Development of A Classification Model For Intrusion Attacks in Internet-Of-Things (IoT) Networks

Adegbite R.M.[1,*]
Kayode A.A [1]
Olaniyan O.O. [1,2]
Arekete S.A.[1,3]

Received: 16/10/2024
Revised: 05/11/2024
Accepted: 7/01/2025

[1] Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria
2 Email Address: olaniyano@run.edu.ng
3 Email Address: areketes@run.edu.ng
* Corresponding Author's Email: adegbite9827@run.edu.ng

https://doi.org/10.20428/jst.v30i2.2517

Development of a Classification Model for Intrusion Attacks in Internet-Of-Things (IoT) Networks

Adegbite R.M
Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria
adegbite9827@run.edu.ng

Kayode A.A
Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria,
kayodeade@run.edu.ng

Olaniyan O.O
Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria
olaniyano@run.edu.ng

Arekete S.A.
Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria
areketes@run.edu.ng

Abstract— Devices connected to the Internet can easily exchange information thanks to the Internet of Things (IoT), a networked system that functions via established protocols. IoT's decentralized architecture presents serious security, privacy, data integrity, and system stability issues despite its revolutionary potential. Although technological innovations like artificial intelligence, the Internet of Things, and big data have significantly raised people's quality of life, they have also increased the likelihood of increasingly complex and serious cyberattacks. By creating a machine learning model for the identification and categorization of intrusion threats in Internet of Things networks, this study seeks to address these issues. In particular, a hybrid strategy that combined Support Vector Machine (SVM) and fuzzy logic techniques was used to improve intrusion detection systems' efficacy in Internet of Things settings. We trained and tested the model using the NSL-KDD dataset from Kaggle. Key performance indicators, such as true positive rate, false positive rate, accuracy, F1 score, precision, and recall, were used to assess the suggested model's performance. With an accuracy of 99% on an imbalanced dataset and 98% on a balanced dataset, the results showed that the hybrid Fuzzy Logic-SVM model performed well across all data splits. These results demonstrate how the model may be used to increase the security and dependability of IoT networks.

Keywords— Internet of Things, Intrusion Detection System, Security, Intrusion, Cyberattacks.

I. INTRODUCTION

The Internet of Things (IoT) is a rapidly expanding system that uses recognized protocols to enable interconnected devices to exchange information [1]. Intelligent IoT gadgets enhance connectivity, revolutionizing markets and improving quality of life through environmental monitoring, smart agriculture, and smart homes. IoT applications span sectors such as smart agriculture, weather monitoring, energy distribution, healthcare, and home automation [2]. Due to its decentralized structure, security, cyber threats, privacy, data integrity, and stability are some of the problems of the Internet of Things (IoT) [1, 3].

The common network traffic intrusion attacks are Denial of Service (DoS), Probe, User-to-Root (U2R), Remote-to-Local (R2L), Brute Force, Phishing web, Distributed Denial of Service (DDoS), Routing Protocol for Low-Power and Lossy Network (RPL), Heartbleed, and Infiltration [4; 3]. Monitoring and analyzing network data to look for traces of intrusion and react when a malicious attack (intrusion) takes place is the process of intrusion detection [2, 3]. Many studies have used both machine learning and deep learning in this field to detect aberrant behaviors coming from both inside and outside the network system. [5]. However, traditional IDSs have limitations in detecting unknown attacks and handling noisy data. Therefore, there arises a necessity for enhanced techniques to accurately distinguish both known and unknown attacks in IoT networks, ensuring high precision and minimal false-positive outcomes [6]. Various machine learning and data mining techniques have been applied to this problem. Some of these include shallow neural networks (SNN), decision trees (DT), random forests (RF), neural networks (NN), k-nearest neighbors (KNN), and support vector machines (SVM).

The combination of two different machine learning techniques, such as fuzzy rough set feature selection and modified KNN classifier, random forest and principal component analysis (PCA), SVM and naïve Bayes, long short-term memory (LSTM) and convolutional neural networks (CNN), K-means clustering and hierarchical clustering, and so on, is indeed one of the many approaches that researchers have explored to enhance intrusion detection accuracy in IoT networks. It is important to note that the choice of which method or combination of methods to use depends on the specific characteristics of the data, the nature of the intrusion detection problem, and the details of its implementation [7]. This study utilized a hybrid approach of fuzzy logic and SVM methods to enhance intrusion detection in IoT networks. By combining fuzzy logic and support vector machines (SVMs) in a hybrid approach for intrusion detection and classification in IoT networks, the system benefits from the strengths of both techniques, leading to an enhanced overall performance of the intrusion detection system.

The structure of this paper is organized as follows: Section 2 covers related work. Our methodology for hybrid model development is explained in Section 3. The result and discussion are described in Section 4. Finally, Section 5 presents the conclusion.

II. RELATED WORKS

A network intrusion detection system using fuzzy logic [8]. The Internet of Things (IoT), emphasizing its capacity to connect objects to the Internet for information exchange and stressing the importance of addressing security issues in IoT networks [9]. The emergence of the Internet of Things (IoT) and its potential benefits across various domains [10]. The

paper emphasized the cybersecurity threats associated with IoT and stressed the need for tailored security solutions. A lightweight intrusion detection system for the Internet of Things [11]. The research attempted to develop a lightweight attack detection strategy utilizing a supervised machine learning-based support vector machine (SVM) to detect an adversary attempting to inject unnecessary data into the IoT network. A heuristic intrusion detection system for the Internet of Things (IoT) [12]. An intrusion detection mechanism using fuzzy rule interpolation [13]. [14] worked on SVM-Based Network Intrusion Detection for the UNSW-NB15 Dataset.

[15] designed an efficient IoT-Botnet attack detection with a sequential feature selection approach to implement a lightweight detection system with high performance for IoT-Botnet attack detection using three different ML algorithms: Naïve Bayes, Decision Tree, and Artificial Neural Network. [16] designed an anomaly-based intrusion detection system using a support vector machine. [17] presented a multi-class neural network model for rapid detection of Internet of Things (IoT) botnet attacks. [18] devised an intrusion detection system employing fuzzy rough set feature selection and a modified KNN classifier to refine the classification of network intrusion attacks through optimal feature selection. An anomaly-based intrusion detection system using fuzzy logic [19]. An effective intrusion detection approach using SVM and Naïve Bayes feature embedding [20].

III. METHODOLOGY

The methodology included designing the system architecture and flowchart, collecting and preprocessing data, and applying classification algorithms.

A. System Architecture and Flowchart

The process began by selecting a training dataset containing network traffic data from 23 IoT devices, which featured various types of network intrusion attacks. The dataset was cleansed to remove noisy or irrelevant information, resulting in a high-quality, refined dataset. Key features, such as *back*, *Neptune*, *ipsweep*, *nmap*, *guess_password*, *ftp_write*, *buffer_overflow*, and *loadmodule*, were selected and fuzzified. These features were instrumental in identifying different types of intrusion attacks. The dataset, sourced from Kaggle, was already divided into two subsets: a training set (80% of the data) and a test set (20%). The proposed hybrid model, which combines fuzzy logic and support vector machine (SVM), was trained using the training subset. Its performance was subsequently tested on the test subset. The system architecture, illustrated in Figure 1, demonstrates how the hybrid model integrates the advantages of fuzzy logic and SVM to detect and classify intrusion attacks. The methodology eliminates unnecessary repetition by organizing processes distinctly within sections. For example, the steps of loading the dataset and preprocessing are consolidated, ensuring clarity and avoiding redundancy. This structure highlights how the output of the fuzzy logic system was used to train the SVM. Through training and testing, the model effectively identified and categorized intrusion attacks into the groups *Normal*, *Probe*, *U2R*, *R2L*, and *DoS*.

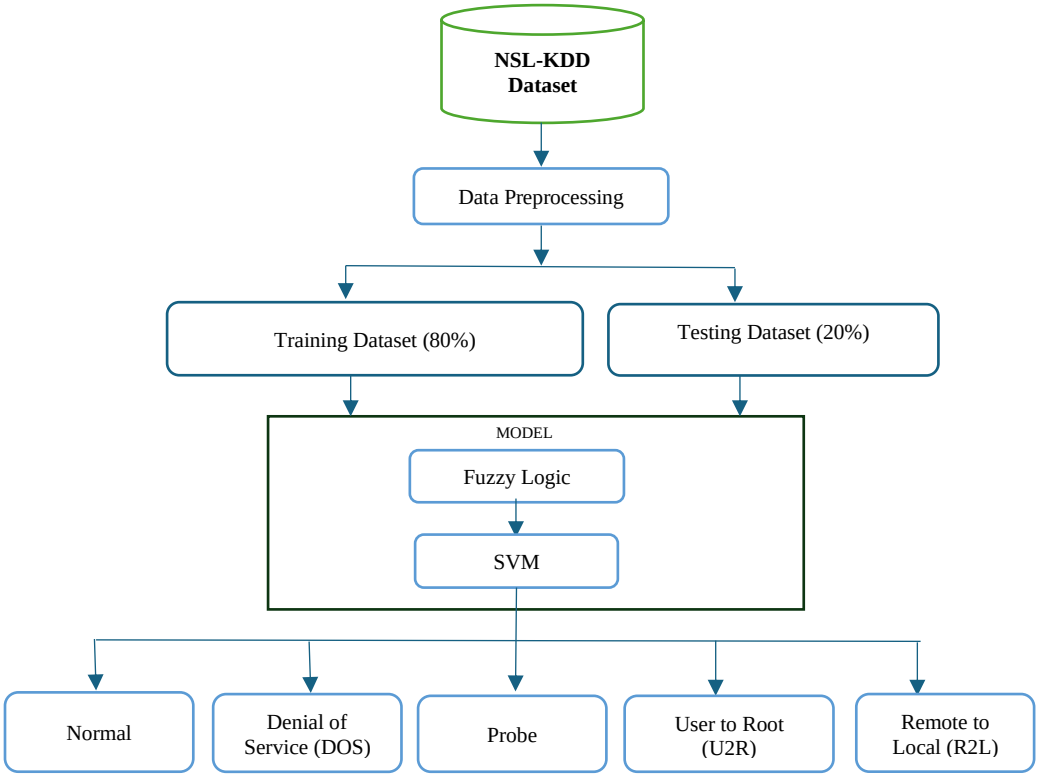

Fig. 1. Proposed Model Architecture

The flowchart for the detection system is shown in Figure 2. The dataset was collected and pre-processed. Subsequently, data was transformed into linguistic variables and fuzzy sets using Fuzzification from the Fuzzy Logic block. Then the model categorized the detected intrusions into five distinct classes: Denial of Service (DoS), Probe, User to Root (U2R), Remote to Local (R2L), or Normal attack.

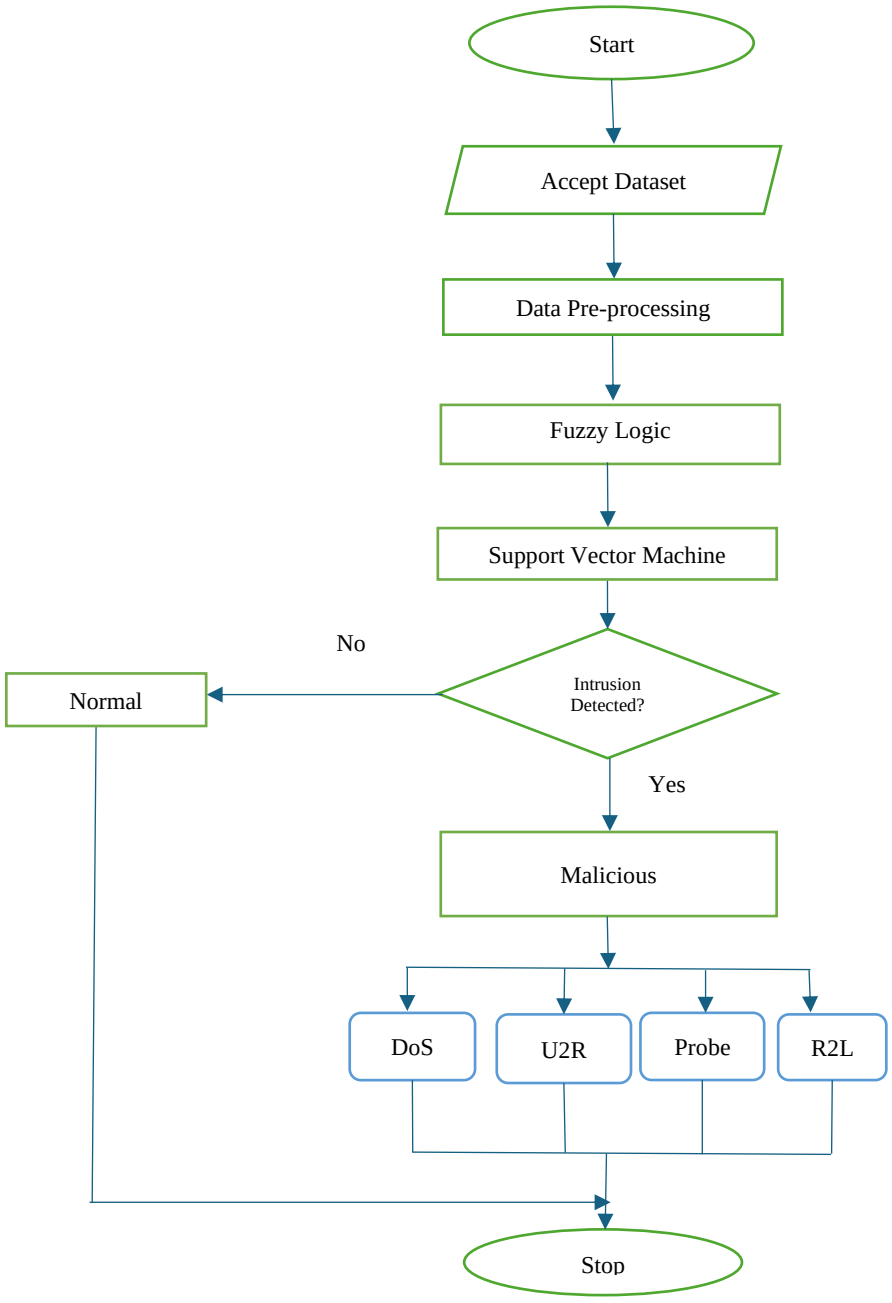

Fig. 2. Proposed Model Flowchart

B. Data Preprocessing

IoT networks often involve diverse and dynamic data sources, including sensor readings, network traffic, and device metadata. These data sources may contain inherent uncertainties due to factors like sensor inaccuracies, environmental variations, or communication errors. By applying fuzzification techniques, IoT data can be transformed into fuzzy variables, allowing for the representation of uncertainty and imprecision. This enables intrusion detection systems to better capture the nuanced

patterns and behaviors associated with both normal intrusion and potential intrusions in IoT networks.

1) Fuzzification: In intrusion detection, diverse and dynamic network traffic patterns necessitate the use of fuzzification to capture features like duration within a fuzzy logic system. The duration feature was fuzzified using triangular membership functions as shown in Figure 3, with the minimum and maximum values defining the boundaries of the fuzzy set and the mean value serving as the central point. This process classified network activities into three categories: Low, Medium, and High. The use of triangular membership functions enabled the derivation of precise membership values for each duration instance.The choice of duration as a feature for fuzzification was based on its critical role in distinguishing between normal and anomalous network activities. A detailed analysis of its distribution supported the selection of membership function boundaries, ensuring that the fuzzification process captured meaningful variations in network behavior. This structured approach eliminates redundancy and provides a clear justification for the selected duration values, emphasizing their importance in the fuzzy logic system and the overall hybrid model.

```
Min value: 2
Peak value: 49.45
Max value: 97
```

Fig. 3. Fuzzy Membership Values for Duration

Pseudocode for Fuzzification of the 'duration' Feature

Define the feature to be fuzzified:
- feature_name = 'duration'

Extract the feature values from the dataset:
- feature_values = multi_data[feature_name] values

Determine the parameters for the triangular membership function:
- min_value = minimum value of feature_values
- max_value = maximum value of feature_values
- peak_value = mean value of feature_values

Compute fuzzy membership values using a triangular membership function:
- fuzzy_membership = fuzz.trimf(feature_values, [min_value, peak_value, max_value])

Add the computed fuzzy membership values as a new column in the dataset:
- multi_data[feature_name + '_fuzzy'] = fuzzy_membership

2) Generated Rules from Fuzzy Membership Values: Based on the computed fuzzy membership values, a series of classification rules were generated.

Table 1. Generated Fuzzy Rules

Duration	
1 – 20	Low
21 – 40	Medium
41 – 97	High

3) Hierarchical Fuzzy Rules: A more structured approach was taken with hierarchical fuzzy rules, which provided a clearer classification strategy based on specific thresholds. If duration ≤ 20: Low; If duration ≥ 21 ≤ 39: Medium; If duration ≥ 40 ≤ 97: High.

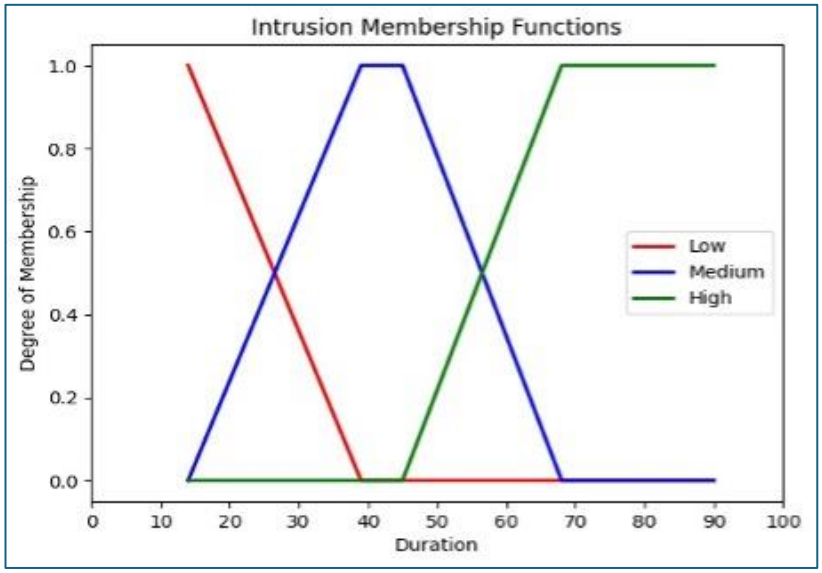

Fig. 4. Triangular Membership Function Graph

C. The Classification Models

After identifying the fuzzification of the dataset, we conducted experiments using a Hybrid IDS to assess their effectiveness in distinguishing between malicious and normal activities. Our Hybrid IDS model consisted of two machine learning algorithms: Fuzzy Logic and SVM.

1) Fuzzy logic: In an intrusion detection system, network traffic data is represented using fuzzy logic, a mathematical framework that allows variables to have

varying degrees of membership in a particular set and can handle uncertainty and imprecision in data. This allows intrusion attack patterns to be identified based on a variety of criteria. The fuzzifier, which transforms precise numbers into words or fuzzy values, serves as a translator in fuzzy logic. The pseudocode for this algorithm is shown below:

Pseudocode for Implementing Fuzzy Logic

a) **Load the Dataset**:
- Output: Network traffic data.
- Action: Load the dataset for analysis.

b) **Preprocess the Data**:
- Handle missing values.
- Encode categorical data into numerical format.
- Normalize or standardize features for consistent scaling.

c) **Scale the Features**:
- Apply feature scaling techniques to ensure numerical features are on a comparable scale.

d) **Split the Dataset**:
- Divide the dataset into training and testing subsets.

e) **Initialize Fuzzy Logic System**:
- Define the fuzzy logic system to be used for classification.

f) **Define Fuzzy Sets and Membership Functions**:
- Create fuzzy sets (e.g., *Low*, *Medium*, *High*) for each feature.
- Assign membership functions to each set, such as triangular or trapezoidal functions.

g) **Define Fuzzy Rules**:
- Establish fuzzy rules to describe the relationships between features and the target output (e.g., intrusion detection outcomes).

h) **Train the Fuzzy Logic System**:
- Output: Training dataset.
- Action: Train the system using fuzzy rules and membership functions.

i) **Make Predictions**:
- Output: Test dataset.
- Action: Predict target labels using the trained fuzzy logic system.

j) **Output**:
- Predicted labels for the test dataset.

2) Support Vector Machine: SVMs are particularly useful for problems with a high-dimensional feature space and can be used to find the optimal hyperplane that separates the data into different classes. To effectively recognize malicious attacks and classify them into various attack classes, the output of the fuzzy logic system is used to train the SVM, enabling it to identify abnormal activities. The pseudocode for this algorithm is shown below:

Pseudocode for Implementing Support Vector Machine (SVM)

a) **Load the Dataset**:
- Input: Network traffic data with labeled instances (e.g., normal and attack classes).
- Action: Load the dataset for processing.

b) **Preprocess the Data**:
- Handle missing values.
- Encode categorical features into numerical form.
- Normalize or standardize the features for consistency.

c) **Split the Data**:
- Divide the preprocessed dataset into training and testing subsets.
- Ensure proper class distribution in both sets.

d) **Initialize the SVM Classifier**:
a. Select an appropriate SVM kernel (e.g., linear, polynomial, RBF) based on the problem requirements.

e) **Train the SVM Classifier**:
- Input: Training dataset and corresponding labels.
- Action: Train the classifier using the output of the fuzzy logic system as feature inputs.

f) **Make Predictions**:
- Input: Test dataset.
- Action: Use the trained SVM classifier to predict the class labels of test instances.

g) **Evaluate Classifier Performance**:
- Compute evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.

h) **Visualize Results**:
- Plot relevant graphs or charts to present classification results and performance metrics (e.g., confusion matrix, ROC curve).

i) **Output**:
- Predicted labels for test data and evaluation metrics.

IV. Result & Discussion

The NSL–KDD dataset was fully downloaded from the Kaggle website (https://www.kaggle.com/datasets/hassan06/nslkdd) for this study. Detecting and classifying different types of intrusion attacks in IoT networks was the main aim of this study. The NSL-KDD dataset is a revised dataset extracted from the KDD '99 dataset to solve some inherent problems (Tavallaee, et al. 2009).

A. Dataset Distribution

The dataset used in this study, sourced from Kaggle, contains 41 attributes/features for each record, including a target

attribute that classifies each record as either an attack type or a normal instance, as shown in Table 2. The NSL-KDD dataset, a subset of the original KDD Cup 1999 dataset, includes 21 attack types in the training set and 17 additional attack types in the testing set. These additional attacks represent new, unseen attacks not included in the training data. All attack types are grouped into four major categories: *DoS*, *Probe*, *R2L*, and *U2R*, as detailed in Tables 3 and 4 for the training and testing sets, respectively. Table 5 summarizes the quantity of attacks in each category. An analysis of the dataset revealed an imbalance in attack categories, particularly for *R2L* and *U2R*, where the test data size exceeded the training data size. This imbalance highlights the challenges in intrusion detection, where certain attack types are underrepresented in the training data but are critical to detect due to their impact.

To address these challenges and improve the training process for underrepresented categories, three experiments were conducted:

- **First Experiment**: The predefined training and testing subsets were used, resulting in a high accuracy of 99%. However, the data imbalance likely contributed to overfitting.
- **Second Experiment**: The training and testing subsets were merged into a single dataset (*kddtri.txt*) to address the imbalance. The combined dataset was split 80/20, achieving an accuracy of 98%, with better generalization and reduced overfitting.
- **Third Experiment**: The combined dataset was split 70/30, maintaining an accuracy of 98%, further demonstrating the model's robustness.

Overall, these experiments highlight the hybrid model's adaptability and effectiveness across varying data distributions. The consistent accuracy across different splits underscores its suitability for practical intrusion detection systems, addressing data imbalance while maintaining high performance.

Table 2. List of NSL-KDD Dataset Files and Their Description

S/N	Name of the file	Description
1.	KDDTrain+.ARFF	The full NSL-KDD train set with binary labels in ARFF format
2.	KDDTrain+.TXT	The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3.	KDDTrain+_20Percent.ARFF	A 20% subset of the KDDTrain+.arff file
4.	KDDTrain+_20Percent.TXT	A 20% subset of the KDDTrain+.txt file
5.	KDDTest+.ARFF	The full NSL-KDD test set with binary labels in ARFF format
6.	KDDTest+.TXT	The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7.	KDDTest-21.ARFF	A subset of the KDDTest+.arff file which does not include records with a difficulty level of 21 out of 21
8.	KDDTest-21.TXT	A subset of the KDDTest+.txt file which does not include records with a difficulty level of 21 out of 21

Table 3. Attack Types in NSL-KDD Train Dataset

Attack	Attack Type
DoS	Back, land, Neptune, pod, smurf, teardrop
Probe	Satan, ipsweep, nmap,portsweep
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, spy
U2R	buffer_overflow, loadmodule, perl, rootkit

Table 4. Attack Types in NSL-KDD Test Dataset

Attack	Attack Type
DoS	Back, land, Neptune, pod, smurf, teardrop, apache2, mail bomb, processtable, udpstorm

Probe	Satan, ipsweep, nmap,portsweep, mscan, saint
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, spy, httptunnel,
U2R	named, sendmail, snmpgetattack, xlock, xsnoop

Table 5. The Quantity of Each Attack Categories in NSL-KDD Training and Test Dataset

	Training data set	**Testing data set**
Class	The quantity of attack	
DoS	45927	7458
Probe	11656	2421
R2L	995	2754
U2R	52	200
Normal	67343	9711
Total	125973	22544

B. Evaluation Metrics

The model's performance was evaluated using the confusion matrix. Confusion Matrix which is particularly valuable for assessing how well the system is able to correctly identify instances of different classes, such as Normal and malicious attacks. The following metrics were calculated:

- True Positive Rate (TPR) is the ratio of True Positives (TP) to the sum of True Positives (TP) and False Negatives (FN). It is calculated by

$$(\text{TPR}) = \frac{TP}{TP+TN} \quad (1)$$

- False Positive Rate (FPR) is the ratio of False Positives (FP) to the sum of False Positives (FP) and True Negatives (TN). It is calculated by

$$(\text{FPR}) = \frac{FP}{FP+TN} \quad (2)$$

- Accuracy is the ratio of the sum of True Positives (TP) and True Negatives (TN) to the sum of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). It is calculated by

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

- Precision (P) is defined as the ratio of true positive samples to predicted positive samples; it represents the confidence of attack detection. It is calculated by

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

- Recall (R) is defined as the ratio of true positive samples to total positive samples and is also called the detection rate. It is calculated by

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

- F1-Score (F) is defined as the harmonic average of the precision and the recall. It is calculated by

$$F = \frac{(2*P*R)}{(P+R)} \quad (6)$$

C. Experimental Results

In this section, we provide the detailed results of the experiments using the proposed model. The dataset was originally divided into separate training and testing sets.

1) Experimental Set Up: The first experiment used the predefined datasets to train the hybrid model and assess its performance on the test data. This setup, however, exhibited some data imbalance, leading to an exceptionally high accuracy rate of 99%. To address potential data imbalance, the second experiment combined the training and testing sets into a single dataset called kddtri.txt. We then applied an 80/20 split, where 80% of the combined data was used for training and 20% for testing. This configuration yielded an accuracy rate of 98%, indicating a slight reduction in accuracy but potentially better generalization and reduced overfitting. In the third experiment, the combined dataset was divided with a 70/30 split, with 70% for training and 30% for testing. This approach also resulted in an accuracy rate of 98%, confirming that the hybrid model maintained high performance across different data splits.

2) Performance Evaluation of Imbalance Dataset: The classification report on the test data provided detailed insight into the hybrid model's performance across different intrusion classes. From Figure 4, the precision, recall, F1-score, and accuracy metrics offered a nuanced understanding of how well the model identified instances within each class.

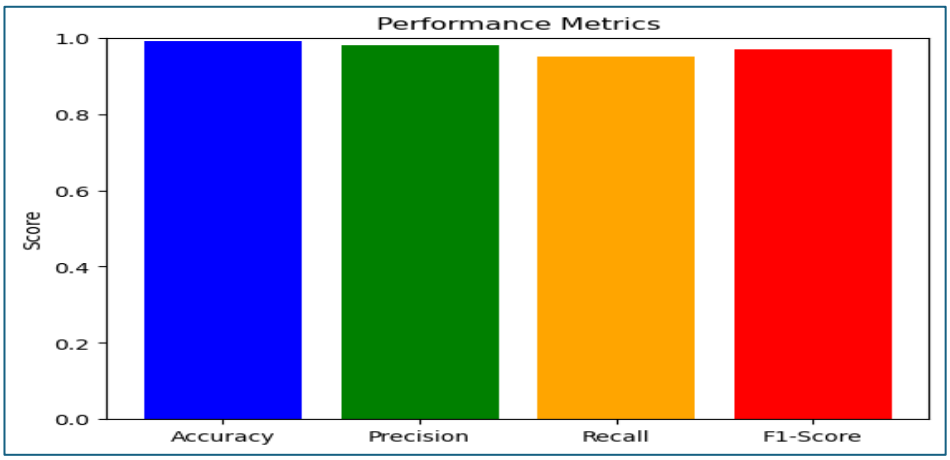

Fig. 5. Overall Performance Evaluation of Imbalanced Dataset

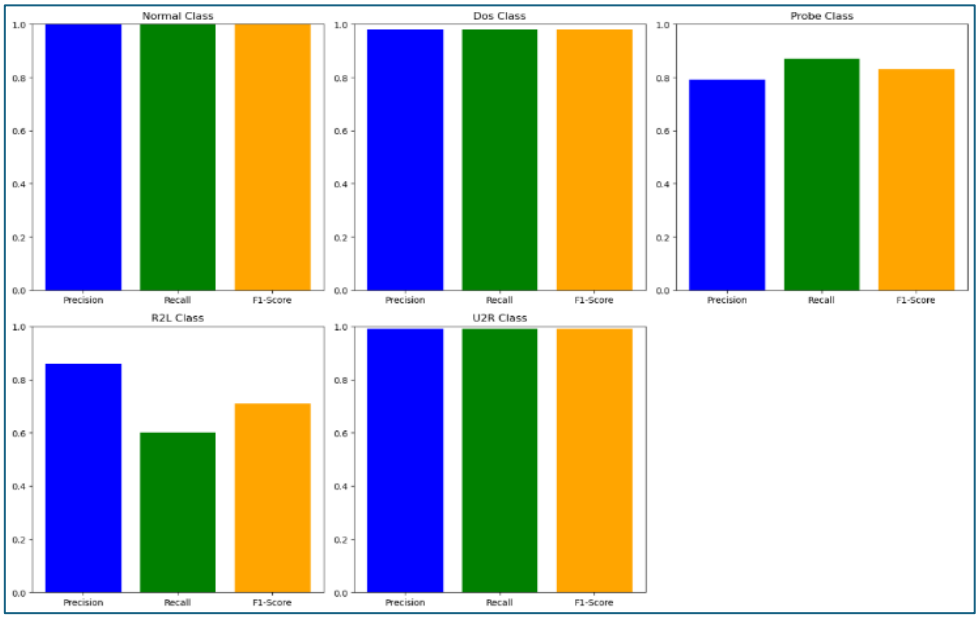

Fig. 6. Performance Evaluation of the Attacks of Imbalanced Datasets

3) Confusion Matrix: The confusion matrix served as a detailed map of the model's predictions against the actual classes, offering valuable insights into the hybrid model's performance across various intrusion categories.

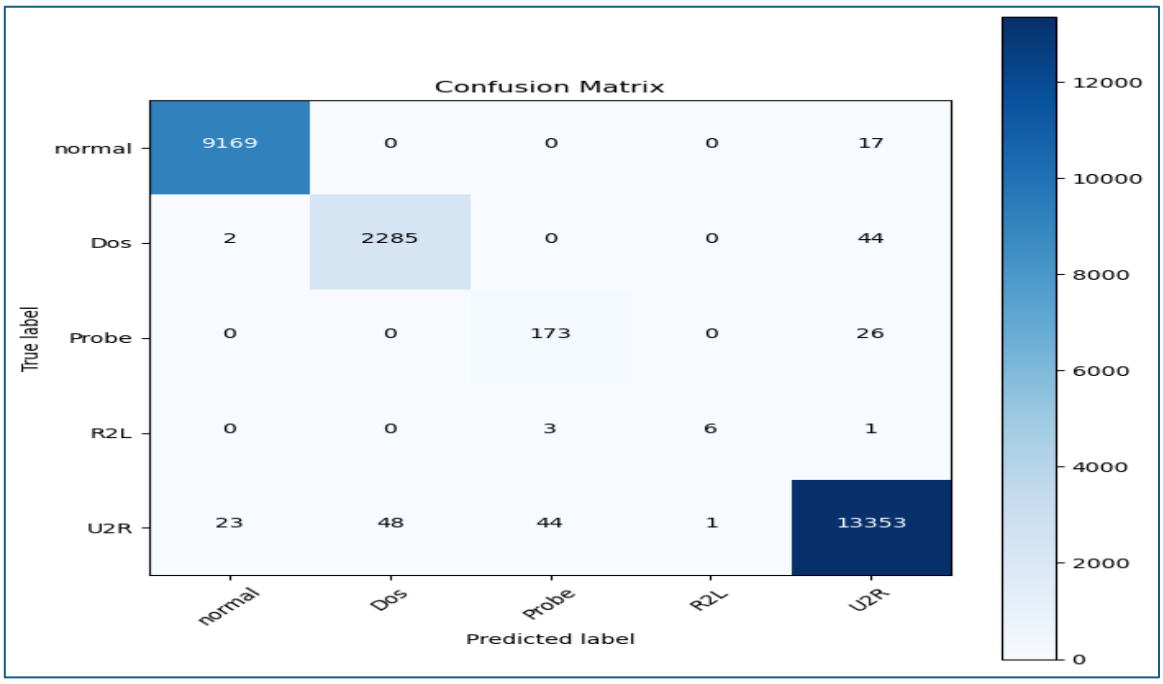

Fig. 7. Confusion Matrix: Decoding Model Performance of Imbalanced Dataset

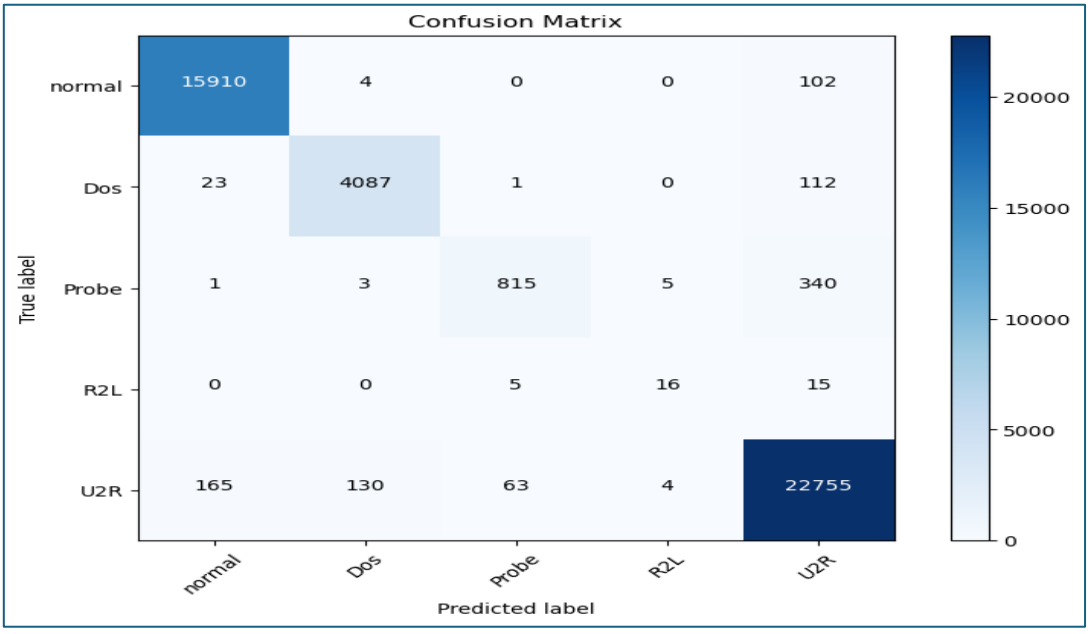

Fig. 8. Confusion Matrix: Decoding Model Performance of Balanced Dataset

4) *The Classification Report:* The classification report gave an overview of a model's performance across different classes, providing metrics like precision, recall, F1-score, and support. Table 3 explained the terms and how they were reflected in the provided classification report. This first experiment exhibited some data imbalance, leading to an exceptionally high accuracy rate of 99% while the second and third experiment resulted in an accuracy rate of 98%, confirming that the hybrid model maintained high performance across different data splits.

```
In [17]: from sklearn.metrics import classification_report, confusion_matrix

         # Predictions on the test set
         y_pred = svm_model.predict(X_test)

         # Class names
         class_names = ['normal', 'Dos', 'Probe', 'R2L', 'U2R']

         # Classification Report
         classification_rep = classification_report(y_test, y_pred, target_names=class_names)
         print("Classification Report on Data Test:\n", classification_rep)

         Classification Report on Data Test:
                       precision    recall  f1-score   support

               normal       0.99      0.99      0.99     10678
                  Dos       0.97      0.97      0.97      2815
                Probe       0.91      0.71      0.80       776
                  R2L       0.80      0.33      0.47        24
                  U2R       0.98      0.98      0.98     15411

             accuracy                           0.98     29704
            macro avg       0.93      0.80      0.84     29704
         weighted avg       0.98      0.98      0.98     29704
```

Fig. 9 Classification Report for Imbalanced Dataset of Intrusion Detection

```
# Classification Report
classification_rep = classification_report(y_test, y_pred, target_names=class_names)
print("Classification Report on Data Test:\n", classification_rep)

Classification Report on Data Test:
              precision    recall  f1-score   support

      normal       1.00      1.00      1.00      9186
         Dos       0.98      0.98      0.98      2331
       Probe       0.79      0.87      0.83       199
         R2L       0.86      0.60      0.71        10
         U2R       0.99      0.99      0.99     13469

    accuracy                           0.99     25195
   macro avg       0.92      0.89      0.90     25195
weighted avg       0.99      0.99      0.99     25195
```

Fig. 10. Classification Report for Balanced Dataset of Intrusion Detection

V. CONCLUSIONS

The Fuzzy-SVM Model proved to be highly effective for IoT network security, handling uncertain and imprecise data with strong classification capabilities. This success highlights the need for dynamic and adaptable defense strategies in the evolving cybersecurity landscape. The model's robust performance metrics indicate its potential for application in other domains with complex data patterns and uncertainties, suggesting a foundation for ongoing improvement and exploration.

REFERENCES

[1] C. Liang, B. Shanmugam, S. Azam, A. Karim, A. Islam, M. Zamani, S. Kavianpour, and N. B. Idris, "Intrusion detection system for the Internet of Things based on blockchain and multi-agent systems," *Electronics (Switzerland)*, vol. 9, no. 7, pp. 1–27, 2020. [Online]. Available: https://doi.org/10.3390/electronics9071120.

[2] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A New Ensemble-Based Intrusion Detection System for Internet of Things," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1805–1819, 2022. [Online]. Available: https://doi.org/10.1007/s13369-021-06086-5.

[3] S. Krishnaveni, S. Sivamohan, S. S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing," *Cluster Computing*, vol. 24, no. 3, pp. 1761–1779, 2021. doi: 10.1007/s10586-020-03222-y.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets, and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019. [Online]. Available: https://doi.org/10.1186/s42400-019-0038-7.

[5] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," *ACM International Conference Proceeding Series*, pp. 1–6, 2018. [Online]. Available: https://doi.org/10.1145/3167918.3167951.

[6] K. Kimani, V. Oduol, and K. Langat, "Cyber security challenges for IoT-based smart grid networks," *International Journal of Critical Infrastructure Protection*, vol. 25, pp. 36–49, 2019.

[7] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Computers & Security*, vol. 65, pp. 135–152, 2017.

[8] R. Shanmugavadivu and N. Nagarajan, "Network intrusion detection system using fuzzy logic," *Indian Journal of Computer Science and Engineering (IJCSE)*, vol. 2, no. 1, pp. 101–111, 2011.

[9] T. Sherasiya, H. Upadhyay, and H. B. Patel, "A survey: Intrusion detection system for internet of things," *International Journal of Computer Science and Engineering (IJCSE)*, vol. 5, no. 2, pp. 91–98, 2016.

[10] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. De Alvarenga, "A survey of intrusion detection in the Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[11] S. Fenanir, F. Semchedine, and A. Baadache, "A machine learning-based lightweight intrusion detection system for the Internet of Things," *Revue d'Intelligence Artificielle*, vol. 33, no. 3, pp. 203–211, 2019. [Online]. Available: https://doi.org/10.18280/ria.330306.

[12] A. ul H. Qureshi, H. Larijani, J. Ahmad, and N. Mtetwa, "A Heuristic Intrusion Detection System for Internet-of-Things (IoT)," *Advances in Intelligent Systems and Computing*, vol. 997, pp. 86–98, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-22871-2_7.

[13] M. Almseidin and S. Kovacs, "Intrusion detection mechanism using fuzzy rule interpolation," *ArXiv Preprint ArXiv:1904.08790*, 2019.

[14] D. Jing and H. B. Chen, "SVM-based network intrusion detection for the UNSW-NB15 dataset," in *Proc. Int. Conf. ASIC*, 2019, pp. 1–4. doi: 10.1109/ASICON47005.2019.8983598.

[15] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine learning-based IoT-botnet attack detection with sequential architecture," *Sensors (Switzerland)*, vol. 20, no. 16, pp. 1–15, 2020. doi: 10.3390/s20164372.

[16] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, and S. Sivamohan, "Anomaly-based intrusion detection system using support vector machine," in *Advances in Intelligent Systems and Computing*, vol. 1056, Springer, 2020, pp. 723–731. doi: 10.1007/978-981-15-0199-9_62.

[17] H. Alzahrani, M. Abulkhair, and E. Alkayal, "A multi-class neural network model for rapid detection of IoT botnet attacks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 688–696, 2020.

[18] B. Senthilnayaki, K. Venkatalakshmi, and A. Kannan, "Intrusion detection system using fuzzy rough set feature selection and modified KNN classifier," *Int. Arab J. Inf. Technol.*, vol. 16, no. 4, pp. 746–753, 2021.

[19] M. Almseidin, J. Al-Sawwa, and M. Alkasassbeh, "Anomaly-based intrusion detection system using fuzzy logic," *2021 International Conference on Information Technology (ICIT)*, pp. 290–295, 2021.

[20] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021. [Online]. Available: https://doi.org/10.1016/j.cose.2020.102158.