2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) | 979-8-3503-9172-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ANTS63515.2024.10898580

AURA: Adaptive Unified Real-Time Analytics for IoT Intrusion Detection

Sanket Mishra, Aravindan V, Rajkanwar Singh, Vikash Kumar Singh
School of Computer Science & Engineering,
VIT-AP University, Amaravati,India
sanketmishra@live.com,aravindanvemula2@gmail.com,rajkanwars15@outlook.com,vikas.1688@gmail.com

***Abstract*—In high-volume, high-velocity contexts, threat identification requires effective real-time data stream analysis. This study offers a novel architecture—real-time processing of high-speed data streams—that is critical for effective threat identification in dynamic contexts. By using a publish-subscribe approach with Apache Kafka, the system is able to manage differences in data volume between many nodes. Experiments on the CICIoV24 and the CICEVSE2024 datasets indicate that the XGBoost model performs better, with high accuracy and robustness against adversarial attacks. Its performance degrades during the HopSkipJump attack, however, defence training can help with it. Moreover, our analysis shows that RandomForest and ExtraTrees perform better in noisy data from the CICIoV24 and XGBoost perform better in noisy data from the CICEVSE24 dataset, emphasizing the importance of selecting algorithms based on performance indicators. The architecture utilizes PyFlink's distributed computation framework to improve computational efficiency for real-time processing and solves idea drift to ensure flexibility in changing data attributes.**

***Index Terms*—Real-Time Data Processing, Apache Kafka, Apache Flink, Deep Learning, Machine Learning, Model Robustness, Drift Detection**

I. Introduction

The growing number of IoT devices has made adaptive and resilient intrusion detection systems (IDS) more important in the rapidly changing field of cyber security. The ongoing increase in cyber attacks has highlighted the need for real-time threat detection systems that can adapt to these difficulties. IoT networks provide particular difficulties in identifying and reducing security threats because of their high-volume, high-velocity data inflow. A thorough architecture that incorporates advanced drift detection algorithms and ensures adversarial robustness has been designed for real-time intrusion detection in Internet of Things (IoT) environments in order to meet these complications. Modern deep learning and machine learning models have been applied to efficiently process fast data streams from IoT devices [18]. This architecture is built on a distributed framework utilizing Apache Kafka and PyFlink, which facilitates scalable and fault-tolerant data transmission and threat classification. Specialized algorithms have been employed to monitor changes in data distributions, allowing for real-time adaptation to evolving threats and ensuring system reliability. Additionally, the architecture is capable of detecting unknown threats by continuously updating its models through adversarial training, thereby providing an extra layer of security. Techniques for ensemble learning have been applied to improve the system's accuracy and robustness to noisy data. In Internet of Things contexts, model retraining procedures guarantee real-time adaptability to new attack vectors. The security of IoT networks has been greatly strengthened by these efforts, leading to a scalable and reliable solution that successfully tackles the dynamic and unexpected nature of cyber threats within the IoT ecosystem.

II. Literature Review

A comparative study [14] was conducted to assess various machine learning algorithms (LightGBM, XGBoost, CatBoost, and LCCDE) for intrusion detection using the CICIoV2024 dataset, where perfect accuracy, recall, precision, and F1-score were reported. However, details regarding training time and train-test splits were not provided. In a replication of LightGBM and XGBoost in a pure Python environment with a 60:40 split, training times were observed to reach several hundred seconds, indicating a need for models with reduced data and features for IoT applications. A system was proposed to address urban traffic congestion utilizing tree-based strategies, including Decision Tree (DT), Random Forest (RF), Extra Tree (ET), and XGBoost, achieving an accuracy of 99.05%, surpassing KNN (96.6%) and SVM (98.01%) [15]. Feature selection through ensemble learning was employed, resulting in enhanced detection accuracy with minimal computational costs. The system was evaluated using the CIC-IDS2017 dataset, and while it showed promise, further exploration of deep learning algorithms for feature selection was suggested to bolster robustness against attacks. Using the CICEVSE2024 dataset, a comparative study was conducted to analyze anomaly detection methods within Electric Vehicle Charging Stations (EVCS), including KNN, RF, SVM, and Neural Networks (NN). Accuracy was reported to range from 77.61% (SVM) to 91.97% (NN), with precision and recall displaying similar trends. While promising metrics were provided, specific details regarding training time and splits were not included. When Federated Learning (FL) models were implemented with Deep Neural

Networks (DNNs), a significant accuracy improvement to 97% was achieved, highlighting FL's potential in decentralized IoT environments. A federated strategy was developed to integrate FL with local models stored on charging station management systems, demonstrating superior detection precision and efficiency. The system's resilience against attacks such as Distributed Denial of Service and Man-in-the-Middle was evaluated using real-world data, though further research was deemed necessary to enhance model robustness through deep learning-based feature selection techniques. The incorporation of AI and ML into real-time threat intelligence frameworks was explored by Ibra Him [4], where real-time data processing for anomaly detection was emphasized. A high-throughput architecture for anomaly detection was proposed by Surianarayanan et al. [2], achieving an accuracy of 98.6% with near-zero latency through the use of Random Forest and Apache Kafka. Mudgal and Bhalla [5] applied Random Forest and k-means clustering to the CIC-IDS 2018 dataset, attaining a precision of 99.66% while reducing latency by 30% using Flink. A Flexible Real-Time Traffic Stream Processing System (FRTSPS) was introduced by Deepthi et al. [6], which outperformed Apache Spark and Apache Storm with a latency reduction of 25%. Another study [7] described a modular IDS architecture designed for real-time data analysis in a 1 Gbps environment. A distributed framework for machine learning-based intrusion detection systems was assessed by Kajiura and Nakamura [8], revealing significant differences in throughput and latency among classifiers. A Distributed Intrusion Detection System for IoT was proposed by Atbib et al. [9], with Decision Tree (86% accuracy) and Random Forest (87% accuracy) outperforming other algorithms. A hybrid model combining Random Forest, XGBoost, and decision trees was presented by Jemili et al. [10], achieving over 97% accuracy across multiple datasets. Finally, an IoT-based cybersecurity framework for the Internet of Drones (IoD) was developed by Ashraf et al. [11], effectively detecting cyberattacks using B-LSTM and LSTM models, which achieved high performance on the CICIDS2017 and KDDCup 99 datasets.

III. Methodology

A. Data Ingestion and Streaming

The data pipeline initiates the real-time acquisition of data from various sources like IoT sensors, network traffic, and other pertinent input channels. The raw data is gathered via a REST API that publishes the information to numerous Apache Kafka topics. Kafka functions as the backbone of the pipeline, offering scalability, fault tolerance, and high throughput within a distributed framework. The API delivers data to Kafka topics, segregating streams based on different categories. These categories are ingested in parallel by various components for further processing.

B. Data Resampling

To handle class imbalance in the dataset, the ImbalancedDatasetSampler in PyTorch is utilized. This sampler addresses the problem by modifying the class distribution, oversampling minority classes, and undersampling majority ones according to specified target counts. Consequently, a balanced dataset is maintained throughout training, improving the model's performance on minority classes. Class distributions are rebalanced during sampling, with sampling weights automatically calculated based on class frequencies. This method removes the necessity of creating a separate balanced dataset.

C. Feature Extraction Using Deep Learning

When data is ingested, the feature extraction stage is triggered using deep learning (DL) models within the system. The pipeline incorporates two models: LSTM (Long Short-Term Memory) and DNN (Deep Neural Networks) for feature extraction. These models are algorithmically refined using techniques like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). PyFlink provides parallel processing, while PSO and GA perform optimization. The feature extraction models operate concurrently, resulting in a race condition between them. The model that completes first publishes its features to a Kafka topic. This real-time strategy ensures that the fastest and most effective model is chosen for subsequent tasks.

D. Machine Learning Model Training

Within the pipeline, several machine learning (ML) models are trained using PyFlink, including Random Forest, XGBoost, LightGBM, Gradient Boosting, and Extra Trees classifiers. These models undergo hyperparameter tuning through techniques like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) to achieve the best performance. Each classifier is trained on a resampled and optimized set of features with five-fold cross-validation employed during the training process. The model chosen for inference is selected according to the procedure described in Algorithm 1.

E. Model Evaluation and Real-Time Predictions

Upon selecting the optimal model, it is deployed to generate real-time predictions on incoming data streams. The predictions produced by the model are continuously updated on a Kafka topic. The model's performance is subject to ongoing monitoring, and in the event of significant performance drift, the model undergoes retraining on the latest data to ensure the maintenance of optimal accuracy.

F. Robustness Testing

Concurrently, Model Poisoning is employed to evaluate the robustness of the trained and selected model. Furthermore, the practice of model assault enhances

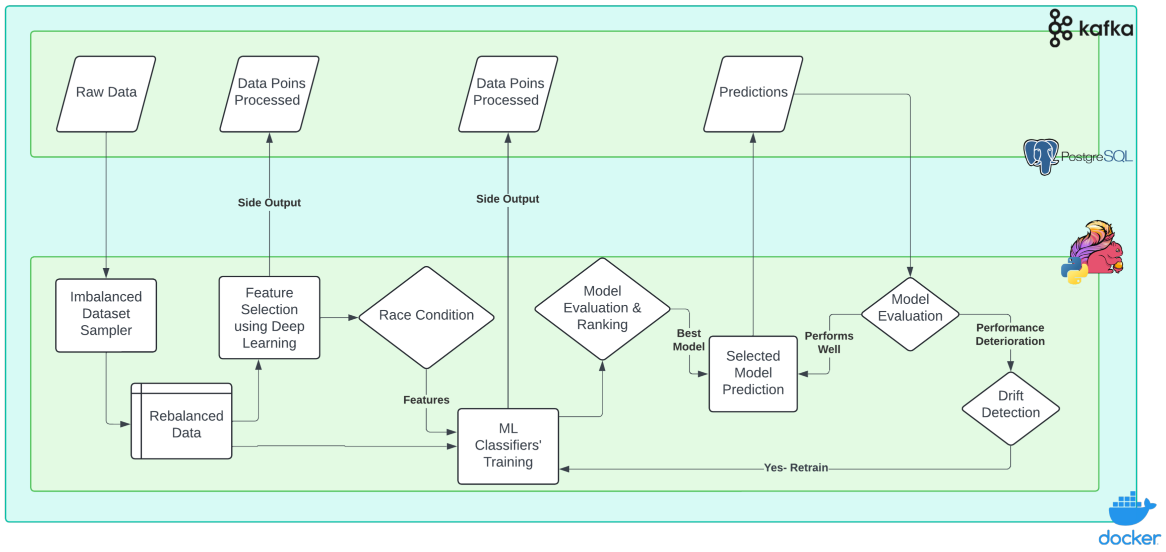

Fig. 1. AURA Framework

Algorithm 1 Rank Model Results

```
1: Input: $ResultsofModels$: Dictionary with model results
2: $df \leftarrow$ pd.DataFrame$(results).T$.reset_index()
3: $df$.rename$(columns = \{`index' : `ModelName'\}, inplace = True)$
4: $df[`CustomRank'] \leftarrow 0.8 \times df[`Accuracy'] + 0.2 \times \left(\frac{60}{df['Time']}\right)$
5: $ranked\_df \leftarrow df$.sort_values$(by = `CustomRank', ascending = False)$.reset_index$(drop = True)$
6: $ranked\_df[`Rank'] \leftarrow ranked\_df$.index $+ 1$
7: Output: Display top 5 rows of $ranked\_df$
8: Save: $ranked\_df$.to_csv$(`ranked\_model\_metrics.csv', index = False)$
```

vulnerability identification, security, and generalizability. Attack simulations elucidate vulnerabilities and facilitate the development of defensive measures. Model attacks also serve as benchmarks for evaluating and contrasting various models and defence strategies, thereby ensuring the creation of more reliable and robust systems.

G. Drift Detection and Model Retraining

The code integrates drift detection to monitor and adapt the model's performance in real-time. The ADWINAccuracy detector tracks changes in accuracy, updating its state with each new data row. If drift is detected, an error is triggered, indicating the row where performance drops. Likewise, the KdqTreeStreaming detector, using a sliding window, monitors changes in feature distributions. An ensemble of both detectors, employing a majority voting mechanism, is used to improve reliability. Drift detection is only invoked when model performance falls below a certain threshold, prompting either retraining or replacement with the next best model. Hyperparameter optimization is skipped to reduce computational load, ensuring faster deployment. This approach ensures the model remains adaptable and efficient as data evolves.

IV. Results and Discussion

A. CICIoV 2024

The AURA framework is tested in the CIC IoV 24 dataset [3]. Since the classes are imbalanced, ImbalancedDatasetSampler is utilised as highlighted in Section III-B and the changes can be seen in Fig. 2A. Based on Algorithm 1, the XGBoost model is selected as the best-performing model. The model was then serialized as inference_model.pkl for the prediction job. Based on the predictions made by the model, a plot of the Decision Boundary can be seen in Fig. 3.

The decision boundary plot of an XGBoost model in fig 3, illustrates the classification behaviour of the model. The x and y-axes represent two features from the dataset, while the color gradient denotes predicted class probabilities. The decision boundary, demarcating the transition between Class 0 (blue) and Class 1 (red) predictions, exhibits an irregular shape, underscoring the model's non-linear decision-making capabilities. This complexity is attributable to

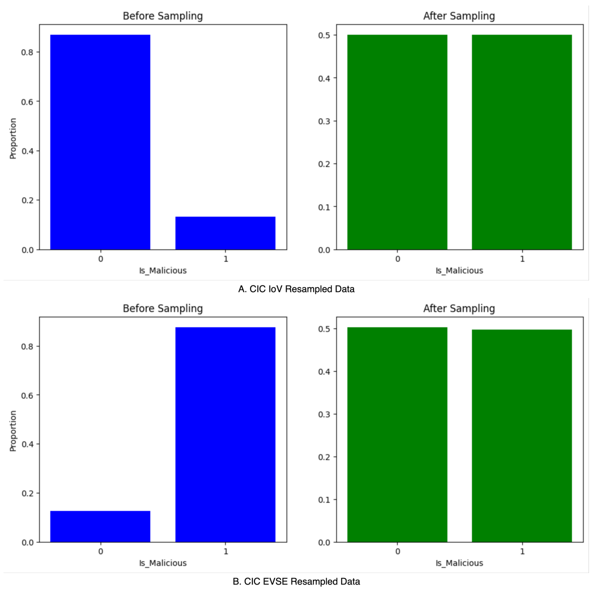

Fig. 2. Resampled Data

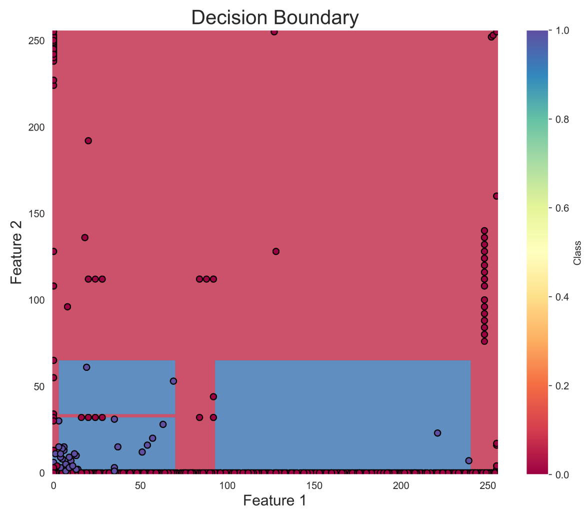

Fig. 3. Decision Boundary of XGBoost Model

the iterative boosting process employed by XGBoost, wherein decision trees are sequentially constructed to refine errors made by previous trees. The data points, represented by black circles, demonstrate how the model separates the classes. Points in proximity to the decision boundary indicate regions of reduced confidence, where predictions are more susceptible to change. Notable instances of misclassification or ambiguity are observed, as evidenced by blue patches within red regions and vice versa. While XGBoost's complexity enables the capture of intricate patterns, it also increases the risk of overfitting, particularly in sparsely populated areas of the feature space. The trade-off between classification accuracy and generalization is thus highlighted, underscoring the importance of carefully balancing model complexity and regularization.

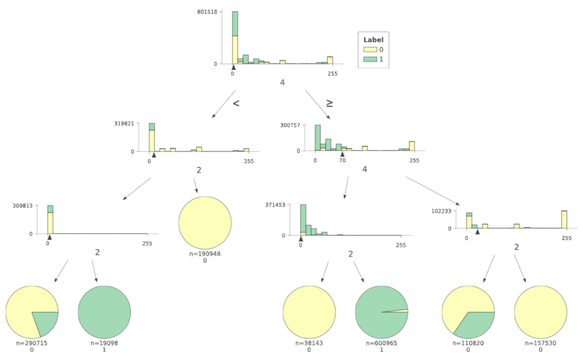

Fig. 4. Visualisation of XGBoost Model

In Fig. 4, the decision tree of the XGBoost model is illustrated, showcasing how the dataset is split based on various features, such as '2' and '4'. Each path is depicted to represent a sequence of splits, guiding samples through decision points. This process continues until the samples are directed to the terminal leaf nodes, where final predictions are generated.

The performance of the five machine learning models is evaluated, utilising 5-fold cross-validation and ranked based on overall accuracy, precision, recall, F1 score, and training time, as shown in Table I.

The overall metrics of all models can be seen in Table I. While there is not a major difference in the accuracy of each of the models, a major difference can be observed in training time. In such scenarios Algorithm 1 is quite useful to select one model for inference.

TABLE I
RANKED MODELS WITH METRICS

Rank	Model	Accuracy	Precision	Recall	F1 Score	Training Time
1	XGBoost	0.960645	0.963521	0.960645	0.960584	3.251693
2	LightGBM	0.960645	0.963521	0.960645	0.960584	8.02442
3	ExtraTrees	0.960645	0.963521	0.960645	0.960584	24.491888
4	RandomForest	0.960645	0.963521	0.960645	0.960584	24.619227
5	GradientBoosting	0.960645	0.963521	0.960645	0.960584	30.115168

To ensure robustness, 5-fold cross-validation was performed for each model. Figure 5 depicts cross-validation performed on the highest ranked model, as per Algorithm 1, using a parallel coordinate plot.

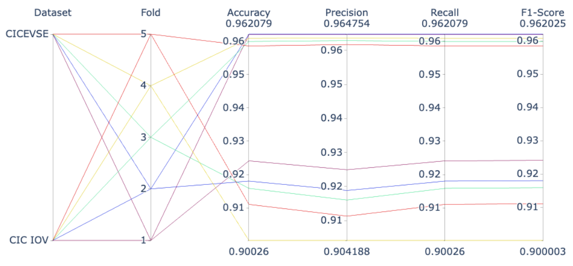

Fig. 5. 5-fold CV on XGBoost Model Training

To determine vulnerabilities, two black box tests were applied to the best-performing model, selected for inference. Their metrics can be seen in Table II.

The Zeroth Order Optimization (ZOO) attack was applied to the best-performing model (XGBoost). Specifically, the ZOO attack proposed by Chen et al. [12] was employed. This attack, which operates without any knowledge of the model's internal structure, relies solely on the model's input-output mappings. The ZOO attack utilizes iterative querying of the model with strategically chosen inputs to estimate the direction towards optimal adversarial examples. In this attack, a coordinate descent algorithm was used to optimize the input vector iteratively. During each iteration, a coordinate was selected for updating, and a binary search was performed to determine the optimal value for that coordinate. The model's performance does not deteriorate on the attack, as can be seen in Fig. 6A.

The HopSkipJumpAttack [13] was applied to the XGBoost model, which operates in a decision-based black-box setting where only output labels are observable. HopSkipJumpAttack estimates the gradient direction through random sampling around the decision boundary and employs an iterative approach with binary search and gradient estimation. This attack was designed to generate adversarial examples by making imperceptible alterations to inputs, thereby causing the model to produce incorrect predictions, as can be seen in Fig. 6B By exploiting the model's sensitivity to these subtle changes, HSJA successfully compromised the model's accuracy. Following the application of HSJA, the model's performance deteriorated. To address this vulnerability, a defense mechanism was subsequently trained. This defence effectively counteracted the attack, leading to the restoration of the model's performance, as seen in Fig. 6C.

TABLE II
PERFORMANCE OF XGBOOST UNDER ATTACK

Attack	Class	Accuracy	Precision	Recall	F1 Score
HOPSKIPJUMP ATTACK	0	0.50	0.5	1	0.67
HOPSKIPJUMP ATTACK	1	0.00	0.00	0.00	0.92
ZOO Attack	0	0.92	0.87	1.00	0.93
ZOO Attack	1	0.92	1.00	0.85	0.92

TABLE III
PERFORMANCE OF XGBOOST AFTER DEFENCE TRAINING

Attack	Class	Accuracy	Precision	Recall	F1 Score
Post Defence	0	0.96	0.92	1.00	0.96
Post Defence	1	0.96	1.00	0.92	0.96

Table III outlines the improved performance of the XGBoost model after defence training, demonstrating restored accuracy (96%) and robustness against the attack, ensuring reliability under adversarial conditions.

To simulate drift, gaussian noise is added to the training data, and deterioration in performance can be seen in table IV. This drift is detected by Adaptive Windowing (ADWIN). ADWIN operates by maintaining a sliding window of the most recent data and continuously monitoring the accuracy of the model on this window. Upon drift detection, the model is retrained.

TABLE IV
MODEL PERFORMANCES ON GAUSSIAN NOISES

Rank	Model	Accuracy	Precision	Recall	F1 Score
1	ExtraTrees	0.9493	0.9529	0.9493	0.9492
2	RandomForest	0.9382	0.9423	0.9382	0.9381
3	GradientBoosting	0.9309	0.9339	0.9309	0.9307
4	XGBoost	0.9231	0.9315	0.9230	0.9227
5	LightGBM	0.9135	0.9138	0.9135	0.9135

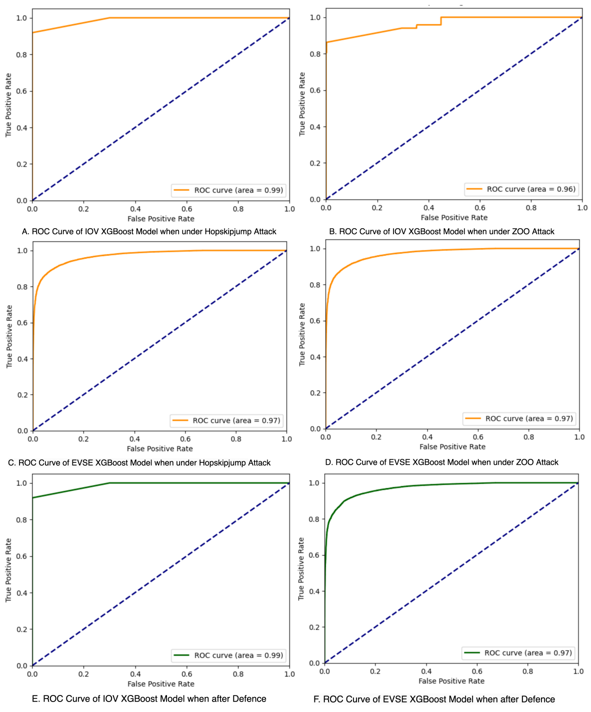

Fig. 6. ROC Curve of XGBoost Model under attack and defence retrained

B. *CIC EVSE 2024*

The AURA framework has been analyzed using the CIC EVSE 2024 dataset [16]. The classes are not balanced, as indicated in Section III-B, therefore Imbalanced Dataset Sampler (2B) is also utilized. Once again, the XGBoost model is the best-performing model, as decided by algorithm 1. The outcomes of the five machine learning models' 5-fold cross-validation evaluation are shown in Table V. The following factors are used to rank the models: recall, F1 score, overall accuracy, precision, and training duration.

The overall metrics of all models can be seen in Table V. While there is not a major difference in the accuracy of each of the models, a major difference can be observed in training time. In such scenarios Algorithm 1 is quite useful to select one model for inference.

TABLE V
RANKED MODELS WITH METRICS ON CICEVSE2024

Rank	Model	Accuracy	Precision	Recall	F1 Score	Training Time
1	XGBoost	0.913869	0.915068	0.913869	0.913795	2.5613472
2	LightGBM	0.912134	0.914059	0.912134	0.912015	5.15944536
3	ExtraTrees	0.909619	0.911001	0.908405	0.908834	15.285908
4	RandomForest	0.904849	0.908438	0.905196	0.904977	17.78417
5	GradientBoosting	0.913956	0.914020	0.913695	0.913516	25.128851

To ensure robustness, 5-fold cross-validation was performed for each model. Figure 5 depicts cross-validation performed on the highest ranked model using a parallel coordinate plot, as per Algorithm 1. Again, to test robustness ZOO and HOPSKIPJUMP attack are utilised.

TABLE VI
PERFORMANCE OF XGBOOST UNDER ATTACK

Attack	Class	Accuracy	Precision	Recall	F1 Score
HOPSKIPJUMP ATTACK	0	0.91	0.89	0.93	0.91
HOPSKIPJUMP ATTACK	1	0.91	0.93	0.89	0.91
ZOO Attack	0	0.91	0.89	0.93	0.91
ZOO Attack	1	0.91	0.93	0.89	0.91

TABLE VII
PERFORMANCE OF XGBOOST AFTER DEFENCE TRAINING

Attack	Class	Accuracy	Precision	Recall	F1 Score
Post Defence	0	0.91	0.89	0.93	0.91
Post Defence	1	0.91	0.93	0.88	0.91

Table VII highlights the improved performance of the XGBoost model after defense training, demonstrating restored accuracy (91%) and robustness against HopSkipJump and ZOO attacks, ensuring reliability under adversarial conditions.

V. CONCLUSION AND FUTURE SCOPE

The work proposes a real-time intrusion detection pipeline was designed for Internet of Things (IoT) data streams using deep learning, machine learning, PyFlink, and Apache Kafka. It is capable of prompt threat detection and performs well in various scenarios with added robustness through drift detection and adversarial framework. Optimization techniques like Particle Swarm Optimization and Genetic Algorithms improved model efficiency. The XGBoost model performed best on the CICIoV dataset. Future improvements could include enhanced protections, support for sophisticated attacks, scalability for larger datasets, explainability mechanisms, and advanced drift detection algorithms to maintain effectiveness in dynamic environments.

REFERENCES

[1] N. Abedzadeh and M. Jacobs, “A Reinforcement Learning Framework with Oversampling and Undersampling Algorithms for Intrusion Detection System,” *Applied Sciences*, vol. 13, no. 20, p. 11275, 2023, doi: 10.3390/app132011275.

[2] C. Surianarayanan, S. Kunasekaran, and P. R. Chelliah, “A high-throughput architecture for anomaly detection in streaming data using machine learning algorithms,” *International Journal of Information Technology*, vol. 16, no. 1, pp. 493-506, Nov. 2023, doi: 10.1007/s41870-023-01585-0.

[3] E. C. P. Neto, H. Taslimasa, S. Dadkhah, S. Iqbal, P. Xiong, T. Rahmanb, and A. A. Ghorbani, ”CICIoV2024: Advancing Realistic IDS Approaches against DoS and Spoofing Attack in IoV CAN bus,” Internet of Things, 101209, 2024.

[4] I. Him. ”Leveraging Artificial Intelligence and Machine Learning for Real-Time Threat Intelligence: Enhancing Incident Response Capabilities,”, 2024.

[5] A. Mudgal, S. Bhatia, ”Big Data Enabled Intrusion Detection with Honeypot Intelligence System on Apache Flink (BDE-IDHIS),” in 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023.

[6] Deepthi, B., et al. ”An efficient architecture for processing real-time traffic data streams using apache flink,” in Multimedia Tools and Applications, vol. 83, no. 13, pp. 37369–37385, 2023.

[7] Oliveira, R., et al. ”Parameterization and Performance Analysis of a Scalable, near Real-Time Packet Capturing Platform,” in Systems, vol. 12, no. 4, pp. 126, 2024.

[8] Maho Kajiura, Junya Nakamura, ”Practical Performance of a Distributed Processing Framework for Machine-Learning-based NIDS,” 2024.

[9] S. Atbib, C. Saadi, H. Chaoui, ”Design of A Distributed Intrusion Detection System for Streaming Data in IoT Environments,” in 2023 9th International Conference on Optimization and Applications (ICOA), 2023, pp. 1-6.

[10] F. Jemili, R. Meddeb, O. Korbaa. ”Intrusion detection based on ensemble learning for big data classification,” in Cluster Computing, vol. 27, no. 3, pp. 3771–3798, 2023.

[11] Ashraf, S., et al. ”IoT empowered smart cybersecurity framework for intrusion detection in internet of drones,” in Scientific Reports, vol. 13, no. 1, 2023.

[12] Chen, P., et al, ”ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models,” in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017.

[13] Jianbo Chen, Michael I. Jordan, Martin J. Wainwright, ”HopSkipJumpAttack: A Query-Efficient Decision-Based Attack,” 2020.

[14] N. Amirudin, S. Abdulkadir. ”Comparative Study of Machine Learning Algorithms using the CICIoV2024 Dataset,” in Platform: A Journal of Science and Technology, vol. 7, no. 1, pp. 1–8, 2024.

[15] Prakash, J., et al. ”A vehicular network based intelligent transport system for smart cities using machine learning algorithms,” in Scientific reports, vol. 14, no. 1, pp. 468, 2024.

[16] E. D. Buedi, A. A. Ghorbani, S. Dadkhah, and R. L. Ferreira, “Enhancing EV charging station security using a multi-dimensional dataset: CICEVSE2024,” in Lecture notes in computer science, 2024, pp. 171–190.

[17] “FL-EVCS: Federated Learning based Anomaly Detection for EV Charging Ecosystem,” IEEE Conference Publication — IEEE Xplore, Jul. 29, 2024. https://ieeexplore.ieee.org/abstract/document/10637543

[18] N. A. Dief, M. M. Salem, A. H. Rabie, and A. I. El-Desouky, “Enhanced transformer long short-term memory framework for datastream prediction,” International Journal of Power Electronics and Drive Systems/International Journal of Electrical and Computer Engineering, vol. 14, no. 1, p. 830, Nov. 2023, doi: 10.11591/ijece.v14i1.pp830-840.