Leveraging Adaptive Deep Learning Model for High Precision Worm Detection

Anoop Lashiyal and P. Satya Sai Charan

Abstract- In the ever-evolving landscape of cybersecurity, network worms present a formidable challenge, with their capacity to adapt and exploit vulnerabilities at an alarming pace. Traditional detection systems often fail to keep up with these rapidly changing attack vectors, rendering them less effective against newly emerging threats. This project pioneers a cutting-edge approach by leveraging adaptive deep learning architectures, particularly Artificial Neural Networks (ANN) and Long Short-Term Memory (LSTM) networks, to elevate the accuracy and efficiency of network worm detection. Through the integration of sophisticated machine learning algorithms, robust data preprocessing techniques, and a framework for real-time adaptability, our solution is optimized to detect both known and novel threats with precision. This adaptive system not only enhances the detection of anomalies but also ensures scalability and efficiency for deployment in dynamic, real-world environments. The resulting solution offers a resilient, high-performance defense mechanism, tailored to address the ever-changing complexities of modern network security challenges.

INTRODUCTION

As cyber threats evolve in sophistication and scale, the demand for cutting-edge detection systems has reached unprecedented levels. Among these threats, network worms—self-replicating malware capable of spreading autonomously across systems—represent a formidable challenge to network security. With their ability to rapidly infiltrate and compromise systems, traditional defense mechanisms often fail to provide timely responses. Modern advancements in machine learning, leveraging algorithms like Decision Trees, Random Forests, and Gaussian Naive Bayes (GaussianNB), have revolutionized anomaly detection by enabling the analysis of large, complex datasets. These models excel at identifying intricate patterns that distinguish benign activities from malicious behaviors, marking a pivotal step forward in cybersecurity.

This project emphasizes the development of adaptive, intelligent systems designed to dynamically evolve alongside emerging threats. By incorporating real-time learning capabilities, these systems continuously update their models to identify and mitigate both known and unknown threats with remarkable precision. Through the integration of advanced data preprocessing techniques and robust machine learning frameworks, this solution ensures scalability, efficiency, and responsiveness in dynamic environments. The outcome is a resilient, high-performance detection mechanism that aligns with the ever-changing landscape of cybersecurity, providing a proactive defense against the proliferation of sophisticated network worms and other malicious entities.

Dataset Information

The dataset utilized for this project was generated using the PerfectStorm tool at the Australian Centre for Cyber Security's (ACCS) Cyber Range Lab [1]. It combines real-world normal activities with synthetic modern attack behaviors. A total of 100 GB of raw network traffic was captured using the Tcpdump tool (e.g., Pcap files). The dataset includes nine categories of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Features and class labels were extracted using Argus and Bro-IDS tools, resulting in 49 features. Two subsets were created: a training set (UNSWNB15 training-set.csv) with 175,341 records and a testing set (UNSWNB15 testing-set.csv) containing 82,332 records, including both normal and attack instances. These features represent network behavior, such as packet size, connection duration, and frequency, while the target variable classifies various network threat types for precise categorization.

Approach

1. In the first stage, the data set available in variable formats are imported and converted into a Data Frame that can be analyzed.
2. Exploratory data analysis on the dataset is performed to get fundamental insights from the data, such as getting to know the datatypes, spotting any outliers, getting to know the basic structure of the dataset, such as the number of rows and columns, and basic statistical inferences such as the correlation between the variables.
3. In the third stage, data preprocessing is performed to improve the quality of both data and prediction. At this stage, usually, Duplicate rows and blank rows in the dataset are removed, and the null or empty values in the dataset are replaced with the mean values of the entire dataset. The categorical or the data having object as datatype are

converted to numerical datatype using One Hot label Encoder. Also, the Principal Component Analysis technique is used for dimensionality reduction since there are some unwanted columns that consume the storage and code execution time and might also result in a less accurate model due to their less correlation with the dependent variable. The data is also split into training and validation data sets, preparing it for the next stage.

4. Training is the most important stage in the process but, surprisingly, is a short one. Suitable Machine learning algorithms are chosen, and the training data set is used to fit the algorithms.
5. For validation, the fit algorithm is used to predict the values for the test data set, and the results are compared with the original to calculate the metrics like accuracy.
6. In the final stage, the results obtained are insights gained are presented in a visual form for quick and better understanding.

ALGORITHM

1. Logistic Regression

Logistic regression is a supervised machine learning algorithm designed for classification tasks, aiming to predict the likelihood of an instance belonging to a specific class. It is a statistical method that examines the relationship between two variables. Primarily used for binary classification, logistic regression employs the sigmoid function to map independent variable inputs to a probability value ranging between 0 and 1. Known for its simplicity and ease of interpretation, it is often used as a baseline model in machine learning.

Logistic Function:

$$P(Y=1 \mid \mathbf{X}) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}}$$

Where:

- $P(Y=1 \mid \mathbf{X})$: Probability that a threat is **Signature-Based** given feature vector $\mathbf{X}$.
- Y: Binary target variable where $Y=1$ denotes a **Signature-Based Threat** and $Y=0$ denotes an **Anomaly-Based Threat**.
- $\mathbf{X}=[X_1, X_2, \ldots, X_n]$: Feature vector representing various attributes of a threat, such as protocol type, port number, frequency of actions, session duration, etc.
- β_0: Intercept term.
- $\beta_1, \beta_2, \ldots, \beta_n$: Coefficients corresponding to each feature $X_1, X_2, \ldots, X_n$.

2. Support Vector Machine

Support Vector Machine is a supervised ML algorithm that can be used for both classification and regression problems. The objective of the SVM is to minimize the cost function and maximize the margin between the support vectors through a hyperplane. Therefore, a hyperplane that can be of any dimension and linear or non-linear is used in SVM to differentiate between two or more classes for classification. Enhances generalization by finding the optimal hyperplane that separates threat types.

Optimization Problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1, \quad \forall i$$

Where:

- $\mathbf{w}$: Weight vector defining the orientation of the hyperplane.
- b: Bias term determining the offset of the hyperplane from the origin.
- y_i: Class label for the i-th threat ($y_i=1$ for **Signature-Based**, $y_i=-1$ for **Anomaly-Based**).
- $\mathbf{x}_i$: Feature vector for the i-th threat.
- $\|\mathbf{w}\|$: Euclidean norm of the weight vector, representing the margin size.

.

3.Gradient boosting

Gradient boosting is an ensemble supervised machine learning algorithm that can be applied to both classification and regression tasks. It extends the capabilities of traditional gradient boosting by incorporating ℓ1 and ℓ2 regularization, enhancing model generalization and mitigating overfitting, where ℓ1 and ℓ2 represent loss functions. This algorithm is often faster than standard gradient boosting due to its parallelized tree construction. Additionally, it efficiently handles missing data, simplifying data preparation. By combining multiple weak learners, gradient boosting creates a robust predictor, making it highly effective at capturing complex, non-linear relationships in threat data.

Sequential Addition of Trees:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta \cdot h_m(\mathbf{x})$$

Where:

- $F_m(\mathbf{x})$: Combined model's prediction after m trees.
- $F_{m-1}(\mathbf{x})$: Combined model's prediction after $m-1$ trees.
- η: Learning rate controlling the contribution of each new tree.
- $h_m(\mathbf{x})$: Prediction of the m-th weak learner (decision tree).

4.Artificial Neural Networks (ANN)

An Artificial Neural Network (ANN) is a machine learning algorithm inspired by the structure of human neurons, suitable for both classification and regression tasks. It comprises three primary layers: an input layer that receives raw data, one or more hidden layers that process and extract patterns, and an output layer that generates predictions based on the interactions within the network. ANN effectively captures complex feature interactions, improving its ability to detect threats. For practical, real-world applications, it requires diverse training data. Despite its robustness—continuing to function even if some units fail—ANNs demand significant computational and storage resources, which can be a drawback.

Backpropagation and Gradient Descent:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}}$$
$$\mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}}$$

Where:

- $\mathbf{W}$: Weight matrix of the neural network.
- $\mathbf{b}$: Bias vector of the neural network.
- η: Learning rate determining the step size during optimization.
- $\mathcal{L}$: Loss function (e.g., binary cross-entropy) measuring the discrepancy between predicted and actual threat classes.

5.Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM), developed by Hochreiter and Schmidhuber, is an advanced form of recurrent neural network (RNN) designed to overcome the limitations of traditional RNNs in learning long-term dependencies. Unlike standard RNNs, which rely on a single hidden state that struggles with retaining information over extended sequences, LSTMs incorporate a memory cell capable of storing information for prolonged periods. This architecture makes LSTMs highly effective for tasks involving sequential data, such as language translation, speech recognition, and time series forecasting. They excel at detecting anomalies in sequences, such as irregular patterns in system events, and are particularly well-suited for network worm detection due to their ability to analyze sequential data and capture temporal relationships.

Convolution Operation:

$$y_{i,j} = \sigma\left(\sum_m \sum_p \sum_q x_{i+p,j+q} \cdot w_{m,p,q} + b_m\right)$$

Where:

- $y_{i,j}$: Output feature map value at position (i, j).
- σ: Activation function (e.g., ReLU).
- $x_{i+p,j+q}$: Input feature at position $(i + p, j + q)$.
- $w_{m,p,q}$: Weights of the m-th convolutional filter.
- b_m: Bias term for the m-th filter.

6.Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a highly effective machine learning approach, particularly suited for tasks in computer vision. These specialized neural networks are designed to process grid-structured data, such as images, with exceptional efficiency. CNNs remain robust even when some units are unresponsive, owing to their extensive and resilient architecture. They are particularly adept at recognizing unique combinations of features, making them excellent for detecting signature-based threats.

Convolution Operation:

$$y_{i,j} = \sigma\left(\sum_m \sum_p \sum_q x_{i+p,j+q} \cdot w_{m,p,q} + b_m\right)$$

Where:

- $y_{i,j}$: Output feature map value at spatial location (i, j).
- σ: Activation function (e.g., ReLU) applied after convolution.
- $x_{i+p,j+q}$: Input feature at position $(i + p, j + q)$ relative to the convolutional filter.
- $w_{m,p,q}$: Weights of the m-th convolutional filter spanning spatial dimensions p and q.
- b_m: Bias term for the m-th filter.

7.K-Nearest Neighbour (KNN)

K-Nearest Neighbors (KNN) is a fundamental yet highly valuable classification algorithm in machine learning. As a supervised learning technique, it is extensively applied in fields such as pattern recognition, data mining, and intrusion detection. Known for its simplicity and ease of use, KNN is a versatile algorithm that operates as a non-parametric method, making predictions based on the similarity between data points. It is relatively robust against outliers and excels at identifying local structures within the data, making it particularly effective for detecting subtle threat patterns.

Majority Voting:

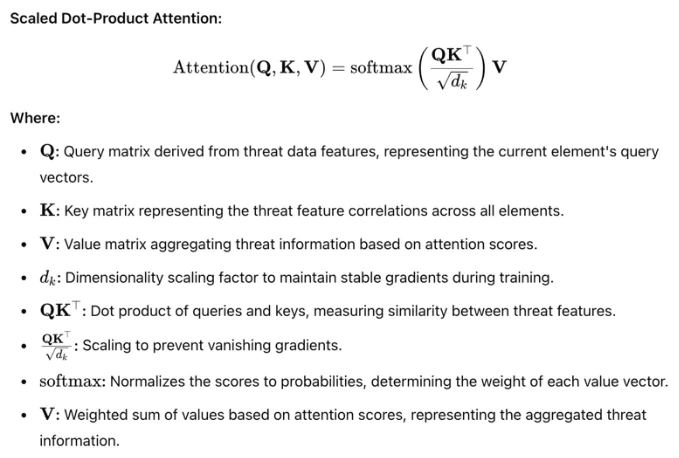
$$\hat{y} = \text{mode}\{y_1, y_2, \ldots, y_k\}$$

Where:

- $\hat{y}$: Predicted class label for the threat.
- $y_1, y_2, \ldots, y_k$: Class labels of the k-nearest neighbors to the threat in question.

8.Transformer-Based Model

Transformers have revolutionized machine learning, particularly in natural language processing, and are now increasingly applied to cybersecurity for advanced threat detection. Leveraging self-attention mechanisms, transformers capture global feature dependencies, enabling the precise identification of complex and evolving threat patterns. Their parallel processing capabilities facilitate faster computation on large datasets, enhancing real-time detection of both signature-based and anomaly-based threats. While their complex architecture requires substantial computational resources and extensive training data, their superior performance in handling dynamic threats positions them as a vital component of modern cybersecurity systems.



Scaled Dot-Product Attention:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Where:

- $\mathbf{Q}$: Query matrix derived from threat data features, representing the current element's query vectors.
- $\mathbf{K}$: Key matrix representing the threat feature correlations across all elements.
- $\mathbf{V}$: Value matrix aggregating threat information based on attention scores.
- d_k: Dimensionality scaling factor to maintain stable gradients during training.
- $\mathbf{QK}^\top$: Dot product of queries and keys, measuring similarity between threat features.
- $\frac{\mathbf{QK}^\top}{\sqrt{d_k}}$: Scaling to prevent vanishing gradients.
- softmax: Normalizes the scores to probabilities, determining the weight of each value vector.
- $\mathbf{V}$: Weighted sum of values based on attention scores, representing the aggregated threat information.

DATA

1. Data Import

As mentioned earlier, the data from the source is provided as two data sets—one dedicated to training the algorithm and one for testing it. The train data set contained 82332 rows and 45 columns, also called features, and the test data set had 175341 rows and 45 features. They are read using the simple pandas read_csv command.

2. Exploratory Data Analysis

In this stage, the data is explored to understand the features and find the underlying trends. The primary functions '.describe()', '.head( )' are used to observe the samples of both the test and train data sets. It was observed that there are four features, 'attack_cat', 'proto', 'service', and 'state', that are formatted as categorical values that should be preprocessed. It was also observed that each record is categorized either as a normal record or an attack, and there are nine diverse types of attacks in the data. They are Generic, Exploits, Fuzzers, Dos, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms as shown in the figure below. The data is then mapped based on the attack's threat type, dividing it into anomaly based and signature-based attack as shown in figure below.

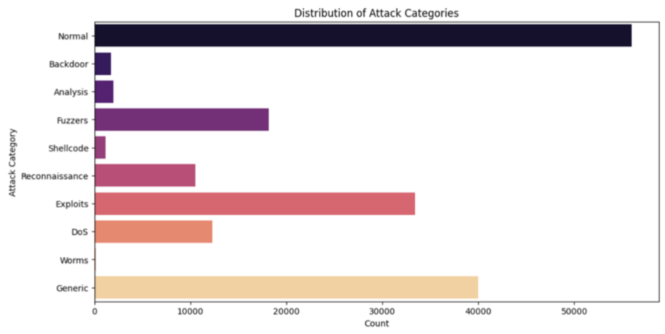


Fig.1 Types of attacks and their density

3. Data Preprocessing

First, the train and test data sets are checked for null values using the command '.isnull()'. Surprisingly, there are no records with incomplete data in the data sets. Synthetic Minority Over-sampling Techniques (SMOTE) is applied to address class imbalance in the dataset, allowing the model to learn better from minority classes. This enabled the dataset to be more information rich and helped in balancing the data as shown in the image below. Since the categorical values cannot be used for data analysis, the four features mentioned in the above sections are converted into numerical values using the 'preprocessing.LabelEncoder()'. In the datasets obtained from the source, the training data set contained fewer records than the testing/validation data set. We combined the data and later split it into a known train test ratio to use a much more common approach. In the dataset, the value to be predicted using the machine learning algorithms (target dataset) is 'label' from the features. This value is dropped from the dataset and is stored in the 'Y' variable. The rest of the features are stored in the 'X' variable. To minimize the bias that can occur because of the difference in the proportionality of the data range, we normalized the data using the standard scalar function.

To reduce the features not required for intrusion detection, we used the concept of Principle Component Analysis. The models are trained in 15 PCA components. So, we reduced the features in the train data from 45 to 15, to observe the run time and accuracy of various ML algorithms. Since the data sets for testing and training are given separately, we just separated and value to be predicted which is the 'label feature' and saved it in different variables. Finally, the datasets are X_train, X_test, Y_train, and Y_test. These data sets are then used to train and validate the algorithms in the next stage.

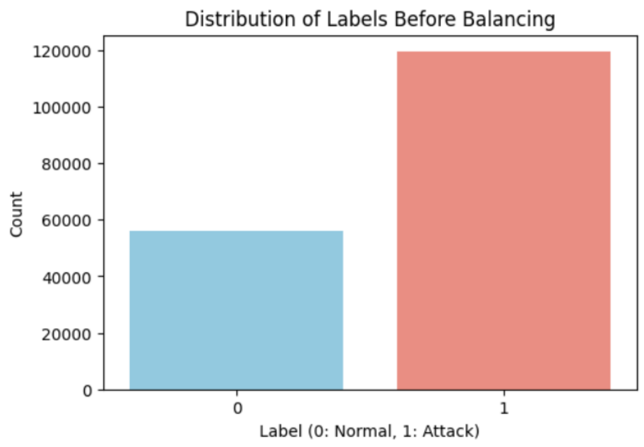


Fig.3 Distribution of labels before balance changes

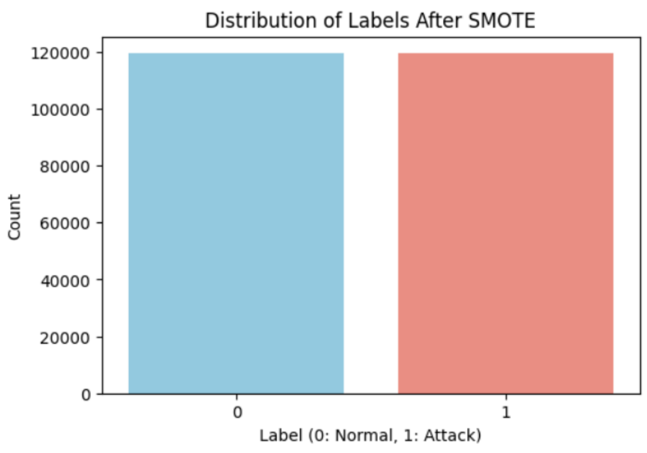
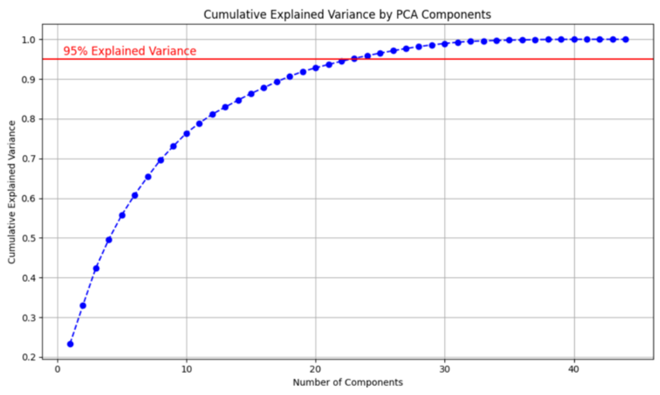


Fig.4 Labels after applying SMOTE



Current Models and Training Approaches

4. Model Evaluation and Performance Metrics

Evaluating models using appropriate metrics is essential to understand their effectiveness in distinguishing between signature-based and anomaly-based threats.

- Accuracy: Proportion of correctly classified instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Precision: Proportion of positive identifications that were actually correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall (Sensitivity): Proportion of actual positives that were identified correctly.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F1-Score: Harmonic mean of precision and recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- ROC-AUC (Receiver Operating Characteristic - Area Under Curve): Measures the ability of the model to distinguish between classes.

$$\text{ROC-AUC} = \int_0^1 \text{TPR}\, d\text{FPR}$$

- Confusion Matrix: Tabular representation of actual vs. predicted classifications, detailing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

5. Logistic Regression

Logistic Regression model is imported from Sklearn library and is trained on the X_train and Y_train datasets. The accuracy, precision, f1-score, recall and ROC-AUC functions are employed giving. For 21 PCA components the accuracy achieved is 82.57.

6. Support Vector Machine (SVM)

The SVM classifier is imported from the Sklearn library and is trained on the X_train and Y_train datasets. The accuracy, precision, f1-score, recall and ROC-AUC functions are employed giving. For 21 PCA components the accuracy achieved is 91.34.

7. Gradient Boosting

From the Sklearn library, the GradientBoostingClassifier is imported and is trained on the X_train and Y_train datasets. The accuracy, precision, f1-score, recall and ROC-AUC functions are employed giving. For 21 PCA components the accuracy achieved is 96.56.

8. Artificial Neural Network (ANN)

Unlike the algorithms above, ANN is a complex algorithm. To train the algorithm we added 1 input layer, 2 hidden layers and one output layer. For the first input layer we used 'Dense' class which is a famous class in tensor flow, and we used 6 output dimensions as there is no rule to decide the neurons in the hidden layer. The input dimensions in this layer are equal to the number of PCA components. We also used 'relu' activation function. For the second hidden layer we used 9 output dimensions (units). In the output layer we used only one output dimensions are the dependent variables is in binary form. The process of training is performed in 2 steps. The first one is compiling the ANN, and the second stage is to fit the ANN to the training set. The Compile used is a method of Tensor flow and the optimizer used is 'adam' with a learning rate of 0.001. This optimizer in general performs Stochastic gradient descent and updates the weight during training to reduce the loss. We fit the data on X_train and Y_train data sets with a batch_size of 32 and nb_epoch =20 to improve the accuracy over time. For 21 PCA components accuracy is 94.03.

9. Long Short-Term Memory (LSTM)

Unlike ANN, which treats inputs as independent features, LSTM incorporates memory elements to learn patterns across time steps, which may lead to better results for sequential datasets like network traffic data. LSTM networks require more computational power and time due to additional components like cell states and gates (input, forget, and output gates) to manage temporal information. For the same dataset, an LSTM may require more epochs and longer training time compared to ANN due to its added complexity. Since LSTMs process data sequentially and include additional computations (gates and cell states), the training time is higher. The Compile used is a method of Tensor flow and the activation function being 'sigmoid'. This optimizer in general performs Stochastic gradient descent and updates the weight during training to reduce the loss. We fit the data on X_train and Y_train data sets with a batch_size of 32 and nb_epoch =30 to improve the accuracy over time. For 21 PCA components accuracy is 94.57.

10. Convolutional Neural Networks (CNN)

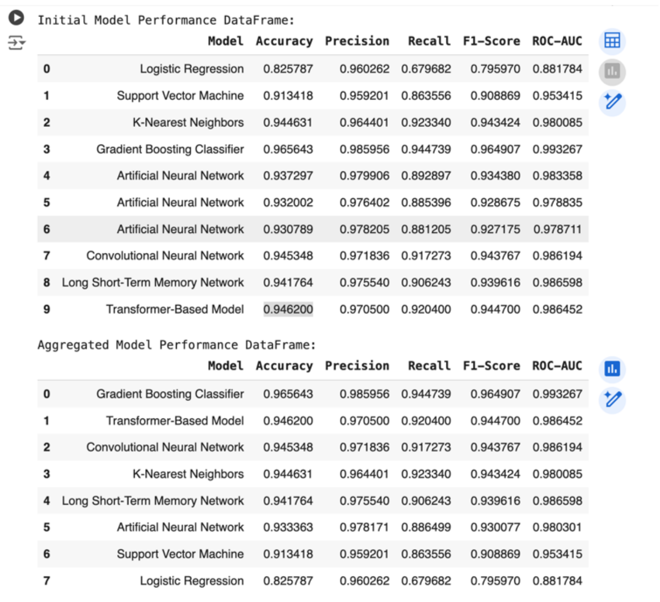
The Compile used is a method of Tensor flow and the activation function being 'relu' and 'sigmoid'. This optimizer in general performs Stochastic gradient descent and updates the weight during training to reduce the loss. We fit the data on X_train and Y_train data sets with a batch_size of 32 and nb_epoch =30 to improve the accuracy over time. For 21 PCA components accuracy is 94.25.

11. Transformer-Based Model

Transformer-Based Model for Real-Time Detection Transformers have revolutionized the field of natural language processing and have recently been adapted for various other domains, including cybersecurity threat detection. The data had to be prepared for real-time threat detection. Used transformer encoder block to make a feed forward network then defined the model and its hyperparameters. We fit the data on X_train and Y_train data sets with a batch_size of 32 and nb_epoch =30 to improve the accuracy over time. For 21 PCA components accuracy is 94.62.

Results and Comparison

ANN, CNN, TBM and LSTM models outperformed traditional methods such as Logistic Regression, Random Forest, and Gradient Boosting. These deep learning models demonstrated superior accuracy, adaptability, and real-time threat detection capabilities.



Initial Model Performance DataFrame:

	Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	Logistic Regression	0.825787	0.960262	0.679682	0.795970	0.881784
1	Support Vector Machine	0.913418	0.959201	0.863556	0.908869	0.953415
2	K-Nearest Neighbors	0.944631	0.964401	0.923340	0.943424	0.980085
3	Gradient Boosting Classifier	0.965643	0.985956	0.944739	0.964907	0.993267
4	Artificial Neural Network	0.937297	0.979906	0.892897	0.934380	0.983358
5	Artificial Neural Network	0.932002	0.976402	0.885396	0.928675	0.978835
6	Artificial Neural Network	0.930789	0.978205	0.881205	0.927175	0.978711
7	Convolutional Neural Network	0.945348	0.971836	0.917273	0.943767	0.986194
8	Long Short-Term Memory Network	0.941764	0.975540	0.906243	0.939616	0.986598
9	Transformer-Based Model	0.946200	0.970500	0.920400	0.944700	0.986452

Aggregated Model Performance DataFrame:

	Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	Gradient Boosting Classifier	0.965643	0.985956	0.944739	0.964907	0.993267
1	Transformer-Based Model	0.946200	0.970500	0.920400	0.944700	0.986452
2	Convolutional Neural Network	0.945348	0.971836	0.917273	0.943767	0.986194
3	K-Nearest Neighbors	0.944631	0.964401	0.923340	0.943424	0.980085
4	Long Short-Term Memory Network	0.941764	0.975540	0.906243	0.939616	0.986598
5	Artificial Neural Network	0.933363	0.978171	0.886499	0.930077	0.980301
6	Support Vector Machine	0.913418	0.959201	0.863556	0.908869	0.953415
7	Logistic Regression	0.825787	0.960262	0.679682	0.795970	0.881784

Fig.6 Model Comparison

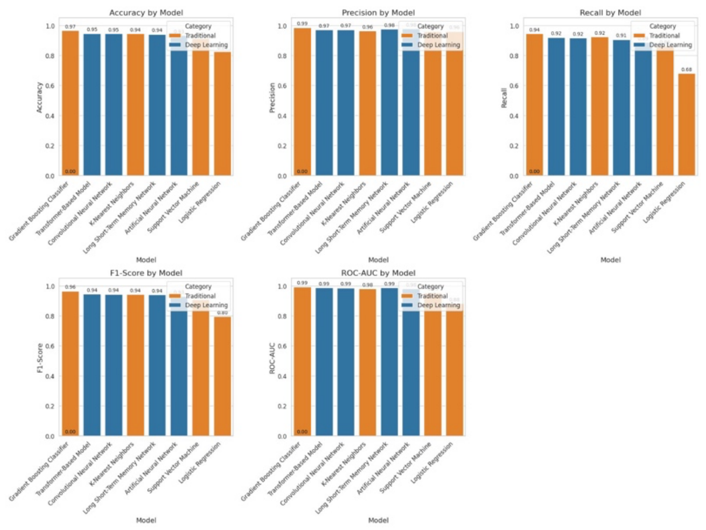


Fig.7 Model Comparision with TDM

As deep learning models have demonstrated superior performance compared to traditional approaches, it is essential to examine their training and validation metrics in greater detail. The alignment between training and validation metrics signifies that the models are not overfitting and are effectively generalizing to unseen data.

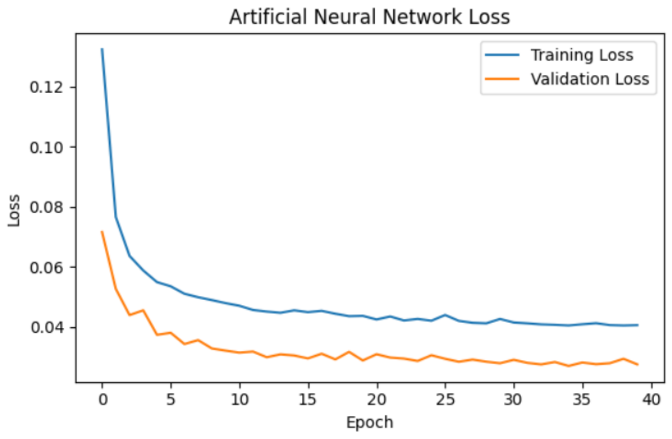


Fig.8 ANN loss with varying epoch value

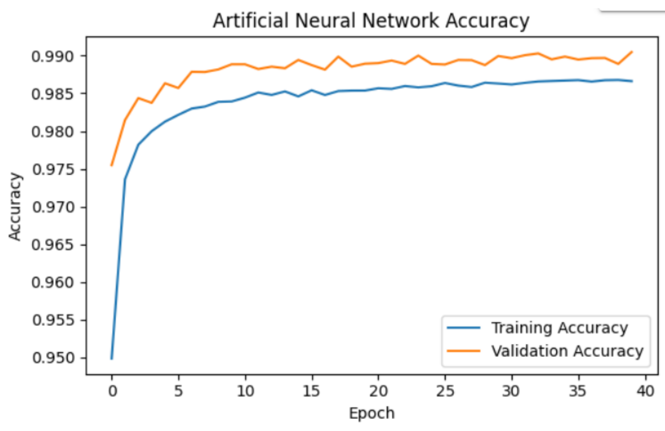


Fig.9 ANN accuracy with varying epoch value

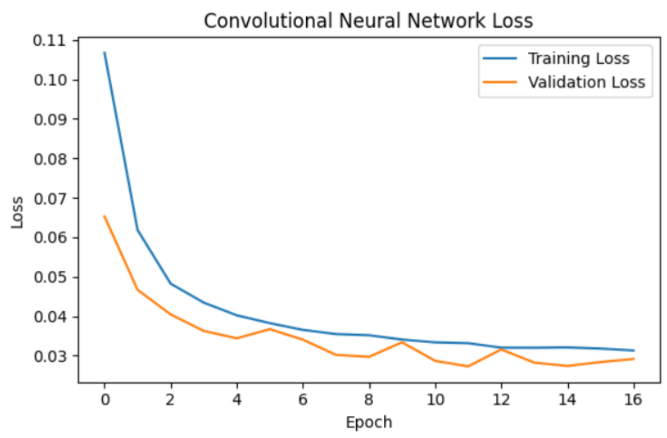


Fig10. CNN loss with varying epoch value

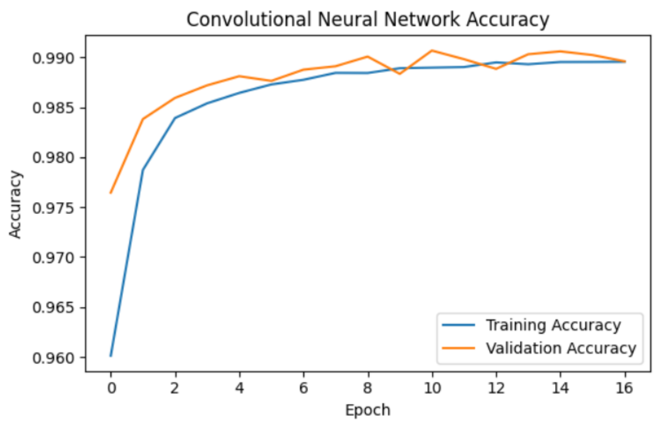


Fig.11 CNN accuracy with varying epoch value

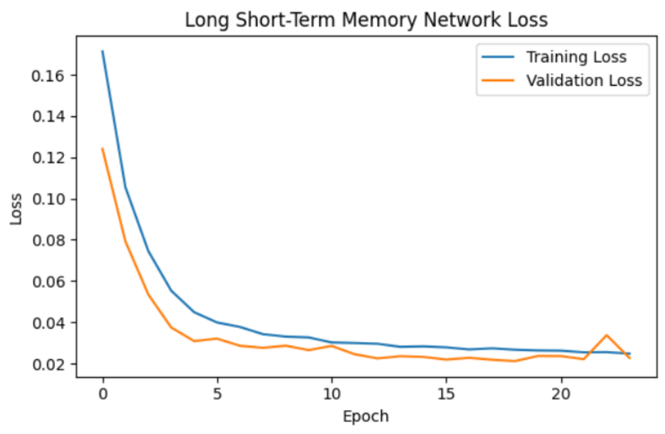


Fig.12 LSTM loss with varying epoch value

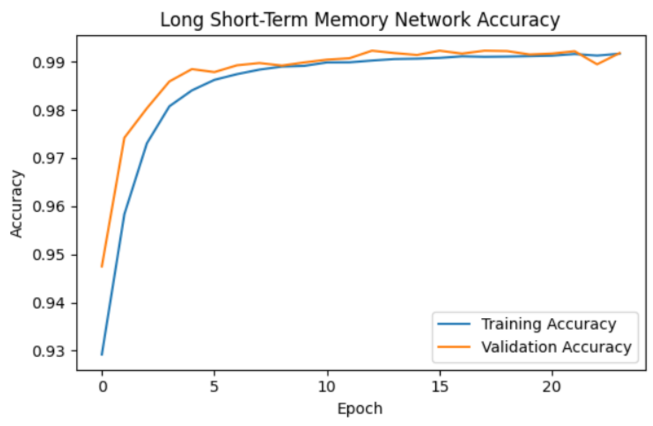


Fig.13 LSTM accuracy with varying epoch value

Now, looking at the confusion matrix of each of these models we can deduse that

[1] All traditional models (Logistic Regression, SVM, KNN, Gradient Boosting) and deep learning models (ANN, CNN, LSTM) achieve high accuracy in correctly classifying threats.The LSTM model outperforms both ANN and CNN slightly, particularly in reducing false positives.

[2] Gradient Boosting outperforms other traditional models by delivering the highest precision and recall, effectively minimizing both false positives and false negatives.In summary, if precision and recall are critical, LSTM seems to be the top-performing model here, with CNN as a close second.

[3] Among deep learning models, the LSTM slightly surpasses ANN and CNN in reducing false positives, enhancing overall classification reliability.

[4] Overall, Gradient Boosting and LSTM emerge as the top-performing models, offering the best balance of accuracy, precision, and recall for effective threat detection.

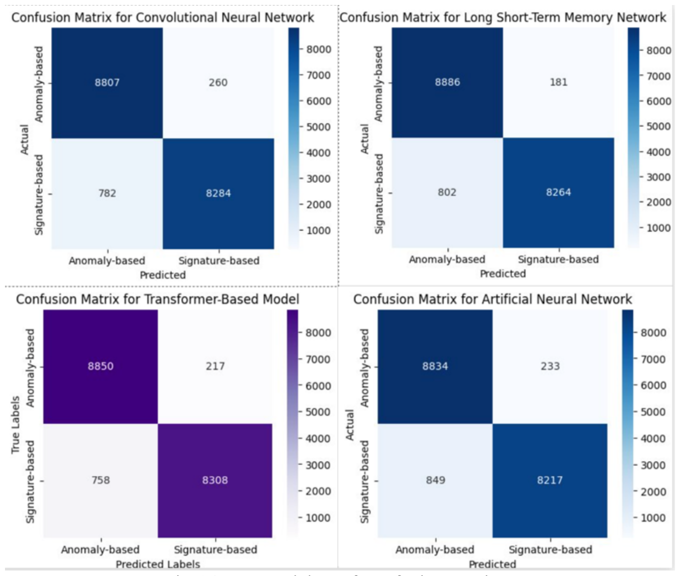


Fig.14 Comparision of confusion matices

Conclusion

ii. Final Conclusion

In summary, the comprehensive evaluation of both traditional and deep learning models for distinguishing between **Signature-Based** and **Anomaly-Based** cybersecurity threats reveals that **Gradient Boosting** and **Long Short-Term Memory (LSTM)** models consistently lead in performance across all key metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. **Gradient Boosting** excels among traditional models by delivering exceptional predictive power and effectively minimizing both false positives and false negatives, making it highly reliable for threat classification. On the deep learning front, the **LSTM** model slightly outperforms other architectures like **Artificial Neural Networks (ANN)** and **Convolutional Neural Networks (CNN)** by adeptly capturing temporal dependencies and reducing false positives, thereby enhancing overall classification accuracy.

Additionally, the **Transformer** model introduces a significant advancement in the detection of complex and evolving threats. Leveraging its self-attention mechanisms, the Transformer excels in identifying intricate patterns and global feature dependencies that may be indicative of sophisticated **Anomaly-Based Threats**. This capability allows the Transformer to perform in-depth analysis of threat data, recognizing subtle anomalies that other models might overlook. Its parallel processing efficiency further supports real-time threat detection, ensuring swift and accurate responses to emerging cyber threats.

While models such as **Support Vector Machines (SVM)** and **K-Nearest Neighbors (KNN)** perform commendably, they are marginally outperformed by **Gradient Boosting**, **LSTM**, and the **Transformer** in handling complex and dynamic threat patterns. Overall, for scenarios where high precision and recall are critical, particularly in detecting sophisticated and evolving threats, **Gradient Boosting**, **LSTM**, and **Transformer** models emerge as the top-performing solutions. These models offer the best balance of accuracy, robustness, and reliability, making them invaluable for effective real-time threat detection in advanced cybersecurity applications.

References

1. S. K. Sharma, A. V. P. Kumar, and S. Bera, “AI-Driven Anomaly Detection in Cybersecurity: Combating Emerging Worm Threats,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2041-2050, 2023.
2. Z. Zhang, M. Li, and Q. Zhou, “Machine Learning-Based Intrusion Detection Systems: A Comparative Study on Models and Datasets,” *IEEE Access*, vol. 11, pp. 48255-48265, 2023.
3. H. Yu, X. Liu, and Y. Lin, “Honeypot-Assisted Worm Detection Using Reinforcement Learning,” *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 1480-1490, 2023.
4. A. Gupta, S. Singh, and M. R. Joshi, “CNN-Based Zero Day Worm Detection Using Regularization Techniques,” *IEEE Access*, vol. 12, pp. 98001-98013, 2024.
5. K. R. Patel, M. K. Rao, and J. Park, “Deep Learning Approaches for Worm Detection in Network Traffic: Current Trends and Future Directions,” *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 150-172, 2024.