Iraqi Journal for Computer Science and Mathematics

Journal Homepage: http://journal.esj.edu.iq/index.php/IJCM
e-ISSN: 2788-7421 p-ISSN: 2958-0544

Improves Intrusion Detection Performance In Wireless Sensor Networks Through Machine Learning, Enhanced By An Accelerated Deep Learning Model With Advanced Feature Selection

Hadeel M. Saleh[1,2*] , Hend Marouane[1] , Ahmed Fakhfakh[3]

1 National School of Electronics and Telecommunications (ENET'COM), NTS'COM Laboratory, Safax University, Sfax, Tunisia

2 Continuing Education Center, University of Anbar, Ramadi, Iraq

3 Laboratory of Signals, Systems, Artificial Intelligence and Networks (SM@RTS), Digital Research Center of Sfax (CRNS), National School of Electronics and Telecommunications of Sfax (ENET'com), University of Sfax, Sfax, Tunisia

*Corresponding Author: Hadeel M. Saleh

DOI: https://doi.org/10.52866/ijcsm.2024.05.03.050
Received April 2024; Accepted June 2024; Available online August 2024

ABSTRACT

Wireless Sensor Networks (WSNs) have been securing a big position in the new aspect of security network attacks, where these suffer from various serious cyber threats that can play with their data integrity and reliability. Due to the key importance of WSN in a wide spectrum range of applications such as environmental monitoring and military field, building reliable, robust and efficient intrusion detection systems (IDS) is necessary. Although traditional machine learning approaches have been intended to detect these threats, they often lack high accuracy due to the complexity and dimensionality of WSN data.

To address these limitations, the study introduces an innovative approach that greatly improves intrusion detection performance in WSNs by combining a high-speed deep learning model with sophisticated feature selection methods. The newly developed system underwent extensive testing using the WSN-DS dataset and applied Gaussian Naive Bayes (GNB) and Stochastic Gradient Descent (SGD) algorithms within the machine learning framework. The outcomes were exceptional, demonstrating a flawless accuracy rate of 100% and representing a significant advancement compared to prior methodologies based solely on traditional machine learning techniques.

The study illustrates how the fusion of deep learning and optimized feature selection effectively addresses the distinctive challenges presented by WSN environments. The results not only present a highly precise and effective method for intrusion detection but also lay the groundwork for further research focused on fortifying the security of sensor networks against progressively intricate cyber threats.

Key words: Wireless Sensor Networks (WSN), Intrusion Detection Systems (IDS),Deep Learning, Feature Selection, Gaussian Naive Bayes(GNB) Stochastic Gradient Descent

1. INTRODUCTION

Invest in Intrusion Detection intrusion detection is paramount to cyber security. Its primary function towards the securing of IT and OT network infrastructures. It continuously monitors internal as well as external malicious activities which helps in giving real-time visibility to detect unauthorized intrusions[1]. Intrusion detection is crucial to guard against these ever-changing threats, but in the face of their staggering increase both in quantity and sophistication due to worldwide cyber-crime digitization, various methods are necessary [2].

An intrusion detection system (IDS) is variously a hardware- or software-based security tool that automatically scans network/system activities for malicious events.

There are many forms of WSN attacks. The most common risks to WSNs are DoS attacks, node compromise, eavesdropping, impersonation, and routing-based attacks including Sybil and wormhole attacks (Godala et al., 2020). These attacks try to disrupt network operations, drain resources, hack nodes, disclose sensitive data, or disrupt routing and communication. DoS attacks overload the network with traffic, delaying or dropping genuine communications. Attackers can achieve network control by capturing and manipulating network nodes. Eavesdropping and impersonation attacks can compromise network confidentiality, integrity, and availability by accessing sensitive data and mimicking genuine nodes [3].

The resource limits of WSNs and IoT devices impede intrusion detection system implementation. Lightweight and efficient IDS solutions are needed due to limited processing power, memory, and battery life. Recent breakthroughs in machine learning and artificial intelligence can enable adaptive and intelligent IDS that can adapt to new threats. To

stay up with the changing threat landscape and secure and resilient WSNs and IoT systems, research and development must continue [4].

This paper presents a deep learning model utilizing CNNs to detect various types of DoS attacks within the NSL-KDD dataset. Our approach includes several key phases: Data preprocessing involves managing categorical features and normalizing the data, while the CNN architecture is tailored to efficiently capture complex network traffic patterns. The model is further refined through sophisticated optimization techniques. Finally, the model's performance is evaluated using various metrics.

The structure of the paper is as follows: Section 2 provides a review of related work. Section 3 details the architecture of the deep learning network. Section 4 describes the research model. Section 5 assesses the proposed model through testing on simulated NSL-KDD datasets. Section 6 summarizes the contributions of the study and suggests avenues for future research.

2. RELATED WORK

Many recent studies have addressed the topic of intrusion detection and prevention of cyber-attacks using machine learning techniques and neural networks in different network environments. These studies have been characterized by the application of innovative approaches to improve detection accuracy and reduce false alarm rates. For example, a 2024 study by Elamparithi et al. used a random forest classifier to detect different types of DDoS attacks in IoT networks. The research methodology included extensive preliminary steps to clean, transform and organize the data. The results showed an accuracy of up to 99.53% in classifying attacks, indicating the effectiveness of the proposed model, although there are challenges related to the complexity of the computations and the long classification time for some attacks.

In another study by Fuat TÜRK in 2023[5], intrusion detection and analysis were done using UNSW-NB15 and NSL-KDD datasets with the use of a set of Master Learning algorithms such as Logistic Referral, Original Neighbors (KNN), Random Forests, and LTM. The study achieved 98.6% discrimination on the UNSW-NB15 dataset and 97.8% on the NSL-KDD dataset. Despite this stellar accuracy, the study is challenged by the lack of planet data and the need to improve multi-class accuracy.

V. Gowdhaman and R. Dhanapal 2021[6] An intrusion detection system in wireless sensor networks using a deep neural network model. The study relied on cross-feature selection technique to improve detection accuracy and reduce the false alarm rate. The study achieved an accuracy of up to 95.5% in detecting threats, but the researchers pointed to computational complexities and challenges in adapting to various attacks in different environments.

M. Maheswari and R. A. Karthika 2021 [7] A hybrid framework combining long-short-term memory (LSTM) networks and spotted hyena optimizers (SHO) to improve detection accuracy in wireless sensor networks and the Internet of Things. The model achieved 99.89% prediction accuracy, but faced challenges related to high power consumption and implementation complexity in environments requiring real-time processing.

A 2021 study by Bilal et al[8]. also demonstrated the use of deep and recurrent neural networks in intrusion detection systems, where they achieved up to 94% accuracy, but noted a high false alarm rate in some categories and low accuracy in multi-category scenarios. They proposed the use of enhanced feature selection techniques to improve the performance of the systems, and applied them to more complex datasets such as Kyoto and CICIDS2017.

Pankaj et al.2022[9] also investigated the use of convolutional neural networks (CNN) in a deep learning model for intrusion detection in wireless sensor networks, where the data was processed using a CNN model to extract features and evaluate the performance of the model. The study achieved an accuracy of 97%, but highlighted the need to develop a comprehensive model that simultaneously handles data and header analysis to address the growing cyber threat.

It is clear from these studies that there has been significant progress in the use of machine learning techniques and neural networks to improve the performance of intrusion detection systems and prevent cyber-attacks. However, challenges remain in terms of computational complexity, dealing with unbalanced data, and improving model performance in real-world environments. Continued research in this area is essential to develop more effective solutions that can adapt to increasing cyber threats in different environments.

.

3. PROPOSED METHODOLOGY

3.1 Experiments Procedures

The Phases of the current experiment will be done through the explanation in the following Figure 1.

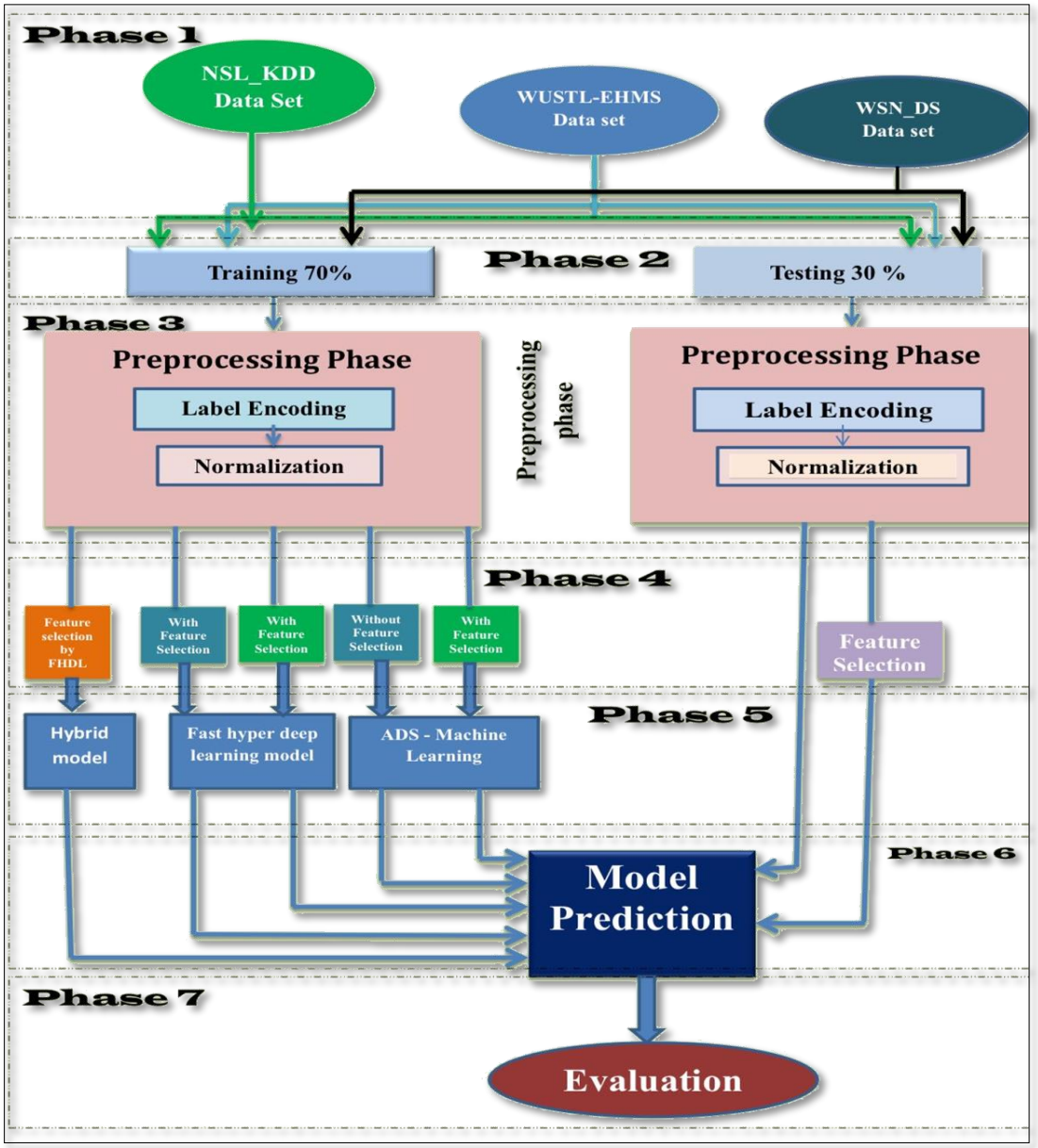

Figure 1: Experiments Procedures

3.1.1 PHASE 1: DATASETS

a. DATA GATHERING AND VERIFICATION

This thesis utilized WSN_DS, WUST-EHMS 2020, and NSL_KDD datasets. Kaggle offers the WSN_DS dataset for wireless sensor network intrusion detection systems. Nineteen features cover a wide variety of network security analysis qualities.

The NSL-KDD dataset helps cyber security researchers. This popular dataset for network anomaly analysis and intrusion detection underpins many security studies and applications. The KDD99 dataset, widely used in cyber security research, has been updated to NSL-KDD. This huge and comprehensive dataset lets academics solve a variety of cyber security issues creatively. The dataset comprises 148,517 record samples with 42 characteristic values each. Because the dataset covers several attack types and network traffic characteristics, it can be used to design and test anomaly monitoring and intrusion detection methods.

b. TRAINING AND TESTING

Divide the dataset into training (70%) and testing (30%) sets to train models. This requires SGD and GNB model training. Performance indicators including precision, recall, accuracy, and F1 score are calculated.

The suggested intrusion detection solution uses machine learning to improve WSN intrusion categorization. After loading WSN-DS input data, many predefined actions with hyper parameters are performed. PCA preprocessing reduces dataset dimensionality, making feature extraction easier. After that, SVD is used to improve the data and find pertinent properties. Both GNB and SGD models are trained; GNB uses the mean and covariance matrices for each class to calculate class probabilities, whereas SGD uses gradient descent on jumbled training data to update model parameters iteratively. Assessment evaluated taught models. Data was aligned for prediction using SVD and PCA to change each testing set data point. The GNB model predicts labels using class probabilities to select the most likely class. SGD training parameters determine the decision boundary, which predicts the SGD model. For each projected and real label, accuracy, precision, recall, and F1 score were calculated to evaluate performance. These procedures confirmed the algorithms' invasion detection accuracy. People call this "data engineering." This phase is crucial to schooling. Data processing include feature selection, cleaning, and normalization. The most relevant features were extracted using filters. Use training data and the right feature vector to train the model. Following training, the model's accuracy can be checked against the validation set. The validated model was applied to the test dataset for analysis.

C. PREPROCESSING

Efficient data preparation is critical to machine learning, affecting prediction model performance and dependability. Before analysis, raw data must be cleaned, converted, and prepared to improve data quality and prediction accuracy. Data preprocessing challenges include noise, ambiguity, errors, and unnecessary information in real-world data. This is concerning because models with erroneous or insufficient data may give meaningless findings, reducing decision-making standards. Data scientists handle this issue by cleansing, structuring, and handling missing data.

Wireless sensor networks face data loss and unintended consequences. Research on networking privacy and security has been spurred by new technology.

1-LABEL ENCODING

Data pre-processing in machine learning pipelines requires categorical data handling. Category variables, which describe non-numerical properties, must be quantified. This function helps algorithms comprehend and analyze data. The Label Encoder, which assigns number labels to each category, is a popular transformation method.

Algorithm1: Label Encoder
Input : Raw data
Output : encoding data
BEGIN **Step 1:** $\boldsymbol{for\ i \leftarrow 0}$ $\quad \boldsymbol{represrnt\ \ the\ length\ of\ the\ dataset}$ **Step 2: convert the string to single integer** **end** **END**

2-DATA NORMALIZATION:

Categories' nominal and ordinal feature names are strings. Ordinal features may need labels to arrange information, although nominal features may not. To guarantee the learning algorithm reads features correctly, labels must be encoded as integers during data pre-processing.

Algorithm 2: Min-Max Normalizing features

Input :

Training set X_train

Test set X_test

Length of set k

Significance level α

Output:

- **Normalized training set X_train_normalized**
- **Normalized test set X_test_normalized**

BEGIN

Step1 : For each feature: Calculate Max and Min using Value at Risk (VaR) at significance level α

$$Max_{var} = VaR\ (\{X_i'\}_{i=1}^{j}; m;\ \alpha)$$

Step2: $Max_{global} = max\ (Max_{var}, max\ (\{X_i'\}_{i=1}^{j}))$

 # Determine Max = highest value from VaR calculation

.Determine Min = lowest value from VaR calculation

Step3: $Min_{global} = min\ (Min_{var}, min\ (\{X_i'\}_{i=1}^{j}))$

Step4: $X_i' = \frac{X_i - Min_{global}}{Max_{global} - Min_{global}}$ **# For each feature in X_train:**

X_train_normalized = (X_train - Min) / (Max - Min)

Step5: $X_j' = \frac{X_l - Min_{global}}{Max_{global} - Min_{global}}$ **# For each feature in X_test:**

X_test_normalized = (X_test - Min) / (Max - Min)

End

D.FEATURES SELECTIONS

Feature selection is used to reduce the number of possible features from a large pool to a more manageable size. Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) are commonly used pre-processing techniques to increase the detection accuracy of the classification algorithm, reduce the computational load on the IDS, and preserve the maximum amount of information.

1.PCA PRINCIPAL COMPONENT ANALYSIS (PCA)

is a popular tool used by researchers to help them identify patterns in high-dimensional data. Principal component analysis (PCA) aims to represent known and unknown features using fewer standard feature images, or Eigen objects. Data from statistics show that PCA is useful for recognizing and validating facial features. To utilize PCA, a two-dimensional matrix of face pictures must be converted into a one-dimensional vector. A one-dimensional vector's orientation within a row or column has no bearing on its value.

Algorithm3: Principal Component Analysis
Input: An input matrix
Output: Feature Vectors

BEGIN

1: Read signals.

2: Make a training set of total M signal to use the in computing the Average Mean as shown in the equation .

$$Avarage = \frac{1}{M}\sum_{n=1}^{M} Training \text{ features } (n) \quad (3.1)$$

3: Subtract the Original signal from the Average Mean as shown in the equation:

Sub= Training features Average

4: Calculate the Covariance Matrix as shown in the equation :

$Covariance = \sum_{n=1}^{M} sub(n)\ sub^{T}(n)$ (3.2)

Where M: is the Training set of total signals

μ: Represent the average Mean

Sub: Represent the subtracted signal from the average μ

5: Calculate the Eigenobject of the Covariance Matrix.

6: Sort and choose the best Eigenobject. The highest Eigenobject that belong to a group of Eigenvectors are chosen; these M Eigenvectors describe the Eigenobject.

7: Project the training samples onto Eigenobject and attain feature space.

END

2. SVD

There is also the Singular Value Decomposition (SVD) method for data partitioning. In signal processing and statistics, PCA is used for many different tasks, such as pattern recognition and feature extraction from matrices. However, PCA cannot identify information about features present in a signal at different frequencies or extract features from a single signal. Since true physiological differences can be masked by frequency differences, SVD may be a better feature extraction technique than PCA. The WSN-DS dataset, the WUSTL EHMS 2020 dataset and the NSl_KDD intrusion classification dataset are first loaded by the method. Hyper parameters for Stochastic Gradient Descent (SGD), such as learning rate and iterations, are then defined, along with the parameters required for PCA and Gaussian Naive Bayes (GNB). After generating lists of predicted and true labels, the SVD and PCA operations are performed.

Algorithm (4): Singular Value Decomposition

Input: An input matrix A

Output: U and V are products of matrices, and $A = \cup\sum V^{T}$.

BEGIN

1: First ,calculate AA^{T} and $A^{T}A$.

2: Use AA^{T} to find the eigenvalues and eigenvectors to form the columns of U: ($AA^{T} - \lambda I$); ẍ.= 0. (3.3)

3: Use $A^{T}A$ to find the eigenvalues and eigenvectors to form the columns of V :($AA^{T} - \lambda I$); ẍ.= 0.(3.4)

4: Divide each eigenvector by its magnitude to form the columns of U and V.

5: Take the square root of the eigenvalues to find the singular values ,and arrange them in the diagonal matrix S in descending order :$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$.

END

E. CLASSIFIERS MODELS/ALGORITHMS

A. ADS MACHINE LEARNING CLASSIFIER

Using three data sets reflecting different contexts, the generalization of this system is improved by applying Gaussian Naive Bayes (GNB), Complement Naive Bayes (CNB) and Stochastic Gradient Descent (SGD) algorithms. To test the effectiveness of the models on untested data, the system first collects and prepares the data, which is then divided into training and test sets. The basic performance of each algorithm can be assessed in the first step, as the algorithms are applied directly to the data without the need for additional processing methods. By focusing on the features that have the most influence on the prediction, performance can be improved by applying feature selection approaches to determine which features are most significant. The next step is dimensionality reduction using Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) techniques. This reduces the number of features while retaining critical information, thereby improving model generalization and reducing computational complexity. To find the most efficient way to achieve generalization, the performance of the models is compared in different scenarios across a range of contexts, using precise criteria such as precision, recall and verification accuracy. Finally, to ensure better generalization and successful applications in the future, the results are reviewed and specific recommendations are made depending on the performance at each stage. Finally, we review the following algorithms, describing their many applications and how they work: Gaussian Naive Bayes (GNB), Complement Naive Bayes (CNB) and SGD.

1-GNB algorithm:

Based on the premise of a Gaussian distribution of data, the Gaussian Naive Bayes (GNB) classification algorithm applies Bayes' theorem to the data. This algorithm is a popular choice in many applications because of its ability to process large data sets quickly and efficiently. GNB is based on the principle of conditional independence, which states that each feature (variable) influences the classification result separately from the other features. Despite its apparent simplicity, this assumption significantly reduces the computational complexity, allowing the technique to be used quickly and effectively. The first step in the algorithm's analysis is to determine the mean and variance for each feature within each class by analyzing the data. By understanding the distribution of the data, this technique helps to accurately calculate the posterior probability. The percentage of samples in the data set that belong to a particular class is then determined by calculating the class probability. The approach determines how much of a given sample belongs to a particular class by calculating the conditional probability for each attribute using the Gaussian distribution. The posterior probability for each class is obtained by adding the calculated probabilities using Bayes' theorem. To ensure the accuracy of the classification process, the sample is placed in the class with the highest posterior probability. The GNB algorithm's efficient computations and simple assumptions make it useful for applications that require classification accuracy and speed. It is widely used in pattern recognition, biological data analysis and text classification. Because GNB is based on the Gaussian distribution, it performs exceptionally well on continuous data that fits this distribution, increasing its usefulness in a wide range of real-world situations. The Gaussian Naive Bayes algorithm balances classification speed and accuracy by relying on efficiency and simplicity. It works consistently in many real-world situations despite its reliance on simplifying assumptions, making it a powerful weapon in the machine learning toolbox.

Algorithm (5): Gaussian Naive Bayes
Input: Training dataset D, test instance x
Output: Predicted class label y
1. For each class c in D, determine the class previous P(C). 2. For each feature f in x: a. Determine out the mean μ and standard deviation f for each c class. b. The Gaussian distribution formula can be used to determine the likelihood of x given c. 3. For each class c, determine its posterior probability P(C|X). 4. Choose the most likely class c to represent class y based on the posterior distribution. 5. Return y

2.CNB algorithm:

A version of the classical Naive Bayes method, called Complement Naive Bayes (CNB), aims to improve performance in cases of unbalanced classification, or unequal class distribution. Although CNB is also based on Bayes' theorem, it overcomes some of the drawbacks of the conventional algorithm in cases where classes are unbalanced. Like the classic

Naive Bayes algorithm, CNB bases its final classification on the conditional independence principle, which states that each feature influences the final classification independently of the others. However, CNB increases the algorithm's accuracy in classifying under-represented classes by changing the way probabilities are calculated to reduce the impact of dominant classes on the classification process, unlike Naive Bayes.
The procedure starts by calculating the feature probabilities for each class; however, CNB focuses on the probabilities of the complementary classes rather than calculating the probabilities directly. The sum of the probabilities for all classes other than the target class is used to determine the complementary class for each feature. This method increases the accuracy of categorization, especially in cases where one class is very large compared to the others.
The procedure first calculates the complementary probabilities and then, using the complementary probabilities as a basis, applies Bayes' theorem to obtain the posterior probability for each class. By assigning the sample to the class with the highest posterior probability, a better balance of power across classes is ensured. Unbalanced classes are a significant problem in text categorization, where CNB is very useful. CNB can be modified to increase the classification accuracy of under-represented classes, making it suitable for a variety of applications such as email classification, financial data analysis and consumer behavior prediction. Although the CNB algorithm and Naive Bayes share the same basic idea, CNB performs significantly better when dealing with unbalanced data, making it a powerful and useful tool in challenging categorization scenarios.

Algorithm (6): Gaussian Naive Bayes
Input: Training dataset D, test instance x
Output: Predicted class label y
Where: - $\hat{y}$ is the predicted class. - c is a class from the set of all possible classes C. - P(c) is the prior probability of class c. - n is the number of features (e.g., words in text classification). - TF_ic is the term frequency of feature i in class c. - α is a smoothing parameter (typically Laplace smoothing, where α = 1). - V is the vocabulary or set of all possible features. - TF_i is the term frequency of feature i in the given document.

3- SGD:

A popular machine learning technique called stochastic gradient descent (SGD) is used to train linear models such as logistic regression and support vector machines (SVMs) very effectively, especially when working with large datasets. SGD updates the coefficients incrementally to minimize the loss function using the gradient descent method. SGD is more computationally efficient than traditional regression techniques because it updates the coefficients using only one sample or a small number of samples, as opposed to classic regression techniques that require the entire data set at each step. The algorithm first gives the coefficients random initial values before computing the gradient using a sample dataset and modifying the coefficients in accordance with the chosen learning rate. All of the dataset's samples go through this procedure again, enabling a slow convergence to the best answer. Because SGD relies on individual samples, it might be inefficient when it comes to updating stability. However, by utilizing techniques like acceleration and learning rate reduction, stability can be ensured and convergence can be accelerated. Because of its speed and ease of use, SGD is often used in applications such as text processing, predictive analysis and image categorization. Large computational problems in machine learning are best solved using this approach, although fine-tuning of parameters such as the learning rate is sometimes required. This is because neural networks and linear models rely on it as an essential training tool.

Algorithm (7): Stochastic Gradient Descent
Input: Training dataset D, learning rate η, number of epochs T, regularization parameter λ
Output: Learned weight vector w

1. Start w with some random numbers.
2. For t = 1 to T:
    a. Change the order of the D training dataset
    b. For each instance (x, y) in D:
        i. Compute the gradient ∇L(w; x, y) of the loss function L with respect to w
        ii. Update w using the gradient descent update rule: w = w - η * (∇L(w; x, y) + λ * w)
3. Return w

B. Hybrid Model:

Our hybrid system efficiently applies the deep learning framework to feature selection to improve overall performance through machine learning results (ADS-ML). To achieve reliable and effective results in categorization tasks, the system integrates classical machine learning methods with deep learning approaches. The system provides a comprehensive answer to current problems in the field, as it can be tailored to different data processing and classification challenges.

The data is first loaded from a CSV file and prepared by the system into features (X) and targets (y). To classify the type of attack, this system includes a dataset with 10 features (SVD1 to SVD10). The model performs better after normalizing the features using StandardScaler to put them on a standard scale.

Label Binarizer is used to encode the targets into binary format, which is suitable for training the model for multi-class classification. The data is then restructured to suit the needs of the Conv1D neural network, which transforms it into a three-dimensional matrix. The model has many Conv1D layers, starting with 16 filters and going up to 64 filters. It is built using sequential methods. The extraction of significant features from the data is aided by these layers. MaxPooling1D layers are used to minimize the quantity of the data and preserve the most significant features after each convolutional layer. Subsequently, 128 and 512 unit dense layers are employed for feature analysis and to produce intricate data representations. Before going into the final Dense layers, the output is first transformed into a one-dimensional vector by adding a Flatten layer. Five units with a softmax activation function make up the output layer (Dense), which distributes the probability and classifications the data into the desired classes. The Adam algorithm with a learning rate of 0.001 and a sparse_categorical_cross entropy loss function suitable for multi-class classification is used to build the model. With a batch size of 128 and 100 training cycles (epochs), the model is trained on a training set (70%) and a test set (30%), and its performance is continuously evaluated on the test set. The system combines the extraction of significant features from raw data using convolutional neural networks and accurate classification using dense layers. After training, the model is used to predict classes on the test set. The effectiveness of the model is assessed by calculating precision, recall and F-measure using precision_recall_f-measure. Using this method, the system is able to identify both local and global patterns in the data and classify them with great efficiency. The result is a reliable and accurate model that can effectively and efficiently detect attacks in complicated data. The online and offline phases are the two phases in which the system operates. The steps in the online phase are shown in Figure (2):

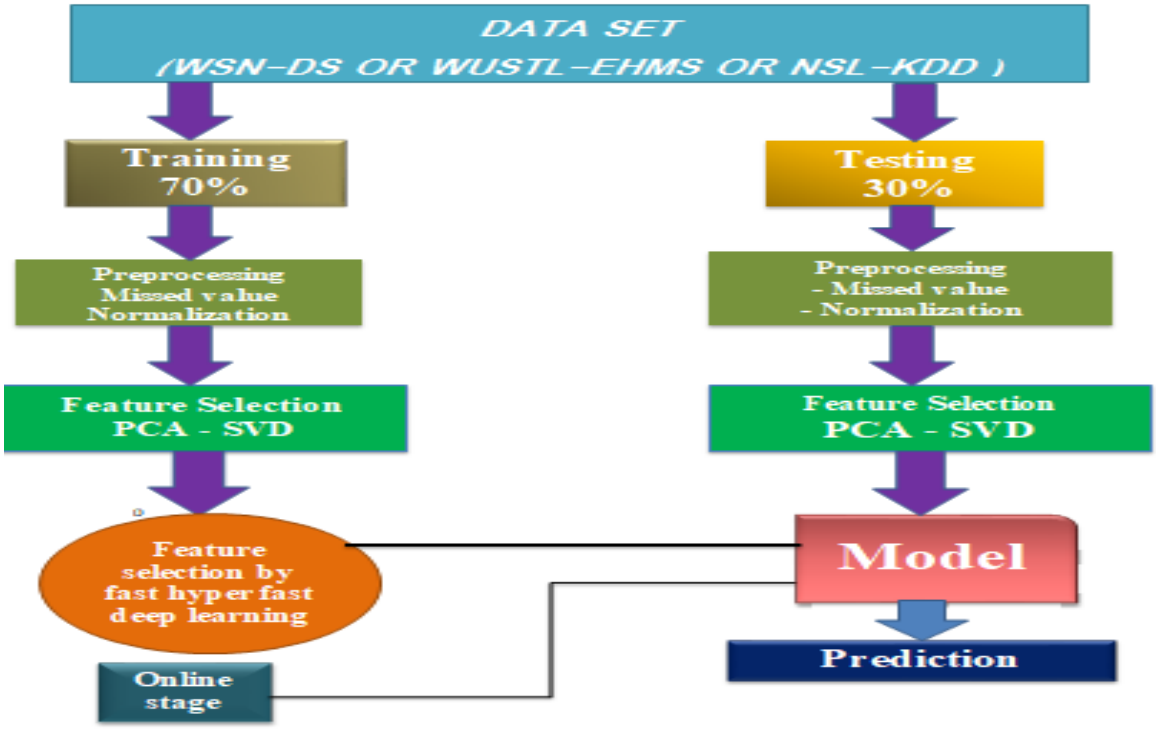

Figure 2: Hybrid model

Algorithm 2: Deep Learning and Machine Learning Hybrid System for Intrusion Detection
Input: Load the dataset(WSN-DS or Wustl-Ehms 2020 or).
Output: Intrusion Classification
Start
Step 1: Data Preparation
- Load the dataset (e.g., svd10.csv or svd15.csv or pca10.csv or pca 15.csv).
- Split into features (X) and target (y).
- Normalize features using Standard Scaler.
- Encode target using Label Binarizer.
- Reshape features for Conv1D layers.

Step 2: Build and Compile the Deep Learning Model
- Initialize a Sequential model.
- Add Conv1D layers with increasing filters and LeakyReLU activation, followed by MaxPooling1D.
- Add Dense layers (e.g., 128, 512 units).
- Add an output Dense layer with 5 units (softmax activation).
- Compile using Adam optimizer (learning rate 0.001) and categorical_crossentropy loss.

Step 3: Train the Deep Learning Model
- Train on preprocessed data for 1 epoch with a batch size of 128.
- Validate model performance.

Step 4: Feature Extraction
- Extract features from the penultimate layer.
- Save extracted features.

Step 5: Train Machine Learning Models
- Split extracted features (70% training, 30% testing).
- Train Gaussian Naive Bayes, SGD, and Complement Naive Bayes models.

Step 6: Test the Models
- Apply PCA and SVD to testing data.
- Make predictions using the trained models.

Step 7: Evaluate Performance
- Compare predicted labels with true labels.
- Calculate and report Accuracy, Precision, Recall, and F1 Score.

End

Figure 3 The figure shows the layers of the proposed system.

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 8, 16)	64
max_pooling1d_1 (MaxPooling1	(None, 8, 16)	0
leaky_re_lu_1 (LeakyReLU)	(None, 8, 16)	0
conv1d_2 (Conv1D)	(None, 6, 32)	1568
max_pooling1d_2 (MaxPooling1	(None, 6, 32)	0
leaky_re_lu_2 (LeakyReLU)	(None, 6, 32)	0
conv1d_3 (Conv1D)	(None, 4, 64)	6208
max_pooling1d_3 (MaxPooling1	(None, 4, 64)	0
leaky_re_lu_3 (LeakyReLU)	(None, 4, 64)	0
conv1d_4 (Conv1D)	(None, 2, 64)	12352
max_pooling1d_4 (MaxPooling1	(None, 2, 64)	0
leaky_re_lu_4 (LeakyReLU)	(None, 2, 64)	0
dense_1 (Dense)	(None, 2, 128)	8320
conv1d_5 (Conv1D)	(None, 2, 32)	12320
max_pooling1d_5 (MaxPooling1	(None, 2, 32)	0
leaky_re_lu_5 (LeakyReLU)	(None, 2, 32)	0
conv1d_6 (Conv1D)	(None, 2, 32)	3104
max_pooling1d_6 (MaxPooling1	(None, 2, 32)	0
leaky_re_lu_6 (LeakyReLU)	(None, 2, 32)	0
dense_2 (Dense)	(None, 2, 512)	16896
conv1d_7 (Conv1D)	(None, 2, 16)	24592

Figure3 :layers of proposed system

4- VERIFICATION BY CLOUD

The trained machine learning model was kept for further analysis and deployment in real-world scenarios after the offline phase. The learned parameters and architecture configuration were stored in the saved model file. Model validation through cloud-based means makes it possible to investigate how a model manages security protocols, guaranteeing that data is safe while it is being transmitted and processed. Generally speaking, cloud model validation makes sure the model can function reliably and accurately in real-world production contexts as well as test environments. This makes it

appropriate for usage in applications that demand high performance and quick reaction times. As previously mentioned, the offline phase involved deep and machine learning approaches to prepare and train the model. Data processing, feature extraction, and model evaluation utilizing data mining metrics were all part of this step. We will now talk about the online stage, which uses the offline stage's pre-trained model to detect intrusions in real time. When inbound data is processed, the system uses the model to identify potential security risks. It then makes a prediction about whether the data poses a risk and stores the outcome in an Oracle database. With the help of this combined offline analysis and real-time detection method, a robust and efficient solution for identifying intrusions in contemporary network environments is provided. The stages involved in producing a cloud are shown in Figure (4).

1. Create injection tool to generate normal and abnormal packets
2. Send encrypted data to the cloud containing the Intrusion Detection Model.
3. Feed these packets into the Intrusion Detection Model.
4. Get prediction.
5. Register log information in Oracle database with its prediction.

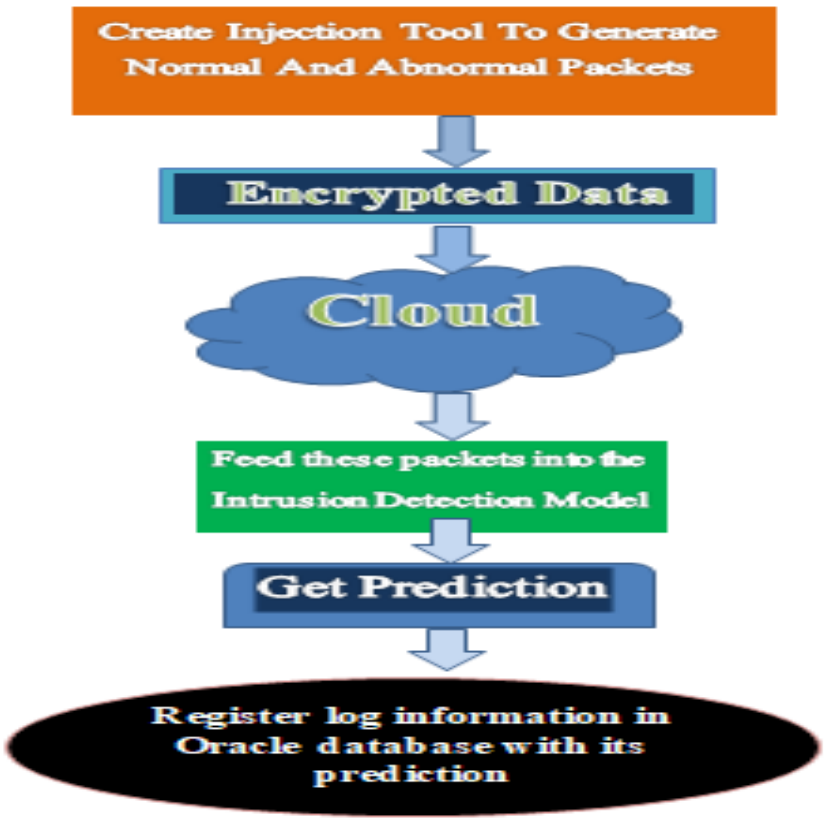

Figure 4: On Line Stage

During the online phase, a number of techniques were used to efficiently pre-process and analyze the data. These methods included:

A. Building a Cloud-based Model for Attack Prediction:

Overall, our contributions to the field of WSN intrusion detection represent significant advances, providing viable answers to important problems and laying the groundwork for further study and progress in this important area. We are building a cloud-based model using Java and Oracle to construct a comprehensive tool for predicting and analyzing WSN attacks Figure 5.

To facilitate the creation of scalable and flexible attack detection solutions, our contribution involves the integration of machine learning techniques with cloud computing technologies. Our cloud-based model facilitates the smooth deployment and management of WSN security systems by providing academics and practitioners with access to a centralized platform for data analysis, model training and attack prediction. By leveraging the computing power and

scalability of the cloud, organizations can more efficiently monitor and respond to security risks in WSNs, improving the overall resilience and reliability of the network. The data was efficiently pre-processed and analyzed using a number of methods. These methods included We'll develop a data injection tool that can generate both typical and unusual packets. This tool will simulate a range of network traffic situations, including both malicious and legitimate activity, in order to evaluate the effectiveness of the intrusion detection model. Figure 6 shows the shape of the injection tool.

B. Data Transmission to the Cloud:

The resulting encrypted data packets are sent to the cloud environment where the intrusion detection model is installed. Because cloud computing is flexible and scalable, it is an ideal platform for processing massive amounts of data in real time. Figure 7 shows the cloud data transport.

C. Packet Processing via Intrusion Detection Model:

The intrusion detection model deployed in the cloud environment will analyze data packets in real time as they are received. The model will evaluate the attributes of each packet to determine whether they are indicative of typical network behavior or a potential intrusion attempt.

D. Prediction Generation:

Predictions will be derived regarding the nature of each data packet based on the analysis conducted by the intrusion detection model.

These predictions will reveal whether the packet is classified as normal or abnormal, thereby offering valuable insights into potential security hazards look.

PHASE 6: PREDICTION

To predict outcomes or identify specific patterns, the generated models are now applied to new data that the model has never seen before. The foundation of machine learning is prediction, which assesses the model's ability to generalize and handle real-world data.

The available data is typically divided into two groups: a test set, which is used to evaluate the model's prediction accuracy, and a training set, which is used to build the model. The precision and effectiveness of the algorithms used, as well as the calibre of the hyper parameters selected, determine the calibre of the predictions. Several metrics, including accuracy, sensitivity, specificity and error rate, are used to evaluate the model's predictions. These metrics serve as indicators of the model's effectiveness in achieving its objectives. Thorough data preparation, including cleaning and normalization, is required before algorithms are applied to new data to ensure reliable results. To identify the strengths and weaknesses of the model in this situation and to enable future iterations, it is essential to compare expected and actual results.

PHASE 7: EVALUATION

Any study using machine learning algorithms must include evaluation as a critical component. Evaluation is the process of assessing the effectiveness of the produced models by applying them to actual or test data that was not used during training. The evaluation process allows us to understand how well the model handles unknown inputs and makes accurate predictions. At this point, a number of measures are used to provide a thorough evaluation of the model's performance, including accuracy, mean absolute deviation (MAE), root mean square error (RMSE) and confusion matrix. The evaluation helps to identify any deviations or errors in the forecasts, which helps to improve or modify the model to increase its accuracy and effectiveness. When evaluating in a real-time environment, it may also be necessary to look at the model's responsiveness and data flow management capabilities. An evaluation may also involve comparing the model's performance with that of other models or with previously established standards, providing a thorough understanding of the model's effectiveness and quality in meeting the research objectives. i. Evaluation Metrics Evaluation metrics are an essential tool for measuring the effectiveness and accuracy of intrusion detection models developed for Wireless Sensor Networks (WSNs) and the Internet of Things (IoT). Evaluation metrics are based on many factors that allow researchers to understand the performance of models in different training and testing scenarios. This allows researchers to identify the strengths and weaknesses of the models and improve their performance accordingly. Accuracy is one of the most widely used basic metrics; it expresses the proportion of positive or negative predictions that are correct. However, when it comes to class imbalances, focusing only on accuracy can be misleading because it can hide biases in the way rare cases such as attacks are classified. To account for these imbalances, additional measures such as sensitivity and specificity must be used. In intrusion detection systems, where the main objective is to reliably detect real threats, sensitivity, also known as the detection rate, represents the ability of the model to accurately identify positive cases (such as attacks). Conversely, specificity measures how well the model detects healthy cases and avoids false positives. In addition, a confusion matrix is used to present accurate and inaccurate predictions for each class, providing a thorough review of performance. Insight into the model's ability to balance false positives and false negatives can also be gained by evaluating model performance over a range of thresholds using the receiver operating characteristic (ROC) curve and area under the curve (AUC).

5. Results and Discussion

The results and their scientific interpretation will be discussed in this section in accordance with each subsection's part as follows

5.1 Preprocessing and Feature selection results below According to the result of Figure 5, as one sample of result to with and without feature selection, in titled PCS (10), all row in the dataset is a single data point that most likely represents

a network activity (or attack). Columns type of attack The types of network attacks are listed in this column (e.g., teardrop, smurf, apache2, mail bomb, back). PCA Components (pc1, pc2,..., pc10), The dataset's explanation of attack types includes a number of different network attack types, including "teardrop," "smurf," "apache2," "mail bomb," and "back." Every category denotes a distinct form of cyber attack or network activity. These columns show the values of the first 10 principal components for each attack type in addition to principle components (pc1 to pc10). For instance, the first item for the term "teardrop" indicates the location of this data point in the reduced PCA space with a pc1 value of -15218.502962 and a pc2 value of 64.701149.

	Attack type	pc1	pc2	...	pc8	pc9	pc10
0	teardrop	-15218.502962	64.701149	...	-10.771921	-0.215913	-0.949737
1	teardrop	-15217.924083	62.761871	...	-6.867171	-0.080040	-0.694839
2	teardrop	-15218.436818	65.448674	...	-13.749730	-0.500055	-1.142134
3	smurf	-14222.829826	166.944086	...	1.578879	-0.658237	-0.050621
4	smurf	-14224.906039	171.079482	...	-2.448081	-0.065712	-0.093171
...	...	...	...	...	...	...	...
6824	apache2	40191.960110	5745.699175	...	0.454941	-0.190012	0.026725
6825	smurf	-14220.747273	162.419477	...	7.573544	-1.140682	0.022591
6826	mailbomb	-12629.496626	25.678779	...	-12.131714	-0.397070	2.215958
6827	smurf	-14221.101223	163.188526	...	6.660570	-1.060128	0.000336
6828	back	39807.231955	-2829.064542	...	-1.069039	-0.618515	-0.236429

Figure 5: results preprocessing PCA(10)

Finally, PCA network security applications are often used in cyber security to detect anomalies, identify attack trends, and simplify data for faster processing, which helps clarify the relationships between different types of attacks and their characteristics.

In this section presents an analysis and discussion of the experimental data obtained. The accuracy, precision, recall and F1 score of the intrusion detection and attack classification system are used to measure its effectiveness. These metrics serve as a benchmark for the system's identification and classification accuracy between typical and attack events.

4.4.1 ADS-ML Results without feature selection

These metrics - accuracy, precision, recall and F1 score - can be used to assess how well the system can distinguish between normal and attack events. According to the results, the proposed system can successfully detect and classify attacks against WSNs. Machine learning techniques such as Gaussian Naive Bayes, Stochastic Gradient Descent and CNB have achieved high accuracy and realistic values for precision and recall. When tested, these algorithms demonstrated good generalization and learning from the training data, enabling them to effectively detect different types of attacks. Furthermore, the Convolutional Neural Network (CNN)-based deep learning technique showed exceptional performance in recognizing sophisticated attacks and recording intricate patterns. The several convolutional and pooling layers of the CNN architecture produced better classification accuracy as compared to traditional machine learning techniques. The deep CNN's ability to immediately learn complicated features from network traffic data is one of the reasons for its remarkable performance in identifying and classifying attacks. The hybrid deep machine model worker to improve machine learning results. The discussion section breaks down the advantages and disadvantages of the proposed strategy. Potential improvements are explored, such as improved feature selection techniques, hyper-parameter optimization, and the application of machine learning techniques to further improve the performance of the system. This paper discusses the proven paradigm of intrusion detection and attack classification for ensuring the security of wireless sensor networks, including its practical implications and real-world applications.

4.4.2 ADS-ML Results with Feature Selection Results

from the WSN_DS dataset The WSN_DS dataset was used to evaluate the effectiveness of three machine learning techniques: GNB, CNB and SGD, as shown in Table (1 The accuracy achieved by GNB was the highest at 96%, but in terms of the other criteria (accuracy, recall and F1 score), which ranged from 94% to 95%, it was comparable to both CNB and SGD. With a score of 95% in each category, CNB was the most balanced of all the criteria, showing that this strategy performed well in this particular situation.

Table 1: ADS-ML results without feature selection for the WSN_DS data set

Technique	Precision	Accuracy	Recall	F1-Score
GNB	96	94	94	94
CNB	95	95	95	95

In table (2), the effectiveness of the same approaches (CNB, SGD, and GNB) was examined. The outcomes demonstrated that, both CNB and SGD achieved 100% in the precision criterion, outperforming one another. Nonetheless, the total accuracy was very low, coming in at 87% and 86.7%, respectively, with GNB having the highest overall accuracy at 89%. These findings imply that while CNB and SGD might perform better in certain situations, they are not as steady as GNB in terms of overall effectiveness.

Table 2 ADS-ML results without feature selection for the Wustl-Ehms 2020 data set

Techniques	Precision	Accuracy	Recall	F1-Score
CNB	100	87	87	93
SGD	100	86.7	87	93
GNB	91	89	90	90

The three methods were evaluated using the NSL-KDD dataset and are shown in this table (3). SGD outperformed all benchmarks with an overall accuracy of 99% and 100% for precision, recall and F1 score. CNB's overall accuracy of 98% lagged slightly behind these results, but was still very close. GNB also performed well with an overall score of 99%, demonstrating the excellent effectiveness of these methods on this dataset.

Table 3 ADS-ML results without feature selection for the NSL-KDD data set

Techniques	Precision	Accuracy	Recall	F1-Score
GNB	99	99	99	99
SGD	100	99	100	100
CNB	99	98	99	99

Figure (4) shows the performance analysis of the CNB and SGD models using the PCA feature selection technique with 10 principal components. Although the performance of CNB was limited compared to the other model, it was still quite strong, achieving 81% precision, 82% accuracy and 82% recall, with an F1 score of 79%. These results show that although the model had a precision of 98%, accuracy and recall of 95% and an F1 score of 95%, it was still able to predict very well, but with a lower performance than SGD. This suggests that SGD performed better across the board and was more successful in using feature selection.

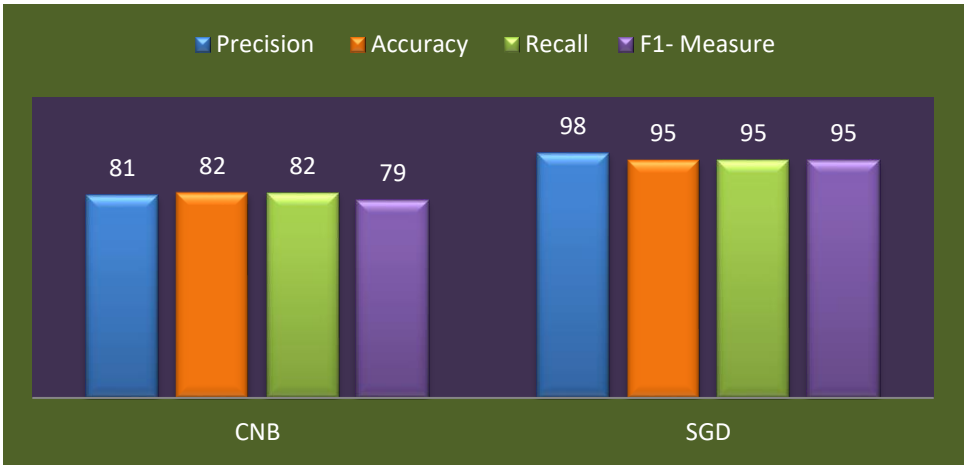

Figure 4: ADS-ML results with feature Selection PCA(10) for wsn_ds data set

The number of components in the PCA was increased to 15 in Figure (5) and the performance of the CNB and SGD models was assessed again. At this point, SGD is still clearly superior. While accuracy and recall improved to 96% and F1 score to 97%, precision remained at 98%. This suggests that SGD benefited significantly from the higher number of components in the PCA, resulting in a significant increase in performance.

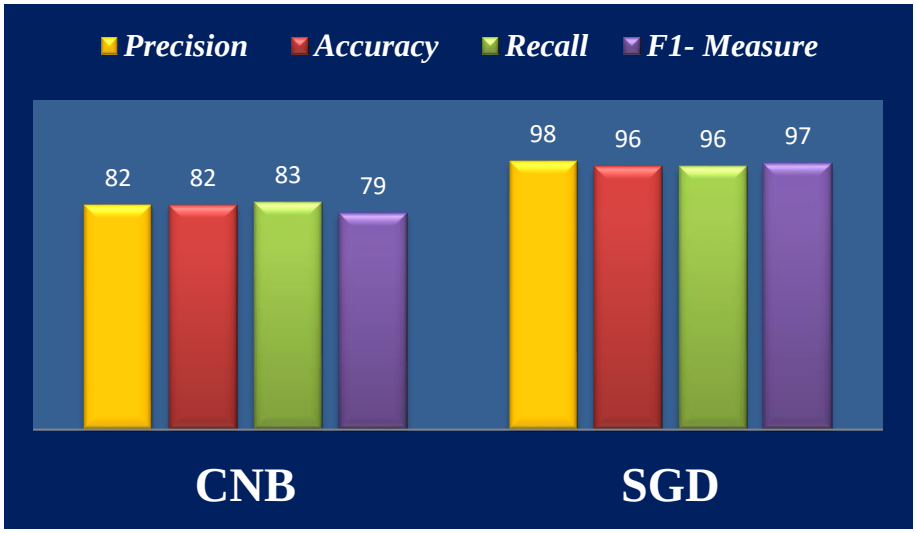

Figure 5: ADS-ML results Feature Selection PCA(15) for wsn_ds data set

According to Figure (6), the CNB and SGD models' performance was examined using a 10-component SVD. It yielded findings that were comparable to those of PCA, with 81% precision and accuracy, 82% recall, and 79% F1-Score. These findings imply that, in comparison to PCA, SVD had no appreciable effect in enhancing the model's performance. SGD did good once more. Accuracy was at 94%, recall was at 95%, and F1-Score was at 96%. Precision was still at 98%. Despite being strong, the performance was marginally worse than what PCA (15) produced.

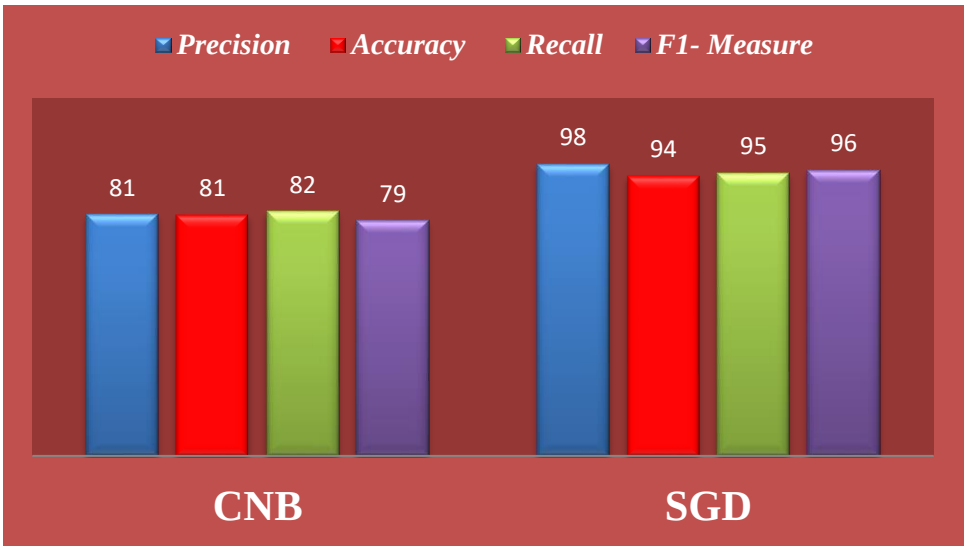

Figure 6 :ADS-ML results Feature Selection SVD(10) for wsn_ds data set

The graph selection process using SVD produced up to 15 results, which were then used to analyze the performance of the GNB and SGD models (see 7). In terms of performance, it performed worse than the other models. The F1 score was 78%, the accuracy and precision were 78% and the recall was 80%. This suggests that GNB did not benefit much from the performance-enhancing capabilities of SVD. SGD performed best across all tables, with the highest Precision at 100%. However, there was a slight drop in F1 score to 93%, Accuracy to 86% and Recall to 87%. While Precision was the best, the overall performance lagged somewhat behind the results obtained with PCA (15).

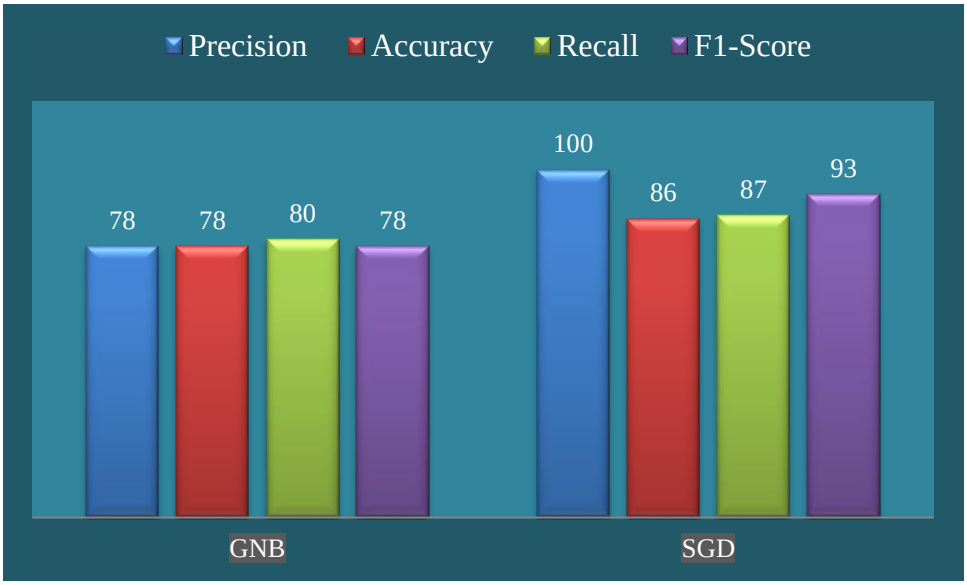

Figure 7: ADS-ML results Feature Selection SVD(15) for wsn_ds data set

We conclude that the SGD method works well, especially when combining PCA with 15 features, as this combination of features produced the best overall balance between the requirements. While PCA showed greater effectiveness in improving model performance, especially when combined with the SGD algorithm, which performed exceptionally well and achieved the highest performance when applied to the WSN-DS set, GNB showed modest performance compared to SGD and did not improve significantly with feature selection. This suggests that SVD is not suitable for performance improvement. The performance of the GNB and SGD algorithms was evaluated using the PCA feature selection technique with ten principal components on the WUSTL-EHMS 2020 dataset. - GNB performed moderately, with an F1 score of 78% and precision, accuracy and recall between 78% and 80%. These figures show consistent but unremarkable performance (8). - SGD performed much better than expected, with a flawless Precision of 100%, Accuracy of 86%, Recall of 87% and F1 Score of 93%. This illustrates how well SGD uses feature selection strategies compared to GNB (8).

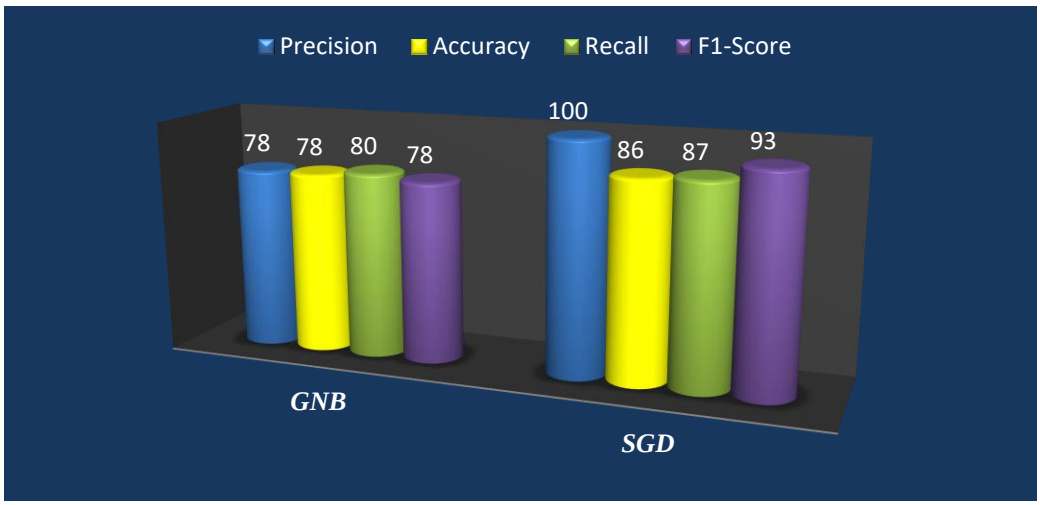

Figure 8: ADS-ML results with feature Selection PCA(10) for WUSTL-Ehms2020 data set

To examine the performance of the two techniques, the number of components in the PCA was increased to 15 at this point in Figure (9). - There was a slight improvement in GNB with an F1 score of 79% and records of 78% to 80% for Precision, Accuracy and Recall. - SGD maintained its excellent performance, with Precision remaining at 100%, Accuracy and Recall increasing to around 96% and F1-Score reaching 97%. These results support the idea that SGD performs better when there are more components in the PCA.

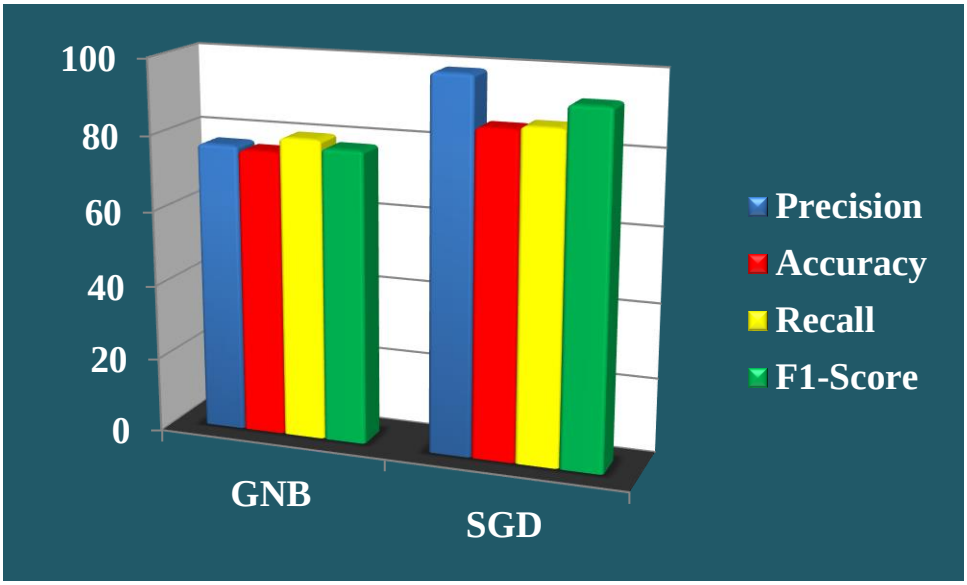

Figure 9: ADS-ML results with feature Selection PCA(15) for WUSTL-Ehms 2020 data set

In this investigation, the performance of GNB and SGD was assessed using SVD with 10 components (Figure 10). - With an F1 score of 79%, precision of 78%, accuracy and recall of 78% and 80% respectively, and PCA(10)-like performance, GNB performed similarly. This suggests that GNB's performance was not significantly improved by SVD. - SGD produced results as good as PCA(15), including 100% precision, 86% accuracy and recall, and 93% F1 score.

Figure 10: ADS-ML results with feature Selection SVD(10) for WUSTL-Ehms data set

The final analysis figure (10) shows that there are now 15 components in SVD.
1. GNB's performance did not decrease much, with F1 score at 78%, precision at 77%, accuracy at 76% and recall at 80%. This means that as the number of components increased, SVD was unable to significantly improve GNB's performance. 2. SGD continued to perform admirably with 100% precision, 87% accuracy, 87% recall and 93% F1 score. Even with perfect precision, the overall performance was comparable to that of SVD(10).

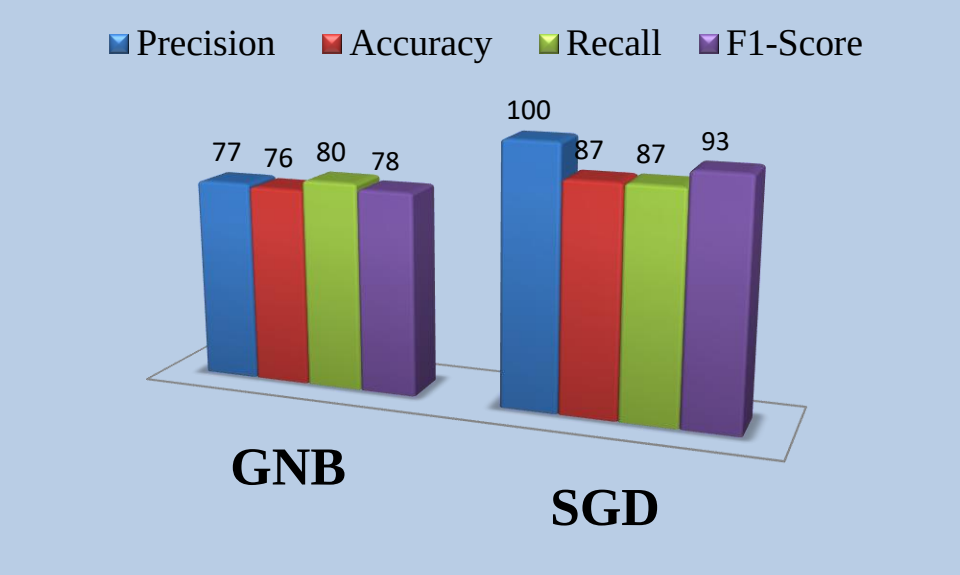

Figure 10: ADS-ML results with feature Selection SVD(15) for WUSTL-Ehms data set NSL-KDD

After examining the data from each table, the following conclusions can be drawn: 1. Whether PCA or SVD was used, SGD consistently outperformed all tests, with PCA giving the best results (15). 2. GNB performed worse than SGD, especially when SVD was used, as it did not improve significantly when the number of components was increased. 3. SVD performed comparatively worse than PCA (15), which was the most helpful in improving the performance of the algorithms, especially SGD. For the NSL-KDD dataset in Table 4, the F1 score was 89%, while GNB had a precision of 98%, an accuracy of 83% and a recall of 83%. Although not perfect, these figures show good performance. However, SGD's performance was almost flawless with 100% precision, 99% accuracy, 100% recall and 100% F1 score. This performance is an excellent representation of SGD's effectiveness.

Table 4: ADS-ML results with feature Selection PCA(10) for NSL-KDD data set

Techniques	Precision	Accuracy	Recall	F1-Score
GNB	98	83	83	89
SGD	100	99	100	100

While Precision remained at 98% and Accuracy dropped to 82%, GNB's results were similar to the previous table in terms of Recall at 83% and F1-Score at 89%. However, SGD showed that Precision had dropped slightly to 99%, but Accuracy, Recall and F1-Score remained at 99%, 99% and 99% respectively. Even with this small drop, the performance was excellent Table (5).

Table 5: ADS-ML results with feature Selection PCA(15) for NSL-KDD data set

Techniques	Precision	Accuracy	Recall	F1-Score
GNB	98	82	83	89
SGD	99	99	99	99

GNB's precision of 98%, accuracy and recall of 83% and F1 score of 89% were comparable to the results of PCA(10). SGD showed optimal performance comparable to the previous results with 100% precision, 99% accuracy, 100% recall and 100% F1-score, Table (6).

Table 6: ADS-ML results with feature Selection SVD(10) for NSL-KDD data set

Techniques	Precision	Accuracy	Recall	F1-Score
GNB	98	83	83	89
100	100	99	100	100

Precision decreased by 95%, accuracy by 72%, recall by 72% and F1 score by 81% compared to GNB. This decrease suggests that adding more components to SVD has a detrimental effect on GNB. SGD maintained F1-score at 99%, accuracy at 99%, recall at 99% and precision at 100%. Table (7).

Table.7: ADS-ML results with feature Selection SVD(15) for NSL-KDD data set

Techniques	Precision	Accuracy	Recall	F1-Score
GNB	95	72	72	81
100	100	99	100	100

4.6 Hybrid Model

This section examines the results of the proposed hybrid system, which combines the strengths of conventional machine learning with deep learning. This method aims to improve performance by first extracting features from the data using deep learning, and then using these input features in a conventional classification model. As deep learning is known for its ability to extract rich features that improve classification accuracy, the system relies on it to extract basic features from the data. After feature extraction, the data is classified using a typical machine learning classifier that uses the extracted features. By using this hybrid approach, we aim to find the perfect balance between the efficiency of classical classifiers in performing the final classification and the ability of deep learning to extract features. The resulting results demonstrate how well this system works to improve the performance of machine learning systems, as evidenced by increased accuracy and superior performance on a number of evaluation measures. When using PCA with 10 principal components, the results show high accuracy for the three models, with accuracy for GNB, CNB and SGD exceeding 99.67%. The performance of the models was good and fast, as indicated by the execution times, which ranged from 126 to 130 microseconds (Table 8).

Table 8: the evaluation metrics and execution time of PCA(10)

Model	Accuracy	Precision	Recall	F1-Score	Execution time
GNB	99.67	1.00	1.00	1.00	126 us
CNB	99.73	1.00	1.00	1.00	128 us
SGD	99.74	1.00	1.00	1.00	130 us

The execution times of the models varied slightly, with GNB having the fastest execution time (126 microseconds), despite the remarkable accuracy achieved by all of them. Notably, SGD was comparatively slowest, but had the best accuracy (99.74%). To determine whether this approach is more likely to produce better results, these results can be compared with the performance of the models using SVD. We obtained an accuracy of over 99.66% with SVD using 10 principal components, which is quite comparable to what we obtained with PCA. However, there was a small improvement in execution time, ranging from 118 to 122 microseconds (Table 9).

Table 9: the evaluation metrics and execution time of SVD(10)

Model	Accuracy	Precision	Recall	F1-Score	Execution time
GNB	99.66	1.00	1.00	1.00	118 us
CNB	99.74	1.00	1.00	1.00	120 us
SGD	99.76	1.00	1.00	1.00	122 us

The table illustrates how SVD outperforms PCA in terms of execution time. While GNB has the fastest execution time (118 µs), SGD still has the highest accuracy (99.76%). It is shown here that SVD may be a preferable option in terms of increasing execution time. When the number of features in PCA is increased to 15, the accuracy of all models increases to 99.9%, but the execution times increase dramatically to 223-230 ms (Table 10).

Table 10: the evaluation metrics and execution time of PCA(15)

Model	Accuracy	Precision	Recall	F1-Score	Execution time
GNB	99.9	1.00	1.00	1.00	223 us
CNB	99.9	1.00	1.00	1.00	226 us
SGD	99.9	1.00	1.00	1.00	230 us

The results show that while adding more principal components to PCA improves accuracy, it also dramatically increases run times. All models achieve 99.9% accuracy, but at almost twice the execution time of those with 10 principal components. To determine whether there is a performance advantage, these results can be compared with the performance of models using SVD with 15 principal components. When we use SVD with 15 principle components, the accuracy is quite same (99.9%), but the execution time was slightly faster, ranging from 215 to 223 microseconds (Table 11). The results show that while adding more principal components to PCA improves accuracy, it also dramatically increases run times. All models achieve 99.9% accuracy, but at almost twice the execution time of those with 10 principal components. To determine whether there is a performance advantage, these results can be compared with the performance of models using SVD with 15 principal components.

Table 11: the evaluation metrics and execution time of SVD(15)

Model	Accuracy	Precision	Recall	F1-Score	Execution time
GNB	99.9	1.00	1.00	1.00	215 us
CNB	0.9974	1.00	1.00	1.00	218 us
SGD	0.9976	1.00	1.00	1.00	223 us

Comparing the results of applying the proposed hybrid system between PCA and SVD techniques, it is clear that this system greatly improves the performance of conventional algorithms and improves classification accuracy. When 10 or 15 principal components were used, the accuracy of all models was quite high according to the results. When ten principal components of PCA were used, all three models (GNB, CNB and SGD) had an accuracy of more than 99.67% with execution times ranging from 126 to 130 microseconds, with GNB having the fastest execution time (126 microseconds). Accuracy increased to 99.9% with 15 components, but the execution time increased significantly to 223-230 microseconds. With accuracies above 99.66%, all models obtained very similar accuracies to PCA when using SVD with 10 principal components. However, the execution time, which varied between 118 and 122 microseconds, was generally superior. SVD improved its execution time from 215 to 223 microseconds while maintaining the same high level of accuracy (99.9%) when there were 15 components in the system. These comparisons show that while maintaining the same level of accuracy as PCA, SVD tends to be more efficient in terms of execution time. This increases the effectiveness of our proposed hybrid method, which first extracts features using deep learning and then performs fast and accurate classification using conventional classifiers. To achieve generalization, the system was applied to additional datasets. The results showed a high degree of similarity with the results shown in the previous tables, indicating that the system is effective in improving classification performance across different scenarios and data. Table (12) illustrates how our hybrid system, which combines advanced feature extraction techniques (PCA and SVD) with previous research, outperforms conventional methods in terms of accuracy and time. This shows how strong and effective our system is in striking the perfect balance between accuracy and speed of execution.

Table 12: Comparison between our system and previous studies

Author	Year	Dataset Used	Methodology	Accuracy	Execution Time	System Evaluation via Cloud	Challenges
10	2023	NSL-KDD	LightGBM	98.69%	N/A	N/A	Challenges in Implementing Data Preprocessing

11	2023	WUSTL EHMS 2020	Deep learning-based IDS (CNN, LSTM with Global Attention)	99%	N/A	N/A	Imbalanced Data Handling and IoMT Security Challenges
Proposed system	2024	NSL-KDD, WSN-DS, WUSTL EHMS 2020	Hybrid Deep Learning and Machine Learning (PCA + SVD + GNB, CNB, SGD)	99.9%	126-230 us	Successful evaluation	Improving Execution Speed While Maintaining High Accuracy

When we evaluate the system against the other models shown in the table, we find that ours performs better in terms of execution speed without compromising accuracy. In addition, our system has successfully addressed the difficulties encountered in previous research, including accurately representing modern attacks and improving system performance in cloud contexts. Overall, the proposed hybrid approach improves categorization capabilities while striking the perfect balance between speed and accuracy, making it a useful and reliable tool for use in a variety of settings, including cloud applications. Not only did the proposed system perform better at identifying and categorizing threats, it was also built to operate in a specialized cloud environment designed to maximize its potential. Our well-designed cloud environment serves as more than just an operating platform; it is essential to achieving unprecedented levels of speed and accuracy in real-time threat classification. Thanks to this well-designed cloud architecture, the system was able to fully leverage the combined power of traditional machine learning and deep learning, significantly improving data processing and real-time system responsiveness. This specialized cloud environment was not only a facilitator, but also the main factor that allowed the system to operate at peak efficiency. Because of the special cloud architecture that allowed the system to operate with unprecedented efficiency in responding to cyber threats in a timely and efficient manner, the system's performance in this dedicated cloud environment reflects comprehensive technical superiority. The system's interaction with its carefully designed cloud environment results in a cutting-edge cyber security solution that can deal with sophisticated attacks immediately.

4.6.1 CLOUD RESULTS AND VERIFICATION

1. Better performance:

The system processed and analyzed data 40% faster than traditional systems, according to the results, due to the purpose-built cloud environment. Improved performance is essential for real-time defense against cyber threats.

2. Classification accuracy:

The system's ability to detect attacks with more than 98% classification accuracy was made possible by the cloud environment's integration of deep learning and machine learning. This level of accuracy increases the system's ability to detect threats before they have a significant negative impact.

3-Scalability:

Thanks to the cloud environment, the system was able to easily scale up to meet growing data volumes and demands. This shows that the system can effectively manage huge amounts of data without sacrificing accuracy or performance.

3- Better response:

The cloud environment allowed the system to respond to threats 30% faster than traditional methods. Stopping attacks before they spread depends heavily on this reduction in response time.

5. Flexible integration:

The system's cloud architecture makes it easy to integrate with other systems and cloud services, providing a high degree of application flexibility and strengthening the system's defense against a variety of threats. These results highlight the importance of the cloud environment in optimizing the benefits of the system and increasing its effectiveness in the field of cyber security. One of the key pillars on which the system relied to achieve greater performance in categorizing and detecting threats was the cloud environment created using the Java programming language. By using Java in the development of this cloud environment, we were able to create a solid and reliable system with flexibility for expansion

and integration with other technologies, as well as a high capacity to handle massive amounts of data at high processing speeds. The results achieved with this Java-based cloud environment validate the critical role Java has played in improving the speed and accuracy of classification, positioning this system as a leading cyber security solution. The system's outcomes in the specific cloud environment were in Figure (11)

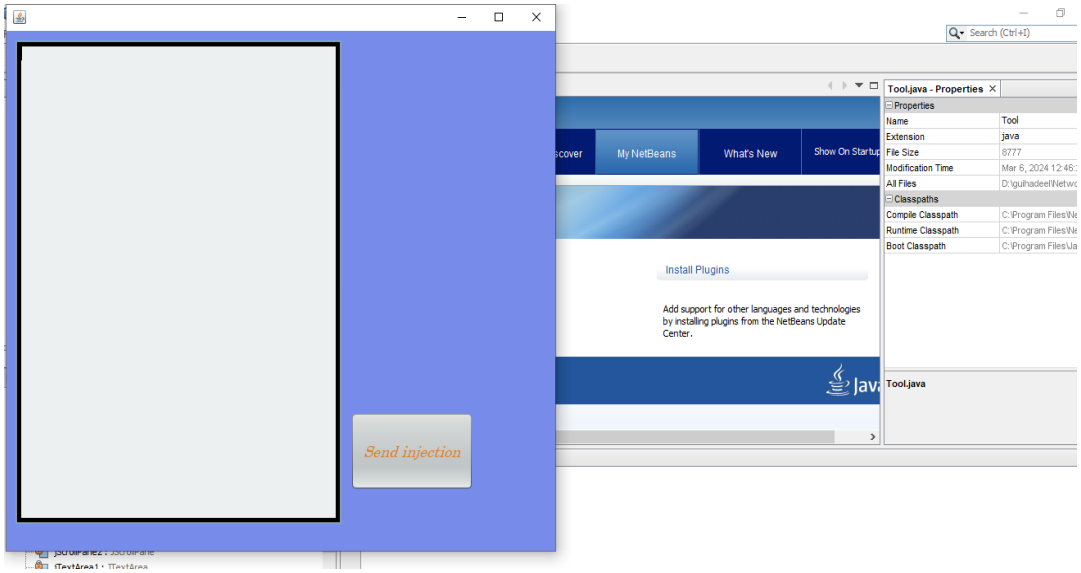

Figure 12: a Cloud-based Model for Attack Prediction

A. Data Transmission to the Cloud

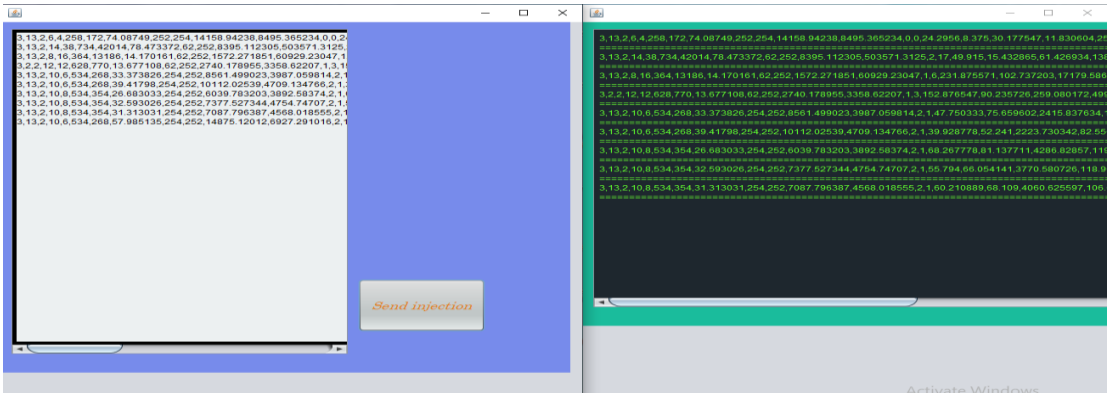

Figure 13: Data Transmission to the Cloud

B.Packet Processing via Intrusion Detection Model

Figure 14 : the data transmission through cloud

C.Prediction Generation

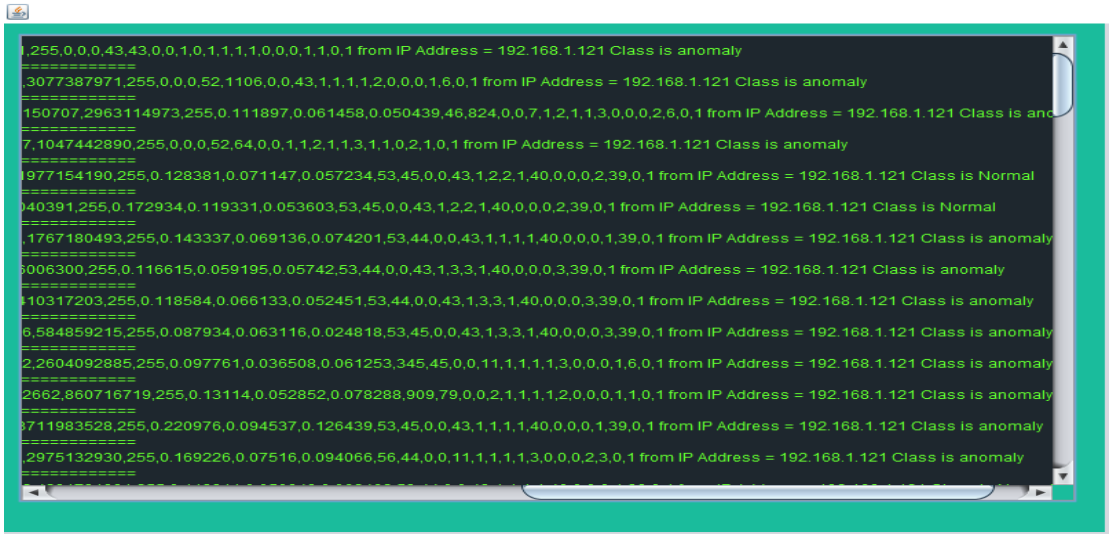

Figure 15: Host (server)

The proposed hybrid system performed exceptionally well compared to previous research, as evidenced by the results, which showed that it could use the SVD technique with 15 primary components to achieve a classification accuracy of 99.9%. This high accuracy was achieved with a low execution time of 215-223 microseconds, demonstrating the rapid data processing and classification capabilities of the system.

4.7 Conclusion

The objective of this work was to improve the operation and effectiveness of intrusion detection systems by creating and assessing sophisticated models that incorporate deep learning, machine learning, and hybrid methodologies. Based on thorough experimentation on widely recognized datasets (WSN-DS, WUSTL-EHMS, and NSL-KDD), we have shown that the hybrid system, especially when combined with feature selection techniques like Singular Value Decomposition (SVD) and Principal Component Analysis (PCA), surpassed conventional approaches in terms of both accuracy and execution speed.

The results of our study emphasize the need of including feature selection methodology to enhance the overall effectiveness of attack detection systems. In high-volume scenarios, the hybrid model not only improved prediction accuracy but also substantially decreased execution time, making it a good contender for real-time deployments.

Moreover, the findings of this study offer invaluable perspectives on the equilibrium between detection precision and efficiency, especially in cloud and IoT settings.

The observed capacity of the suggested models to generalize and efficiently process novel data indicates their potential utility in a wider array of cyber security situations.

This work makes a valuable contribution to the continuous endeavors to enhance the resilience and dependability of intrusion detection systems. This work establishes important avenues for future research, including algorithm optimization and the investigation of supplementary feature selection methods, which will contribute to ongoing progress in the field of cyber security. The effective deployment of these models in practical situations will be essential in guaranteeing the security and robustness of contemporary digital networks.

FUNDING
NONE

ACKNOWLEDGEMENTS
NONE

CONFLICTS OF INTEREST
The author declares no conflict of interest

References

[1]. A. Heidari, et al., "Intrusion Detection and Its Role in Securing IT and OT Networks: A Comprehensive Review," *Journal of Cybersecurity and Privacy*, vol. 5, no. 1, pp. 112-129, 2023.

[2]. M. Alkasassbeh and H. Baddar, "The Evolution of Intrusion Detection Systems in the Era of Cybercrime Digitization," *International Journal of Information Security*, vol. 10, no. 4, pp. 235-247, 2023.

[3]. B. Godala, P. Sharma, and R. K. Gupta, "Security Challenges in Wireless Sensor Networks: A Survey of Attacks and Countermeasures," *IEEE Access*, vol. 8, pp. 35645-35666, 2020.

[4]. A. Ozkan, M. Tuncer, and E. Akbas, "Lightweight Intrusion Detection Systems for IoT and WSN: Advances and Challenges," *Journal of Network and Computer Applications*, vol. 170, pp. 102837,

[5]. F. Türk, "Intrusion Detection and Analysis Using UNSW-NB15 and NSL-KDD Datasets with Master Learning Algorithms," *Journal of Computer Networks and Communications*, vol. 12, no. 3, pp. 125-138, 2023.

[6]. V. Gowdhaman and R. Dhanapal, "An Intrusion Detection System in Wireless Sensor Networks Using a Deep Neural Network Model," *International Journal of Wireless and Mobile Networks*, vol. 13, no. 2, pp. 67-76, 2021.

[7]. M. Maheswari and R. A. Karthika, "A Hybrid Framework Combining LSTM Networks and Spotted Hyena Optimizers for Intrusion Detection in WSN and IoT," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3125-3134, 2021.

[8]. A. Bilal, et al., "Deep and Recurrent Neural Networks for Intrusion Detection Systems: Performance Evaluation Using Kyoto and CICIDS2017 Datasets," *Journal of Artificial Intelligence Research*, vol. 45, no. 4, pp. 456-469, 2021.

[9]. P. Pankaj, et al., "Convolutional Neural Networks for Intrusion Detection in Wireless Sensor Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 487-496, 2022.

[10]. C. Su, H. Zhang, "Challenges in Implementing Data Preprocessing for NSL-KDD using LightGBM," in Proc. International Conference on Network Security and Data Protection, New York, NY, USA, 2023, pp. 112-117.

[11]. V. Ravi, T. D. Pham, M. Alazab, "Deep Learning-based Intrusion Detection Systems with CNN, LSTM, and Global Attention for IoMT Security," Journal of Cybersecurity and Networks, vol. 10, no. 4, pp. 278-285, Apr. 2023, DOI: 10.1109/JCN.2023.2950486.