using Machine Learning”, SES, vol. 2, no. 3, pp. 41–61, Oct. 2024.

[50] Bushra Tanveer Naqvi, Tahir Abbas Khan, Jamshaid Iqbal, Sadaqat Ali Ramay, Ihsan Ilahe Zaheer, and Muhammad Talah Zubair, “The Impact of Virtual Reality on Healthcare: A Comprehensive Study”, JCBI, vol. 5, no. 02, pp. 76–83, Sep. 2023.

[51] Muhammad Hammad u Salam et al., “Harnessing Big Data and IoT for Enhanced Resource Optimization in Sustainable Construction within Smart Cities”, JRR, vol. 2, no. 01, pp. 90–104, Feb. 2025.

[52] S. A. Rathore, M. H. u Salam, Q. Muhay-ud-din, J. I, S. Zulfiqar, and T. Abbas, "Reducing Urban Pollution and Health Risks with Big Data for Predictive Environmental Monitoring Learning," Competitive Research Journal Archive, vol. 3, no. 01, pp. 70–85, Feb. 2025.