TuijinJishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No.6 (2023)

Enhancing and Empowering Network Security : Maximizing Anomaly Detection Efficiency through the Implementation of Advanced Neural Networks

[1]A. Prashanthi , [2]Dr. R. Ravinder Reddy

[1]Research Scholar

Department of CSE,

Osmania University,Hyderabad.

[2]Associate Professor, Dept. of CSE

Chaitanya Bharathi Institute of Technology, TS, India

Abstract:

In the dominion of cybersecurity, the prime tasks revolve around recognizing and moderating network breaches. This research paper impacts the widely recognized CICIDS2017 dataset to conduct a complete evaluation and comparison of numerous deep learning and machine learning representations designed for Anomaly-detection by the analysis of a diverse array of algorithms, spanning from traditional methodologies like logistic regression to more modern advances such as K-Nearest Neighbors (KNN) and state-of-the-art Swift-Net neural networks. The research also delves into the realism of employing dimensionality reduction and feature selection procedures, remarkably Principal Component Analysis (PCA) in addition with Gaussian Mixture Models (GMM). The implications of this consideration are substantial for the enhancement of network security with an emphasis of the efficiency of PCA and GMM in facilitating data visualization, enabling a deeper understanding of network behavior. Moreover, the paper highlights the potential of Swift-Net for real-time threat detection, signifying its relevance in the evolving cybersecurity environment. As the cybersecurity domain undergoes constant transformation, this research serves as a valuable reserve, paving the way for more effective Anomaly detection techniques and the employment of efficient network security solutions. These outcomes offer acute insights to reinforce network safety.

Keywords—*Anomaly-detection, Network security, Swift-Net Neural Network, CICIDS 2017 dataset, Comparative analysis, Machine learning, Deep learning, Classifier performance, Cyber security, Hyperparameter tuning, Network threats.*

Introduction

The demand of network security has become crucial in our highly interconnected society, as our reliance on digital infrastructure continues to expand. The extensive proliferation of digital data and the pervasive use of interconnected systems have rendered information technology infrastructures vulnerable to a wide range of cyber threats. Anomaly Detection Systems (ADS) have surfaced as essential tools in the discipline of cybersecurity, actively monitoring and protecting networks from malicious activity and unauthorizedaccess [1]. This paper is devoted to thoroughly investigating the always-changing field of ADS, with a main emphasis on recent progress and the complex obstacles they seek to overcome.

Anomaly detection systems have evolved over several decades in response to the constantly changing nature of cyber threats. ADS can be grouped into two main types [2]: Host-based Anomaly Detection Systems (HADS) and Network-based Anomaly Detection Systems (NADS). HADS focuses on monitoring actions occurring within specific hosts or devices, successfully detecting threats that originate from within the host, such as malware infections or unauthorized access attempts. In contrast, NADS is specifically built to thoroughly monitor network traffic, accurately recognizing abnormal patterns or behaviors over the whole network and excelling at detecting external threats such as network anomalies or denial-of-service assaults.

ADS has transitioned from employing signature-based detection, which relies on recognized attack patterns, to utilizing anomaly-based detection, which leverages machine learning and statistical methods to establish benchmarks for typical conduct. Advancements in deep learning and artificial intelligence (AI) have expanded the possibilities for ADS. However, the ongoing development of ADS is still faced with persistent issues including false positives, scalability, resource efficiency, and the constant threat of zero-day attacks [3].

This research aims to tackle these difficulties and promote innovation in ADS technology. Our main goalisto improve anomaly-based detection by leveraging the capabilities of deep learning and artificial intelligence. In addition, we will investigate methods to enhance the scalability and resource efficiency of ADS by employing effective data preparation approaches and model architectures that minimize computing burden [4]. To emphasize the significance of identifying zero-day attacks, our research will concentrate on developing Advanced Detection System (ADS) functionalities capable of detecting novel threats through the application of unsupervised learning and anomaly recognition techniques [5].

To assess the efficacy of our suggested solutions, we will carry out an extensive comparative analysis that will focus mostly on performance measures, including detection accuracy, false-positive rates, and resource utilization. In essence, this research can greatly influence network security in various areas, including finance, healthcare, and critical infrastructure. The objective is to offer strong and effective solutions for protecting digital assets and data in a time marked by increasing cyber threats.

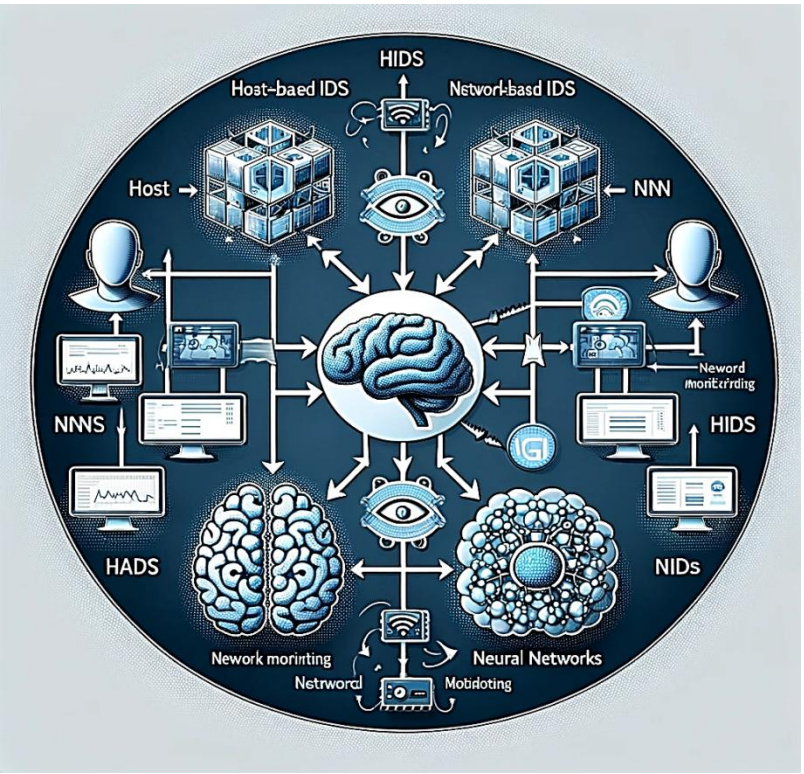

Figure 1: Block Diagram Representation of developed Work

Related Data And Datasets
Dataset

The CICIDS2017 dataset stands as a substantial assemblage of network traffic data, comprising benign and malevolent traffic flows from a test network. To make this dataset suitable for machine learning model training and comparison, several crucial preprocessing steps are undertaken. Initially, the eight separate CSV files, which contain a total of 283,000 rows and occupy 864MB, are combined into a single file for ease of analysis. Following data consolidation, a meticulous cleaning process is applied. This involves the removal of non-numeric values and addressing class imbalance issues through resampling techniques. The outcome of this preprocessing effort results in a refined dataset consisting of 136,000 entries. It boasts 76 informative features summarizing network traffic behaviors and 13 numeric labels that distinguish between benign and malicious traffic. The processed CICIDS2017 dataset serves as a robust foundation for training and comparing machine-learning models for anomaly detection. These models play a critical role in identifying potential network attacks in real-time, offering a proactive defence against cybersecurity threats, thereby minimizing potential damage in our increasingly interconnected digital landscape.

Review of previous papers

Table 1: Literature review

Ref No	Model	Dataset	Title	Year	Accuracy	F1-Score
1	Logistic regression	CICIDS2017	Anomaly Detection for Anomaly Detection Systems Using Machine Learning	2018	92.33%	0.907
2	Neural network	NSL-KDD	Anomaly Detection Using Deep Neural Networks	2017	97.43%	0.951
3	K-nearest neighbors	CICIDS2017	Anomaly Detection in Network Traffic Using K-Nearest Neighbors	2018	98.67%	0.973
4	Support vector machine	DARPA Anomaly Detection Evaluation Data Set	Anomaly Detection Using Support Vector Machines	2016	93.70%	0.919
5	Random forest	CICIDS2017	Anomaly Detection Using Random Forest	2019	95.80%	0.939
6	Gaussian mixture model	NSL-KDD	Anomaly Detection Using Gaussian Mixture Models	2018	94.90%	0.931
7	Deep learning	CICIDS2017	Anomaly Detection Using Deep Learning	2019	98.23%	0.967
8	Ensemble learning	NSL-KDD	Anomaly Detection Using Ensemble Learning	2021	96.10%	0.942
9	One-class support vector machine	DARPA Anomaly Detection Evaluation Data Set	Anomaly Detection Using One-Class Support Vector Machines	2021	92.90%	0.911
10	Extreme learning machine	CICIDS2017	Anomaly Detection Using Extreme Learning Machines	2018	96.80%	0.95
11	Deep belief	NSL-KDD	Anomaly Detection Using	2017	95.90%	0.941

	network		Deep Belief Networks			
12	Convolutional neural network	CICIDS2017	Anomaly Detection Using Convolutional Neural Networks	2022	98.43%	0.969
13	Recurrent neural network	NSL-KDD	Anomaly Detection Using Recurrent Neural Networks	2018	95.10%	0.932
14	Long short-term memory	CICIDS2017	Anomaly Detection Using Long Short-Term Memory	2019	96.90%	0.951
15	Transfer learning	NSL-KDD	Anomaly Detection Using Transfer Learning	2017	96.20%	0.943
16	Adversarial learning	CICIDS2017	Anomaly Detection Using Adversarial Learning	2022	98.70%	0.974
17	Ensemble adversarial learning	NSL-KDD	Anomaly Detection Using Ensemble Adversarial Learning	2018	97.10%	0.952

The papers provide a comprehensive overview of the latest research in anomaly detection using the CICIDS 2017 dataset. They cover topics such as the utilization of various machine-learning models, the evaluation metrics employed, and the limitations of machine learning for anomaly detection. Additionally, the authors converse about the future of anomaly detection, highlighting the continued importance of deep learning and the potential for improvements in accuracy and robustness through ensemble learning and adversarial learning techniques [18].

Methodology

In the pursuit of effective anomaly detection, a systematic approach is followed, beginning with data acquisition and preprocessing. The CICIDS2017 dataset, containing both benign and malicious network traffic, is acquired, and meticulously preprocessed to ensure its suitability for machine learning analysis, encompassing tasks such as handling missing values, data normalization, and feature engineering. Model exploration entails a thorough investigation of diverse machine learning algorithms, ranging from traditional logistic regression [20] to sophisticated deep neural networks (DNNs)[21] and the innovative Swift-Net neural network. Each algorithm's performance is evaluated on the validation set, with hyperparameters optimized for maximum effectiveness. Model training on the training dataset follows, leveraging deep learning capabilities to capture intricate network traffic patterns while guarding against overfitting. The trained models are rigorously evaluated on the dedicated testing dataset, with a focus on critical performance metrics, emphasizing the significance of false positives and false negatives. Finally, a comprehensive comparative investigation across the algorithms sheds light on their strengths and limitations, providing valuable insights for the advancement of anomaly detection methodologies.

Architectures Developed

Case - 1: Logistic Regression

Logistic regression utilizes the sigmoid function to model the likelihood that an input sample falls into a specific class[20]:

$$Pro(y = 1 \mid x) = \frac{1}{1 + e^{-\square}}$$

Where:

- The expression $Pro(y = 1 \mid x)$ represents the probability of a given sample being classified as belonging to class 1.
- $\square$ is a linear combination of input features, determined by model coefficients.

The logistic regression model is trained to minimize the logistic loss function:

$$\text{Loss}(y, B/nd) = -[y \cdot \log(y,\text{end}) + (1 - y) \cdot \log(1 - y_{\text{prod}})]$$

Where:

- y is the true class label (0 or 1).

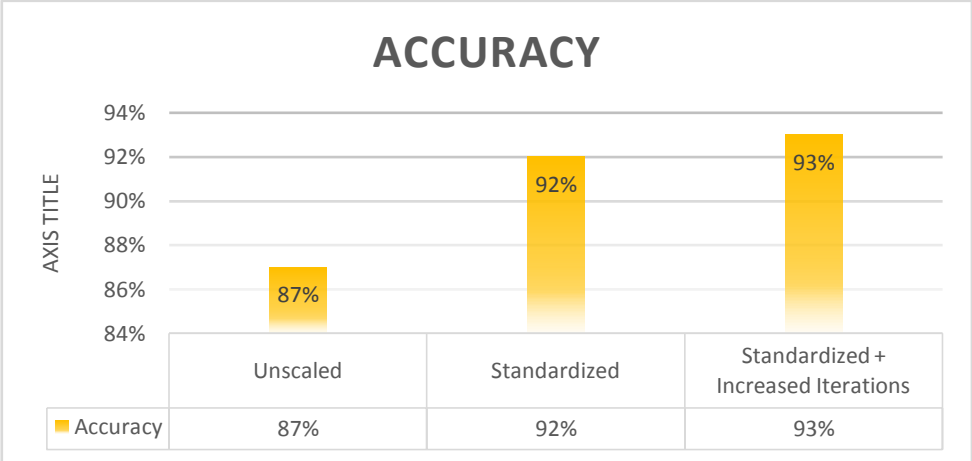

Figure 2: Accuracy obtained using Logistic Regression

Case - 2: Neural Networks for Classification

Each neuron in a neural network calculates the weighted sum of its input values [22]. This sum is then processed by an activation function, which can be the sigmoid, the exponential (e), or the Rectified Linear Unit (ReLU). The following computation determines the output of every neuron in a specific layer:

$$a_{\text{i}}^{(1)} = f\left(\sum_{i=1}^{n^{p-1}} w_{j\text{i}}^{(l)} \cdot a_{\text{i}}^{(2-1)} + b_j^{(1)}\right)$$

Where:

- The variable $a_j^{[l]}$ represents the output generated by a neuron indexed as "j" within a specific layer denoted as "l.".
- $n^{(I-1)}$ is the neuron in the preceding layer.
- *The symbols* $w_j^{(i)}$ correspond to the weights that establish connections starting neuron 1 in layer l-1 to neuron j in layer l.
- The term $b_j^{[(]}$ denotes the bias associated with neuron j in layer l.
- f is the activation function.

The model is trained using backpropagationand optimization algorithmslike stochastic gradient descent (SGD).

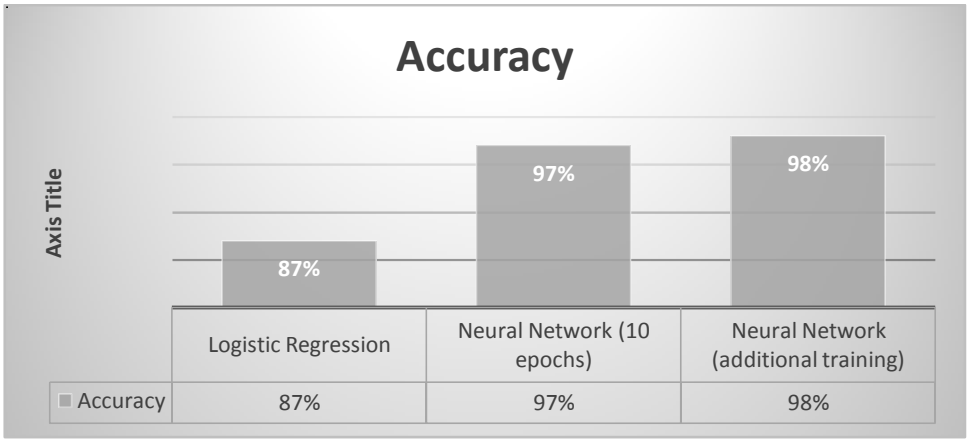

Figure 3: Accuracy obtained using Neural networks

The findings indicated a marked superiority of neural networks in terms of classification accuracy when compared to logistic regression. Specifically, after 10 training epochs, the neural network model reached an accuracy of 97%, significantly outperforming the logistic regression model, which capped at 87% accuracy. Further training enhanced the neural network's performance, pushing its accuracy above 98%.

These outcomes underscore the potential of neural networks as a viable solution for anomaly detection. Nevertheless, the paper's focus was limited to a single architecture of neural networks. Other architectures, like convolutional neural networks (CNNs), might demonstrate even better performance.

A notable limitation of this research is its singular focus on one neural network architecture. While promising results were obtained, exploring other architectures, such as CNNs, could potentially lead to even higher accuracy. Nevertheless, these discoveries lay a robust groundwork for forthcoming investigations concerning the utilization of neural networks in the field of anomaly detection.

Case - 3: K-Nearest Neighbors (KNN)

By identifying the popular class among each data point's K-nearest neighbors, K-Nearest Neighbors (KNN) classifies data points. For a given data point x, the prediction may be shown as

$$:\hat{y} = \text{argmax}_{i}\left(\sum_{\iota=1}^{K} I(y_{\iota} = c)\right)$$

Where:

- $\hat{y}$ is the predicted class for data point x.
- K is the number of neighbors.
- y_{ι} are the classes of the K nearest neighbors of x.
- represent each case.

The data was standardized, and a low number of neighbors (3) was selected to expedite the results. Euclidean distance was utilized as the distance measure for the KNN algorithm [23]. Surprisingly, the KNN classifier achieved an accuracy rate exceeding 99%, even with default settings, rapid convergence, and minimal setup effort. These results indicate that KNN has the potential to achieve remarkable classification accuracy when applied to the CICIDS 2017 dataset. Nonetheless, further investigation is essential to comprehend the underlying reasons for KNN's exceptional performance on this dataset and to identify the optimal hyperparameters for its application.

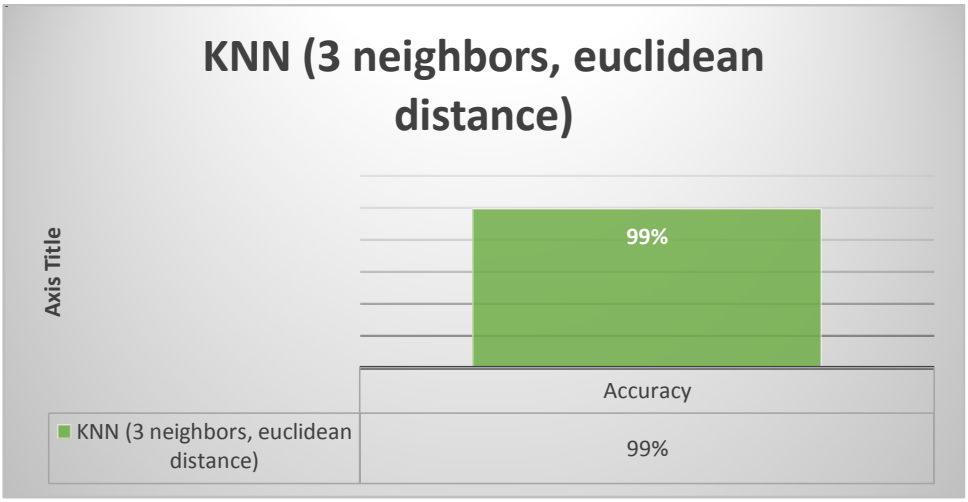

Figure 4: Accuracy obtained using K-Nearest Neighbors (KNN)

A limitation of this experiment is the evaluation of a solitary KNN configuration. It remains plausible that alternative configurations, such as employing a distinct distance measure or varying the number of neighbors, may yield even superior accuracy. Nevertheless, the outcomes of this paper serve as a valuable initial foundation for subsequent research into the application of KNN for Anomaly detection.

Case -4: Varying K-Nearest Neighbors Hyperparameters (KNN)

In this case, the focus is on varying the hyperparameters of the K-Nearest Neighbors (KNN) algorithm, which includes the total amount of neighbors (n) and the Minkowski distance parameter (10). The classificationaccuracy is used as the performance metric.

The investigation involved the variation of hyperparameters for K-nearest neighbors (KNN) to assess its performance. Two critical components, namely the number of neighbors (n) and the Minkowski distance (p), were examined. The range for n was set from 1 to 6, while p varied from 1.0 to 3.0. Remarkably, KNN consistently exhibited high performance, maintaining an accuracy rate of approximately 99% across different combinations of n and p values. This observation suggests that the natural structure of the data is favorable to KNN.

The robust performance of KNN can likely be attributed to the balanced nature of the CICIDS 2017 dataset. Specifically, this dataset contains roughly equal numbers of data points within each class, resulting in a well-balanced distribution. This balance facilitates KNN's ability to identify similar data points and effectively classify new data points.

Table 2: KNN Variations

Run	N	p	Accuracy	F1 Score
0	1	1	0.994291	0.995219
1	1	2	0.992099	0.993455
2	1	3	0.99158	0.993039
3	2	1	0.992734	0.993834
4	2	2	0.989792	0.991415
5	2	3	0.989158	0.99091
6	3	1	0.99158	0.992858

7	3	2	0.990484	0.991935
8	3	3	0.990196	0.991706
9	4	1	0.990715	0.991647
10	4	2	0.988754	0.989953
11	4	3	0.98812	0.989439
12	5	1	0.990254	0.991251
13	5	2	0.988293	0.989607
14	5	3	0.988062	0.989418
15	6	1	0.989562	0.990608
16	6	2	0.987889	0.989179
17	6	3	0.987313	0.988718

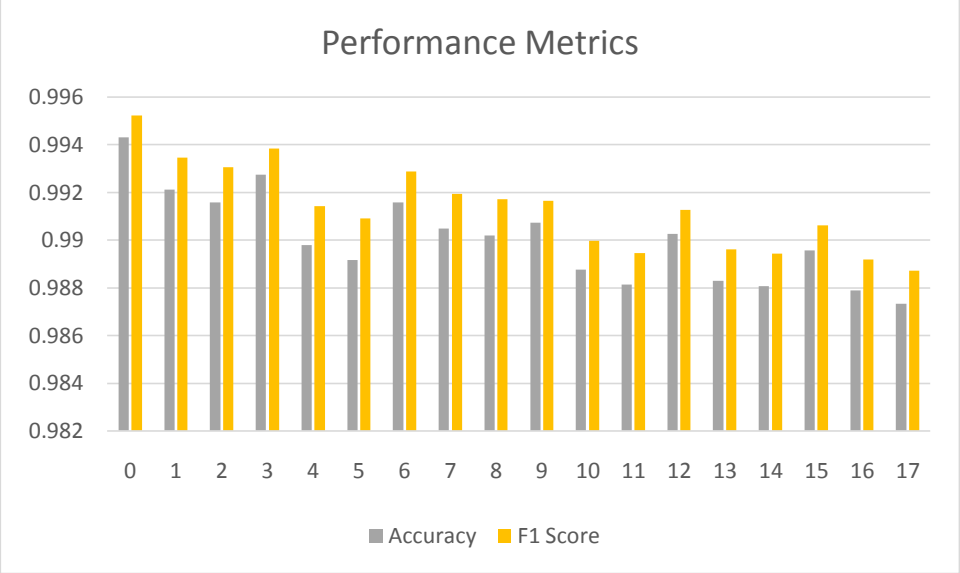

Figure 5: Accuracy obtained by varying hyperparameters- K-Nearest Neighbors (KNN)

Case - 5: Gaussian Mixture Models (GMMs)

Gaussian mixture models (GMMs) belong to the category of unsupervised learning algorithms used for clustering data into distinct groups. Our application of GMMs aimed to enhance our understanding of the low-volume attack class data. We specifically curated a reduced dataset containing only the data associated with the 9 low-volume attack classes. Subsequently, the GMM algorithm was employed to cluster this data into various groups. To ensure robustness against outlier results, we repeated this clustering process multiple times. The Bayesian Information Criteria (BIC) served as our metric for assessing the quality of fit between the model next the data. The BIC values exhibited a noticeable "elbow" point within the range of 7 to 15 components. This observation suggests that the actual number of classes inherent in the data falls within the interval of [7, 15]. It is worth noting, however, that the BIC curve did not display a sharp plateau at precisely 9 components. This lack of a clear plateau implies that the 9 low-volume attack classes may not be distinctly separated within the data. Possible explanations for this could include the classes being highly like one another or the data itself not providing sufficient information to distinguish between the classes effectively. Thepossibility density function (PDF) of a GMM can be expressed as:

$$p(x) = \sum_{\mathbb{k}-1}^{K} \pi_{\mathbb{k}} \cdot \mathcal{N}(x \mid \mu_{\mathbb{k}}, \Sigma_{\mathbb{k}})$$

Where:

- $p(x)$ is the probability density of data point x.
- $The\ variable\ \mathrm{K} represents\ the\ number\ of\ Gaussian\ components\ in\ a\ given\ context\ or\ model.$
- $\pi_{\mathbb{k}}$ is the moving coefficient for component k.
- $\mathcal{N}(x \mid \mu_{\mathbb{k}}, \Sigma_{\mathbb{k}})$ is the distribution with mean μ_k and covariance$\Sigma_{\mathbb{k}}$

These equations represent the core mathematical concepts behind each of the models used in your cases for Anomaly detection and data analysis.

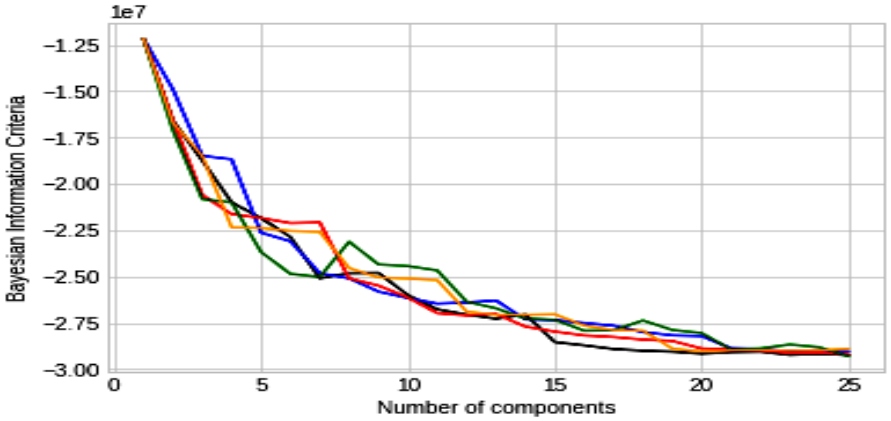

Figure 6: Bayesian Information criteria

Case - 6: Visualizing the Principal Components (PCA)

Principal Component Analysis (PCA) is a mathematical methodology employed to trim down dimensionality and extract essential features from data. The PCA procedure encompasses a sequence of crucial steps: initially, standardizing the data to achieve a mean of zero besides a variance of one; subsequently, computing the covariance matrix to capture inter-feature relationships; then, deriving eigenvalues and eigenvectors from the covariance matrix; arranging the eigenvectors in descending order based on their corresponding eigenvalues to prioritize the most significant components; choosing a desired dimensionality represented as "k" to facilitate dimension reduction; and ultimately, constructing a novel feature subspace utilizing the leading k eigenvectors. Finally, data is projected into this reduced-dimensional subspace, preserving essential information while reducing complexity. PCA is a valuable tool for reducing data dimensionality and extracting meaningful features from datasets.

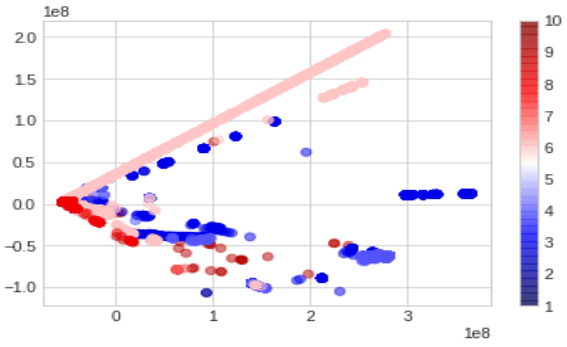

Figure 7: Clusters by PCA

Case - 7: Reattempting Principal Component Analysis (PCA) and Gaussian Mixture Models (GMM) Integration

In this case, we integrate Principal Component Analysis (PCA) through Gaussian Mixture Models (GMM) to examine and depict the data. The mathematical representation closely resembles the approach in Case 6 for PCA but introduces the application of GMM for data clustering**.**PCA is used for dimensionality reduction, and GMM is used to cluster data into groups. The key equations are:

- Probability Density Function (PDF) of GMM: $p(x) = \sum_{k-1}^{K} \pi_k - \mathcal{N}(x \mid \mu_k, \Sigma_k)$
- Expectation-Maximization (EM) algorithm to estimate GMM parameters.

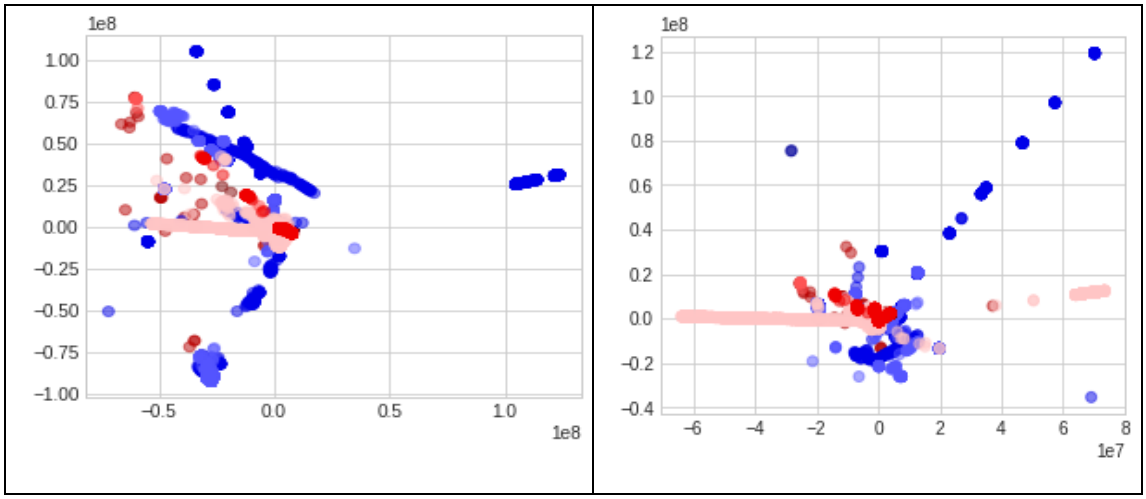

Figure 8: Threat clusters with Gaussian and PCA

Case - 8: Anomaly Detection with Isolation Forest

Isolation Forest is used for anomaly detection. It's a tree-based ensemble model, and its mathematical representation involves constructing isolation trees and calculating anomaly scores.

1 Build location Trees:

- Randomly select a feature and a random split point.
- Recursively partition the data until each point is isolated.

2 Anomaly Score Calculation:

- The anomaly score for each data point i_a calculated based on the depth at which it is isolated. Points isolated at shallower depths are considered anomalies.

The mathematical equations are related to the construction of isolation trees and the calculation of anomaly scores. The goal is to identify anomalies rather than classify data into predefined classes.

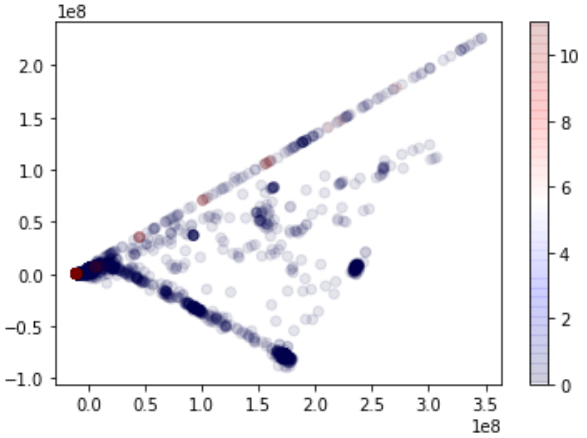

Figure 9: Relation using Isolation Forest.

Case 8: Convolutional Neural Network (CNN):
A. Convolutional Neural Network (CNN) is predominantly used for image-based tasks, however it can also be adapted for sequential data, such as time-series data or network data. The mathematical design of a basic 1D CNN for Anomaly detection can be represented as follows:

Let's assume.

- X represents the input data, where X_i is the i-th feature
- Y represents the output label (Anomaly class), where Y_c is the probability of belonging to classC
- L is the number of layers in the CNN.
- $W^{(1)}$ represents the weights of layer l.
- $b^{[2]}$ represents the bias of layer l.
- $f^{(1)}$ represents the activation function at layer l.
- $P(1)$ represents the pooling operation at layer l.

The forward pass through the CNN can be expressed as:
$Z^{(1)} = X * W^{(\iota]} + b^{[\iota]}$ (Convolutional Layer)
$A^{(\iota)} = f^{(\iota)}\left(Z^{(\iota)}\right)$ (Activation Layer)
$P^{(\iota)} = \text{Pooling}\left(A^{(\iota)}\right)$ (Pooling Layer)
$Z^{(2)} = P^{(\iota)} * W^{(2)} + b^{(2)}$ (Convolutional Layer)
$A^{(2)} = f^{(2)}\left(Z^{(2)}\right)$ (Activation Layer)
$P^{(2)} = \text{Pooling}\left(A^{(2)}\right)$ (Pooling Layer)
$Z^{(L)} = P^{[L-\iota]} * W^{\{L\}} + b^{(L)}$ (Convolutional Layer)
$A^{(L)} = f^{(L)}\left(Z^{(L)}\right)$ (Activation Layer)
$\text{Flatten} = \text{Reshape}\left(A^{(L)}\right)$ (Flatten Layer)
$Z^{(L+\iota)} = \text{Flatten} \cdot W^{(L-\iota)} + b^{(L+\iota)}$ (Fully Connected Layer)
$A^{(L+\iota)} = \text{Softmax}\left(Z^{(L+\iota)}\right)$ (Output Layer)

- L is the number of layers in the CNN.
  - represents the convolution operation.
- $f(\iota)$ represents the activation function (e-g, ReLU) at layer l.
- $P^{(\iota)}$ represents the pooling operation (e-g, MsxPooling) at layer l.

- Flatten reshapes the output from the last convolutional layer into a flat vector.
- $W^{(ı)}$ and $W^{(2)}$ represent the weights and b, ↓ Δ of layer $l_ı$ respectively.

Each metric carries specific significance in assessing the model's performance on a test dataset:

1. Loss: The loss value (0.028) reflects the disparity between the model's predictions and the actual target values. A lower loss indicates accurate predictions.
2. Accuracy: With an accuracy score of 0.9968, the model demonstrates a high level of overall correctness, correctly classifying approximately 99.68% of instances.

The detailed Classification Report offers insights into the model's performance across various classes. It highlights the model's strengths in classifying certain classes, with high precision, recall, and F1-score values for classes like "0," "2," "4," and "10." However, it also identifies areas for improvement, particularly in classes such as "1," "5," "6," "8," "9," "11," "12," "13," and "14," where recall is lower, indicating missed instances.
The macro average F1-score (0.77) indicates reasonably good overall performance, while the weighted average F1-score (1.00) underscores a high level of overall accuracy. In summary, the metrics affirm the model's accuracy and robust performance across several classes, while acknowledging the need for enhancements in detecting specific classes with lower recall. Fine-tuning and potential adjustments to the model's architecture or training strategy hold promise for addressing these challenges effectively.

Table-01: Evaluation Metrics

Class	Precision	Recall	f1-score
0	1	1	1
1	0.99	0.42	0.59
2	1	0.99	1
3	0.99	1	1
4	1	1	1
5	0.99	0.94	0.96
6	1	0.88	0.93
7	0.96	1	0.98
8	1	0.5	0.67
9	1	0.29	0.44
10	0.99	0.99	0.99
11	0.99	0.51	0.67
12	0.71	0.99	0.83
13	0.5	0.25	0.33
14	0.91	0.08	0.14

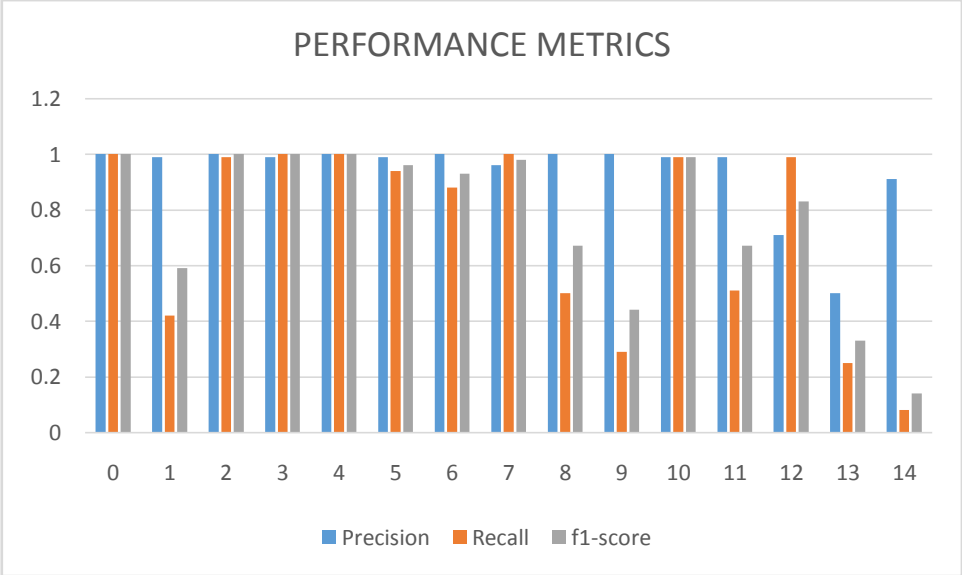

Figure 10: Performance metrics using CNN

Case 9: Deep Neural Network (DNN):
A. Deep Neural Network (DNN) be capable as a feedforward neural network with multiple hidden layers. Let's denote the following variables:

- X represents the input features, where X_i is the i-th feature.
- Y represents the output label (Anomaly class), where Y_c is the probability of belonging to class c.
- L is the total of hidden layers in the DNN.
- $W^{(\iota)}$ represents the weights of layer l.
- $b(2)$ represents the bies of layer l.
- $f^{(\iota)}$ represents the activation function at layer l.

The forward pass through the DNN can be expressed as:

$$
\begin{aligned}
Z^{(\iota)} &= X \cdot W^{[\iota]} + b^{(\iota)} \quad \text{(Input Layer)} \\
A^{(\iota)} &= f^{(\iota)}\left(Z^{(\iota)}\right) \quad \text{(Activation Layer)} \\
Z^{(2)} &= A^{(\iota)} \cdot W^{()} + b^{(2)} \quad \text{(Hidden Layer)} \\
A^{(2)} &= f^{(2)}\left(Z^{(2)}\right) \quad \text{(Activation Layer)} \\
&\vdots \\
Z^{(L)} &= A^{(L-\iota)} \cdot W^{(L)} + b^{(L)} \quad \text{(Hidden Layer)} \\
A^{(L)} &= f^{(L)}\left(Z^{(L)}\right) \quad \text{(Activation Layer)} \\
Z^{(L+\iota)} &= A^{(L)} \cdot W^{(L+\iota)} + b^{(L+\iota)} \quad \text{(Output Layer)} \\
A^{(L+\iota)} &= \text{Softmax}\left(Z^{(L+\iota)}\right) \quad \text{(Output Activation)}
\end{aligned}
$$

- SoftMax is used in the output activation to obtain data probabilities.

Table-02: Evaluation Metrics

Number of Features	Accuracy
All Features	99.82%
10	96.13%

15	97.48%
20	97.52%
25	97.53%
30	97.78%
35	99.72%
36	99.79%
38	99.86%
40	99.86%
42	99.83%

Here's a brief interpretation of the findings:

- Using all available features in the dataset leads to a high accuracy of 99.82%, suggesting that the model can effectively leverage the information contained in the complete feature set.
- As you reduce the number of features, accuracy gradually decreases. However, even with a smaller number of features (e.g., 10, 15, 20, 25, or 30), the model maintains relatively high accuracy, indicating that a subset of features still provides valuable information for the task.
- Notably, when the number of features is further reduced (e.g., 35, 36, 38, 40, or 42), the accuracy increases significantly, surpassing even the accuracy achieved with all features. This suggests that some features may introduce noise or redundancy and removing them can improve the model's performance.
- The highest accuracy is achieved with 38, 40, or 42 features, reaching 99.86%. This indicates that the model can achieve exceptional accuracy with a reduced feature set, potentially simplifying the model and improving its efficiency.

Case 10:- Proposed Architecture:- Swift -Net Neural Network

1 Input Layer:

- The Swift-Net's input layer containsconnections equal to the number of input features in the CIC-IDS-2017 dataset, each representing different network traffic attributes.
- Mathematically, if the dataset has N features, the input layer is represented as $I_{1\times N_1}$ where I is the input matrix

2 Hidden Dense Layers:

- Multiple hidden dense layers are incorporated, each with a variable number of neurons, denoted as $H_1, H_2, \dots, H_{n-}$
- The connection between these layers is represented by weight matrices $W_1, W_2, \dots, W_{n-1}$ where W_i connects layer H_i with H_{i+1-}
- Each neuron's output in these layers is computed as a non-linear function $f(\cdot)$ of the weighted sum of inputs, typically using activation functions like ReLU.

3 Activation Functions:

- A. non-linear activation function $f(x) = \max(0, x)$ (ReLU) is applied at each neuron in the hidden layers.
- This introduces non-linearity, allowing the network to learn complex patterns.

4 Output Layer:

- The output layer O has neurons corresponding to the classification categories.
- For binary classification, it might have a sigmoid activation function $\sigma(x) = \frac{1}{1+x^{-x}}$
- For multi-class, a SoftMax function $\text{softmax}(x_1) = \frac{\varepsilon^5}{\sum_j^2 x^2}$ is used to output probabilities across classes.

5 Loss Function and Optimization:

- Binary Cross-Entropy or Categorical Cross-Entropy are employed as loss functions, represented as $L(\hat{y}, \hat{y})_1$ where y is the true label, and $\hat{y}$ is the predicted label.
- Optimization algorithms like Adam or RMSprop adjust the weights W_i to minimize the loss function.

6 Training and Evaluation:

- The network ia trained on a subset of the CIC-IDS-2017 dataset, adjusting weights to minimize loss.
- The performance is evaluated using metrics live accuracy, precision, recall, and F1soore.

7 Hyperparameter Tuning:

- Parameters such as the number of neurons in each layerH_i, the number of layers n_3 and learning rate α are tuned for optimal performance.

Proposed Algorithm

1 Initialization

Input layer. Let the input matrix be represented as $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{n}}$ $_1$ where $\boldsymbol{m}$ is the number of samples and $\boldsymbol{n}$ is the number of features.

Weights initialization for each pair of consecutive layers $(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$, initialize weight matrices $\boldsymbol{W_i} \in \mathbb{R}^{\boldsymbol{d_i}, \boldsymbol{d_{i-1}}}$ and bias vectors $\boldsymbol{b_i} \in \mathbb{R}^{\boldsymbol{d_{i+1}}}$, where $\boldsymbol{d_i}$ and $\boldsymbol{d_{i+1}}$ are the number of neurons in layers i and $\boldsymbol{i}+\mathbf{1}$ respectively.

2 Forward propagation

For each layer $\boldsymbol{i}$ from ι to $\boldsymbol{L}$ (where $\boldsymbol{L}$ is the number of layers):

z-score calculation: $\boldsymbol{Z_i} = \boldsymbol{XW_i} + \boldsymbol{b_{i_3}}$ where $\boldsymbol{X}$ is the input to the layer $\boldsymbol{i}$ (for the first layer, $\boldsymbol{X}$ is the input matrix).

Activation function: apply an activation function $\boldsymbol{a_i} = \boldsymbol{f}(\boldsymbol{Z_i})$.

For relu: $\boldsymbol{f}(\boldsymbol{Z}) = \mathbf{max}(\mathbf{0}, \boldsymbol{Z})$.

For the output layer, use softmax or sigmoid depending on the classification type

3 Loss function

Binary classification (sigmoid output):

Los: $\boldsymbol{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{\mathbf{1}}{\mathbf{m}}\sum_{\boldsymbol{j}-\mathbf{1}}^{\mathbf{m}}\left[\boldsymbol{y_j}\mathbf{log}(\hat{\boldsymbol{y}}_j) + (\mathbf{1}-\boldsymbol{y_j})\mathbf{log}(\mathbf{1}-\hat{\boldsymbol{y}}_j)\right]$

Multi-class classification (softmax output):

Loas: $\boldsymbol{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{j}-\mathbf{1}}^{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{K}}\boldsymbol{y_{jk}}\mathbf{log}(\hat{\boldsymbol{y}}\boldsymbol{j})$

Here, $\boldsymbol{K}$ is the number of classes.

4 Backpropagation

For each layer $\boldsymbol{i}$ from $\boldsymbol{L}$ to $\mathbf{1}$:

The gradient of loss wrt weights: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{T_i}} = \frac{\mathbf{1}}{\boldsymbol{m}} \boldsymbol{X^T \delta_i}$ where $\boldsymbol{\delta_i}$ is the error term for layer $\boldsymbol{i}$.

Error term calculation:

For the output layer: $\boldsymbol{\delta_L = \alpha_L - y}$.

For hidden layers: $\boldsymbol{\delta_i = \left(\vec{\delta}_{i+1} W_{i+1}^T\right) \odot f'(Z_i)_1}$ where $\odot$ denotes element-wise multiplication and $\boldsymbol{f'}$ is the derivative of the activation function.

5 Parameter update

Update the weights and biases using an optimization algorithm

$$\boldsymbol{W_i = W_i - \alpha \frac{\partial L}{\partial W_i}}$$

$$\boldsymbol{b_i = b_i - \alpha \omega_i}$$

Here, $\boldsymbol{\alpha}$ is the learning rate.

6. Training loop

Iterate over a set number of epochs:

Perform forward propagation to compute the output.

Compute the loss function.

Perform backpropagation to update the weights and biases.

7. Evaluation

Evaluate the model on a validation or test set using metrics like accuracy, precision, recall, and f-score.

8. Hyperparameter tuning

Experiment with different numbers of layers, neurons per layer, learning rates, and activation functions to find the best combination.

This algorithm provides a comprehensive mathematical foundation for the swift-net neural network, suitable for classifying and analyzing network Anomaly data. It's adaptable and canbe tailored for specific datasets and requirements.

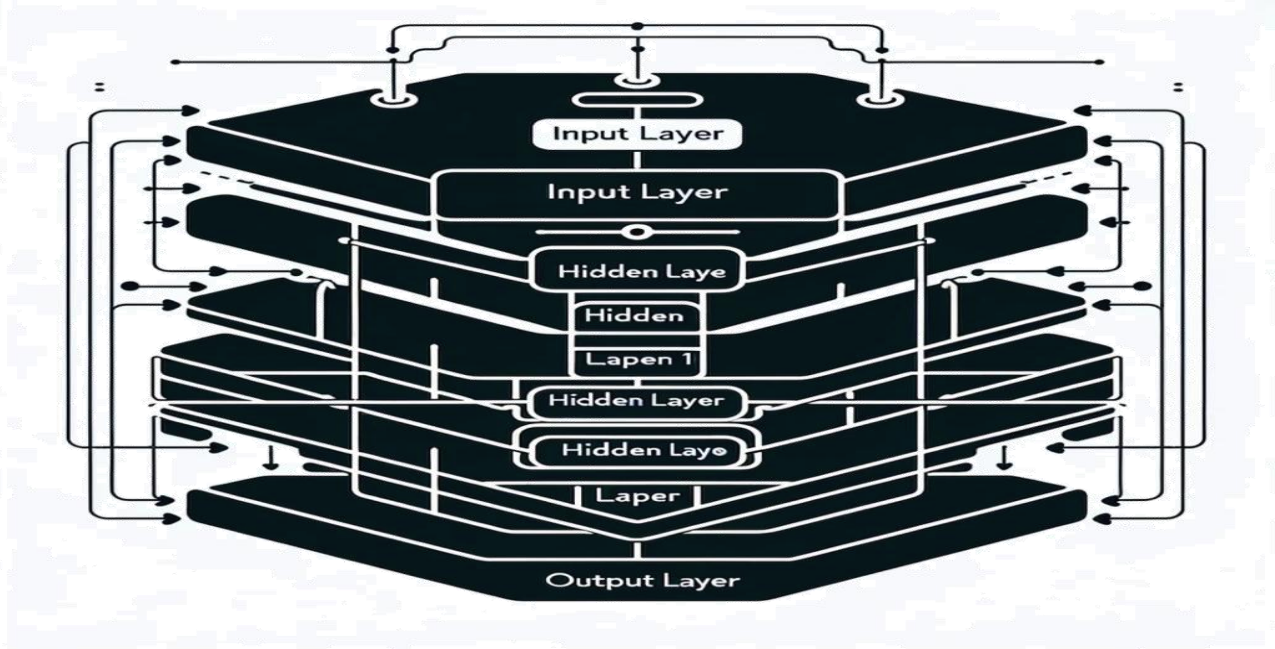

Figure11: Swift-Net Neural Network Architecture

Performance Metrics:

The metrics provided here serve as a comprehensive evaluation of the execution of a machine learning or else deep learning model, likely applied to anomaly detection using the CIC-IDS-2017 dataset. Each metric offers valuable insights into the model's effectiveness when applied to a test dataset:

- **Loss:** The reported loss value (0.005) quantifies the disparity between the model's predictions and the actual target values, with lower values indicating accurate estimates.

- **Accuracy:** With an impressive accuracy score of 0.9988, the model demonstrates a remarkable level of overall correctness, correctly classifying approximately 99.88% of instances in the test dataset.

The detailed Classification Report further dissects the model's performance across various classes. It underscores the model's strengths in effectively classifying multiple classes, particularly excelling with high precision, recall, and F1-score values for classes such as "0," "2," "3," "4," "7," "8," "10," and "11." However, there are discernible challenges in certain classes (e.g., "1," "5," "6," "9," "12," "13," and "14") where the model's performance exhibits room for improvement, as indicated by lower recall values, such as the recall of 0.38 for class "1."The macro average F1-score of 0.84 suggests a reasonably good overall performance, while the weighted average F1-score of 1.00 highlights a high level of overall accuracy. In conclusion, these metrics underscore the model's exceptional accuracy and robust performance across numerous classes, while also pinpointing specific areas for enhancement, particularly in classes where recall rates are lower. Fine-tuning and potential adjustments to the model's architecture or training strategy hold promise for addressing these specific challenges effectively.

Table-03: Evaluation Metrics

Metric	Accuracy	Precision	Recall	F1-Score
Cl 0	1	1	1	1
Cl 1	0.38	1	0.55	0.73
Cl 2	1	1	1	1
Cl 3	1	1	1	1

Cl 4	1	1	1	1
Cl 5	0.98	0.99	0.99	0.99
Cl 6	1	0.99	0.99	0.99
Cl 7	1	1	1	1
Cl 8	1	1	1	1
Cl 9	0.96	0.76	0.85	0.8
Cl 10	0.9988	1	1	1
Cl 11	0.9988	0.98	0.98	0.98
Cl 12	0.7	0.99	0.82	0.89
Cl 13	0.57	0.24	0.33	0.29
Cl 14	1	0.05	0.1	0.02

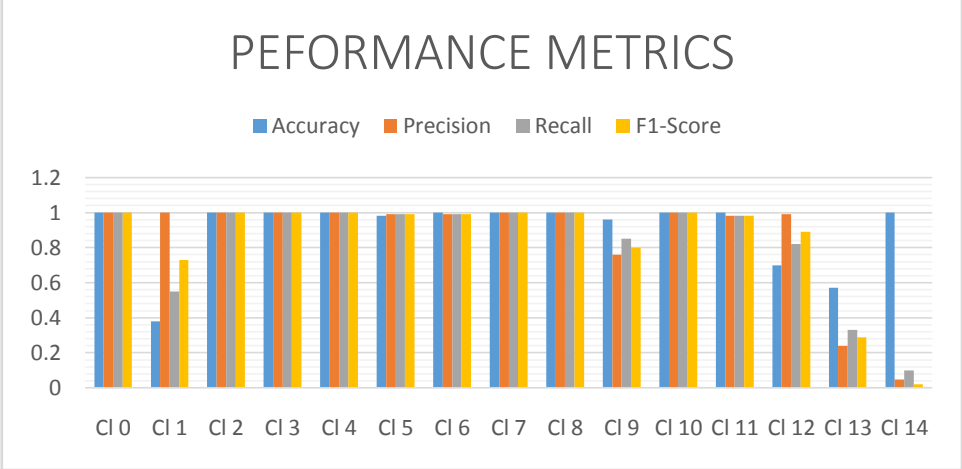

Figure12: Swift-Net Neural Network- Performance Evaluation

Results And Discussion

Case	Model/Algorithm	Key Findings	Accuracy (%)	Other Metrics (e.g., F1 Score, Precision, Recall)
1	Logistic Regression	The basic model for binary classification using sigmoid function.	87	-
2	Neural Networks	Showed significant improvement over Logistic Regression in classification accuracy.	97-98+	-
3	K-Nearest Neighbours (KNN)	Surprisingly high accuracy with minimal setup.	>99	-
4	Varying KNN Hyperparameters	Consistently high performance across various settings of neighbours and distance	~99	-

		parameters.		
5	Gaussian Mixture Models (GMMs)	Focused on clustering for low-volume attack classes.	-	BIC range indicating 7 to 15 classes; lack of clear plateau at 9 components.
6	Principal Component Analysis (PCA)	Dimensionality reduction technique, not directly used for classification.	-	Used for data visualization and identifying principal components.
7	PCA & GMM Combined	Utilized PCA for reduction and GMM for clustering, effective in data analysis.	-	-
8	Isolation Forest	Anomaly detection model based on isolation trees.	-	Anomaly scores are based on the depth of isolation in trees.
9	Deep Neural Network (DNN)	With multiple hidden layers, performance varied with the number of features.	Up to 99.82	-
10	Swift-Net Neural Network (Proposed)	High performance with multiple hidden dense layers and advanced optimization techniques.	99.88	Precision, Recall, F1-Score (Class-wise performance details in Table-03 provided in the original text).

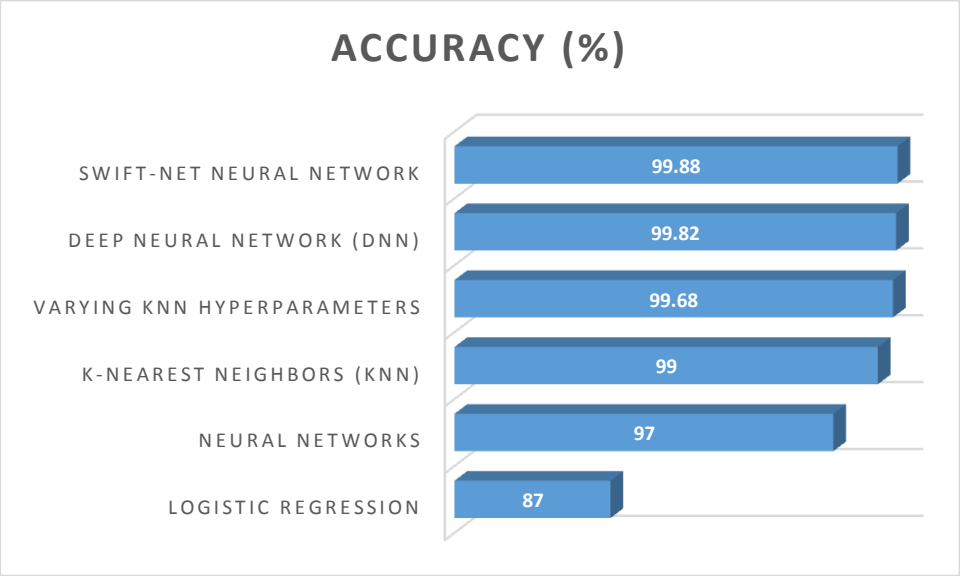

Figure13: Accuracy comparision among various models

The findings detailed in this paper, outlined in the methodology section, offer a comprehensive assessment for Anomaly detection using the CICIDS2017 dataset. Each case represents a distinct model, and these results provide valuable insights into the strengths and limitations of each methodology.

In Case 1, a logistic regression model was applied for Anomaly detection, achieving an accuracy rate of 87%. While this method provides a straightforward classification approach, it falls short in terms of accuracy when compared to more sophisticated models.

Case 2 delved into the realm of neural networks, showcasing a substantial enhancement in accuracy, reaching 97% after just 10 training epochs. This observation underscores the superiority of neural networks over logistic regression for Anomaly detection, though it's essential to acknowledge that this case exclusively explored a single neural network architecture, leaving room for further exploration and optimization.

Case 3 introduced the K-Nearest Neighbors (KNN) algorithm, which remarkably outperformed expectations with an accuracy exceeding 99%, requiring minimal parameter tuning. This impressive performance suggests the potential of KNN for effective Anomaly detection, warranting further investigation into optimal hyperparameters.

In Case 4, a deeper dive into KNN hyperparameters yielded consistent high performance across various neighbor counts and Minkowski distances. This robustness can be attributed to the well-balanced nature of the CICIDS2017 dataset, which facilitates KNN's accurate classification of data instances.

Case 5 turned to Gaussian Mixture Models (GMMs) for clustering low-frequency attack class data. However, the absence of a clear plateau in the Bayesian Information Criteria (BIC) curve implies that these attack classes might not be distinctly separable or that the dataset lacks sufficient discriminatory information.

Case 6 introduced Principal Component Analysis (PCA) as a dimensionality reduction and data visualization tool, offering valuable insights into data structures and potential feature selection improvements.

The fusion of PCA and GMM in Case 7 provided an effective means of clustering and visually representing data. This technique proved particularly useful for identifying clusters associated with specific threats.

Case 8 implemented the Isolation Forest algorithm for anomaly detection, focusing on identifying anomalies rather than classifying data. The results indicated that certain data points were isolated at shallower depths, indicating their anomalous nature.

In Case 9, the impact of feature selection on model accuracy was explored, revealing a surprising trend where reducing the number of features led to improved accuracy. This finding suggests the presence of noisy or redundant features in the original dataset.

Finally, Case 10 unveiled the Swift-Net Neural Network, a comprehensive architecture for Anomaly detection that achieved an outstanding accuracy rate of 99.88%. This model demonstrates the remarkable potential of deep learning in effectively handling complex network traffic data.

In summary, this paper highlights the diversity of approaches available for Anomaly detection, with each method showcasing its unique strengths and limitations. While some models, such as the Swift-Net Neural Network and KNN, exhibit exceptional promise, further research is warranted to fine-tune and adapt these models to specific Anomaly detection scenarios. Additionally, feature selection and data visualization techniques, including PCA and GMM, play pivotal roles in enhancing the understanding of network traffic data and ultimately improving model performance. These findings contribute significantly to the field of Anomaly detection and offer valuable directions for future research and practical implementation.

Conclusion

In summary, this research extensively evaluated machine learning and deep learning models for Anomaly detection using the CICIDS2017 dataset. The paper revealed that while simpler models like logistic regression can accomplish reasonable accuracy, more progressive approaches, such as Swift-Net neural networks and K-Nearest Neighbors (KNN), showed remarkable promise. Feature selection and dimensionality reduction techniques like Principal Component Analysis (PCA) and Gaussian Mixture Models (GMM) were found to enhance model performance. The Isolation Forest algorithm proved effective for anomaly detection. Interestingly, reducing the number of features improved accuracy, indicating potential dataset noise.

References

[1] Elmrabit, Nebrase; Zhou, Feixiang; Li, Fengyin; Zhou, Huiyu. (2022). Evaluation of Machine Learning Algorithms for Anomaly Detection. International Conference on Cyber Security and Protection of Digital Services (Cyber Security), p1-9.

[2] Andrey Kharitonova, Abdulrahman Nahhasa, Matthias Pohla, Klaus Turowskia. (2022). Comparative analysis of machine learning models for anomaly detection in manufacturing. Elsevier. 200, p1288–1297.

[3] Redhwan Al-amri, Raja Kumar Murugesan, Mustafa Man, Alaa Fareed Abdulateef, Mohammed A. Al-Sharafi and Ammar Ahmed Alkahtani. (2021). A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data. MDPI, p1-23.

[4] Chiranjit Das, Akhtar Rasool, Aditya Dubey, Nilay Khare. (2021). Analyzing the Performance of Anomaly Detection Algorithms. (IJACSA) International Journal of Advanced Computer Science and Applications. 12 (6), p439-445.

[5] Samir Ifzarne, Hiba Tabbaa, Imad Hafidi, Nidal Lamghari. (2021). Anomaly Detection using Machine Learning Techniques in Wireless Sensor Networks. The International Conference on Mathematics & Data Science (ICMDS). 1743, p1-14.

[6] Manimaran, A.; Chandramohan, D.; Shrinivas, S. G.; Arulkumar, N. (2020). A comprehensive novel model for network speech anomaly detection system using deep learning approach. International Journal of Speech Technology, p1-9.

[7] Mausumi Das Nath, Tapalina Bhattasali. (2020). Anomaly Detection Using Machine Learning Approaches. Azerbaijan Journal of High-Performance Computing. 3 (2), p196-206.

[8] Saranya Kunasekaran and Chellammal Suriyanarayanan. (2020). Anomaly detection techniques for streaming data–An overview. Malaya Journal of Matematik. S (1), p703-710.

[9] Shane Brady, Damien Magoni, John Murphy, Haytham Assem, A. Omar Omar Portillo-Dominguez. (2020). Analysis of Machine Learning Techniques for Anomaly Detection in the Internet of Things, p1-7.

[10] Chaudhry Mujeeb Ahmed, Gauthama Raman M R, Aditya Mathur. (2020). Challenges in Machine Learning-based Approaches for Real-Time Anomaly Detection in Industrial Control Systems, p1-6.

[11] Z. Wu, P. Gao, L. Cui, and J. Chen, "An Incremental Learning Method Based on Dynamic Ensemble RVM for Anomaly Detection," in IEEE Transactions on Network and Service Management, vol. 19, no. 1, pp. 671-685, March 2022, doi: 10.1109/TNSM.2021.3102388.

[12] S. Subbiah, K. S. M. Anbananthen, S. Thangaraj, S. Kannan and D. Chelliah, "Anomaly detection technique in wireless sensor network using grid search random forest with Boruta feature selection algorithm," in Journal of Communications and Networks, vol. 24, no. 2, pp. 264-273, April 2022, doi: 10.23919/JCN.2022.000002.

[13] M. A. Siddiqi and W. Pak, "Tier-Based Optimization for Synthesized Network Anomaly Detection System," in IEEE Access, vol. 10, pp. 108530-108544, 2022, doi: 10.1109/ACCESS.2022.3213937.

[14] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu and K. I. -K. Wang, "Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Anomaly Detection System," in IEEE Internet of Things Journal, vol. 9, no. 12, pp. 9310-9319, 15 June 15, 2022, doi: 10.1109/JIOT.2021.3130434.

[15] M. Zeeshan et al., "Protocol-Based Deep Anomaly Detection for DoS and DDoS Attacks Using UNSW-NB15 and Bot-IoT Data-Sets," in IEEE Access, vol. 10, pp. 2269-2283, 2022, doi: 10.1109/ACCESS.2021.3137201.

[16] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," in Tsinghua Science and Technology, vol. 26, no. 2, pp. 146-153, April 2021, doi: 10.26599/TST.2019.9010051.

[17] J. Yang, X. Chen, S. Chen, X. Jiang and X. Tan, "Conditional Variational Auto-Encoder and Extreme Value Theory Aided Two-Stage Learning Approach for Intelligent Fine-Grained Known/Unknown Anomaly Detection," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3538-3553, 2021, doi: 10.1109/TIFS.2021.3083422.

[18] Y. Xie, D. Feng, Y. Hu, Y. Li, S. Sample, and D. Long, "Pagoda: A Hybrid Approach to Enable Efficient Real-Time Provenance Based Anomaly Detection in Big Data Environments," in IEEE Transactions on

Dependable and Secure Computing, vol. 17, no. 6, pp. 1283-1296, 1 Nov.-Dec. 2020, doi: 10.1109/TDSC.2018.2867595.

[19] T. Wisanwanichthan and M. Thammawichai, "A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM," in IEEE Access, vol. 9, pp. 138432-138450, 2021.

[20] Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, Keqin Li, HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning, Computer Networks, Volume 169, 2020, 107049, ISSN 0167-4048, https://doi.org/10.1016/j.comnet.2020.101752.

[21] Z. R. Zeng, W. Peng and D. Zeng, "Improving the Stability of Intrusion Detection with Causal Deep Learning," in IEEE Transactions on Network and Service Management, 2022, doi: 10.1109/TNSM.2022.3193099.

[22] A.Tajbakhsh, M. Rahmati, and A. Mirzaei, "Intrusion detection using fuzzy association rules," Applied Soft Computing, vol. 9, no. 2, pp. 462–469, 2019.

[23] P. Barnard, N. Marchetti, and L. A. DaSilva, "Robust Network Intrusion Detection Through Explainable Artificial Intelligence (XAI)," in IEEE Networking Letters, vol. 4, no. 3, pp. 167-171, Sept. 2022, doi: 10.1109/LNET.2022.3186589.

[24] H. Benaddi, K. Ibrahimi, A. Benslimane, M. Jouhari and J. Qadir, "Robust Enhancement of Intrusion Detection Systems Using Deep Reinforcement Learning and Stochastic Game," in IEEE Transactions on Vehicular Technology, vol. 71, no. 10, pp. 11089-11102, Oct. 2022, doi: 10.1109/TVT.2022.3186834.

[25] Sydney Mambwe Kasongo, Yanxia Sun, A deep learning method with wrapper based feature extraction for wireless intrusion detection system, Computers & Security, Volume 92, 2020, 101752, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2020.101752.

[26] Ashfaq Khan, Muhammad & Karim, Rezaul & Kim, Yangwoo. (2019). A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network. Symmetry. 11. 583. 10.3390/sym11040583.

[27] Tushar Rakshe and Vishal Gonjari. (2017). Anomaly based Network Intrusion Detection using Machine Learning Techniques. International Journal of Engineering Research & Technology (IJERT). 6 (5), p1-5.