EfficientDense-ViT: APT Detection via Hybrid Deep Learning Framework with Hybrid Dipper Throated Sine Cosine Optimization Algorithm (HDT-SCO)

Khaled Almasoud[1,*]

[1]Chief Information Security Officer, General organization for Social Insurance, Riyadh, Saudi Arabia
Email: khaled.almasoud@hotmail.com

Abstract

Advanced Persistent Threats (APT) are intelligent, sophisticated cyberattacks that frequently evade detection by gradually interfering with vital systems or focusing on sensitive data. It is proposed herein the new approach of the Hybrid Dipper Throated Sine Cosine Optimization Algorithm (HDT-SCO) for APT detection in association with the EfficientDense-ViT model. It handles the class imbalance issue with advanced processing Adaptive Synthetic Minority Oversampling Technique (ADASYN), including min-max scaling for normalization, and median imputation for missing values. In terms of feature engineering, ResNet-152 and Symbolic Aggregate Approximation (SAX) are adopted for statistical, deep, and time series feature extraction. HDT-SCO optimizes the selection of relevant features to refine by integrating into it the three approaches: PCA, RFE, RF Feature Importance, and L1 Regularization (Lasso). Compared to current detection techniques, the best detection model shows high performance and efficiency through the hybrid deep learning model known as EfficientDense-ViT, which is a combination of EfficientNet, DenseNet, and Vision Transformers (ViT) that can detect APTs reliably. This method shows considerable improvement in both accuracy (0.98741 for the 70/30 split and 0.99143 for the 80/20 split) and efficiency as compared to existing models in the detection of APTs in cybersecurity.

Keywords*:* Cyber Security; APT Detection; Hybrid optimization; HDT-SCO; Deep learning; Vision Transformers (ViT); EfficientDense-ViT

1. Introduction

In the realm of cybersecurity, APTs are one of the most difficult challenges due to their covert and enduring traits. Usually, the APTs are executed by a talented enemy, who is the infiltrator of the network, which sustains even months, while the attacker accomplishes their objective such as sensitive information theft or infrastructure sabotage [1]. KFSN or Quad- Feature Selector Network has been defined as a neural network structure aimed at feature selection in machine learning-oriented tasks where such selection of features enhances the performance of the model [2-3]. A QFSN aims to accomplish this by utilizing a certain topology associated with the network which is used to sieve through several inputs that are available and select the most relevant ones [4]. Dense-ViT (Dense Vision Transformer) is a modification of the Vision Transformer (ViT) model which aims to perform image processing by means of transformer models which are primarily developed for processing natural language texts [5]. The Dense-ViT model extends the basic ViT model by addressing the inbuilt constraints of attention with dense attention and expanding the connectivity of the network thus the performance of the model are enhanced in processing image data [6].

EfficientDense-ViT addresses two significant challenges within the APT detection space: feature selection and computational efficiency. The first main block of the architecture is the Quad-Feature Selector Network (QFSN), which features a dynamic selection of the most informative aspects from a huge variety of input data [8]. This process helps in data dimensionality reduction by eliminating irrelevant or redundant features while preserving vital information for accurate attack detection [7-8]. Thus, by concentrating only on the most critical features, QFSN improves the model's performance while simultaneously decreasing overfitting risk; thus, it leads to a faster as well as an accurate detection process. The second component that constitutes the EfficientDense-ViT framework is combining Vision Transformer (ViT) with DenseNet: a convolutional neural network that

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

emphasizes feature reuse and efficient gradient flow through its architecture [9]. The dense connectivity pattern of DenseNet allows this architecture to learn rich features at various degrees of abstraction: low-level patterns and high-level contextual information are captured quite well [10]. Consequently, this will allow the system to take advantage of ViT's picture classification efficiency without having to make up for its computational costs. It ensures that there is hybridization, thus ensuring that the model is not only computationally sustainable but also suitable to address the complex ability of APTs to attack [11].

EfficientDense-ViT builds on the strengths of these two recent architectures to design an all-inclusive detection system. Since both DenseNet and ViT modules are employed to process the input data, the architecture is capable of identifying both space and time trends in the attack data, thus aiding in the Efficient Advanced Persistent Threat detection across various scenarios [12]. In addition, the present implementation employs QFSN which keeps the model relevant to the most important features, improving its capability to tell what is a normal activity, and what is a malicious one [13]. This paper is expected to argue that the EfficientDense-ViT is a revolutionary approach in combating APTs. The proposed framework, through advanced feature selection, effective processing, and hybrid neural architectures provides a timely and precise APTs detection method. The experiment results will also indicate that EfficientDense-ViT is more efficient than the already existing state of the arts in methods that are relevant to cybersecurity, thus providing a major solution to the ever-changing threat in the cyberspace. The primary contribution of this study is:

- It integrates complex preprocessing techniques for quality data that can be used with models, including the normalization application of min-max scaling, median imputation handling missing data cases, and applying ADASYN for class imbalance.
- Improving the depiction of network traffic and the behavior of the system with high-tech deep learning techniques such as symbolic aggregate approximation for time-series data and ResNet-152, used in extracting high-level deep features.
- The study presents the Hybrid Dipper Throated Sine Cosine Optimization Algorithm (HDT-SCO) for feature selection, which combines several optimization methods such as L1 Regularization (Lasso), PCA, RFE, and RF Feature Importance to improve feature extraction and model performance.
- The study suggests the EfficientDense-ViT model, a hybrid architecture that integrates Vision Transformers (ViT), DenseNet, and EfficientNet for the detection process, offering a reliable method for precisely and effectively detecting APT attacks.

The structure of this paper is: The literature review is covered in Section 2, the technique is presented in sufficient detail in Section 3, the results and discussion are covered in Section 4, and the conclusion is summarized in Section 5.

2. Proposed Methods

To increase network security, this method evaluates and improves internet security standards. Five programs work together to solve important security challenges. Using the AES Enhancement Algorithm, dynamically generated keys strengthen encryption. This also mitigates key static risks. Initializing keys, replacing them, moving rows, merging columns, and dynamically changing keys are required. To detect prospective attacks, the Dynamic Intrusion Detection Algorithm computes "anomaly scores" from AES output and dynamically alters weights. User verification is safe using biometric-based multi-factor authentication. Combining biometric data with additional authentication factors achieves a flexible authentication score. To protect communication, the Safe Key Exchange Algorithm for Public Key Infrastructure (PKI) exchanges public keys and derives shared keys using modular exponentiation. Based on implemented measures and SDLC stages, the Secure SDLC Integration Algorithm automatically adjusts security settings. Five additional security measures might improve the answer. The Biometric-Based Multi-Factor Authentication Algorithm verifies users' identities. The Secure SDLC Integration Algorithm incorporates software safety features. The AES Enhancement Algorithm improves security. Internet security improvement is well planned. This approach, which uses equations, performance assessments, and rigorous measurements, provides a solid foundation for improving network safety and assessing the pros and cons of current security solutions. Assessments using mathematical methods are fair and precise, revealing the efficacy of any security system.

3. Literature review

Authors [14] have developed machine-learning tools designed for the classification and detection of cyber espionage activities using open-source information. The Deploy three methods of research such as Deep and Machine learning consists of multilayer perceptrons, random forests, decision trees, and convolutional neural networks. Among the mixed ensemble ML methods, the approach that combined XGBoost and random forest

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

classifiers yielded a 0.52% false positive rate and an ultimate prediction accuracy of 98.92%. The experiment's findings demonstrated that the model outperformed the others on every dataset.

Researchers [15] have proposed a new DL-based technique utilizing network flow to identify APT attacks. First, the network data was divided according to IP-based network flows; secondly, reconstructing IP information of each flow; and finally, applying DL algorithms to infer features which could distinguish APT attack IPs from others. Besides, a proposed hybrid DL system using GCN and BiLSTM techniques. This mechanism performed the best evaluation result from others.

To increase the angle of effectiveness of APT attack detection in deep learning systems, authors [16] presented a novel attention network-based technique in 2023. The capture concealed and annihilate mechanism was studied, implemented, and tested using a CNN-LSTM deep learning network on full network traffic data. Instead of feeding the results directly to a classification, the input was examined closely and evaluated by attention network. Lastly, the attention network's classified output data identified the APT attacks. The results depicted how the suggested approach enhanced the efficiency of network traffic analysis and APT virus detection.

Researchers [17] have proposed a solution through the use of Deep learning Bayesian nets, C5.0 decision trees all of which in this case fall under machine learning strategies utilizing the NSL-KDD dataset and carried out in-depth research concerning advanced persistent threats APTs. These models were estimated using a 10-fold cross-validation technique. Consequently, the accuracy recording (ACC) for the C5.0 decision tree algorithm, the Bayesian system, and the six layers of deep neural algorithms displayed values of 95.64%, 88.37%, and 98.85%, respectively. In addition, the designed FDR values (3R, 4R and 5R system) were 2.56 for deep learning algorithm of 6 layers, 10.47 for bayesian networks and 1.13 for C5.0 decision tree algorithm, for the target area of FPR.

A new method for feature selection and data balance optimization in the context of APT detection was created. A hybrid HHOSSA was created that takes advantage of the Sparrow Search Algorithm and Harris Hawk Optimization. In addition, hybrid HHOSSA optimization was also suggested for that purpose to improve weighted average Bi-LSTM and light GBM. To increase the classifier efficiency, HHOSSA-SMOTE adapted the balanced variations of the two movement’s inclined directional lateral movements and information exfiltration. The accuracy of the result had been very high on the resultant output, at 94.468% and with sensitivity at 94.665% and specificity at 95.23% [18-19].

Authors [20] presented three layers architecture in 2023, which includes Graph Convolutional Networks, Inference and Multi-layer Perceptron.The IP information profiles were built on the aggregation and clustering of flow networks which were developed from a particular IP address. An MLP layer was employed to perform aggregation and feature extraction of the IPs based on the traffic of the flow network in question. The GCN layer carried out feature analysis and reconstruction of the Ips via behavior extraction from the IP information files.

Researchers [21-23] have proposed the FIE method in 2024 which was built based on two other methods: representation learning and feature intelligent extraction. More specifically, the proposed FIE approach combined BiLSTM for feature intelligent extraction with the Attention network to get odd behaviors of APT IP embeddings from network data. The proposed RL approach in the study used contrastive learning and data rebalancing as its two main methods for improving the classification of APT and normal IPs. In particular, this rebalancing approach with testing datasets helped to properly train the data.

Authors [24] presented a new and efficient tool to detect advanced persistent threats (APT) based on deep convolutional neural network, called Smart Flower Cosine Algorithm. SFCA was used to augment the weight of DeepCNN, which executed the task of detection of APT attacks. It used the Sine Cosine Algorithm with superior Smart Flower Optimization Algorithm in doing so while obtaining this. Sample quantiles also used for optimization using the Quantile normalization in pre-processing. While selecting relevant features, Fuzzy distance calculation was used. On the other hand, a deep learning approach proffered in this very work successfully tested the result of finding APT attacks.

Researchers [25] introduced a novel multi-dimensional hybrid Bayesian belief network architecture for classifying malware as either benign or dangerous. The research offered a hybrid strategy to address the general diversity of malware behavior characteristics using three analysis methods: dynamic analysis Bayesian belief network, static analysis Bayesian belief network, and event assessment Bayesian belief network. The designed framework accomplished the detection of APT malware with an exceptional sensitivity of 92.62 percent and a remarkably low false positive rate of 0.0538 percent, in a setting where security controls could not be considered robust.

3.1 Problem statement

An APT is a targeted and aggressive cyber-attack against a given system or organization with a primary goal of system infiltration. Defense systems that rely on conventional methods of detection have little chances of

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

successfully detecting APTs due to their rich embedded tactics that are made up of several stages and that can lie within a network owner or an organization for a long time. This makes the organization liable to data compromise, loss of money and even damage to the organization's good name. Additionally, [21] improved the machine learning APT recognition and classification model by employing a number of algorithms and achieved a remarkable prediction accuracy of 98.92% at a false positive rate of 0.52%. This approach was applicable and yielded satisfactory results across out of the shelf datasets but it was based on a deep classifier which tends to be compute expensive. Also, [22] described a deep learning-based work using network flow in which hybrid BiLSTM-GCN model reached the highest accuracy among the evaluation scores. However, this approach requires extensive pre-processing which can affect scalability. Then, [23] added a deep learning APT detection system applying CNN with LSTM, which enhanced APT detection effectiveness. However, the extra attention layer might add to the complexity of the model and the time taken to process the model. Then, a Smart Flower Cosine Algorithm driven DeepCNN was suggested, which achieved efficient APT detection with fuzzy based feature extraction but still required heavy pre-processing to reach the best performance. Lastly, utilized a hybrid Bayesian belief network to detect malware, which produced a false positive rate of 0.0538% and a classification accuracy of 92.62%. The multi-dimensional techniques ensure accuracy but they also require considerable computational resources owing to the different layers in Bayesian analysis approaches.

4. Proposed Methodology

This research proposes the Hybrid Dipper Throated Sine Cosine Optimization Algorithm (HDT-SCO with EfficientDense-ViT (HDSEDV) model for APT attack detection. In the preprocessing phase, cleaning is done by imputing missing values by median imputation and normalizing the data using min-max scaling. Then the use of ADASYN balances the data where cases of class imbalance are encountered on the data set. In the process of feature engineering, statistical features including network traffic data and system behavior data were extracted. High-level deep features are extracted using ResNet-152, which consists of convolution layers, residual connections, MaxPooling, and fully connected layers. The time series feature extraction is performed using Symbolic Aggregate Approximation (SAX). Feature selection is then performed with a hybrid optimization technique namely the Hybrid Dipper Throated Sine Cosine Optimization Algorithm (HDT-SCO), where the first step is the PCA to diminish the data dimensionality by mapping it into a new space defined by the principal components that explain the most variance. The least significant features are recursively eliminated using Recursive Feature Elimination (RFE), which enhances the model's effectiveness and interpretability. RF Feature Importance is used to rank features based on how they contribute to splitting decision trees. L1 Regularization (Lasso) is used to select features by making some coefficients equal to zero. The workflow ends with DL-Based APT Attack Detection using the EfficientDense-ViT model which is a new architecture that combines EfficientNet, DenseNet, and Vision Transformers (ViT) for attack detection, which offers a strong basis for identifying advanced persistent threats. Figure 1 depicts the designed model's overall process architecture.

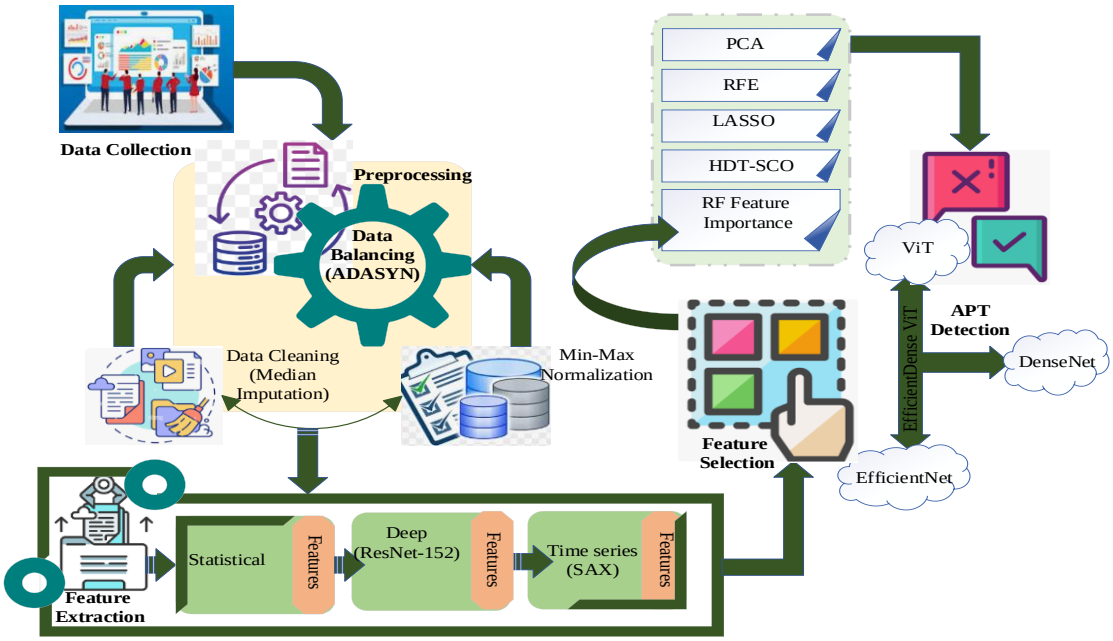

Figure 1. Overall workflow of the developed framework

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

4.1 Data Collection

DAPT 2020 was created using network traffic collected over a period of five days, or three months in real time. The purpose of this dataset was to assist scholars in comprehending the connections among various attack vectors, detecting anomalies, and revealing any obscure associations that could aid in the early detection of an APT attack.

4.2 Pre-processing

Pre-processing stage- Data Cleaning with Median Imputation and Data Normalization through Min-max normalization are used:

4.2.1 Data Cleaning

Median Imputation: In median imputation, the median value of the respective feature is used to replace any missing or null values in the data. The median is used as the average value here because the former is less sensitive than the latter to outliers to have a good robust imputation. This process helps avoid biasing and guarantees the consistency of the dataset. Median imputation can be very helpful in numerical data where missing values are very frequent, but the underlying distribution should be the one to prevail. Let the dataset X with features $X_1, X_2, \cdots, X_n$, where some values are missing. These missing values are replaced with the median of the respective feature in the dataset. The median imputation for a feature X_j (for the missing values) is determined in eq. (1).

$$X_j^{imputed} = Median\ (X_j) \tag{1}$$

Where X_j denotes the j-th feature column. Finally, the feature's median value is used in place of any missing values in a specific row for feature j.

4.2.2 Data Normalization

The min-max normalization is used to normalize the data and this scaling approach scales the values of the dataset range between 0 and 1 based on the minimum and maximum values of each feature. The min-max normalization for each feature X_j is determined in eq. (2).

$$X_j^{normalized} = \frac{X_j - min(X_j)}{(X_j) - min(X_j)} \tag{2}$$

Where the minimum and maximum value in the feature X_j is represented as $min\ (X_j)$ and (X_j) . Before the data is fed into the following phases of the EfficientDense-ViT framework, these processes aid in cleaning and normalizing it.

4.2.3 Data Balancing

A data balancing technique called adaptive synthetic minority oversampling approach, or ADASYN, corrects class imbalance in datasets by producing synthetic samples for the minority class. X_{min} and X_{maj} are the minority and majority classes in the dataset, respectively, while $N_{min}$and N_{maj} are the number of samples in the minority and majority classes. The first step is to identify the minority and majority classes. ADASYN's objective is to produce k synthetic samples for every minority class sample according to its degree of difficulty (density of nearby majority samples). The number of synthetic samples (N_{syn}) that should be made for each minority sample depends on the number of majority class samples in the near vicinity of each minority sample.

The density measure, often known as the difficulty level, counts the number of majority samples immediately surrounding each minority class sample, x_i. Finding the k-nearest neighbors (k-NN) Euclidean distance is usually how this is done. The number of majority neighbors divided by the total number of neighbors determines the difficulty level δ_i of a sample x_i.

$$\delta_i = \frac{Number\ of\ majority\ class\ neighbors\ of\ x_i}{k} \tag{3}$$

Furthermore, the synthetic sample N_{syn} is generated for each minority sample x_i. These synthetic samples are produced by interpolating between x_i and one of its KNNs from the majority class. The interpolation is determined in eq. (4).

$$x_{new} = x_i + \lambda \cdot (x_{NN} - x_i) \tag{4}$$

Where x_{NN} represents the randomly selected K-nearest neighbor from the majority class. λ denotes the random value between 0 and 1, that controls the interpolation among x_{NN} and x_i. Finally, the dataset is balanced with more minority class examples after the synthetic samples for the minority class are generated.

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

4.3 Feature Extraction

Here, the time-series, deep, and statistical characteristics for the APT attack detection framework are used to extract features.

4.3.2 Statistical Features

The statistical features incorporate the following attributes of the data: network traffic, flow duration, byte count, packet count, entropy of packet size, system behavior, CPU utilization, memory utilization, and disk utilization.

i) Network Traffic: The flow statistics features, including flow time, byte count, and packet count, are assessed to obtain relevant network traffic data.

ii) Flow duration: By measuring the time lag between the start and finish of the network flow, its duration may be determined. Let f_{start} and f_{end} stand for the flow's start and end times, which are determined by applying eq. (5).

$$d = f_{end} - f_{start} \tag{5}$$

Where d is the network traffic

iii) Byte Count: It is quantified by count the total number of bytes that are moved during the network flow. Eq. (6) is used to calculate the bytes that represent the summary of each packet in the flow.

$$b = \sum_{i=1}^{n} \quad B_i \tag{6}$$

Where B_i is the i^{th} packet size, and n is the overall packets

iv) Packet count: The total number of packets is a representation of a network flow. The symbol for the packet count is n.

v) Entropy of packet size: To find irregularities in data delivery, entropy measurements of packet sizes are employed. It determines how irregular or unpredictable the network flow's packet sizes are. Eq. (7), which uses the probability distribution $p(k)$, yields the packet size.

$$h(K) = -\sum_{k \in K} \quad p(k)p\,(k) \tag{7}$$

Where K is the collection of unique packet sizes.

vi) System Behaviour: The resource utilization statistics are analyzed with CPU, memory, and disk storage.

vii) CPU utilization: This is used to derive the properties of the CPU usage in eq. (8)

$$u_{cpu} = \frac{T_{user}+T_{system}}{T_{user}+T_{system}+T_{idle}} \times 100 \tag{8}$$

Where, T_{user}, T_{system}, and T_{idle} are the time spent for the modes of user, system, and idle.

viii) Memory utilization: It is used to extract the properties of the memory use percentage, calculated using eq. (9).

$$u_{mem} = \frac{M_{used}}{M_{total}} \times 100 \tag{9}$$

Here, M_{used}, and M_{total} are the used memory and total memory.

ix) Disk utilization: Its properties are derived using eq. (10), which is a measure of the percentage of disk usage.

$$u_{disk} = \frac{D_{used}}{D_{total}} \times 100 \tag{10}$$

Where D_{used} and D_{total} are the used disk and total disk space.

4.3.2 Deep features

A deep convolutional neural network (CNN) called ResNet-152 is used to extract deep features from data, especially images. Its pre-trained layers are used in the process to produce a high-level, lower-dimensional feature representation. Let the input data be represented as a tensor X of dimensions (H, W, C) where H is the height, C is the number of channels, and W is the width. ResNet-152 is used to extract the features using a sequence of pooling layers, residual blocks, and convolutional layers. The forward pass is determined in eq. (11)

$$F = f_{ResNet}(X) \tag{11}$$

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Where f_{ResNet} is the feature extraction function of ResNet-152, F represents the extracted feature vector of size d. Following network processing, the global average pooling (GAP) layer condenses the spatial dimensions into a single feature vector:

$$F_{deep} = GAP(F) = \frac{1}{H'W'}\sum_{i=1}^{H'} \sum_{j=1}^{W'} F[i,j] \quad (12)$$

Where, F_{deep} indicates the final dep feature vector, the dimensions after convolutional operations are said to be H' and W'.

4.3.3 Time Series Features

To extract symbolic representations from time-series data, a technique called Symbolic Aggregate Approximation (SAX) [34] transforms the data into a collection of discrete symbols. Assume that time series data is represented as $T = \{t_1, t_2, \cdots, t_n\}$, where n is the number of points. First, set the time series' mean to zero and its standard deviation to one by normalizing it.

$$T_{norm} = \frac{T-\mu}{\sigma} \quad (13)$$

Where μ denotes the mean of T, and the standard deviation of T is said to be σ. In the Piecewise Aggregate Approximation (PAA), the normalized time series T_{norm} is divided into w equal-sized segments. The mean value is computed for each segment in eq. (14).

$$P_i = \frac{1}{\frac{n}{w}}\sum_{j=(i-1)\frac{n}{w}+1}^{i\frac{n}{w}} T_{norm}[j], \qquad i = 1,2,\cdots,w \quad (14)$$

Where $P = \{P_1, P_2, \cdots, P_w\}$ is the PAA representation. Then, symbolic conversion is performed by mapping each segment's mean P_i into the discrete symbol based on the pre-defined breakpoints which divides the Gaussian distribution into α equal-sized regions. The mapping is determined in eq. (15).

$$S_i = symbol\ (P_i), \qquad i = 1,2,\cdots,w \quad (15)$$

Where, the symbolic representation of the time series is $S = \{S_1, S_2, \cdots, S_w\}$. The sequence S is the last SAX representation, condensing the time series into a string of symbols. These methods are crucial for further analysis throughout the framework because this method extracts significant characteristics from time-series data (symbolic representations), statistical features, and images (deep features).

4.4 Feature Selection

Feature selection is another important stage in machine learning and deep learning. There the primary motive of reducing the number of features, while dealing with the dataset is going further to select the most significant characteristics that will help the machine in the prediction task. These methods combine various feature selection methods to enhance the selection process and bring out the best result. All the above PCA, RFE, Lasso, and RF techniques fall within the scope of the proposed Quad-Feature Selector Network. For feature dimensional reduction, PCA is primarily used to reduce the dimension of the features retrieved; then, the feature having the lowest relevance score or coefficient is eliminated using the RFE. The RF approach is used to calculate the feature importance score for the target variable for the high-priority features and the hybrid HDT-SCO algorithm is used to select the feature for APT attack detection.

4.4.1 Principal Component Analysis (PCA)

PCA [35] allows to minimize the feature space size with creation of new features, orthogonal to each other. Ranking the components is defined by the variation in the original data. The data is projected using this dimension reduction technique with the aim to preserve as much variance as possible. For a dataset z with N samples and M features, determine the covariance matrix $\bar{c}$ using eq. (16).

$$\bar{c} = \frac{1}{N-1} z^t z \quad (16)$$

where z is denoted as centered (every feature's mean is zero). The covariance matrix's eigenvalues and eigenvectors are determined using eq. (17).

$$\bar{c} = v \wedge v \quad (17)$$

where v is denoted as an eigenvector matrix and $\wedge$ is considered a diagonal eigenvalue matrix. Then sort the eigenvectors by the eigenvalues in decreasing order of the eigenvalues. The feature transformation is done by choosing the first few eigenvectors K (principal components) associated with the largest eigenvalues, which explain the most variance. The original data onto this reduced set of components is projected using eq. (18).

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

$$z_{new} = zv_K \quad (18)$$

where v_K K is denoted as the top K eigenvectors. PCA makes the data less complex, simplifying the model and preventing overfitting.

4.4.2 Recursive Feature Elimination

RFE is a feature selection approach that relies on feature deletion and wrapping. This model is retrained on fewer features until the most relevant features are achieved, after that, the least relevant features are removed. Then, a Random Forest or linear model is chosen to determine feature weights. In linear models, feature significance j is determined by the coefficient's absolute amount ϕ_j using eq. (19).

$$(j) = |\phi_j| \quad (19)$$

The model is then trained once again after removing the feature with the lowest significance score or coefficient. This process should be repeated until the desired number of features is obtained. The received importance scores are then used to sort the remaining features. To improve the model's accuracy and interpretability, RFE entails removing the least significant characteristics.

This approach calculates the feature importance score for the target variable using the RF algorithm, an ensemble learning technique. An RF model is trained using the dataset. The degree to which a feature reduces the entropy, or Gini index, of the trees is how RF determines the feature's significance. A trained RF chooses the most relevant features based on the feature significance ratings. The relevance of each feature j is calculated by assessing its contribution to the reduction of impurities (e.g., the Gini impurity or entropy) in decision trees. Eq. (20) is used to calculate the feature's relevance score j.

$$(j) = \frac{1}{n}\sum_{t=1}^{t} \Delta I_{j,T}$$

(20) where $\Delta I_{j,T}$ is denoted as an attribute j responsible for the entire impurity reduction in the tree T and n is considered several trees in the RF. The features with the greatest importance ratings should next be chosen after sorting the features according to their importance scores using Random Forest Feature Importance.

4.4.3 L1 Regularization (Lasso)

L1 Regularization, or the Least Absolute Shrinkage and Selection Operator (Lasso), applies a penalty on the loss function equal to the absolute value of the coefficients. By setting some feature coefficients to zero, this penalty selects features and encourages sparsity. Moreover, the best characteristics are selected and attempt to make the model sparse. In Lasso regression, the aim is to reduce the cost function using Eq. (21).

$$j(\alpha) = \frac{1}{2N}\|y - z\alpha\|_2^2 + \lambda\|\alpha\|_1 \quad (21)$$

Where $\|y - z\alpha\|_2^2$ is denoted as mean squared error, $\|\alpha\|_1$ is considered the L1 norm of the coefficients, and $\lambda$is denoted as a regularization parameter. Some of the coefficients α_j are forced to be exactly zero by the L1 regularization term. The characteristics associated with non-zero coefficients are considered significant. The L1 Regularization (Lasso) eliminates the less significant features, lowers the chance of overfitting, and improves interpretability by shrinking their coefficients to zero.

4.4.4 Hybrid Optimization for Feature Selection

The feature selection is done based on the hybrid dipper throated optimization and the Sine Cosine optimization (HDT-SCO) algorithm for APT detection.

i) DTO Algorithm

The DTO simulates the real process of identifying the locations and velocities of swimming and flying birds in order to locate food. Equation (22) is used to update the swimming birds' position and speed.

$$BL_{nd}(t+1) = BL_{best}(t) - C_1 \cdot |C_2.BL_{best}(t)| - BL_{nd}(t) \quad (22)$$

where C_1 and C_2 are the adaptive values, t is the iteration number, and $BL_{nd}(t)$ and $BL_{best}(t)$ indicate the bird's normal and optimal locations, respectively. During the optimization process, these values are modified according to the number of iterations and random values. Conversely, use equations (23) and (24) to update the flying bird's location.

$$BL_{nd}(t+1) = BL_{nd}(t) + BS(t+1) \quad (23)$$

$$BS(t+1) = C_3 BS(t) + C_4 r_1\big(BL_{best}(t) - BL_{nd}(t)\big) + C_5 r_1(BL_{Gbest} - BL_{nd}(t)) \quad (24)$$

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

where each bird's updated speed is represented by $BS(t+1)$, the global best location is indicated by BL_{Gbest}, the weight value is C_3, the random number in [0,1] is indicated by r_1, and the constants $and\ C_4$ and C_5 are present.

ii) Sine Cosine Optimization Algorithm

To find the best solution locations for the feature selection, the Sine Cosine (SC) Optimization technique is essential. The following random variables are used to define SC operations: the direction of motion, the position of the movements, the switching of sine and cosine components, and the emphasis or lack thereof on the destination effect. The candidate solutions' updating procedure is carried out using Equation (25).

$$P(t+1) = \{P(t) + r_5 \cdot sin\ sin\ (r_6)|r_7 S^*(t) - S(t)| \qquad r_4 < 0.5\ \ P(t) + r_5 \cdot cos\ cos\ (r_6)|r_7 S^*(t) - S(t)| \qquad r_4 \geq 0.5 \tag{25}$$

where t represents the quantity of search iterations. Two important solutions are tracked by the method: the current solution, denoted by S, and the optimal solution, represented by $\llbracket S^*$. These values assigned to the random variables r_4, r_6, and r_7 lie between 0 and 1. Due to their ability to affect the locations of the solutions, these random variables are essential to the method. In particular, the algorithm's equation shows that the position of the current solution is influenced by the best solution location found so far. This influence raises the possibility of finding the best answer and makes it easier to explore the search space. The search process is further improved by dynamically updating the value of r_4 by the following equation while the SC algorithm proceeds with its iterations.

$$r_4 = a - \frac{a \times t}{t_{max}} \tag{26}$$

The constant is denoted by a, while the current and maximum iterations are denoted by t and t_{max}, respectively. The SC algorithm is a reliable metaheuristic method among the many existing algorithms. It differs in that it can use one ideal solution to direct the other options. This approach's notable reduction in memory usage and convergence time sets it apart from other techniques. It is important to understand that as the SC algorithm comes across more local optima, its efficacy could be compromised. The DTO algorithm and the SC optimizer are both incorporated into the novel feature selection technique.

4.5 DL-based APT Attack Detection

The EfficientDense-ViT model leverages the strengths of EfficientNet, DenseNet, and Vision Transformer (ViT) to detect Advanced Persistent Threats (APTs).

4.5.1 EfficientNet

The EfficientNet detects the APT attack from the selected feature as the input at this stage using compound scaling of depth, width, and resolution. It processes the input X via a series of convolution layers which yield feature map F_{EFF}.

$$F_{EFF} = f_{EfficientNet}(X) \tag{27}$$

Where, $F_{EFF} \in R^{H' \times W' \times C'}$ represents the downsampled feature maps. The EfficientNet scales the network through compound scaling in eq. (28).

$$d = \alpha \cdot r, \qquad w = \beta \cdot r, \qquad r = \gamma \cdot r \tag{28}$$

Where, d, w, r are the depth, width, and resolution scaling factors, and α, β, γ are the constants optimized for EfficientNet.

4.5.2 DenseNet

DenseNet captures feature dependencies by connecting each layer to every other Layer, which produces densely connected feature maps F_{Dense}. Each layer l in DenseNet is connected to all previous layers.

$$F_l = g([F_0, F_1, \cdots, F_{l-1}]) \tag{29}$$

Where, $g(\cdot)$ is the composite function and $[\cdot]$ represents the concatenation of feature maps. The output feature map from DenseNet $F_{Dense} \in R^{H' \times W' \times C'}$ is determined in eq. (30).

$$F_{Dense} = f_{DenseNet}(X) \tag{30}$$

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

4.5.3 Vision Transformer (ViT)

The ViT detects the APT attack features from sequences by dividing the input into patches and processing using a self-attention framework. In patch embedding, the input image X is splitted into N patches of size $P \times P$. Then, each patch is flattened and embedded into the lower-dimensional space.

$$Z_0 = [E(x_1)]; E(x_2); \cdots; E(x_N)] + E_{pos} \quad (31)$$

Where $E(x_i)$ denotes the linear embedding of patch x_i, the positional encoding is said to be E_{pos}. Then, the embeddings Z_0 are passed through L transformer layers, and each of these is applied by multi-head self-attention (MSA) and feedforward networks (FFN).

$$Z' = MSA(Z) + Z \quad (32)$$

$$Z = FFN(Z') + Z' \quad (33)$$

The output from ViT is determined in eq. (34)

$$F_{ViT} = Z_L \quad (34)$$

Where, $F_{ViT} \in R^{N \times D}$, the embedding dimension is said to be D. Figure 2 shows the architecture of the EfficientDense-ViT model

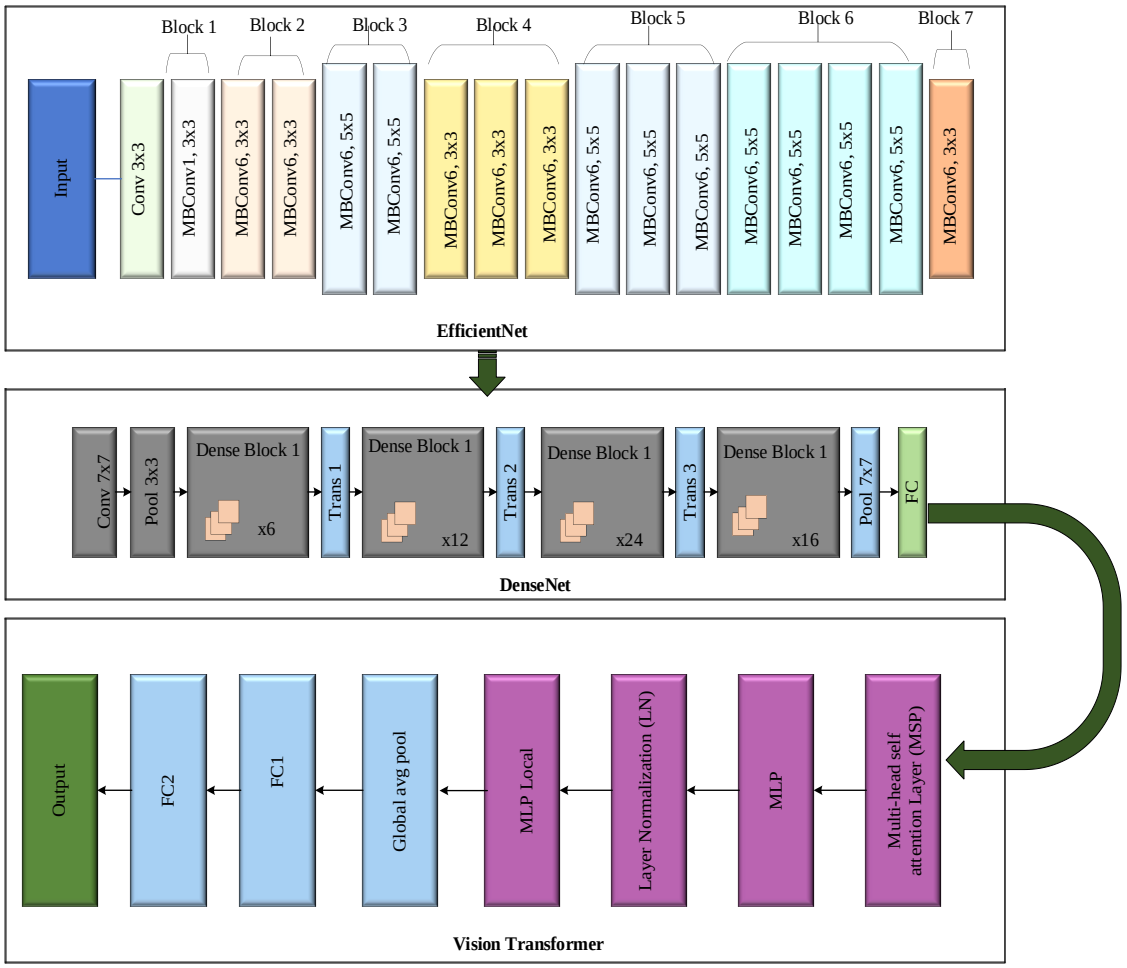

Figure 2. EfficientDense-ViT model architecture

4.5.4 APT Detection

The feature mapping by the EfficientNet, DenseNet, and ViT are concatenated into a single feature vector F_{fusion}.

$$F_{fusion} = [F_{EFF}; F_{Dense}; F_{ViT}] \quad (35)$$

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

These fusion captures both the spatial contextual information from the input selected features. Next, these fused features F_{fusion} are passed to the fully connected layer for APT attack detection. The output is the probability score of each class (APT or benign). The fully connected layer is determined in eq. (36).

$$y_{logits} = W \cdot F_{fusion} + b \tag{36}$$

Where W and b are the weights and biases of the classifier. The softmax activation is determined in eq. (37).

$$y = softmax(y_{logits}) \tag{37}$$

Where the predicted probabilities for each class is y. After that, the model is trained using the Cross-entropy loss function.

$$L = -\frac{1}{N}\sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij}^{true} \cdot log\ log\ (y_{ij}^{pred}) \tag{38}$$

Where N is the samples, C is the number of classes and the true and predicted probabilities for class j of sample i are y_{ij}^{true} and y_{ij}^{pred}. The predicted class is determined in eq. (39).

$$\hat{y} = argmax_{j}^{Y_j} \tag{39}$$

Where $\hat{y}$ is the detected class label? The EfficientDense-ViT model leverages the strengths of EfficientNet, DenseNet, and Vision Transformer (ViT) to detect Advanced Persistent Threats (APTs).

5. Result and Discussion

The findings of this study's APT attack detection observation, which was based on industry-standard datasets, are explained in detail in this section. Overall, the outcomes are contrasted with current methods. To test and assess the mathematical modeling of the proposed method, the dapt2020 dataset [25] (https://www.kaggle.com/datasets/sowmyamyneni/dapt2020) is used in Python, and the model is tested using the standard simulation parameters.

5.1 Performance Analysis

The comparison of the overall model's performance on a specific dataset is evaluated using two data splits: 70/30 and 80/20. The performance metrics compared include Accuracy, Precision, Sensitivity, Specificity, F-measure, Negative Predictive Value (NPV), Matthews Correlation Coefficient (MCC), False Positive Rate (FPR), and False Negative Rate (FNR). The acquired outcomes are exhibited in Fig.5 to Fig.13, respectively. The existing techniques used to validate the developed model performances are CNN-LSTM [23], Bi-GRU , and Autoencoder.

5.1.1. Accuracy

For APT detection, accuracy is defined as the percent instances correctly recognized out of all examined ones-these are the threats and the rest being non-threats-accounting for the overall efficacy of the technique. It shows the overall capability of the mechanism to distinguish harmful activity from benign activity.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Instances} \tag{40}$$

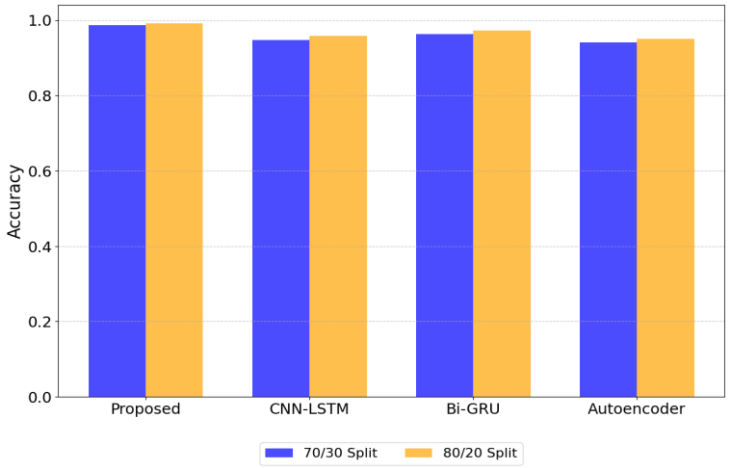

Figure 3. Comparison of Accuracy

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Figure 3 Accuracy Comparison. The proposed model obtains the highest accuracy for both data splits, which is represented by the values of 0.98741 in the case of the 70/30 split and 0.99143 in the case of the 80/20 split. It means that it correctly predicts the larger number of cases and surpasses other models with a higher distinguish between positive and negative class accuracy. Comparing these, the results for CNN-LSTM, Bi-GRU, and the Autoencoder models reflect a lower precision. For instance, CNN-LSTM achieved 0.94766 and 0.95893; Bi-GRU achieved 0.96299 and 0.97299; the Autoencoder, scored 0.94123 and 0.95123 respectively for the 70/30 split and the 80/20 split.

5.1.2 Precision

Precision, also known as Positive Predictive Value, assesses how well the system detects APT threats out of all the positive examples that have been detected. It highlights how the model can accurately identify threats while reducing false alerts.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{41}$$

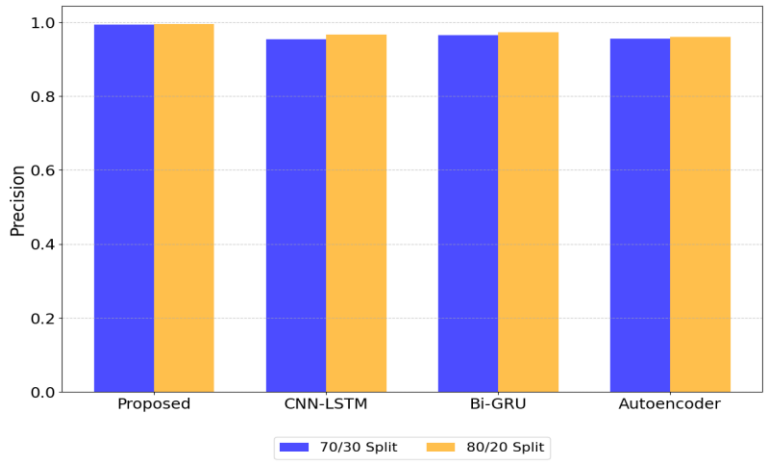

Figure 4. Comparison of Precision

Figure 4 presents a comparative analysis of precision. The proposed model attains the highest precision, with values of 0.99407 and 0.9949 for the 70/30 and 80/20 splits, respectively. CNN-LSTM achieves precision values of 0.95368 and 0.96631, Bi-GRU scores of 0.96542 and 0.97229, and Autoencoder gets 0.95523 and 0.95986 for the respective data splits.

5.1.3 Sensitivity

The model's sensitivity measures how well it can detect APT threats. Its primary goal is to identify malicious activity occurring within the network, to minimize missed detections.

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{42}$$

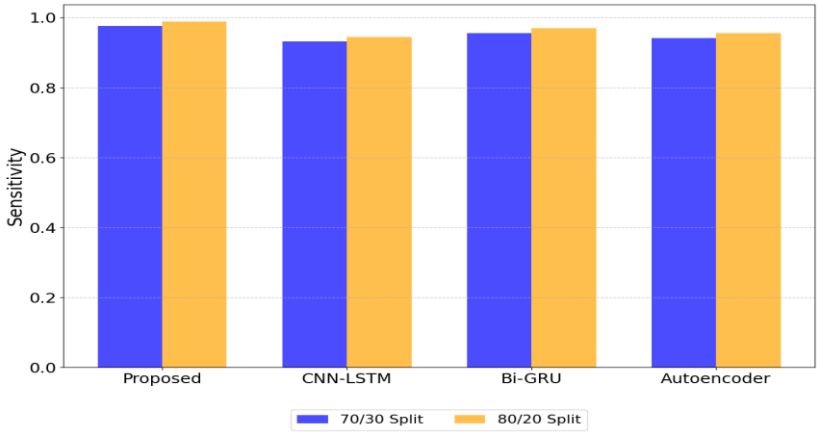

Figure 5. Comparison of Sensitivity

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Figure 5 depicts the comparison of sensitivity. The proposed model is quite sensitive; specifically, it has values of 0.97614/0.988293 for the 70/30 and 80/20 splits. It means the model is very efficient in pointing out true positive cases without any danger of leaving them out. In this case, the CNN-LSTM has the least sensitivity, calculated as 0.93148 and 0.94456; Bi-GRU gives relatively better performance, registering sensitivity values of 0.95514 and 0.96996, and the Autoencoder.

5.1.4 Specificity

The model's specificity assesses how well it can detect benign (non-threat) behaviors. It shows how successfully the system stays away from sounding false alarms in the absence of threats.

$$Specificity = \frac{True\ Positives}{True\ Negatives + False\ Positives} \tag{43}$$

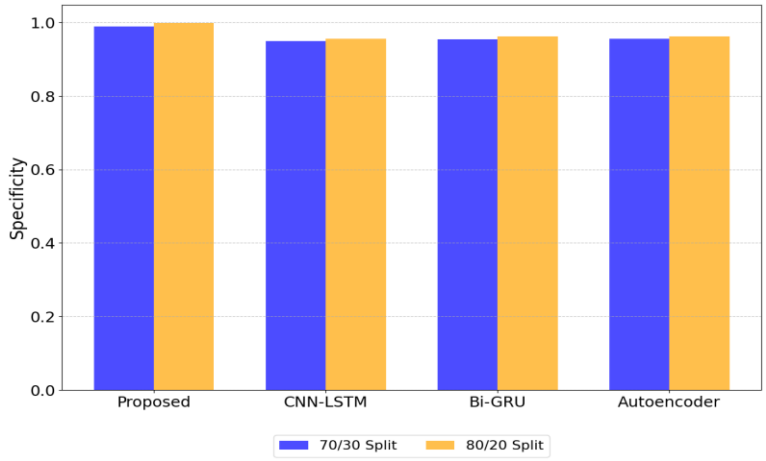

Figure 6. Comparison of Specificity

Figure 6 indicates the performance analysis of specificity. The suggested technique is very specific, with values of 0.98822 and 0.998 in the 70/30 and 80/20 splits, respectively. This clearly shows how well the model does at identifying the negative cases correctly, thereby minimizing false positives. The other models remain very good but slightly reduce in specificity: CNN-LSTM to 0.94831 and 0.95592, Bi-GRU to 0.95321 and 0.96134, Autoencoder to 0.95521 and 0.96109 for the respective data split.

5.1.5 F-measure

For the detection of APT threats, the F-measure provides a harmonic mean by balancing sensitivity and precision. This measure is especially helpful when it comes to APT detection, as it's crucial to both identify threats and prevent false alarms.

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{44}$$

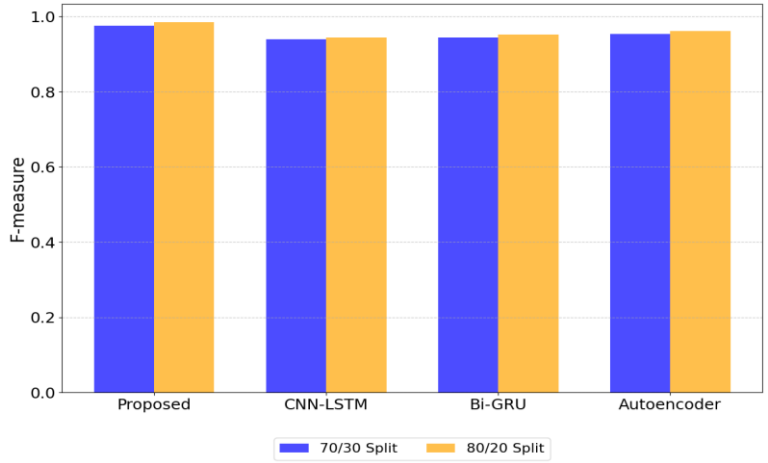

Figure 7. Comparison of F-measure

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Figure 7 indicates the comparison assessment of the F-measure. The values of the F-measure for the proposed model are the highest among all, reaching 0.97525 for the 70/30 split and 0.98485 for the 80/20 split. This shows that this model performs very well in both precision and recall. The rest have lower values of the F-measure: CNN-LSTM with 0.93871 and 0.94421, Bi-GRU attains 0.94452 and 0.95226, and finally, the Autoencoder reaches 0.95362 and 0.96088.

5.1.6 NPV

NPV evaluates how well the system detects benign activity. It shows how many activities that are labeled as non-threatening are genuinely harmless.

$$NPV = \frac{True\ Negatives}{True\ Negatives + False\ Negatives} \tag{45}$$

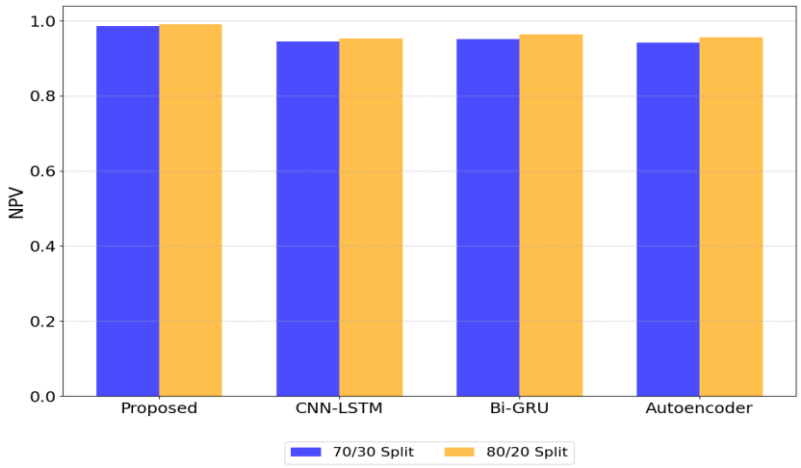

Figure 8. Comparison of NPV

Figure 8 illustrates the comparison of NPV. The proposed model exhibits high NPV values of 0.98547 and 0.99008 for the 70/30 and 80/20 splits, respectively, indicating its reliability in predicting negative cases. The other models, while also performing well, show slightly lower NPV values: CNN-LSTM scores 0.94487 and 0.95271, Bi-GRU achieves 0.95148 and 0.96336, and the Autoencoder records NPV values of 0.94142 and 0.95529.

5.1.7 MCC

MCC is an accurate measure of the overall performance of the APT detection model that considers all four components of the confusion matrix viz. TP, TN, FP, and FN. It also provides fair evaluation, especially in cases that have data imbalanced as in the case of uncommon APT incidents.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{46}$$

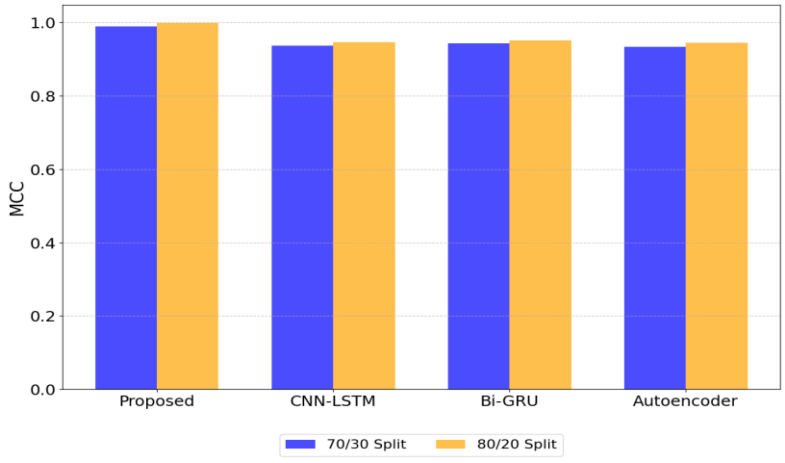

Figure 9. Comparison of MCC

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Figure 9 shows the MCC comparison. The proposed model achieves exceptionally high MCC values of 0.99021 and 0.99897 for the 70/30 and 80/20 splits, respectively, indicates a strong correlation among the predicted and observed classifications. The other models, though effective, have lower MCC values: CNN-LSTM records 0.93687 and 0.94621, Bi-GRU achieves 0.94412 and 0.95163, and the Autoencoder attains 0.93345 and 0.94496.

5.1.8 FPR

FPR measures how often harmless actions are mistakenly classified as APT threats. False alarms have to be reduced in APT detection systems to avoid unnecessary intervention.

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{47}$$

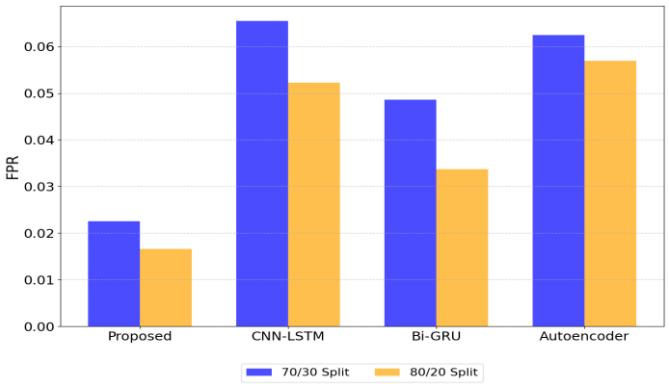

Figure 10. Comparison of FPR

Figure 10: Comparison analysis of FPR. It can be seen that the suggested mechanism offers the minimal values of FPR, which are 0.02254 and 0.01655 for splits 70/30 and 80/20, respectively, which correspond to almost negligible false positives. Other models demonstrated greater FPR values: CNN-LSTM achieved 0.06548 and 0.05229, Bi-GRU reached 0.04862 and 0.03365, and the Autoencoder achieved 0.06251 and 0.05694 for the considered data splits.

5.1.9 FNR

FNR calculates how frequently the system misses real APT threats. The detection model must have a low FNR to effectively detect and stop harmful activity.

$$FNR = \frac{False\ Negatives}{False\ Negatives + True\ Positives} \tag{48}$$

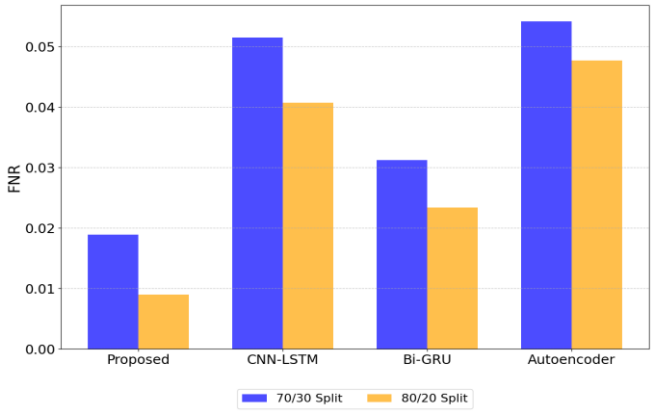

Figure 11. Comparison of FNR

Figure 11 shows the FNR comparison. The developed technique obtains the lowest FNR values, with 0.01894 and 0.00894 for the 70/30 and 80/20 splits, respectively, indicating a minimal rate of false negatives. The rest of the models achieve higher FNR values: CNN-LSTM scores 0.05148 and 0.04069, Bi-GRU reaches 0.03125 and 0.02341, and the Autoencoder records 0.05417 and 0.04768 for the respective splits. Overall performance analysis of the model for different learning rates is tabulated in Table 1.

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

Table 1: Overall performance comparison with existing techniques (70/30 and 80/20)

Split Data70/30									
Model	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**	**NPV**	**MCC**	**FPR**	**FNR**
Proposed	0.98741	0.99407	0.97614	0.98822	0.97525	0.98547	0.99021	0.02254	0.01894
CNN-LSTM	0.94766	0.95368	0.93148	0.94831	0.93871	0.94487	0.93687	0.06548	0.05148
Bi-GRU	0.96299	0.96542	0.95514	0.95321	0.94452	0.95148	0.94412	0.04862	0.03125
Auto encoder	0.94123	0.95523	0.94186	0.95521	0.95362	0.94142	0.93345	0.06251	0.05417
Split Data 80/20									
Proposed	0.99143	0.9949	0.988293	0.998	0.98485	0.99008	0.99897	0.01655	0.00894
CNN-LSTM	0.95893	0.96631	0.94456	0.95592	0.94421	0.95271	0.94621	0.05229	0.04069
Bi-GRU	0.97299	0.97229	0.96996	0.96134	0.95226	0.96336	0.95163	0.03365	0.02341
Autoencoder	0.95123	0.95986	0.95562	0.96109	0.96088	0.95529	0.94496	0.05694	0.04768

5.1.10 Execution Time

Execution time is when a system or algorithm must finish a certain operation or procedure. It is a crucial parameter for evaluating the model's computational effectiveness and applicability for large-scale or real-time cybersecurity applications.

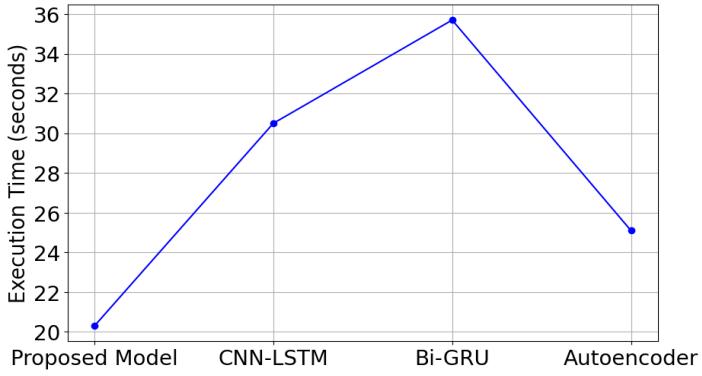

Figure 12. Comparison of Execution Time

The execution time comparison is displayed in Figure 12. The Suggested Model runs the quickest at 20.3 seconds, showing an optimized hybrid architecture for performance. The Autoencoder model presents at 25.1 seconds and would allow for a much more simplified architecture with focus on anomaly detection. For more complex time series analysis, the CNN-LSTM model, which aligns convolutions with recurrent layers, runs slowly at 30.5 seconds. Last but not least, the Bi-GRU method took the longest processing time, running at 35.7 seconds. Overall, the Proposed Model maintains effective APT detection while achieving the best execution time.

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

6. Conclusion

This paper proposes the application of a novel APT detection technique using the HDSEDV model. In this proposed methodology, advanced preprocessing techniques such as handling class imbalances are performed, using ADASYN; normalization is done through min-max scaling and median imputation to handle missing values. Features engineering has been performed by using SAX for time-series data and ResNet-152 for deep feature extraction. The HDT-SCO algorithm integrates L1 Regularization (Lasso), RF Feature Importance, PCA, and RFE for optimizing feature selection. Compared with those state-of-the-art models, the final model, EfficientDense-ViT, which integrates EfficientNet, DenseNet, and Vision Transformers (ViT), shows major improvements over accuracy (0.98741 for the 70/30 split and 0.99143 for the 80/20 split) and efficiency. Even though the results from the proposed hybrid model have been encouraging, the limitations of this approach include its computationally high complexity and the challenges encountered while attempting real-time detection in dynamic situations. Further research could concentrate on improving the model for quicker detection, lowering the computational cost for deployment in resource-constrained situations, and incorporating unsupervised or semi-supervised learning to automatically discover new attacks.

Reference

[1] Laurenza, G. (2020). Critical infrastructures security: improving defense against novel malware and Advanced Persistent Threats.

[2] Alevizos, L., Eiza, M. H., Ta, V. T., Shi, Q., & Read, J. (2022). Blockchain-enabled intrusion detection and prevention system of APTs within zero trust architecture. Ieee Access, 10, 89270-89288.

[3] Lee, K., Lee, J., & Yim, K. (2023). Classification and analysis of malicious code detection techniques based on the APT attack. Applied Sciences, 13(5), 2894.

[4] Sakthivelu, U., & Vinoth Kumar, C. N. S. (2024). A multi-step APT attack detection using hidden Markov models by molecular magnetic sensors. Optical and Quantum Electronics, 56(3), 282.

[5] Sharma, A., Gupta, B. B., Singh, A. K., & Saraswat, V. K. (2023). Advanced persistent threats (apt): evolution, anatomy, attribution and countermeasures. Journal of Ambient Intelligence and Humanized Computing, 14(7), 9355-9381.

[6] Javed, S. H., Ahmad, M. B., Asif, M., Almotiri, S. H., Masood, K., & Ghamdi, M. A. A. (2022). An intelligent system to detect advanced persistent threats in industrial internet of things (I-IoT). Electronics, 11(5), 742.

[7] Reynolds, M. (2021). The art of attack: Attacker mindset for security professionals. John Wiley & Sons.

[8] Wu, J., & Wu, J. (2020). Security risks from vulnerabilities and backdoors. Cyberspace Mimic Defense: Generalized Robust Control and Endogenous Security, 3-38.

[9] Riggs, H., Tufail, S., Parvez, I., Tariq, M., Khan, M. A., Amir, A., ... & Sarwat, A. I. (2023). Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure. Sensors, 23(8), 4060.

[10] Zimba, A., Chen, H., Wang, Z., & Chishimba, M. (2020). Modeling and detection of the multi-stages of Advanced Persistent Threats attacks based on semi-supervised learning and complex networks characteristics. Future Generation Computer Systems, 106, 501-517.

[11] Roy, S. (2024). Cyber Deception against Adversarial Reconnaissance in Enterprise Network using Semi-Indistinguishable Honeypot (Doctoral dissertation).

[12] Mallaboyev, N. M., Sharifjanovna, Q. M., Muxammadjon, Q., & Shukurullo, C. (2022, May). Information security issues. In Conference Zone (pp. 241-245).

[13] Nichols, R. A. (2020). Analysis of Factors to Reduce Advanced Persistent Threat (APT) Exploitation Risk: A Delphi Study. Capella University.

[14] Xiong, C., Zhu, T., Dong, W., Ruan, L., Yang, R., Cheng, Y., ... & Chen, X. (2020). CONAN: A practical real-time APT detection system with high accuracy and efficiency. IEEE Transactions on Dependable and Secure Computing, 19(1), 551-565.

[15] Zhang, C., Jia, D., Wang, L., Wang, W., Liu, F., & Yang, A. (2022). Comparative research on network intrusion detection methods based on machine learning. Computers & Security, 121, 102861.

[16] Yue, H., Li, T., Wu, D., Zhang, R., & Yang, Z. (2024). Detecting APT attacks using an attack intent-driven and sequence-based learning approach. Computers & Security, 140, 103748.

[17] Joloudari, J. H., Haderbadi, M., Mashmool, A., GhasemiGol, M., Band, S. S., & Mosavi, A. (2020). Early detection of the advanced persistent threat attack using performance analysis of deep learning. IEEE Access, 8, 186125-186137.

[18] Kumari, I., & Lee, M. (2023). A prospective approach to detect advanced persistent threats: Utilizing hybrid optimization technique. Heliyon, 9(11).

[19] Nguyen, H. C., Xuan, C. D., Nguyen, L. T., & Nguyen, H. D. (2023). A new framework for APT attack detection based on network traffic. Journal of Intelligent & Fuzzy Systems, 44(3), 3459-3474.

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024

[20] Oliveira, N., Praça, I., Maia, E., & Sousa, O. (2021). Intelligent cyber attack detection and classification for network-based intrusion detection systems. Applied Sciences, 11(4), 1674.
[21] Vo, M. T., Nguyen, T., Vo, H. A., & Le, T. (2021). Noise-adaptive synthetic oversampling technique. Applied Intelligence, 51(11), 7827-7836.
[22] Panda, M. K., Subudhi, B. N., Veerakumar, T., & Jakhetiya, V. (2023). Modified ResNet-152 network with hybrid pyramidal pooling for local change detection. IEEE Transactions on Artificial Intelligence.
[23] Elhaik, E. (2022). Principal Component Analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated. Scientific Reports, 12(1), 14683.
[24] Thongprayoon, C., Jadlowiec, C. C., Leeaphorn, N., Bruminhent, J., Acharya, P. C., Acharya, C., ... & Cheungpasitporn, W. (2021). Feature importance of acute rejection among black kidney transplant recipients by utilizing random forest analysis: an analysis of the UNOS database. Medicines, 8(11), 66.
[25] https://www.kaggle.com/datasets/sowmyamyneni/dapt2020?select=csv

DOI: https://doi.org/10.54216/JCIM.150212
Received: May 11, 2024 Revised: July 15, 2024 Accepted: October 27, 2024