Comparative Analysis of Machine Learning based Intrusion Detection in Realistic IoT Networks

Rana Alharbi, Chuadhry Mujeeb Ahmed, Newcastle University, UK

E-mail: R.A.A.Alharbi2@newcastle.ac.uk

Abstract. The Internet of Things (IoT) is rapidly growing and expanding into various sectors, such as healthcare, transportation, smart homes, and more. Despite the benefits of using IoT devices, they present several challenges. Given the significant role these devices play in our lives, it is crucial to address issues related to their security and privacy. These devices are limited in resources, which complicates their security and the protection of the data that they manage. The paper aims to examine intrusion detection systems using the Gotham2025 dataset, generated through the Gotham testbed, which consists of 78 emulated IoT devices utilising various protocols, including MQTT, CoAP, and RTSP, to assist in safeguarding IoT networks from attacks. We conduct a comparative analysis between five machine learning algorithms, including Random Forest, XGBoost, Logistic Regression, Naive Bayes, and Deep Neural Network. We demonstrate that the Random Forest Classifier was the top-performing model, achieving an F1-score of 0.99 in classifying attacks.

1 Introduction

The term Internet of Things is defined as a network of devices, enabling information exchange online [1]. The idea of connecting devices other than standard computers to the Internet has existed for a long time. For instance, an internet connected toaster was introduced in 1990. The term Internet of Things was originally introduced in 1998 by Kevin Ashton [2]. Furthermore, the number of internet connected devices is growing rapidly. Estimates by Cisco suggest that by 2030, 500 billion devices will be connected to the Internet [3]. The Internet of Things is rapidly gaining popularity and expanding into various sectors, including healthcare, transportation, smart homes, and more. It includes the use of baby and home monitoring cameras, video-enabled doorbells, autonomous vehicles, smartwatches, and so on [4]. However, this advancement presents challenges. Despite the benefits associated with increased usage, risks concerning security and privacy arise, which make safeguarding these IoT networks a vital priority. Part of these safeguarding techniques is the use of machine learning.

Machine Learning (ML) techniques are gaining popularity in security [5, 6]. It primarily relies on automating activities based on inferred knowledge from the data it utilises. From attack detection to authentication and access control in IoT networks, malware analysis, anomaly and intrusion detection, and IoT device identification, machine learning is creating a more secure IoT environment [7, 8]. Previous studies have considered old datasets and tested with a single ML technique. A comprehensive comparison between different ML techniques and the latest device dataset is missing. Our paper aims to compare a

broad range of machine learning based intrusion detection systems using the latest Gotham2025 dataset [9], generated through the Gotham testbed[10], which consists of 78 emulated IoT devices utilising various protocols, including MQTT, CoAP, and RTSP, to help safeguard IoT networks from attacks. The remainder of the document is organised as follows:

- Section 2 (Background and Related Work): Discusses the background and related work, including vulnerabilities in the Internet of Things, datasets, and intrusion detection systems.
- Section 3 (Methodology) describes the methodology, characteristics of the dataset, experimental setup, selected machine learning models, and evaluation metrics.
- Section 4 (Experimental Results) presents the experiments and discusses the results of each evaluated machine learning model and comparatively assesses model performance.
- Section 5 (Conclusions and Future Work) concludes the paper and outlines directions for future work.

2 Background and Related Work

This section highlights IoT vulnerabilities, explains datasets and why they are needed to detect these threats and introduces Intrusion Detection Systems.

2.1 Internet of Things Vulnerabilities

Inadequate security of IoT devices leads to disastrous security incidents that can affect organisations' reputation and finances. Furthermore, they may result in the disclosure of users' sensitive data or data destruction [11]. A significant attack took place in 2016 involving the Mirai botnet, which targeted IoT devices and utilised them to attack the Domain Name Server Dyn in a Distributed Denial of Service (DDoS). Antonakakis et al. conducted an analysis of the Mirai botnet attack and concluded that the inadequate security of IoT devices is the primary cause of the attack [12, 13].

These vulnerabilities include unauthorised access, open ports, and unprotected network interfaces [14]. Outdated firmware is another vulnerability in IoT networks, where many IoT devices fail to update to the latest versions, rendering them susceptible to attacks due to this insecure firmware. Additionally, the use of weak passwords, particularly in situations where an organisation uses the same weak passwords across multiple devices, can lead to one compromised device, causing a chain of compromised devices. Attackers often exploit the limited resources of IoT devices, making them one of their primary targets [15]. Due to constraints in storage and power capabilities, these devices typically require lightweight encryption methods to mitigate vulnerabilities such as exposed communications and flaws in the data transmission processes of IoT devices [16, 15].

Attacks exploiting these vulnerabilities typically appear as denial-of-service (DoS) attacks, where devices are overwhelmed with requests, affecting their availability. It can be accomplished with the help of network protocols such as UDP or ICMP flood attacks, ping of death, and SYN/ACK attacks. On the other hand, Distributed Denial-of-Service (DDoS) has the same impact, utilising multiple entities to attack the same target. Brute force is another attack that attempts unauthorised access by guessing weak passwords [15]. Network scanning, which includes the detection of open ports or exposed network interfaces, represents another threat to IoT networks [14].

What makes securing the Internet of Things a challenging process is the nature of the devices that constitute an Internet of Things network. These networks may include a variety of devices, ranging from small temperature sensors that measure room temperature and communicate with other devices to cameras or self-driving vehicles [17]. These IoT networks differ in size, the number of connected devices, their energy consumption levels, the operating system used, and the protocols employed [4]. This heterogeneity, arising from IoT devices differing in size, capabilities, characteristics, hardware constraints, and communication protocols [5], presents one of the key challenges in maintaining IoT security [2, 18].

2.2 Datasets

The increasing adoption of IoT devices across various sectors has led researchers to create datasets that can be utilised to train machine learning models for effectively detecting different intrusions and attacks against these networks, thus enhancing performance when these models are applied in real-world scenarios. The effectiveness of the model is heavily influenced by the nature and quality of the dataset. Datasets are created using testbeds that simulate IoT environments, and they can be generated by creating realistic IoT environments with actual IoT devices. Different data collection methods have multiple factors,

including network size, the number of devices utilised, the type of conducted attacks, and the total duration of device traffic collection. The types of devices included in the testbed impact the dataset's quality, especially as technology evolves and real-world devices change. Another critical factor is that the broader the functionalities offered by an IoT device, the wider the attack vector on that IoT network [14].

One of the most popular datasets for testing Intrusion Detection Systems (IDS) in cybersecurity is KDD-99, created in 1990 and improved on by NL-KDD in 2009. Most research has relied on this dataset to evaluate their IDS. However, this dataset does not represent emerging threats and vulnerabilities, which means that testing on an outdated dataset will not accurately reflect how these IDS perform when deployed in the real world. Consequently, the IDS will not generalise well to new threats and attacks [19]. Other datasets tailored for IoT Intrusion Detection Systems include: MedBIoT, IoTID20, IoT-23 and CICIoT2023 [20].

For a dataset to be effective, it must include a variety of new attacks and threats utilising real IoT devices to create a realistic network environment, as some datasets employ traditional computers instead of IoT devices to simulate attacks [21]. Among the recent datasets are CICIoT2023[21] and Gotham2025 [9] Datasets. The Gotham2025 Dataset is designed to assist research in large-scale IoT networks. It provides heterogeneity by offering various types of devices, protocols, and attack surfaces to simulate real-world IoT environments. The nature of the dataset mimics real-world scenarios by collecting data in separate files rather than dividing data after collection, which some researchers do to test federated learning approaches [9].

2.3 Intrusion Detection Systems (IDS)

Traditional IDS, such as signature-based systems, effectively detects known attacks. However, they struggle to identify new or unseen threats. Machine learning and deep learning models have become the foundation of many modern IDSs, successfully addressing the limitations of traditional systems. Key challenges in implementing IDS include associated computational costs, particularly as we deal with IoT devices that have limited capabilities. We must reduce these costs while ensuring high accuracy and robustness. Another challenge lies in the nature of the dataset, which often presents a skewed distribution, and the choice of relevant features in intrusion detection, which in turn influences model complexity [22]. Furthermore, the appropriate selection of features significantly impacts training time and IDS performance [23]. Machine learning techniques are valuable for identifying patterns in data and detecting suspicious behaviour, whether using supervised learning models with labelled data or unsupervised ones, which lack labels and is useful for identifying zero-day attacks. However, it can produce many false positives. Deep learning and reinforcement learning are among the methods employed in IDS implementation [19].

3 Methodology

To achieve the objectives of this study, we follow the methodology in Figure 1: we first inspect the Gotham dataset, select the machine learning algorithms, choose the evaluation metrics, evaluate each machine learning model individually and comparatively analyze the performance of the selected ML models. This chapter outlines the methodology and gives a detailed explanation of the Gotham dataset, the preprocessing steps, the model selection process, the evaluation metrics used, and the experimental setup designed to assess model performance.

3.1 Dataset and Preprocessing

We used the GothamDataset2025 [9], which contains labelled network traffic data representing a diverse range of attack scenarios and normal traffic. The normal traffic is collected over two hours, whereas each attack is gathered over a period of one to one and a half hours. They collected traffic separately, ensuring that the data for each device was reflected in its own file to facilitate decentralized analysis and learning. Since the data is spread over multiple files and we are working in a centralised setting, we combined all traffic into a single dataframe to evaluate Machine Learning performance.

The type of attacks collected in the Gotham Dataset 2025 are grouped into: **Normal, Network Scanning, Brute Force, Infection, C&C Communication, DoS.** Table 1 shows attacks before grouping. The dataset is unbalanced. Moreover, grouping the attacks into six classes made DOS attacks dominant in the number of items across the training data. Figure 2 shows the class breakdown after the data have been grouped into six labels, and Table 2 shows the number of attacks after the grouping. The preprocessing was performed using the author's code for the dataset provided on GitHub [9]. Their code serves as a preprocessing step that includes data splitting, feature extraction, along encoding to ensure

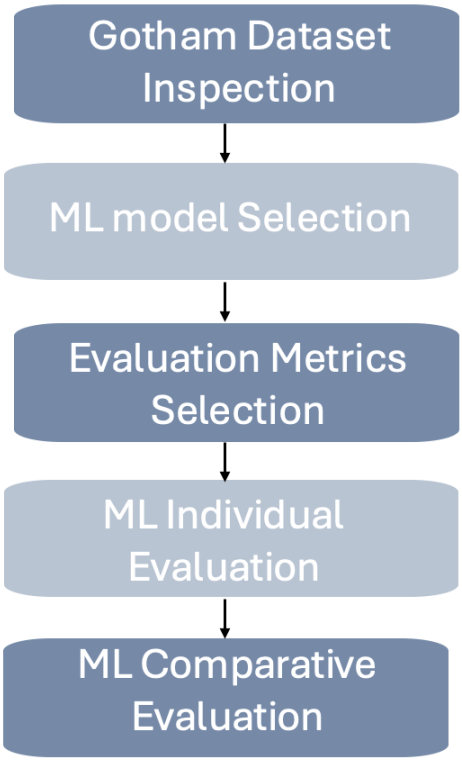

Figure 1: Overview of the Methodology.

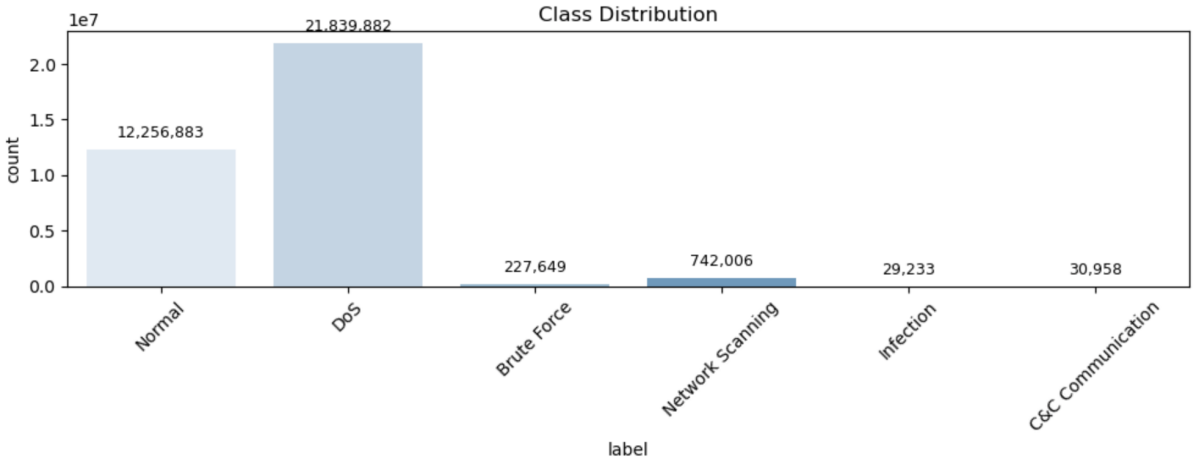

Figure 2: Class Distribution for Dataset.

Label	Count
Benign	12256883
Mirai UDP Flooding	8897895
Mirai TCP Flooding	6548173
Mirai GRE Flooding	5911401
TCP Scan	737764
CoAP Amplification	274837
Telnet Brute Force	227649
Merlin TCP Flooding	120000
Merlin ICMP Flooding	57580
Merlin UDP Flooding	29996
Merlin C&C Communication	29356
Ingress Tool Transfer	21587
Unknown	7670
File Download	7196
UDP Scan	4242
Mirai C&C Communication	1074
C&C Communication	528
Reporting	450
Total	35134281

Table 1: Dataset before grouping.

Label	Count
Brute Force	227649
C&C Communication	30958
DoS	21839882
Infection	29233
Network Scanning	742006
Normal	12256883
Total	35161611

Table 2: Dataset after grouping.

consistent data handling across both their deep learning methods and our machine learning approaches. These preprocessing steps applied to the Gotham dataset include extracting protocol types and source and destination ports from the frame. Protocols feature, converting ports into categorical values, separating IP flags and TCP flags into individual binary features, standardizing the timestamp, handling missing values, scaling numerical features, and detailed attack labels are mapped into six categorical labels, which are then encoded into numerical values suitable for training machine learning models. Some adjustments were made to the code, as some traffic was incorrectly categorised during preprocessing; for instance, benign traffic and certain C&C communications were labelled as 'Other' or unknown traffic.

The Gotham dataset is tailored to smart city environments and was collected using the Gotham testbed [10]. It includes emulated traffic from various device types, such as air quality and building monitors, which include readings from fifteen and twenty-seven sensors, respectively, including humidity, temperature, and other types of sensors. Other device types include cooler motors, which use acceleration and speed sensors to monitor fan vibration. These devices communicate using several lightweight IoT protocols: Message Queuing Telemetry Transport (MQTT) [24], a standard lightweight messaging protocol for devices with limited resources, Constrained Applications Protocol (CoAP protocol) [25] is used for machine-to-machine communications such as building automation and smart energy, and Real Time Streaming Protocol (RTSP) [26] which is used in streaming audio and video. The Gotham dataset is relatively new and has not been explored. It introduces heterogeneity in the types of devices included in the smart city environment and generates revenue to facilitate Federated learning approaches that limit data sharing and protect privacy due to the manner in which this traffic is collected.

3.2 Selected Machine Learning Algorithms

The selection of the algorithm was based on its relevance and impact in the field. Random Forest was chosen due to its widespread use and recognition as one of the most cited algorithms in the literature. Although XGBoost and Logistic Regression have fewer citations, they still demonstrate relevance in Intrusion Detection Systems [23]. Naive Bayes Classifier was also included, as it has been utilised in various IoT security solutions [14].

- **Random Forest Classifiers:** Random forest is one of the ensemble methods in which the final decision of the classifier is based on the predictions of multiple machine learning algorithms. The Random Forest consists of a collection of decision trees, where the dataset is divided into various subsets. Each decision tree takes a section of the features and some records in a process called Bootstrap. Each tree makes a decision based on the subset of the assigned dataset. The final decision of the Random Forest is determined by a majority vote, where the most frequently chosen classification becomes the final prediction in a process known as aggregation. Bootstrap and aggregation together are referred to as bagging. An alternative method of training on different subsets is known as pasting. Using multiple decision trees helps to mitigate the problem of high variance present in a single decision tree [27].

- **XGBoost Classifier:** XGBoost is a machine learning technique that stands for Extreme Gradient Boosting, which is an optimised version of Gradient Boosting. It offers speed, scalability, and portability. It involves training predictors in a sequential manner, where each predictor adjusts its predecessor's error and fits it to the residual errors of the previous predictor. XGBoost prevents overfitting by incorporating early stopping in its techniques [27].

- **Logistic Regression:** Logistic Regression is based on calculating the probability that an instance belongs to a particular class. It handles multiclass classification by producing a vector of probabilities. The softmax function is applied, which gives a probability distribution over classes. For instance, given an input x, it computes a score for each class, where the class with the highest probability represents the predicted class [27].

- **Naive Bayes Classifier:**

  Naive Bayes is similar to Bayesian classifiers, but it simplifies the complexity present in them. To reduce this complexity, Naive Bayes employs a conditional independence assumption, which assumes that there is no dependence among all features describing x given the class label. This assumption decreases the number of parameters, thus lowering the complexity [28]. It relies on Bayes' theorem, which updates the probability distribution after observing some data [27].

3.3 Evaluation Metrics

Evaluation metrics are designed for Machine Learning and Deep Learning algorithms performance assessment. The expected behaviour of models is to achieve high True Positives (when a model makes a positive prediction and it is accurate) and high True Negatives (when a model makes a negative prediction and it is accurate) while minimising False Positives (when a model makes a positive prediction that is incorrect) and False Negatives (when a model makes a negative prediction that is incorrect). The application of each metric depends on the specific use case in which it is employed. Accuracy alone is inadequate for evaluating models, particularly when the datasets are skewed. For instance, in a binary classification scenario, if class A has 90% of the data samples while class B accounts for only 10%, a classifier that predicts all instances as class A would achieve an accuracy of 90%. However, this percentage does not provide a good indication of how the classifier behaves when it is employed in the real world. For instance, when we aim to reduce false positives, we should prioritise precision. On the other hand, when we want to decrease false negatives, we should concentrate on recall. Various metrics can provide differing insights into the model performance [27]. The models discussed in section 3.2 will be evaluated using the different metric evaluations, including Accuracy, precision, and recall, with the focus on F1-Score since we are working with an unbalanced dataset. Confusion matrices are utilised as well to show false positives and false negatives that exist in each class. The metrics employed in this paper[21]:

Accuracy metric refers to the percentage of correct predictions from total predictions, both correct and incorrect, see Equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision refers to positive prediction accuracy, which indicates how many of the predicted positive instances were correct, as shown in Equation 2.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall, (True Positive Rate and sensitivity), indicates how many of the positive instances the model correctly identifies, as shown in Equation 3.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-Score, consider both false negative and false positive using precision and recall as shown in Equation 4.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

The **confusion matrix** is a good indicator of how many instances are being misclassified. The matrix shows a count of each class and how many of these instances are being misclassified. The columns in the matrix represent the predicted class, where the rows represent the actual class.

3.4 Experimental Setup

All of the machine learning models were implemented using the Scikit-learn [29]. The Machine Learning models utilised in this paper include Random Forest Classifiers, XGBoost Classifier, Logistic Regression and Naive Bayes Classifier [23] [14]. SVM was also among the machine learning models chosen, as it ranks second to Random Forest [23]. However, it struggled to handle the large dataset effectively, as shown by our experiments, where training took days without completion on both our local computing systems and Google Colab. Training on our local system equipped with a 2.3 GHz 8-Core Intel Core i9 processor and 16 GB RAM took considerable time to run various experiments. After conducting experiments locally, we purchased a paid subscription to Google Colab Pro, which improved performance and reduced the time needed to run different experiments. Several challenges were encountered concerning interrupted experiments and device overheating during training, which impacted the experimental process.

4 Results

In this section, we will discuss the performance of each selected model in classifying the various attack classes using the evaluation metrics discussed in section 3.3: accuracy, precision, recall, and the resulting confusion matrix. Following the individual analysis, we conduct a comparative evaluation to identify the top-performing model.

4.1 Random Forest Classifier

The Random Forest Classifier performs well with the dataset, achieving nearly perfect predictions for four of the six classes. As illustrated in the confusion matrix in Figure 3, Normal, C&C Communications, Infection, and DoS demonstrate near-perfect predictions, with at most 0.45%, which is a small percentage misclassified as other labels. However, Network Scanning and Brute Force show some confusion. The classification report in Table 3 indicates an overall accuracy of 100% and a macro-average F1-Score of 0.99, suggesting that Random Forest is effective with unbalanced data and robust in classifying network activities.

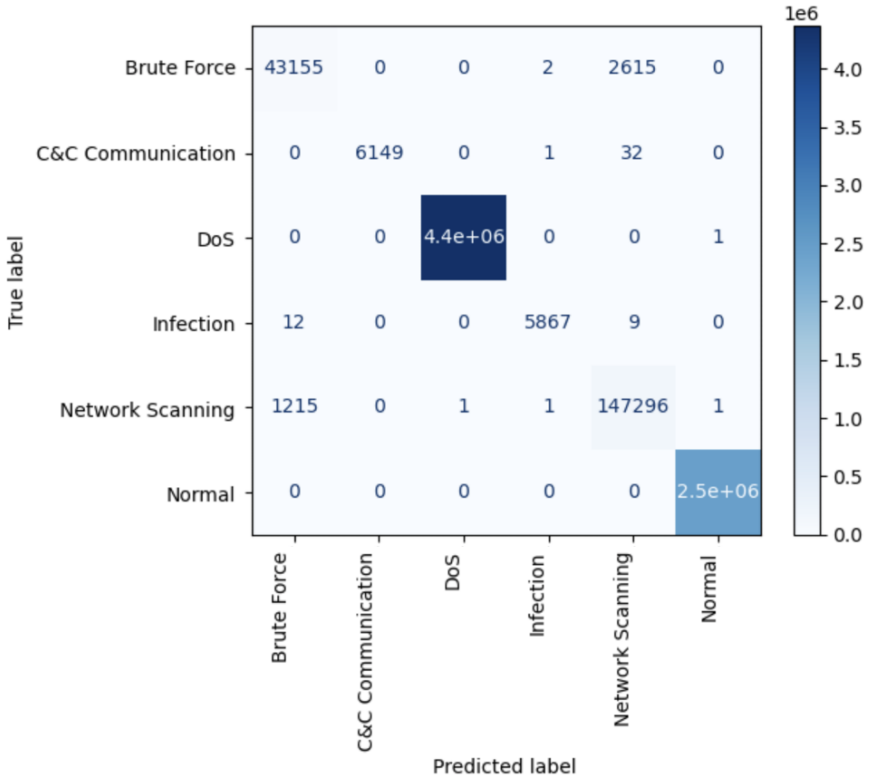

Figure 3: Random Forest Classifier Confusion Matrix

Class	Precision	Recall	F1-Score	Support
0 Brute Force	0.97	0.94	0.96	45,772
1 C&C Communication	1.00	0.99	1.00	6,182
2 DoS	1.00	1.00	1.00	4,367,650
3 Infection	1.00	1.00	1.00	5,888
4 Network Scanning	0.98	0.99	0.99	148,514
5 Normal	1.00	1.00	1.00	2,451,315
Accuracy			**1.00**	7,025,321
Macro Avg	0.99	0.99	0.99	7,025,321
Weighted Avg	1.00	1.00	1.00	7,025,321

Table 3: Classification Report (Random Forest Classifier)

4.2 XGBoost Classifier

The XGBoost Classifier performs slightly worse than the Random Forest Classifier. As illustrated in the confusion matrix in Figure 4, normal traffic, C&Cs communications, and DoS demonstrate some confusion, with at most five instances misclassified as other labels. However, network scanning is the least accurate among all six classes regarding misclassification. The classification report in Table 4 indicates an overall accuracy of 100% and a macro-average F1-Score of 0.97, suggesting that the XGBClassifier is less effective than Random Forest with unbalanced data and in the robustness of classifying network activities.

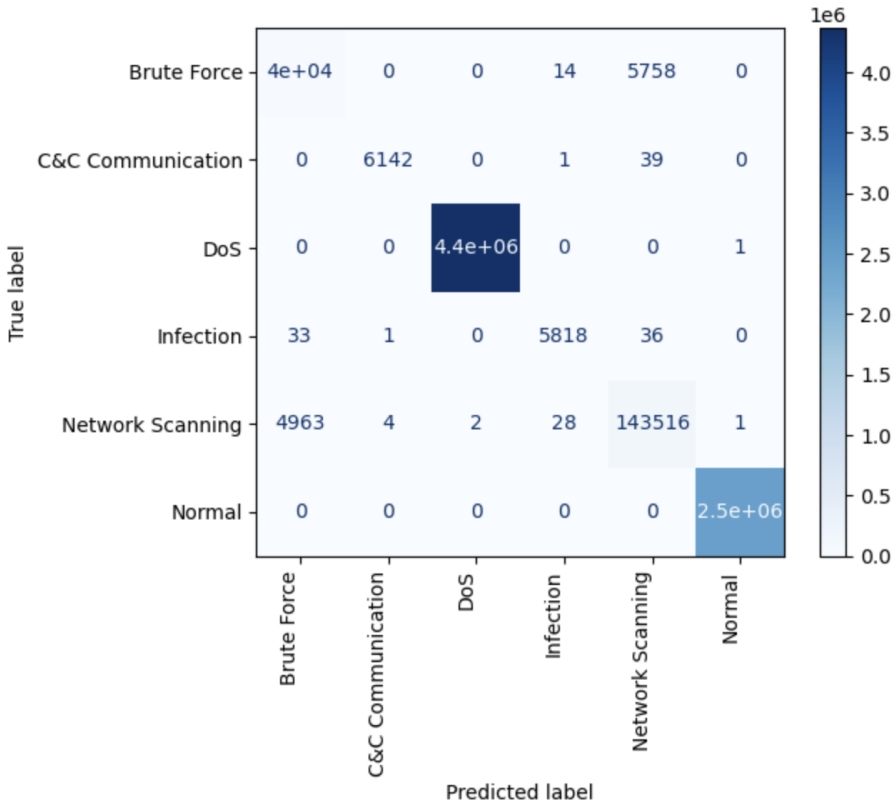

Figure 4: XGB Classifier Confusion Matrix

Class	Precision	Recall	F1-Score	Support
0 Brute Force	0.89	0.87	0.88	45,772
1 C&C Communication	1.00	0.99	1.00	6,182
2 DoS	1.00	1.00	1.00	4,367,650
3 Infection	0.99	0.99	0.99	5,888
4 Network Scanning	0.96	0.97	0.96	148,514
5 Normal	1.00	1.00	1.00	2,451,315
Accuracy			**1.00**	7,025,321
Macro Avg	0.97	0.97	0.97	7,025,321
Weighted Avg	1.00	1.00	1.00	7,025,321

Table 4: Classification Report for XGBClassifier

4.3 Logistic Regression

The Logistic Regression model ranked second to last among the tested machine learning models. As illustrated in the confusion matrix in Figure 5, Brute Force, Network Scanning, Infection, and C&C Communication are frequently confused with other labels and are misclassified. However, Normal traffic and DoS perform better, indicating that logistic regression does not handle class imbalance well. The classification report in Table 5 shows an overall accuracy of 99% and a macro-average F1-score of 0.72. However, accuracy is not particularly relevant in this context. As presented in the table, Brute Force has an F1-score of 0.36, while Infection has an F1-score of 0.19, and Normal traffic and DoS achieve an F1-score of 1. This suggests that Logistic Regression performs well with classes that have a larger number of instances and poorly with minority classes. Nonetheless, the confusion matrix in figure 3 demonstrates that despite DoS achieving an F1-score of 1, the model still misclassifies 2574 instances as Brute Force,

Infection, Network Scanning, and Normal traffic. This indicates that even the class with many instances is still subject to misclassification.

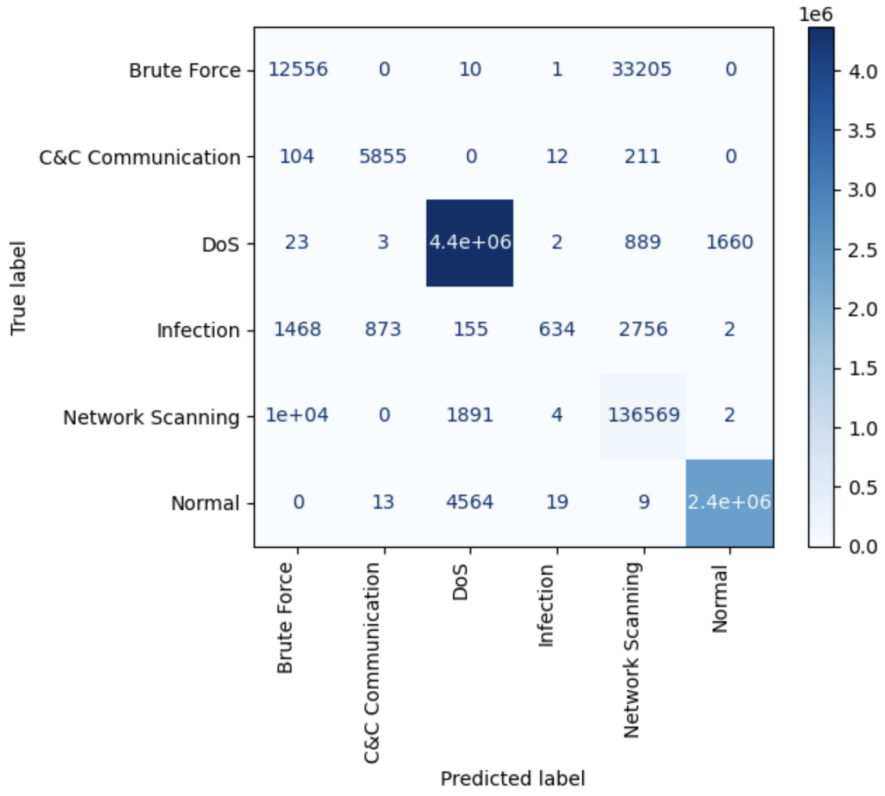

Figure 5: Logistic Regression Confusion Matrix

Class	Precision	Recall	F1-Score	Support
0 Brute Force	0.52	0.27	0.36	45,772
1 C&C Communication	0.87	0.95	0.91	6,182
2 DoS	1.00	1.00	1.00	4,367,650
3 Infection	0.94	0.11	0.19	5,888
4 Network Scanning	0.79	0.92	0.85	148,514
5 Normal	1.00	1.00	1.00	2,451,315
Accuracy			**0.99**	7,025,321
Macro Avg	0.85	0.71	0.72	7,025,321
Weighted Avg	0.99	0.99	0.99	7,025,321

Table 5: Classification Report for (Logistic Regression)

4.4 Naive Bayes Classifier

Among the tested machine learning algorithms, the Naive Bayes Classifier demonstrated the lowest overall performance. The model showed inconsistent behaviour across all classes, as illustrated in the confusion matrix in Figure 6. Dos is often misclassified as other classes of attacks but is frequently mistaken for normal traffic. The classification report in Table 6 shows no clear correlation between the number of samples and the precision, recall and F1-score of each class. For instance, C&C Communications, with 6,182 samples, achieved the highest F1-score, while Normal traffic and DoS, with 2,451,315 and 4,367,650

samples, respectively, had lower F1-score. The report indicates an overall accuracy of 56%, a macro-average F1-Score of 0.43 and a weighted average F1-score of 0.55. These results suggest that the Naive Bayes classifier is ineffective for the Gotham dataset in distinguishing these attacks.

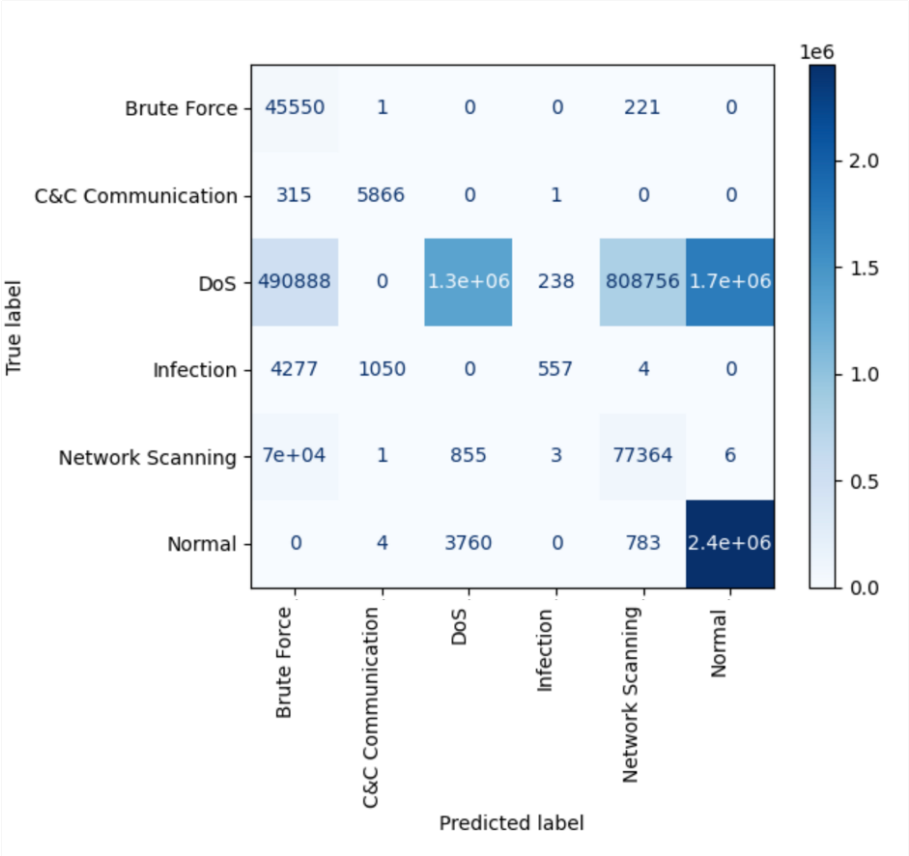

Figure 6: Naive Bayes Confusion Matrix

Class	Precision	Recall	F1-Score	Support
0 Brute Force	0.07	1.00	0.14	45,772
1 C&C Communication	0.85	0.95	0.90	6,182
2 DoS	1.00	0.31	0.47	4,367,650
3 Infection	0.70	0.09	0.17	5,888
4 Network Scanning	0.09	0.52	0.15	148,514
5 Normal	0.59	1.00	0.74	2,451,315
Accuracy			0.56	7,025,321
Macro Avg	0.55	0.64	0.43	7,025,321
Weighted Avg	0.83	0.56	0.55	7,025,321

Table 6: Classification report (Naive Bayes Classifier)

4.5 Centralised Deep Neural Network

Researchers in [9] implemented a simple fully connected deep neural network with three hidden layers and a ReLU activation function. Table 7 indicates that the deep neural network model achieved 0.91% as a Macro Average F1-Score, with the DoS and Normal traffic classes achieving an F1-Score of 1. The lowest scores among all six classes were observed in the Brute Force and Infection classes. The confusion matrix in Figure 7 illustrates that many instances are being misclassified, with the class that has the fewest

instances misclassified being Normal traffic, followed by DoS, which correlates with the large number of instances.

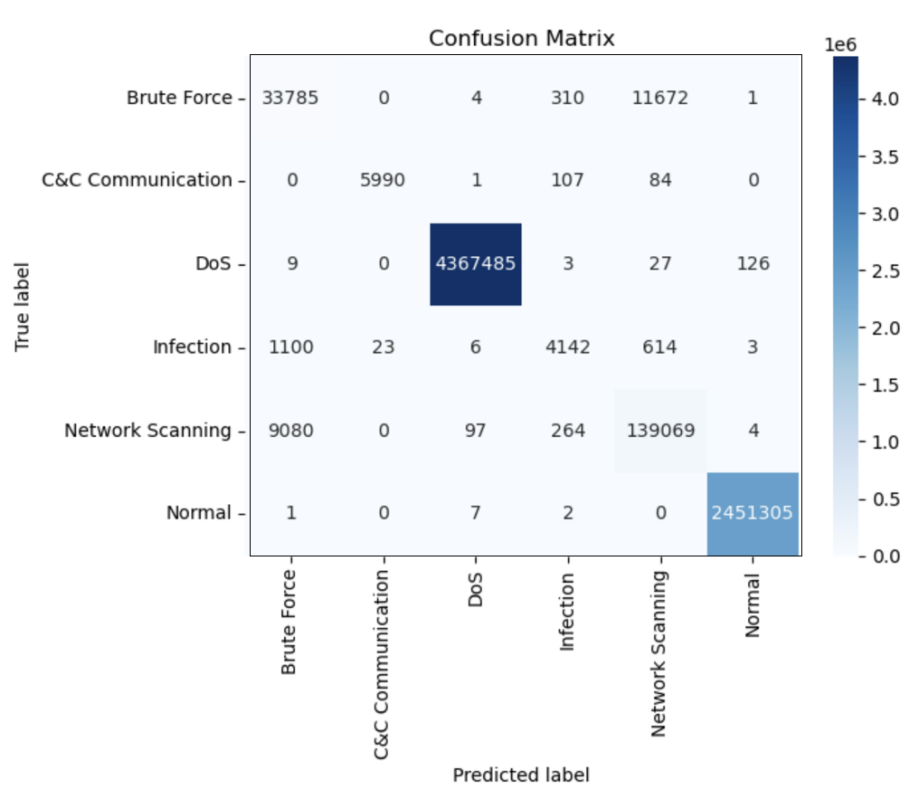

Figure 7: Deep Neural Network (DNN) Confusion Matrix

Class	Precision	Recall	F1-score	Support
0 Brute Force	0.77	0.74	0.75	45,772
1 C&C Communication	1.00	0.97	0.98	6,182
2 DoS	1.00	1.00	1.00	4,367,650
3 Infection	0.86	0.70	0.77	5,888
4 Network Scanning	0.92	0.94	0.93	148,514
5 Normal	1.00	1.00	1.00	2,451,315
Accuracy			1.00	7,025,321
Macro avg	0.92	0.89	0.91	7,025,321
Weighted avg	1.00	1.00	1.00	7,025,321

Table 7: Classification report (Deep Neural Network)

4.6 Comparative Evaluation of Model Performance

The performance across the five different algorithms is summarised in Table 8 and Figure 8, revealing variations in performance across classes. Random Forest and XGBoost were among the top-performing models. The random forest model achieved the highest overall accuracy and F1 score, with near-perfect classification performance for almost four out of the six classes, demonstrating its effectiveness in the Gotham dataset. XGBoost followed, achieving perfect F1 scores in three out of the six classes. A deep neural network performs well, earning an overall F1-score of 91%. However, it underperforms in two attack classes. In contrast, Logistic Regression and Naive Bayes had the lowest performance, indicating limitations in attack classification. Logistic Regression performed better than Naive Bayes, achieving an overall F1-score of 72% and an accuracy of 99%. Naive Bayes, On the other hand, achieved an accuracy of 56%, a macro F1-score of 43%, and a weighted F1-score of 55%. It records the lowest performance per class, with 14%, 15%, and 17% F1-scores demonstrating how it is ineffective in distinguishing these attacks.

Class	RF	XGBoost	DNN	LR	NB
0 Brute Force	0.96	0.88	0.75	0.36	0.14
1 C&C Communication	1.00	1.00	0.98	0.91	0.90
2 DoS	1.00	1.00	1.00	1.00	0.47
3 Infection	1.00	0.99	0.77	0.19	0.17
4 Network Scanning	0.99	0.96	0.93	0.85	0.15
5 Normal	1.00	1.00	1.00	1.00	0.74
Accuracy	1.00	1.00	1.00	0.99	0.56
Macro F1	0.99	0.97	0.91	0.72	0.43
Weighted F1	1.00	1.00	1.00	0.99	0.55

Table 8: F1-scores across different tested models.

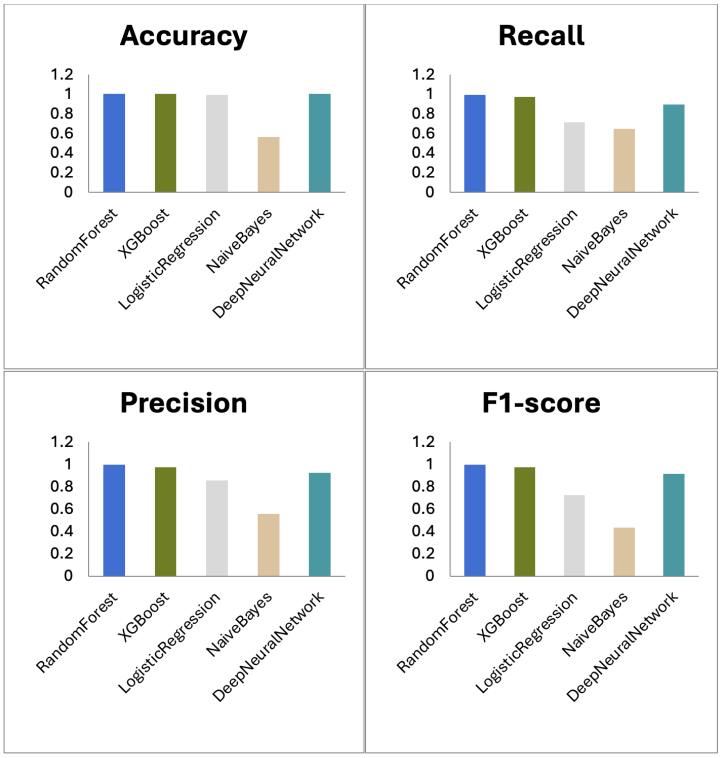

Figure 8: Models Performance.

5 Discussion

This study utilizes the Gotham dataset, which is based on a smart city environment that has not been previously studied. It focuses on network heterogeneity and large-scale IoT networks. It has 22 features; the model's limited feature set helps lower computational demands. The dataset relies on distributed data collection, where each device maintains its own file to enable federated learning. However, the study has certain limitations. First, this work depends on supervised learning, which may struggle to identify new attack types not included in the training dataset. Such models could have difficulty generalising to unseen data. Another potential limitation is the fact that the Gotham dataset is highly imbalanced, which can cause models to be biased and struggle to identify minority classes when deployed. Finally, these models may not operate effectively on small IoT devices due to their high computational cost. Therefore, this limits the applicability of these models in resource-constrained environments.

IoT devices are typically limited in resources, restricting their ability to run complex models; these models must be efficient in terms of memory, storage, and power consumption. For example, DNNs have a large number of parameters that need to be stored for effective decision-making, along with the memory required for operational processes [30]. Additionally, there is a trade-off between accuracy and execution time, especially for IoT problems; for instance, achieving high accuracy might come at the cost of longer execution times. Therefore, models must make quick predictions and respond quickly, particularly in time-sensitive IoT systems [31]. For example, SVM was one of the explored models, where it exhausted resources during training, where no model was produced; for that exact reason, Vakili et al. [31] excluded SVM from their experiments due to its high execution time. Theoretically, SVM has a worst-case time complexity of $O(n^3)$ where the execution time scales as the cube of the number of data points n, which justifies the long training periods required for such models. On the other hand, Random Forest and XGBoost have a time complexity of $O(k \log n)$, where the execution time is proportional to the logarithm of the input size, making them significantly faster than SVM. Logistic regression and Naive Bayes come in the middle with linear time complexity $O(nd)$, which means their runtime grows linearly with input size [32]. Han et al. [32] observed that inference times of these models when deployed in the real world match their theoretical Big-O complexity. They concluded that Logistic regression is efficient in resource-constrained environments, while Naive Bayes has low inference time. However, our study indicates that fast execution times are less relevant when these two models had the lowest accuracy, precision, recall, and F1-score compared to the others. This emphasises the importance of balancing accuracy, lightweight design, and fast, reliable responses suitable for IoT devices.

6 Conclusions and Future Work

We present a comparative analysis of machine learning based intrusion detection systems using the Gotham2025 dataset [9] to help safeguard IoT networks from attacks. The goal is to understand the performance of standard algorithms on realistic and recent IoT data. We evaluated ML algorithms in terms of Accuracy, Precision, Recall, and F1-score as evaluation metrics using the Gotham dataset individually and conducted a comparative analysis between the five machine learning algorithms: Random Forest, XGBoost, Logistic Regression, Naïve Bayes, and Deep Neural Network. Our results show that Random Forest outperforms the other models, achieving a high Macro-Average F1 Score, which demonstrates its effectiveness in detecting attacks on IoT-specific data.

All the settings of our experiments in which these machine learning models are tested are centralised. Recently, the field of Federated Learning has mainly been utilised to preserve privacy [33]. Instead of collecting data from all devices in a centralised data storage, it allows distributed training among IoT devices, which protects privacy by eliminating the need to share their data [9]. For Future work, we intend to examine the potential of testing machine learning models and exploring other deep learning models in federated settings.

References

[1] Ghaida Alqarawi, Bashayer Alkhalifah, Najla Alharbi, and Salim El Khediri. Internet-of-things security and vulnerabilities: Case study. *Journal of Applied Security Research*, 18(3):559–575, 2023.

[2] Debasis Bandyopadhyay and Jaydip Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69, 2011.

[3] Yousaf Bin Zikria, Rashid Ali, Muhammad Khalil Afzal, and Sung Won Kim. Next-generation internet of things (iot): Opportunities, challenges, and solutions. *Sensors*, 21(4), 2021.

[4] Amira Zrelli. Hardware, software platforms, operating systems and routing protocols for internet of things applications. *Wirel. Pers. Commun.*, 122(4):3889–3912, February 2022.

[5] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[6] Chuadhry Mujeeb Ahmed, Gauthama Raman M R, and Aditya P. Mathur. Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. In *Proceedings of the 6th ACM on Cyber-Physical System Security Workshop*, CPSS '20, page 23–29, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. Machine learning in iot security: Current solutions and future challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1686–1721, 2020.

[8] Chuadhry Mujeeb Ahmed, Martin Ochoa, Jianying Zhou, and Aditya Mathur. Scanning the cycle: Timing-based authentication on plcs. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, page 886–900, New York, NY, USA, 2021. Association for Computing Machinery.

[9] Othmane Belarbi, Theodoros Spyridopoulos, Eirini Anthi, Omer Rana, Pietro Carnelli, and Aftab Khan. Gotham dataset 2025: A reproducible large-scale iot network dataset for intrusion detection and security research, 2025.

[10] Xabier Sáez-de Cámara, Jose Luis Flores, Cristóbal Arellano, Aitor Urbieta, and Urko Zurutuza. Gotham testbed: A reproducible iot testbed for security experiments and dataset generation. *IEEE Transactions on Dependable and Secure Computing*, 21(1):186–203, 2024.

[11] Mamoona Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117, 2021.

[12] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC, August 2017. USENIX Association.

[13] Chuadhry Mujeeb Ahmed, Sridhar Adepu, and Aditya Mathur. Limitations of state estimation based cyber attack detection schemes in industrial control systems. In *2016 Smart City Security and Privacy Workshop (SCSP-W)*, pages 1–5, 2016.

[14] Barjinder Kaur, Sajjad Dadkhah, Farzaneh Shoeleh, Euclides Carlos Pinto Neto, Pulei Xiong, Shahrear Iqbal, Philippe Lamontagne, Suprio Ray, and Ali A. Ghorbani. Internet of things (iot) security dataset evolution: Challenges and future directions. *Internet of Things*, 22:100780, 2023.

[15] Mohammad Shahin, Mazdak Maghanaki, Ali Hosseinzadeh, and F. Frank Chen. Advancing network security in industrial iot: A deep dive into ai-enabled intrusion detection systems. *Advanced Engineering Informatics*, 62:102685, 2024.

[16] Mohamed Faisal Elrawy, Ali Ismail Awad, and Hesham F. A. Hamed. Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing*, 7(1):21, 2018.

[17] Jahanvi Sharma, Anju Sangwan, and Rishi Pal Singh. A review on evolving domains of internet of things: Architecture, applications, and technical challenges. *International Journal of Communication Systems*, 36(18):e5613, 2023.

[18] Gauthama Raman MR, Chuadhry Mujeeb Ahmed, and Aditya Mathur. Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation. *Cybersecurity*, 4(1):27, 2021.

[19] Andrea Pinto, Luis-Carlos Herrera, Yezid Donoso, and Jairo A. Gutierrez. Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure. *Sensors*, 23(5), 2023.

[20] Usman Adedayo Adeniyi and Akinyemi Moruff Oyelakin. A survey on promising datasets and recent machine learning approaches for the classification of attacks in internet of things. *Journal of Information Technology and Computing*, 4(2):31–38, Dec. 2023.

[21] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13), 2023.

[22] Muhammad Azmi Umer, Chuadhry Mujeeb Ahmed, Muhammad Taha Jilani, and Aditya P Mathur. Attack rules: an adversarial approach to generate attacks for industrial control systems using machine learning. In *Proceedings of the 2th Workshop on CPS&IoT Security and Privacy*, pages 35–40, 2021.

[23] Md Mahbubur Rahman, Shaharia Al Shakil, and Mizanur Rahman Mustakim. A survey on intrusion detection system in iot networks. *Cyber Security and Applications*, 3:100082, 2025.

[24] Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. MQTT Version 5.0. `https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html`, March 2019. OASIS Standard.

[25] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[26] Anup Rao, Rob Lanphier, and Henning Schulzrinne. Real Time Streaming Protocol (RTSP). RFC 2326, April 1998.

[27] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition, 2019.

[28] Tom M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression. `https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`, 2020. Supplementary chapter to *Machine Learning*.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Rahul Mishra and Hari Gupta. Transforming large-size to lightweight deep neural networks for iot applications. *ACM Comput. Surv.*, 55(11), February 2023.

[31] Meysam Vakili, Mohammad Khosheghbal Ghamsari, and Masoumeh Rezaei. Performance analysis and comparison of machine and deep learning algorithms for iot data classification. *ArXiv*, abs/2001.09636, 2020.

[32] Yu Han, Aaron Ceross, and Jeroen H. M. Bergmann. AI for regulatory affairs: Balancing accuracy, interpretability, and computational cost in medical device classification, 2025.

[33] Mark Devine, Saeid Pourroostaei Ardakani, Mohammed Al-Khafajiy, and Yvonne James. Federated machine learning to enable intrusion detection systems in iot networks. *Electronics*, 14(6), 2025.