Comparative Analysis of ML Models for Intrusion Detection in IIoT Systems

Mihir Patel, Mitalee Vaghasia, Dhruv Thakkar, Anuja Nair, *Member, IEEE*, Tarjni Vyas, Shivani Desai, Sudeep Tanwar, *Senior Member, IEEE*
Department of Computer Science and Engineering, School of Technology, Nirma University, Ahmedabad, Gujarat, India
Emails: 22bce185@nirmauni.ac.in, 22bce190@nirmauni.ac.in, 21bce300@nirmauni.ac.in, anuja.nair@nirmauni.ac.in, tarjni.vyas@nirmauni.ac.in, shivani.desai@nirmauni.ac.in, sudeep.tanwar@nirmauni.ac.in

***Abstract*—The Industrial Internet of Things (IIoT) has brought about numerous advantages in the manufacturing sector, but at the same time it has also increased the risk of cyber attacks due to the increased connectivity and complexity of industrial control systems (ICS). Intrusion Detection Systems (IDS) are crucial for protecting IIoT networks as it analyses and filters network packets in order to identify different kinds of cyber threats. In this paper machine learning algorithms have been applied along with Principal Component Analysis (PCA) and Autoencoders (AE) for data dimensionality reduction. The classifiers explored are Decision Trees, Random Forest, K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP). Also, clustering algorithms were applied for feature generation in order to enhance classification performance. We propose a solution that combines K-Nearest Neighbors (KNN) with Autoencoder 48 for dimensionality reduction to maximize the security of manufacturing plants. Our model outperforms others, as demonstrated through comparison, showing the best results in accuracy, precision, recall, and F1-score, highlighting its potential to enhance IDS in IIoT systems.**

***Index Terms*—CICIDS2017, Network Analysis, Packet Classification, Intrusion Detection, Industrial Control System, IoT**

I. Introduction

The Industrial Internet of Things (IIoT) has reshaped manufacturing systems by integrating sensors, actuators, and smart devices throughout the factory floor, allowing for monitoring in real time, maintenance prediction, and automation [1]. However, increasing connectivity raises substantial cybersecurity issues. IIoT systems, which combine industrial control systems (ICS) with connected devices, are attractive targets for cyber-attacks due to their complexity and weaknesses. DDoS attacks, ransomware and targeted attacks against ICS can disrupt production lines, jeopardise safety, and result in financial losses. To address these threats, Intrusion Detection Systems (IDS) are required to monitor network traffic, detect irregularities, and identify potential attacks in real time. These systems play an crucial role in protecting IIoT-enabled manufacturing plants against increasing cyber threats.

Considering the growing complexity of IIoT systems, conventional signature-based IDS approaches are now inadequate. These systems struggle to adapt to the constantly shifting dynamics of IoT environments, where threats such as DoS attacks and data alteration are common. As a result, there is an increasing demand for advanced IDS solutions that use machine learning, deep learning, and other AI techniques. These technologies allow us to detect novel threats sooner and increase the accuracy and efficiency of security measures in real time [2].

Maseer *et al.* [3] conducted a comparative study on machine learning algorithms for industrial intrusion detection using the CICIDS2017 dataset. Supervised methods, including ANN, DT, k-NN, NB, RF, SVM, and CNN, were evaluated, where the k-NN classifier outperformed others. In contrast, unsupervised techniques like k-means and EM showed limited effectiveness.

Chindove *et al.* [4] proposed an adaptive intrusion detection system leveraging preprocessing techniques such as Gini Importance, Permutation Importance, and PCA for feature selection. Recurrent Neural Networks (RNN) and Random Forest (RF) models achieved macro F1-scores of 0.73 and 0.87, respectively. The study highlighted challenges like class imbalance and the need to balance model complexity with training duration.

Yin *et al.* [5] proposed an MLP classifier combined with Birch clustering for anomaly detection on the CICIDS2017 dataset. The approach used Information Gain (IG) for feature selection and random undersampling to address class imbalance, achieving a high accuracy with 12 clusters. This method outperformed similar approaches in F1-score and accuracy, emphasizing the effectiveness of combining clustering and MLP for intrusion detection in industrial control systems.

Elmasri *et al.* [6] assessed machine learning algorithms, finding that the LOF (Local Outlier Factor) approach outperformed KNN with an average accuracy of 90.5%. PCA for feature reduction improved model efficiency and reduced training time. However, the study faced limitations such as training on only normal data samples and a potential for high false positive rates.

Rakesh *et al.* [7] evaluated various models, including LSTM-CNN stacks, Naïve Bayes, KNN-RF stacks, and XGBoost using ensemble methods. XGBoost showed flawless performance with 48 features. The study utilized stacking, bagging, and boosting for model integration, and feature selection through Recursive Feature Elimination (RFE). However, the reliance on a single feature selection method was a limitation, as it may exclude valuable features.

Building on the limitations found in the reviewed literature, such as class imbalance, insufficient feature selection, high false positive rates, and difficulties adapting to real-world IoT scenarios, this study presents a comprehensive solution that

addresses these complications. Using the CICIDS2017 dataset, we aim to improve machine learning-based IDS models for industrial intrusion classification. To address the scalability and performance issues, we explore dimensionality reduction approaches such as PCA and Autoencoders (32 and 48 features), followed by clustering to produce additional features for classification. We tested classifiers like Decision Trees, Random Forest, KNN, and a custom MLP model inspired by Yin *et al*. [8], with optimized parameters, to identify industrial IoT intrusions and determine the best-performing model for accurate detection.

A. Motivation

The motivation for this proposed approach is as follows.

1) The fast expansion of industrial IoT systems has increased their vulnerability towards advanced cyber threats, making robust intrusion detection systems crucial.
2) With industries heavily reliant on IoT-enabled automated machinery, even little disruptions caused by attacks like DDoS can result in huge economic losses and safety issues.
3) Current intrusion detection technologies tend to lack the precision and scalability required to address the increasing complexity and quantity of network traffic in industrial settings.
4) Improving the ability to identify specific types of cyberattacks in IIoT systems helps enable targeted countermeasures and stronger overall security.

B. Research Contributions

This paper has the following research contributions:

- We propose a model using the CICIDS2017 dataset for predicting industrial IoT intrusion types, addressing the need for real-time intrusion detection in IIoT systems.
- We apply feature transformation techniques, including PCA and AutoEncoders (AE32, AE48), to enhance classification performance.
- The best performance was achieved by combining AE48 with KNN, showing superior results in classification.

C. Organization

The paper is further organized as follows: Section II proposes the system model and the problem formulation, Section III explains the proposed framework, Section IV discusses the results, and Section V provides the conclusion and future works.

II. System Model and Problem Formulation

A. System Model

The accurate analysis of network traffic is essential for immediate response and remedy from the cybersecurity threats. For the purpose of accurate and real-time analysis of network traffic we need to constantly monitor each and every packet that is transmitted over the IoT network. The system model is depicted in Fig. 1.

Let the users Industrial IoT network be denoted by V, and the attacker be denoted by A. Under the represented scheme A sends some malicious packet P intended for a particular device on V, before P reaches the destination device it has to pass through the IoT server, which has an Intrusion detection system embedded. This intrusion detection system uses some software like CICFlowMeter to capture and analyze the network traffic passing through the server, this software monitors various features of packets such as packet size, flow duration, protocols, payload statistics, etc which are mentioned in the CICIDS2017 dataset (D).

$$SOURCE \xrightarrow[\mathbf{P^i}]{Captured\ by\ IDS} DESTINATION \quad (1)$$

where $\mathbf{P^i} \in \mathbb{R}^{78}$.

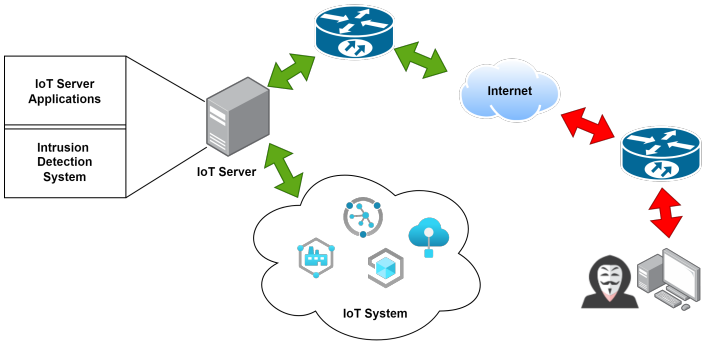

Fig. 1: A diagram of the system model.

Now this packet $\mathbf{P^i}$, is sent to a classification module to classify this packet into one of the category. After this classification module detects some or the other type of threat, a responder is notified about the abnormal packet and the responder now has to deal with the potentially malicious packet.

After the packet $\mathbf{P^i}$ is classified and it belongs to some of the attacks the concerned authorities can be notified, who can take the remedial steps to minimize the physical, cyber, and economic effects that the attacker might be trying to cause. To remedial steps might include timing out the source IP address, in case of any DoS/DDoS attack, banning the source IP in case of any Web Based Attack, and Shutting down unused ports while facing a Port Scan. Other measures can be taken in case of various other attacks.

B. Problem Formulation

Accurate classification of packets being analyzed from intrusion detection systems is critical for the mitigation and recovery of the systems; this can also help us prevent these types of attacks in the future. During the classification of the packets received from the intrusion detection system, the misclassified packets compose a heavy penalty on the system to take the unnecessary steps to identify the actual attack or they can ignore the real attack by classifying the malicious packets as normal or benign packets. These ignored packets can cause a heavy damage to the Industrial IoT systems under consideration. There were originally 15 classes of attacks, but we merged some of the classes under the same category.

Attack Type (Original)	Attack Type (Mapped)
BENIGN	BENIGN
DDoS	DoS/DDoS
DoS Hulk	DoS/DDoS
DoS GoldenEye	DoS/DDoS
DoS slowloris	DoS/DDoS
DoS Slowhttptest	DoS/DDoS
PortScan	Port Scan
FTP-Patator	Brute Force
SSH-Patator	Brute Force
Bot	Bot
Web Attack – Brute Force	Web Attack
Web Attack – XSS	Web Attack
Web Attack – SQL Injection	Web Attack
Infiltration	Infiltration
Heartbleed	DoS/DDoS

TABLE I: Mapping of Original Attack Types to Mapped Categories

We need to classify a given packet $\mathbf{P^i}$ into one of the 7 families 6 of which belong to some attack class, and one of them are benign packets. This problem of classification can be defined by the following equation. Suppose a system S is used to classify the packet, then the classification will result in the following values, $\mathbf{y_0^i}$, $\mathbf{y_1^i}$, $\mathbf{y_2^i} \ldots \mathbf{y_8^i}$ representing the probabilities of the packet belonging to respective class.

$$C_i = \arg\max_j \ (y_0^i, \ y_1^i \ldots \ y_6^i) \tag{2}$$

Considering the penalties we have to apply during false alarms and the case of missing out on any malicious packets we need to penalize our model to improve the model and reduce the cost of overheads and damages caused by missed malicious packets, we provide the following objective function. This objective function also takes into account the class imbalance in the dataset.

$$\mathcal{L}(y^i, \ \hat{y}^i) = \frac{\lambda_1}{N} * \sum_{i=1}^{N}(-\sum_{c=1}^{C} w_c.y_c^i * \log(\hat{y}_c^i)) + \lambda_2.\frac{FP+FN}{N} + \ \lambda_3.\frac{||\Theta||^2}{2} \tag{3}$$

In the above-mentioned equation the first term is for the normal categorical cross entropy loss, with the class weights accommodated in the equation, the second term is the penalty for false alarms, which we need to reduce to improve the robustness and the latency of the system. The last term in the equation refers to the regularization to help fight against over fitting of the dataset. We need to minimize $\mathcal{L}$ to get the best model for classification of the network packets in IIoT.

III. The Proposed Framework

The proposed framework, as shown in Fig. 2, consists of three layers: the data collection layer, the artificial intelligence layer and the application layer.

A. Data collection layer

To evaluate intrusion detection systems for industrial IoT security in manufacturing plants, we propose a realistic data collection framework using a custom-designed IoT testbed. This testbed includes an Operational Network to simulate normal industrial activities and an Attack Network to generate malicious traffic.

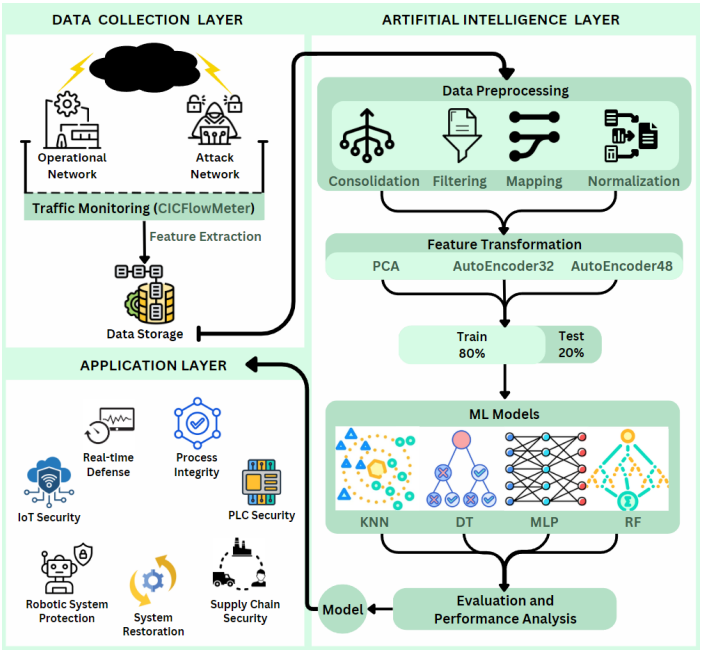

Fig. 2: Proposed framework

The Operational Network mirrors a manufacturing plant with IoT devices, sensors, actuators, and SCADA systems performing regular tasks, such as monitoring and controlling industrial processes. In parallel, the Attack Network introduces controlled cyberattacks like denial-of-service (DoS) and unauthorized access attempts to simulate real security threats.

CICFlowMeter is deployed within the Operational Network to capture and analyze the network traffic. This tool monitors and extracts network flow features, including packet size, flow duration, communication protocols, and payload statistics. Once captured, these network traffic features are sent to the proposed system model for classification.

B. Intelligence Layer

1) Dataset description: The CICIDS2017 dataset [9] is designed for network security and intrusion detection, covering six major attack profiles executed using relevant tools:

- Brute Force: Cracks passwords and discovers hidden web content.
- Heartbleed: Exploits a bug in the OpenSSL library within the TLS protocol.
- Botnet: Controls multiple devices for malicious tasks.
- DoS/DDoS: Disrupts services by overwhelming resources.
- Web Attacks: Includes Brute Force, XSS, and SQL Injection targeting web vulnerabilities.
- Infiltration: Exploits software vulnerabilities to gain internal network access.

The CICIDS2017 dataset organizes attack traffic across specific days, as summarized in Table II:

TABLE II: Daily Labels in the CICIDS2017 Dataset

Day	Traffic Labels
Monday	Benign
Tuesday	Brute Force (SFTP, SSH)
Wednesday	DoS (Slowloris, Slowhttptest, Hulk, GoldenEye), Heartbleed
Thursday	Web Attacks (Brute Force, XSS, SQL Injection), Infiltration (Dropbox download, Cool Disk)
Friday	DDoS (LOIT), Botnet (ARES), Port Scans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL, B)

2) Data Preprocessing: The preprocessing of the CICIDS2017 dataset begins by consolidating data from five separate files, each corresponding to a specific day and attack scenario, into a single dataframe. Missing and infinite values are removed, and features with uniform values (entropy = 0) are dropped, as they provide no useful information. To standardize numeric features, we apply the StandardScaler from sklearn. Each feature P_j^i is scaled using the mean $\overline{P_j}$ and standard deviation σ_{P_j} of the j^{th} feature in the training dataset, as given by:

$$P_j^{i'} = \frac{(P_j^i - \overline{P_j})}{\sigma_{P_j}} \tag{4}$$

This ensures all features are on a comparable scale, critical for effective classification. The dataset's original 15 classes are mapped into 6 broader categories—*Benign, Bot, Brute Force, DoS/DDoS, Infiltration, Port Scan* and *Web Attack* as observed in Table I—to address class imbalance and simplify the classification task. The final cleaned and standardized dataframe, containing 78 features, is then saved for subsequent modeling.

3) Selected Model: In this study, we applied three different feature transformation methods to the CICIDS2017 dataset to improve classification performance: PCA, AutoEncoder32 (AE32), and AutoEncoder48 (AE48). Initially, we performed PCA, reducing the feature set from 78 (after dropping irrelevant features) to 17 features per packet while retaining 99.94% of the original information. The dataset was then split into training and testing sets (80% and 20%, respectively). The transformed features were passed into various machine learning models, including KNN, Decision Trees (DT), Random Forest (RF), and Multilayer Perceptron (MLP). We trained and evaluated these models using metrics such as accuracy, precision, recall, F1-score, AUC-ROC, and the confusion matrix.

While PCA effectively reduced the feature set and retained significant information, we explored AutoEncoders to assess whether a higher latent dimension could improve performance. We began with AutoEncoder32 (AE32), setting the latent dimensionality to 32, then used the transformed features in the same models. Finally, we applied AutoEncoder48 (AE48) to investigate if a higher latent dimension yielded better results.

Ultimately, the best classification results were achieved by applying AE48 followed by KNN, demonstrating the effectiveness of this combination for the given task. To better understand its effectiveness, we first describe the AE48 configuration and training setup, followed by the implementation details of the KNN classifier.

While training, AE48 uses the mean squared error (MSE) loss function, with a learning rate of 0.001, 30 epochs, and a batch size of 256. The mean squared error loss function is given by:

$$\mathcal{L}(X^{ip},\ X^i) = \frac{1}{N} * \sum_{j=0}^{N} (X_j^{ip} - X_j^i)^2 \tag{5}$$

where $\mathbf{X^{ip}}$ represents the input data, $\mathbf{X^i}$ is the reconstructed output from the AutoEncoder, and N is the number of features or data points.

Once trained, AE48's reduced features were prepared for classification using machine learning models. The feature transformation process is as follows:

Suppose we capture a packet $\mathbf{P}^i$, which has the following parameters represented by $\mathbf{P}_0^i, \mathbf{P}_1^i, \ldots, \mathbf{P}_{77}^i$ as mentioned in the CICIDS2017 dataset proposed by Sharafaldin *et al.* [9]. These captured parameters are then preprocessed and passed to an autoencoder of latent dimension 48, represented by $\mathbf{AE}_{\theta_{48}}$ where θ_{48} are the parameters used by the autoencoders, which converts these parameters from $\mathbf{P}_j^i$ to $\mathbf{L}_j^i$. This conversion is represented by the following equation:

$$\mathbf{P}^i \xrightarrow{\mathbf{AE}_{\theta_{48}}} \mathbf{L}^i \tag{6}$$

where, $\mathbf{P}^i \in \mathbb{R}^{78}$, $\mathbf{L}^i \in \mathbb{R}^{48}$.

Now these reduced features corresponding to each packet are sent to the classifier algorithm, denoted by f_{KNN}, which is the best model to classify these packets according to our experimentation. The KNN algorithm works by determining the k-nearest neighbors of the input sample based on a distance metric. In our implementation, we used **Euclidean distance** to measure similarity between data points. For two feature vectors x and x', the Euclidean distance is given by:

$$d(x, x') = \sqrt{\sum_{j=1}^{n} (x_j - x'_j)^2} \tag{7}$$

Here, n represents the number of features, and x_j and x'_j are the j-th feature values of the respective feature vectors.

Using this distance metric, the k-nearest neighbors are identified, and the class of the input packet is determined based on a majority vote among the neighbors. Our model was configured with 5 neighbors (number of nearest neighbors considered for classification) and 12 parallel jobs (number of processors used during the fitting process) using the `KNeighborsClassifier` from scikit-learn.

This KNN model then classifies the input packet P_i into one of the 7 classes: *Benign, Bot, Brute Force, DoS/DDoS, Infiltration, Port Scan, Web Attack*, represented by $\mathbf{y_0}, \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}, \mathbf{y_4}, \mathbf{y_5}, \mathbf{y_6}$, which are the probabilities that the packet corresponds to the respective class. The final class to which the packet belongs is represented by Y, given by the following formula:

$$Y^i = \arg\max\left(y_0^i, y_1^i, y_2^i, y_3^i, y_4^i, y_5^i, y_6^i\right) \quad (8)$$

In addition to direct classification methods, we experimented with a clustering-based approach to evaluate its impact on performance. After feature reduction, the data was clustered into 7 groups using Bisecting K-means from scikit-learn, and the cluster assignments were added as an additional feature for classification. However, this approach did not yield any significant improvement over the direct AE48 + KNN combination. The clustering results were unsatisfactory, further reinforcing the robustness and effectiveness of our primary method.

IV. Results and Discussion

A. Experimental Setup

The proposed framework is implemented on Kaggle using Python version 3.10.14. Pandas 2.2.3 handles the dataset, and Numpy 1.26.4 performs logical mathematical calculations. Libraries like Matplotlib are used for visualization. The Nvidia Tesla P100 GPU (16GB VRAM) is used for computational purposes. It has 3,584 CUDA cores and 29 Gigabytes of GDDR6 RAM, which helps for faster and more efficient processing.

B. AI-based Evaluation and Results

We evaluated the models using various metrics, such as accuracy, precision, recall, F1-Score, and area under the receiver operating characteristic curve (AUC-ROC score). The models used for the classification are Decision Trees, Random Forests, KNNs, and Multi-Layered Perceptrons. We used various types of algorithms such as Principal Component Analysis and Auto Encoders of various sizes for the reduction of dimension of the input data. The original data consisted of 78 features, using PCA we reduced the 78 features to 17 features retaining about 99.94% of the information. We also used autoencoders with latent feature dimensions of 32 and 48.

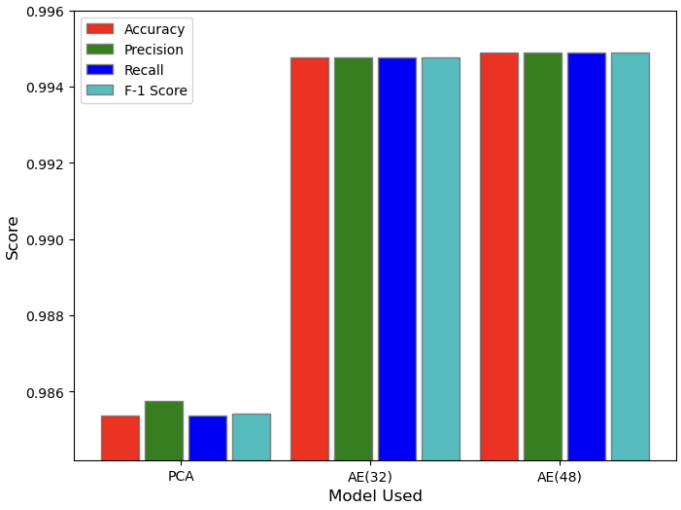

Fig. 3: A graph comparing feature reduction algorithms that we experimented with.

Fig. 3 shows the comparison of various reduction methods among AE32, AE48 and PCA. Among all the methods AE48 was the most promising one with KNN, so we now compared various ML and DL models followed by feature reduction with AE48.

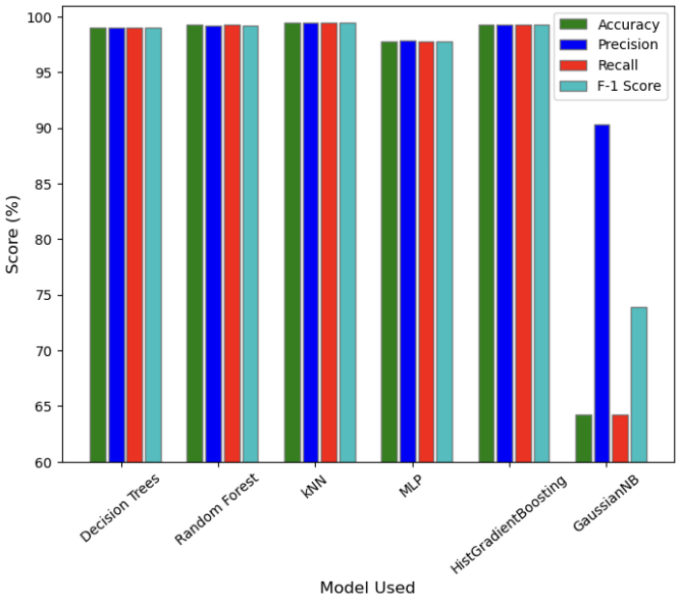

Fig. 4: A graph comparing various ML models' performance after feature reduction with AE48.

Fig. 4 shows the comparison of various metrics used for the evaluation of the models that we selected for our study. This graph clearly shows that the KNN model used after the dimensionality reduction by autoencoder of latent dimension 48 has the best accuracy, precision, recall, and F-1 Score. This specific methodology attained an accuracy of 99.489%, a precision of 99.490%, recall of 99.489%. The second best methodology for this classification was KNN followed by autoencoder of latent size 32, as the latent size reduces some of the information if lost while encoding the features. The other methods include using various ML algorithms such as Decision Trees and Random forest after dimensionality reduction by Principal Component Analysis (final feature dimensions of 17). For PCA, the IncrementalPCA was used, which retained about 99.94% of the information. The MLP classifier followed by an autoencoder of size 48 showed an accuracy of 97.814% which was very low as compared to other ML models such as KNN, Decision Trees, and Random Forest. The architecture of the MLP was proposed by Yin *et al.* [8], changing the number of neurons in the last second layer from 256 to 128.

Fig.5 shows the AUC-ROC characteristics of various classes of attacks. We can see that the AUC-ROC score of most of the classes of attacks is close to one. Hence we can conclude that the model can differentiate between various types of attacks with a high sense of confidence, the model also generalizes well because the metrics mentioned above are reported on the test data unseen by the model during the training phase.

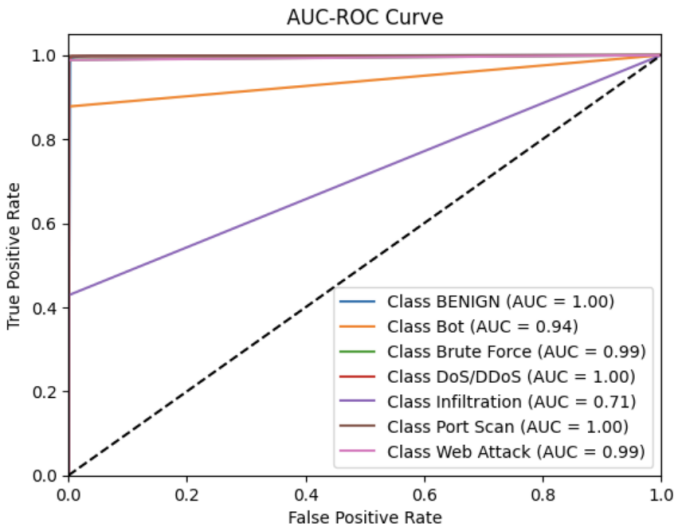

Fig. 5: The ROC curve for various types of attacks, generated by the AE(48) + KNN.

Fig.5 shows the ROC curve with an average area under the curve (AUC) of 0.9471. From this ROC curve, we can deduce that this model can keep the true positive rate while keeping the false positive rates low, which is crucial for various industrial applications so that there aren't any false alarms and none of the malicious frames are ignored. The steep accent and large AUC suggest that the model performs well. The AUC-ROC score can be calculated from the following formula for each class.

$$\mathbf{AUC - ROC} = \int tpr \;\; d(fpr) \tag{9}$$

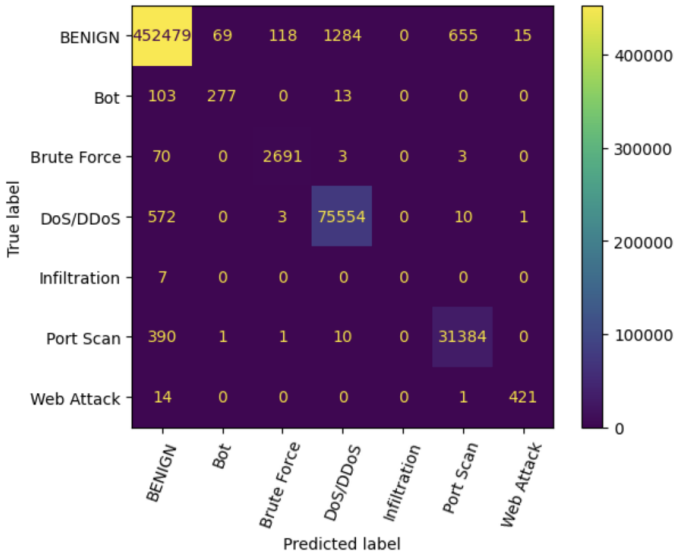

Fig. 6: The confusion matrix on test dataset generated by the AE(48) + KNN.

Fig.6 shows the confusion matrix generated by AE(48) + KNN method used on the test dataset. Most of the samples lie on diagonal elements of the confusion matrix showing that most of them are classified correctly. The model is not a perfect fit for all the cases, as seen from the confusion matrix. Some of the instances in class Bot are misclassified as Bengin. Also, some mismatches along the matrix show that the proposed algorithm is not perfect and there is a scope for improvement. The abnormality in "Infiltration" class can be seen because of very low representation of class about 36 datapoints as compared to over 28,00,000 datapoints in total.

Comparing various classification feature reduction and clustering algorithms, we found the best model to be the one using latent size 48 autoencoders, and then a simple KNN classification algorithm is the most effective one, achieving an accuracy of 99.489%.

V. Conclusion

This paper addresses the growing need for robust Intrusion Detection Systems (IDS) in IIoT-enabled manufacturing plants, where increased connectivity exposes critical systems to diverse cyber threats. By employing the CICIDS2017 dataset, we explored the impact of dimensionality reduction techniques, including PCA and Autoencoders, and evaluated various machine learning models for intrusion detection. The combination of Autoencoder with a latent dimension of 48 and KNN emerged as the most effective, achieving superior metrics across accuracy, precision, recall, and F1-score. Our findings underscore the potential of combining feature transformation and machine learning to enhance the performance and reliability of IDS in industrial environments. While the results demonstrate a high degree of accuracy and generalizability, limitations such as misclassifications in certain attack classes and the challenges of real-time scalability remain. These findings pave the way for future research to focus on refining detection methods, enhancing system performance, and addressing deployment challenges in complex IIoT ecosystems.

References

[1] K. Barton, F. Maturana, and D. Tilbury, "Closing the loop in iot-enabled manufacturing systems: Challenges and opportunities," in *2018 Annual American Control Conference (ACC)*, pp. 5503–5509, IEEE, 2018.

[2] S. H. Mekala, Z. Baig, A. Anwar, and S. Zeadally, "Cybersecurity for industrial iot (iiot): Threats, countermeasures, challenges and future directions," *Computer Communications*, 2023.

[3] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE access*, vol. 9, pp. 22351–22370, 2021.

[4] H. Chindove and D. Brown, "Adaptive machine learning based network intrusion detection," in *Proceedings of the International Conference on Artificial Intelligence and its Applications*, pp. 1–6, 2021.

[5] Y. Yin, J. Jang-Jaccard, F. Sabrina, and J. Kwak, "Improving multilayer-perceptron (mlp)-based network anomaly detection with birch clustering on cicids-2017 dataset," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 423–431, IEEE, 2023.

[6] T. Elmasri, N. Samir, M. Mashaly, and Y. Atef, "Evaluation of cicids2017 with qualitative comparison of machine learning algorithm," in *2020 IEEE Cloud Summit*, pp. 46–51, IEEE, 2020.

[7] L. Rakesh, L. Upadhyay, and P. M. Reddy, "Evaluation of network intrusion detection with machine learning and deep learning using ensemble methods on cicids-2017 dataset," in *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp. 1429–1433, IEEE, 2023.

[8] Y. Yin, J. Jang-Jaccard, F. Sabrina, and J. Kwak, "Improving multilayer-perceptron(mlp)-based network anomaly detection with birch clustering on cicids-2017 dataset," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 423–431, 2023.

[9] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.