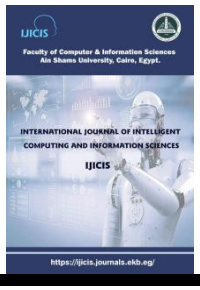
IJICIS, Vol.24, No.2, 18-27
DOI:10.21608/ijicis.2024.273450.1325

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



MACHINE LEARNING FOR DETECTING INTERNET OF THINGS NETWORK CYBER-ATTACK

Imad Tareq AL-Halboosi*

Department of Computer Sciences,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
emadtariq1982@gmail.com

Salsabil El-Regaily

Department of Computer Sciences,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
salsabil_amin@cis.asu.edu.eg

Bassant M. Elbagoury

Department of Computer Sciences,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
Faculty of Computer Sciences and Computer Engineering
King Salman International University
drbassantcs@gmail.com

El-Sayed M. El-Horbaty

Department of Computer Sciences,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
shorbaty@cis.asu.edu.eg

Received 2024-03-01; Revised 2024-04-04; Accepted 2024-04-14

Abstract: *With the proliferation of internet of things devices, guaranteeing the security of these networked systems has become a top priority. Cyberattacks on IoT devices pose considerable risks to individuals and companies because they generate massive amounts of sensitive data across numerous linked devices, making data privacy and integrity a key concern. Machine learning models can help classify different types of cyber-attacks in IoT networks based on logs of activities, analyze behaviors, and predict malicious or unusual activities. This research employs a parallel method utilizing Machine Learning techniques such as LDA, SVM, SVM+LDA, and QDA, on the WUSTL-IIOT database and compares it with traditional methods. The data is partitioned into smaller training datasets and trained in parallel. Experiments show that this parallel training system detects and forecasts cyber threats more accurately. The detection speed with the parallel ML models was high, and the best accuracy was 100% using the SVM+LDA model.*

Keywords: *AI, Cybersecurity, Internet of Things, Machine Learning.*

1. Introduction

In the new AI era, the use of IoT devices in many industries has grown widespread due to significant developments in computer hardware and software technologies, resulting in an exponential rise in data

***Corresponding Author**: Imad Tareq AL-Halboosi

Computer Sciences Department, Faculty of Computer and Information Science, Ain Shams University, Cairo, Egypt

Email address: emadtariq1982@gmail.com

collection. IoT technology has been integrated into a variety of industries, including manufacturing, mobile phones, networks, home devices, healthcare, and smart environments, making sensitive data security critical. As large-scale assaults increase, there is an urgent need for systems to detect and protect against cyber attackers. These flaws potentially jeopardize the security of smart environments developed around IoT systems [1]. DoS, MITM, DDoS, Ransomware, and Botnet are common security risks in IoT networks. Furthermore, IoT devices frequently use distinct protocols and specs, making intrusion detection a difficult operation [2]. As a result, there is an urgent need for robust security solutions customized to IoT devices. To solve this issue, Intrusion detection systems (IDSs) serve a vital role in detecting and preventing security breaches in IoT environments [3]. Machine learning (ML) approaches can be utilized for prediction and detection by learning features in advance. IDS can be classified as unsupervised or supervised based on the classifier's training method. Supervised learning learns labelled training samples as much as possible to predict data outside the training sample set, whereas unsupervised learning learns unlabeled training samples to discover structural information in the training sample set [4]. Researching ML aims to speed and enhance detection and prediction in current attacks. Parallelization represents the future of computing, driven by low cost and energy consumption in parallel computing architecture development. This article employs parallelism with machine learning to address data sets and detect cyber-attacks using parallel support vector machine (SVM), quadratic discriminant analysis (QDA), and linear discriminant analysis (LDA) optimization algorithms, based on the new WUSTL-IIOT database, which was implemented to detect and predict attacks. This work leverages the new WUSTL-IIOT database to detect cyber assaults on IoT applications. We made the following research contributions:

• Data analysis is presented, and the performance of ML techniques is assessed in parallel and sequence learning modes and compared.
• There is no previous study that used this model on a WUSTL-IIOT data set.
• We used machine learning as SVM, LDA, and QDA on the WUSTL-IIOT database in parallel to get a higher computation speed to detect cyber-attacks in IoT. The remainder of this work is organized as follows: The associated literature review in Section 2. The datasets are described in Section 3. Section 4 describes the proposed models. Section 5 discusses the evaluation criteria and experimental results. Section 6 conclusions and future works.

2. Literature Review

The growing use of the IoT has opened up many new study avenues, one of which is the detection and categorization of cyber-attacks in IoT devices, and numerous studies have been proposed to protect IoT networks from malicious assaults. In this section, we shall review some of the most recent studies. In [5], the utility of LDA is demonstrated as an attribute reduction method for big data classification. The classifier's performance was evaluated before and after using LDA as a dimensionality reduction technique. The classifier was tested on six different datasets from the UCI ML repository, and the results showed that after LDA was applied to the dataset, the best accuracy was 90.45%. The work in [6] demonstrates using a parallel SVM intrusion detection framework with feature reduction for unbalanced datasets. SVM algorithm that combines clustering and classification in parallel. The proposed method is evaluated using the NSL-KDD dataset and has an accuracy of 99.53%. The work in [7] presents E2I3DS, an ML-based intrusion detection for industrial IoT. This approach decreases the number of required features for the WUSTL-IIOT-2021 dataset from 48 to 11. Experiments using the 11-feature dataset revealed that the suggested system had 99% accuracy. In [8] suggested an anomaly detection system for fog nodes in a smart city using ensemble methods, including RF and ET bagging techniques. Additionally, the NIMS botnet and UNSW-NB15 datasets containing simulated IoT sensor data were employed achieve

classification accuracy of 99.34%. In [9], several IDS models are implemented utilizing six ML algorithms (RF, SVM, DT, KNN, NB, and LR). These models were assessed using the WUSTL-IIoT-2021 dataset. Among the six models, the model developed using the RF algorithm attained an accuracy of 99.97%, followed by SVM at 99.6%, 99.96%, 99.89%, 87.42%, and 99.37% respectively. The authors of [10] propose an intrusion detection model based on the Particle Swarm Optimization (PSO) and Bat algorithm (BA) for feature selection, as well as the RF classifier for the classification of malicious behaviors in IIoT-based network traffic. The proposed model's performance is evaluated using the WUSTL-IIOT-2021 dataset with 16 features. The top score was 99.99%. This paper used the parallel approach with machine learning (ML) as SVM, LDA, and QDA, in the WUSTL-IIOT datasets.

3. Dataset Description

This paper uses WUSTL-IIOT dataset; IoT applications are used in ML based intrusion detection systems. The WUSTL-IIOT -2021 Washington University in St Louis developed a cybersecurity-focused network-driven dataset of IoT applications [11]. Through the simulation and modeling of real-world industrial systems, the architecture used to simulate actual industrial applications includes a variety of IoT sensors and actuators, a logger, an HMI, a PLC, and an alarming device. There are 1,194,464 samples in the data collection, including 87,016 for malicious and 1,107,448 for benign samples. It is made up of 2.7 GB of data that was collected over 53 hours. The dataset contain 41 attributes chosen based on how their values changed throughout the attack phases. Attacks such as DOS, command injection, reconnaissance, and backdoors are used in the test bed [12], divided into five categories, as shown in Table 1.

Table 1. Shows the overall number of records in the WUSTL-IIOT dataset and the different types of records.

IoT traffic	Type of event	Data record
Attack	Normal	1107448
	DoS	78305
	Reconn	8240
	CommInj	259
	Backdoor	212

4. Proposed Models

This section describes the proposed parallel ML models for detecting cyber attacks in IoT devices a model that uses parallel computing techniques to detect cyber attacks. This model uses SVM, LDA, and QDA algorithms to analyze data and detect abnormal signals that may indicate attacks.

4.1. Parallel-Trained Models

We train several $\boldsymbol{N}$ SVM and LDA in parallel on several $\boldsymbol{N}$ data chunks, then use the predicted probabilities as the new data chunk. Finally, we minimize the number of workers to $\boldsymbol{N/2}$ at each level.The total size of probabilities minimizes the data size as we train the following N new-level models on the previous N previous model's probabilities. Additionally, the number of parallel models operating concurrently is lowered by half at each stage. The process is depicted in Figure 1.

Number of parallel models = $\boldsymbol{N_{process}}/2$
Data Size = Length ($\boldsymbol{models' probabilities}$)

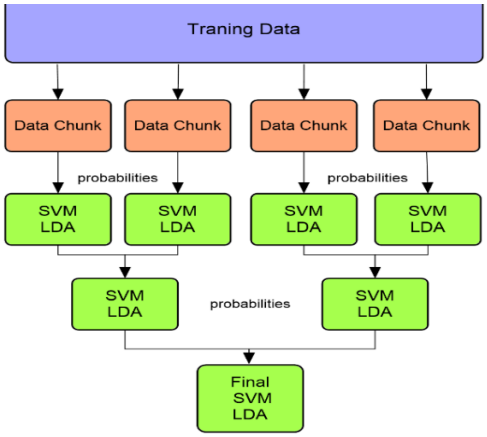


Figure 1. Parallel SVM and LDA Training

Figure 1 shows a schematic with several options for partitioning training data and merging the results. The support vector sets from two SVM classifiers are combined, and the identification of support vectors in each combined subset is repeated until only one set of vectors remains. This strategy has the advantage of not requiring each SVM to handle the whole training dataset, and it allows for the training of numerous SVM classifiers over a distributed computer network, considerably speeding up the training process. While this cascade arrangement frequently produces adequate accuracy in a single pass, obtaining the optimal may necessitate feeding the result of the final layer back to the initial one. As a result, it is critical to understand when feedback is required and how to effectively collect support vectors.

Algorithm1: Predict_Layer

Inputs: Input: $\boldsymbol{X}$ train, $\boldsymbol{Y}$ train, $\boldsymbol{N_{process}}$ will be used in this layer, with parallel Boolean flag, layer models list of model blocks in the layer

Outputs: result array, $\boldsymbol{y}$ array of labels

1. X layer = split dataset ($X, N_{process}$)
2. Y layer = split dataset ($y, N_{process}$)
3. Tmp layer result = list of None for in range of $N_{process}$
4. if with parallel:
   a. Parallel with a length of $N_{process}$:
      i. For index in the range of $N_{process}$:
         1. layer models [index] fit model block with (X layer [index], y layer [index])
         2.tmp layer result [index] = layer models [index] predict model block with (X layer [index])
5. Else:
   b. For index in the range of $N_{process}$:
      i. layer models[index] fit model block with (X layer [index], y layer [index])
      ii. tmp layer result [index] = layer models [index] predict model block with (X layer [index])
6. layer result = []
7. extend layer result with (X samples) for X samples in tmp-layer result
8. convert layer result to an array
9. return layer result, y.

4.2. Parallel Prediction Module Based on Voting Classifiers

We employ a parallelized voting classifier, which employs several classification cells that run in parallel, each of which may contain one or more classifiers and output a voted weight that we later use to classify the input sample.

Voting Rule: We predict the class labels using the predicted probabilities $\boldsymbol{p}$ for the classifier, with $\hat{\boldsymbol{y}}$ representing the predicted class for the input sample.

$$\hat{y} = arg\,max \sum_i^m W_i\, P_{ij} \quad (1)$$

$$Where\ W_i\ Is\ the\ weight\ assigned\ to\ the\ classifier\ j\ .$$

Let $\boldsymbol{x}$ be a classification sample belonging to class labels $\boldsymbol{i}$ where $\boldsymbol{i} \in [\mathbf{0}, \mathbf{1}]$, as in a multi-classification problem. Assume having two cells with two parallel classifiers as the following: $\boldsymbol{Cif_{00}}$. The prediction sequence for every classifier in a cell computes a decision probability, as shown in Figure 2.

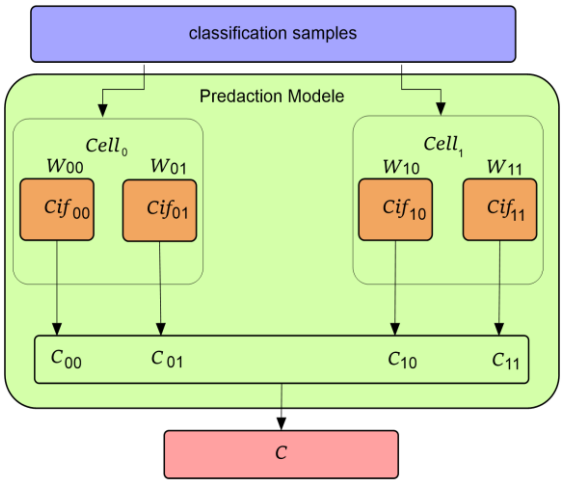


Figure 2. Shows the parallelized voting classifier with several parallel classification cells running.

Let the classifiers' prediction probabilities be as follows:

$\boldsymbol{C_{00}(x) \rightarrow [0.3, 0, 7]}$, $\boldsymbol{C_{01}(x) \rightarrow [0.8, 0, 2]}$

$\boldsymbol{C_{10}(x) \rightarrow [0.4, 0, 6]}$, $\boldsymbol{C_{11}(x) \rightarrow [0.2, 0, 8]}$

Calculate the final decision by weighting each classifier vote as a soft voting technique. $\boldsymbol{W_{ik}}$ as the associated weight with each classifier. The calculated average of the total votes by the classifiers using uniform unbiased weights:

$\boldsymbol{P(i_0|x)} \frac{\mathbf{0.3+0.8+0.4+0.2}}{\mathbf{4}} = \mathbf{0.425}$

$\boldsymbol{P(i_1|x)} \frac{\mathbf{0.7+0.2+0.8+0.6}}{\mathbf{4}} = \mathbf{0.575}$

$\hat{\boldsymbol{y}} = \boldsymbol{max(0.425, 0.575) = 0.575}$ The final predicted class is $\boldsymbol{i = 1}$.

If weighting were not average, the results would change.

Assume normalized weight vector

$\boldsymbol{W = [0.10, 0.40, 0.40, 0.10]}$. We compute the final probability as follows:

$$\boldsymbol{P(i\_0 \mid x) = (0.3 * 0.10) + (0.8 * 0.40) + (0.4 * 0.40) + (0.2 * 0.10) = 0.53}$$

$$P(i\_1 \mid x) = (0.7 * 0.10) + (0.2 * 0.40) + (0.6 * 0.40) + (0.8 * 0.10) = 0.47$$

$\hat{y} = max(0.53, 0.47) = 0.53$ The final predicted class is $i = 1$.

4.3. Parallel Prediction Module

We run each classifier's prediction in parallel and then reduce the probabilities in parallel to get the final probabilities for each cell. The final decision is then computed by selecting the maximum predicted probability. Figure 3 for the parallel estimation module is shown below.

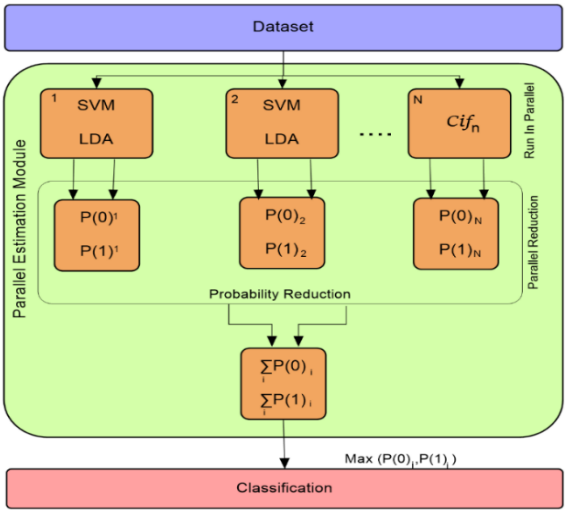


Figure 3. Shows the parallel estimation module

4.4. Data Pre-Processing

Before the classification task, the data is pre-processed. The following tasks are completed during the pre-processing stage.

- We are removing ['StartTime,' 'LastTime,' 'SrcAddr,' 'DstAddr,' 'sIpId,' 'dIpId'] as mentioned on the original doc of the dataset.
- Removing 'Target' as we will work multi-classification, and thus contain the binary label for the dataset.
- we will remove any highly correlated features that are above 0.95, The final features columns will be ['Mean,' 'SrcPkts,' 'Sport,' 'Dport,' 'SrcBytes,' 'DstPkts,' 'DstBytes,' 'TotBytes,' 'SrcLoad,' 'Loss,' 'DstLoad,' 'SrcLoss,' 'DstLoss,"pLoss,' 'SrcJitter,' 'SIntPkt,' 'DstJitter, 'DIntPkt,' 'DstJitAct, "Proto,' 'Dur,' 'TcpRtt,' 'IdleTime,' 'TotAppByte,' 'sDSb,' 'dTtl,' 'sTtl,' 'SAppBytes,' 'SrcJitAct'].
- Min-Max Scalar: a data-preprocessing step used by several machine learning methods for numerical features. The lowest and maximum features are equivalent to zero and one, respectively. The Min-Max Scaler decreases data within a defined range, often between zero and one. To modify data, it scales attributes to a given range. It fits the values within a specific range while keeping the original distribution's shape. The Min-Max scaling is carried out using the:

X_std = ((x-x.min (axis=0)/ (x.max (axis=0) - x.min (axis=0)) X_Scaled = x-std*(max-min) +min. (2)

5. Evaluation Criteria and Experimental Results

For every assignment, different indicators and metrics can be used to evaluate any learning model, including accuracy, F1 score, recall, and precision. These measures were created using data from True Positive (***TP***), False Positive (***FP***), True Negative (***TN***), and False Negative (***FN***). The incorrectly recognized legitimate and attack vectors were FP and FN, respectively. The terms ***TP*** and ***TN*** relate to the number of genuine attack vectors that have been effectively classified.

- Accuracy: Accuracy: represents the percentage of correctly categorized samples and applications in a dataset. The higher the accuracy, the more precise the classifier.

$$\text{Accuracy} = (\text{TN} + \text{TP})/(\text{TP} + \text{FN} + \text{FP} + \text{TN}) \quad (3)$$

- Precision: refers to the accurate detection of benign and positive samples and applications in the dataset. A classifier with higher precision performs better and is more desirable.

$$\text{Precision} = \text{TP}/(\text{FP} + \text{TP}) \quad (4)$$

- F1-Score: The F1 is calculated by computing the harmonic mean of a classifier's recall and precision.

$$\text{F1} - \text{score} = 2.(\text{Precision} * \text{recall}) / (\text{precision} + \text{recall}) \quad (5)$$

- Recall: This measure calculates the proportion of true positive predictions among all possible positive forecasts.

$$\text{Recall} = \text{TP}/\text{TP} + \text{FN} \quad (6)$$

5.1 Conducted Experiment

We use parallelization to get a higher computation speed with experimental multi-classification WUSTL-IIOT datasets that comprise 1,107,448 attack and normal data records. The training set had 955008 vectors, whereas the test set contained 238753 vectors and 29 features.The normal and attack vectors from five classes (Normal, DoS, CommInj, Reconn, and Backdoor) are included in the data. These datasets were split into 80% training and 20% testing sets, lowering the danger of overfitting while allowing for extensive model efficacy monitoring over the full dataset. We used class weights since the data were imbalanced. We used two classification cells containing two sub-classifiers, LDA and a SVM. All experiments were carried out on a CPU: Intel(R) Xeon(R) CPU @ 2.20GH, 4 cores, 2 threads per core, 32 GB RAM.

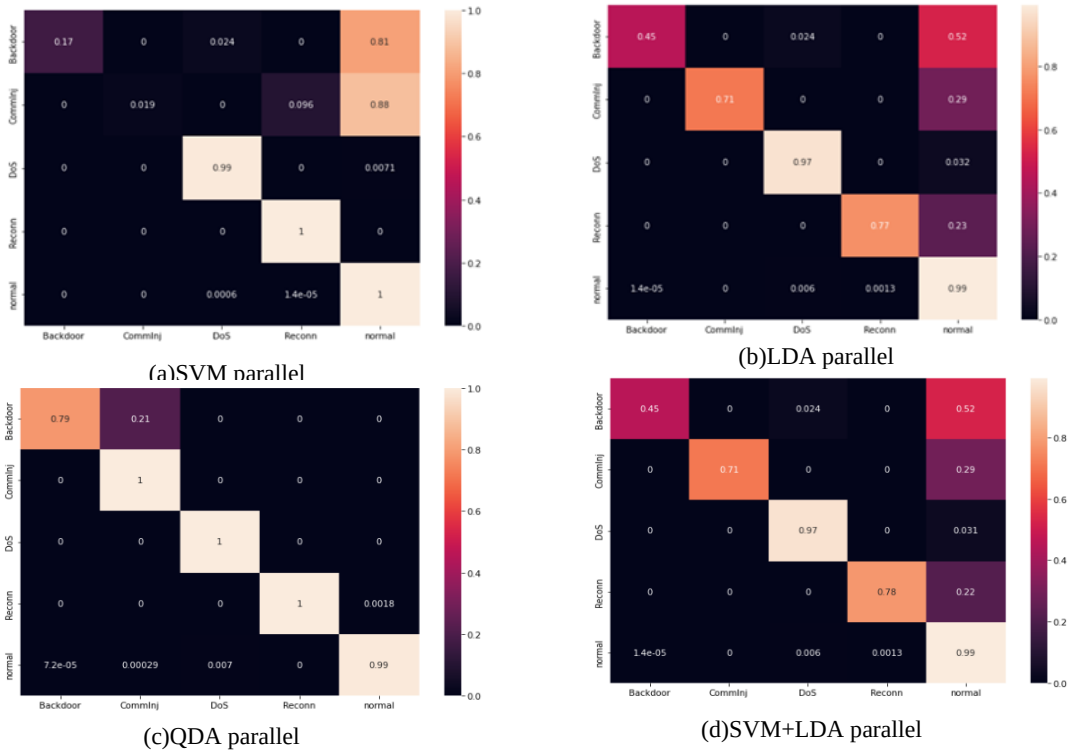
5.2. Experiments and Results

In this section, we offer experimental results on cyber-attack detection to the WUSTL-IIOT datasets to compare the proposed parallel SVM, LDA, QDA, and SVM+LDA with the one based on a sequential structure. We evaluate the train's performance and predict accuracy, recall, loss, precision, and F1-score. The experiment is prepared as follows: We first chunk our data into four training datasets to train SVM classifiers and randomly took 238753 samples that were previously labeled as test sets. The remaining

samples from the training datasets are 955008, with a feature vector of 29. All algorithms are written in Python and employ a polynomial kernel. To assess the performance of these methods, we run four tests using the same datasets: SVM, LDA, QDA, and SVM+LDA. The evaluation results of the parallel machine learning approach for multi-classification are presented in Table 2. In particular, the results display training time, prediction, accuracy, recall, precision, and F1-score. Figure 4 illustrates the confusion matrix. The results show that the least training time is using the LDA model, which is faster in detection attacks, but the best accuracy we got was using (SVM + LDA), where it was 100% using parallel models. Table 3 presents the results of these experiments based on sequential mode.

Table 2. Show the number of layers with the processor, time overall in seconds (s), prediction, and accuracy in parallel mode.

model	Layer	Time (s)	Train	Predict	Accuracy	Precision	Recall	F1-score
SVM	4	3392.61	21000	190.138	99.86	99.71	63.56	66.24
	2	2409.62						
	1	15197.81						
LDA	4	4.355	8.946	0.928	98.92	91.77	77.80	83.66
	2	1.812						
	1	2.054						
QDA	4	43.221	80.135	1.276	99.30	80.76	95.52	85.14
	2	11.931						
	1	24.900						
SVM+LDA	4	3983.31	413.35	192.916	100	98.48	92.54	94.84
	2	82.68						
	1	65.27						



(a)SVM parallel (b)LDA parallel

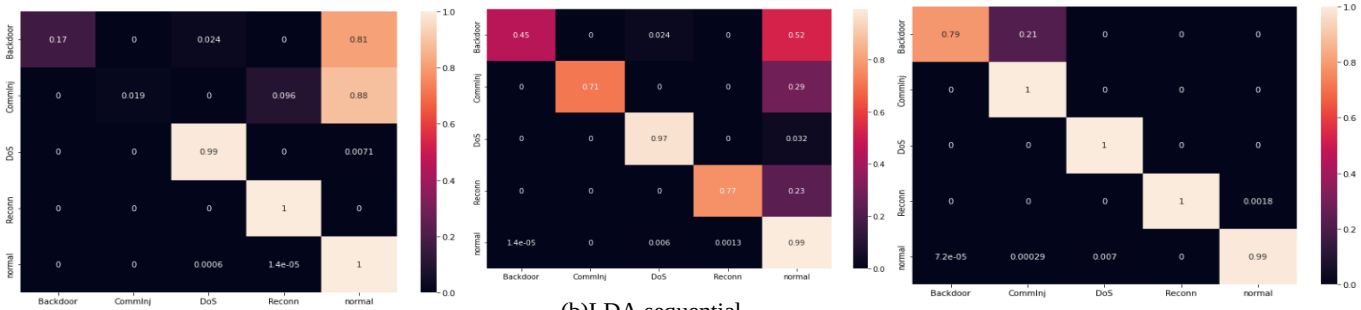
(c)QDA parallel (d)SVM+LDA parallel

Figure 4. (a), (b), (c) (d)The confusion matrix for five classes in the WUSTL-IIOT datasets using parallels validation: Normal, DoS, CommInj, Reconn, and Backdoor, all values between 45 and 100.

In Table 3. The sequential mode results show that the least training time is using the LDA model, which is faster in predicting attacks, but the best accuracy we got was using QDA, which was 99.99%. Figure 5 illustrates the confusion matrix.

Table 3. Show the number of layers with processor, time overall in seconds (s), predict and accuracy in sequential mode.

model	Layer	Time (s)	Train	Predict	Accuracy	Precision	Recall	F1-score
SVM	4	9866.304	279.024	394.463	99.86	99.71	63.56	66.24
	2	5369.713						
	1	12724.91						
LDA	4	6.026	9.482	0.602	98.92	91.77	77.80	83
	2	1.685						
	1	1.660						
QDA	4	81.102	123.485	1.276	99.99	98.51	94.94	96.46
	2	20.452						
	1	21.842						



(a)SVM sequential (b)LDA sequential (c)QDA sequential

Figure 5. (a), (b), (c) The confusion matrix for 5 classes in the WUSTL-IIOT datasets using sequential validation: Normal, DoS, CommInj, Reconn, and Backdoor, all values between 17 and 100.

Our experiment used the parallel approach on the new WUSTL-IIOT database. In our experiments, we used each model (SVM, LDA, QDA, SVM+LDA) in parallel and sequential modes and compared the results. The best result we obtained in the parallel approach was using the SVM+LDA model with an accuracy of 100, with the sequential the best result for the QDA model with an accuracy of 99.99. the results As shown in tables [2],[3] respectively.

6. Conclusions and Future Work

This paper proposes a methodology for detecting cyber-attacks based on a parallel model. Our goal was to protect IoT devices from malicious cyber-attacks. The experiments used four parallel machine learning models to detect unwanted cyber-attacks. Our experiments on the modern data set WUSTL-IIOT were conducted. Results were compared used accuracy, precision, F1 score, and recall. we used the WUSTL-IIOT database with parallel machine learning, with high speed and accuracy in detecting cyberattacks in IoT devices. Experiments used the limited memory and CPU resources of the test time. Experiments demonstrated that this parallel training system detects and predicts cyber threats with greater accuracy.

In future work, we hope to improve the models' reliability when malicious edge nodes are on the network. Furthermore, we will concentrate on the filtering process used to detect poisoning attempts.

The machine learning approach will likely cause privacy difficulties if there are untrusted servers or clients. These are just a few ideas to consider for future experiments. Before implementation, it is important to thoroughly research and evaluate any proposed approach's feasibility, effectiveness, and security implications.

Acknowledgments: We would like to thank Eng. Mohamed Ahmed, for his tremendous efforts in coding this research, mohamed.ahm.cs@gmail.com.

References

1. A. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2017.
2. I. Idrissi, M. Azizi, and O. Moussaoui, "IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review," 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), p. 1, 2020.
3. A. Chauhan, R. Singh, and P. Jain, "A Literature Review: Intrusion Detection Systems in Internet of Things," Journal of Physics: Conference Series, vol. 1518, p. 1, 2020.
4. Z. A. E. Houda, B. Brik, and S.-M. Senouci, "A Novel IoT-Based Explainable Deep Learning Framework for Intrusion Detection Systems," IEEE Internet of Things Magazine, vol. 5, p. 20, 2022.
5. Ahmed Hussein Ali, Zahraa Faiz Hussain, and Shamis N. Abd, “Big Data Classification Efficiency Based on Linear Discriminant Analysis,” Iraqi J. Comput. Sci. Math., pp. 7–12, 2020, doi: 10.52866/ijcsm.2019.01.01.001.
6. J. Zhao et al., “An effective parallel SVM intrusion detection model for imbalanced training datasets,” ICEIS 2020 - Proc. 22nd Int. Conf. Enterp. Inf. Syst., vol. 2, no. Iceis 2020, pp. 225–232, 2020, doi: 10.5220/0009390302250232.
7. Alani MM. An explainable efficient flow-based Industrial IoT intrusion detection system. Computers and Electrical Engineering. 2023 May 1;108:108732.Vigoya L, Fernandez D, Carneiro V, Nóvoa FJ. IoT Dataset Validation Using Machine Learning Techniques for Traffic Anomaly Detection. Electronics. 2021 Nov 19;10(22):2857.
8. Alrashdi, I.; Alqazzaz, A.; Aloufi, E.; Alharthi, R.; Zohdy, M.; Ming, H. AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 305–310.
9. Eid AM, Nassif AB, Soudan B, Injadat MN. IIoT network intrusion detection using machine learning. In2023 6th International Conference on Intelligent Robotics and Control Engineering (IRCE) 2023 Aug 4 (pp. 196-201). IEEE.
10. Gaber T, Awotunde JB, Folorunso SO, Ajagbe SA, Eldesouky E. Industrial internet of things intrusion detection method using machine learning and optimization techniques. Wireless Communications and Mobile Computing. 2023 Apr 30;2023:1-5.
11. M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain. "WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research," Washington University in St. Louis, USA, October 2021, https://www.cse.wustl.edu/~jain/iiot2/index.html.
12. M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin, “TRUST XAI: Model-Agnostic Explanations for AI With a Case Study on IIoT Security,” IEEE Internet Things J., vol. 10, no. 4, pp. 2967–2978, 2023, doi: 10.1109/JIOT.2021.3122019.