Enhanced Adaptive Hybrid Convolutional Transformer Network for Malware Detection in IoT

Abdulaleem Ali Almazroi
Department of Information Technology-Faculty of Computing and Information Technology in Rabigh,
King Abdulaziz University, Rabigh, 21911, Saudi Arabia

***Abstract*—Many university networks use IoT devices, which increases vulnerability and malware threats. The complex, multi-dimensional structure of IoT network traffic and the imbalance between benign and dangerous data make traditional malware detection techniques ineffective. The Adaptive Hybrid Convolutional Transformer Network (AHCTN) is a novel model that uses CNNs for spatial feature extraction and Transformer networks for global temporal dependencies in IoT data. Unique preprocessing methods like Category Importance Scaling and Logarithmic Skew Compensation handle unbalanced data and severely skewed numerical characteristics. The Unified Feature Selector combines statistical and model-based feature selection methods and guarantees that only the most relevant characteristics are utilized for classification. DWS and LRW handle data imbalance. Our feature engineering approaches, such as Flow Efficiency and Packet Interarrival Consistency, improve prediction accuracy by capturing essential data correlations. The integration of advanced machine learning techniques ensures precise malware classification and enhances cybersecurity by addressing vulnerabilities in IoT-driven academic networks. The AHCTN model was carefully tested using the IoEd-Net dataset, which contains a variety of IoT devices and network activity. The AHCTN outperforms previous models with 98.9% accuracy. It also performs well in Log Loss (0.064), AUC (99.1%), Weighted Temporal Sensitivity (97.1%), and Anomaly Detection Score (96.8%), recognizing uncommon but essential abnormalities in academic network data. These findings demonstrate AHCTN's robustness and scalability for academic IoT malware detection.**

***Keywords*—IoT security; malware detection; convolutional transformer network; cybersecurity; machine learning; network anomaly detection**

I. Introduction

Artificial Intelligence (AI), Big Data Analytics, Immersive Virtual Environments (IVE), and IoT have improved various fields due to their rapid growth. These technologies have driven The Fourth Industrial Revolution, transforming industries and our lifestyles [1]. In particular, IoT has changed device connection and data exchange. Connectivity boosts efficiency, automation, and convenience. Malware now threatens IoT installations. Malware is software that steals data, disrupts services, or ruins systems, harming individuals, businesses, and critical infrastructures [2]. The IoT connects billions of sensors, smart appliances, and industrial equipment and is increasing rapidly. However, IoT's numerous networked devices pose concerns. Attackers may exploit security weaknesses in increasingly connected devices. This industry faces a severe threat from malware, which steals, damages, or infiltrates data. Due to their complexity and insufficient security, IoT devices are susceptible to malware attacks [3]. Cybercriminals use IoT vulnerabilities for DDoS and crypto-mining. Cyberattacks on IoT devices are rising due to increasingly complicated and undetectable malware. Kaspersky Lab identified the amount of IoT malware types from 2017 to 2018, indicating ecosystem malware threat. Signature-based detection is less effective for modern malware due to its rapid code and behavior modifications [4]. Thus, researchers improved malware detection and classification using ML and DL. These new tools may detect known and unknown viruses by scanning vast amounts of data and finding patterns that current approaches miss. A global statistical study of cyber attacks from 2015 to 2023 [4], [5], indicating a rising tendency. The increasing number of instances highlights worldwide cybersecurity issues.

Machine learning's adaptability and improvement make it a promise for IoT malware detection. Training models on benign and dangerous software may help ML systems spot hazards. Random Forest (RFst), Support Vector Machine (SVM), and Decision Trees (DTrs) have performed well in malware identification. Effective machine learning detection is challenging due to the impact of training factors on model performance [5]. Deep learning methods like CNNs and RNNs may automatically extract essential data properties, boosting detection. Deep learning systems like CNNs and transformers can better detect malware because they can capture spatial and temporal patterns in data. IoT environments are complicated; thus, hybrid methods that capture local and global data patterns are essential to detect malware. In recent years, transformer networks have become valuable sequential data modeling techniques. Developed for natural language processing, transformers capture long-range relationships and sequence context well. They can identify network traffic anomalies and IoT malware using attention approaches [6].

Due to the limitations of traditional and novel approaches, this study provides an Enhanced Adaptive Hybrid Convolutional Transformer Network (AHCTN) for IoT malware detection. AHCTN uses CNNs and transformers to identify malware. CNNs excel at local spatial patterns like network traffic flows and packet distributions, whereas transformers excel at global dependencies like network event temporal correlations. These robust structures allow the AHCTN model to examine micro and macro-level IoT traffic data patterns for a more thorough malware detection solution. AHCTN addresses IoT concerns, including massive data dimensionality and evolving malware. Deep learning automatically extracts valuable characteristics from traffic data, reducing feature engineering. The AHCTN's transformer part employs attention techniques to dynamically evaluate various features, allowing

the model to zero in on the most critical malware detection patterns. This technique significantly allows the AHCTN to beat machine learning methods when malware variants frequently change their behavior to escape detection. AHCTN parameters are optimized for performance in this study. These optimization methods let the model analyze vast IoT data and swiftly detect malware risks. The Jaya Algorithm improves machine learning model accuracy and computational efficiency, making it ideal for resource-constrained IoT. Key contributions of this work:

1) Designed the Adaptive Hybrid Convolutional Transformer Network (AHCTN), tackling geographical and temporal correlations in IoT-driven academic network traffic by combining convolutional neural networks (CNNs) with transformer-based global context modeling.
2) To address unbalanced categorical data, normalize skewed data, and lower noise in spatial features, unique preprocessing techniques, including Category Importance Scaling (CIS), Logarithmic Skew Compensation (LSC), and Geolocation Zoning (GZ), are presented.
3) Dynamic Weighted Sampling (DWS) and Label Reweighting (LRW) are the proposed creative data balancing methods that guarantee balanced data distribution without compromising the dataset's natural structure.
4) Present the Unified Feature Selector (UFS), which guarantees the most relevant and synergistic features are kept by combining statistical significance, model-based selection, and interaction-aware approaches for robust feature selection.
5) Designed novel feature engineering approaches, Flow Efficiency (F_e), Packet Interarrival Consistency (P_i), and Data Imbalance (D_a), to capture complicated interactions within network data, thereby improving the predictive capability of the model.
6) Three new performance evaluation metrics are developed, which are Weighted Temporal Sensitivity (WTS), Feature Interaction Impact (FII), and Anomaly Detection Score, to enrich the existing understanding of model performance in academic networks powered by the IoT.
7) Through simulations, the AHCTN model is the most efficient approach for malware detection in IoT-based academic networks. It exceeds current models with a remarkable accuracy of 98.9%.

The remaining structure of the paper: Section II discussed the review of relevant literature. The proposed method structure is described in detail in Section III. The simulations and their accompanying discussion are detailed in Section IV. The last section concludes with a discussion of future work.

II. Related Work

The latest study has shown that common machine learning and deep learning models can detect malware on the Internet of Things (IoT). Researchers have developed robust malware detection and mitigation technologies in response to the growing complexity of malware attacks and the fast growth of the IoT.

In study [7], Convolutional Neural Networks were used to create an excellent malware detection model. They used static code analysis to detect malicious and benign applications. CNN model classified malware with 91.01% accuracy. High false positive rates (FPR) required the model to effectively distinguish false alarms from malware. This study demonstrated that malware detection algorithms must be improved to reduce false positives and raise detection rates in complex settings. Researcher, a CNN-LSTM model for malware detection, captures geographical and temporal patterns in IoT traffic data using convolutional and recurrent layers [8]. With FPR at 0.2%, the model was 99.6% correct. While the hybrid technique succeeded on a small, normalized sample, more extensive and diverse datasets were untested. LSTM processing power limits real-time applications. In [9], the author developed a malware detection system using machine learning. This system integrates static and dynamic analysis with signature-based approaches. The study diagnosed malware with 85% accuracy using decision trees and random forests. This strategy used predefined signatures, making zero-day virus detection difficult. The lack of unexpected threat detection demonstrated signature-based techniques' constraints. Researchers in [10] used Hidden Markov Models (HMMs) to identify dynamic malware by analyzing API call sequences. Their strategy enhanced limited dataset detection accuracy. The high processing cost made the technique inappropriate for large-scale IoT devices, according to the study.

Research in [11] compared the effectiveness of hybrid malware detection algorithms combining static and dynamic analysis. Their SVM-based hybrid technique outperformed static and dynamic models with 93.5% accuracy. Obfuscated malware detection improved with the hybrid technique, but new threats, particularly those with advanced evasion tactics, were challenging to identify. Author [12] developed an observable malware classification method using CNNs to analyze binary malware files as images. Their 94.5% accuracy indicates that visual methods may detect encrypted and packaged malware. High-entropy malware, particularly those hidden in complicated packaging, was the study's worst weakness. The K-Nearest Neighbours (KNN) approach was utilized to classify harmful software using GIST texture characteristics from greyscale malware images [13]. Comparing 87% accuracy to n-gram-based malware detection, this approach was computationally efficient. It struggled with comparable binary malware families. Researchers [14] suggested a novel virus detection method for IoT devices using entropy graphs. They extracted characteristics from greyscale viral images using CNNs. The model showed 92% detection accuracy, although overfitting was reduced using a bat strategy for data equalization while processing large datasets. The study did not address real-time malware detection.

Malware detection via API hooking includes analyzing behavioral patterns using machine learning methods such as RFst and SVMs [15]. The program detected malware activity with 90% accuracy. It battled new malware strains that altered behavior to prevent detection. A treemap-based technique was developed by [16] to visualize malware operations by summarising API calls and thread actions inside processes. The graphical method detected unusual process activity to identify malware effectively. Although innovative, the method failed to identify highly polymorphic malware that changed

behavior over time. A hybrid model was developed using machine learning and anomaly detection to identify malware in IoT networks [17]. Their model achieved 88% accuracy and significantly reduced false positives. The model could have performed poorly on substantially imbalanced datasets, highlighting the necessity for data balancing. Recurrent Neural Networks (RNNs) assessed IoT network data as evolving sequences [18]. Achieving 89% accuracy in time-based anomaly detection is promising. Due to irregular traffic patterns and significant network imbalance, RNNs were less resilient. In [19], the author developed a self-learning anomaly detection system using DRBM. Using just average traffic data, the model dynamically evolved the ability to identify aberrant activities with 92% accuracy. Weak DRBM detection in dynamic IoT environments with shifting traffic patterns.

Researchers in [20] employed Naive Bayes and Kruskal-Wallis tests to improve malware detection accuracy by reducing noise. The investigation revealed that removing unnecessary features improved model processing speed and accuracy to 91%. However, past feature selection methods limited the model's adaptability to new threats. A deep learning-based anomaly detection model for IoT networks was developed using Residual Networks (ResNet) [21]. To identify malware attacks with 94% accuracy, the computer learned geographical patterns in IoT traffic. Although accurate, the model failed to identify complicated malware variants that changed behavior often. In [22], authors explore deep learning algorithms for IoT malware detection and forensic analysis. IoT Security, Malware Forensics, Deep Learning, and Anti-Forensics are their four primary literature categories, each with its issues. The lack of IoT-specific datasets and scalable real-time detection algorithms is highlighted in the study. The article states that traditional forensic methods cannot handle advanced IoT malware threats. Future directions include advanced anti-forensic countermeasures. Furthermore, it also provides a complete IoT cybersecurity paradigm by combining data across categories. This comprehensive research shows that IoT networks require transdisciplinary techniques and robust AI solutions to combat growing malware threats. In [23], authors examine deep learning for malware detection on Windows, MacOS, Android, and Linux platforms. Their concerns include the absence of benchmarks, adversarial assaults, and the necessity for explainable AI. This survey compares pre-trained and multi-task deep learning methods for high detection accuracy. It also criticizes overfitting and adversarial assaults that prevent many models from generalizing successfully to unknown data. It recommends thorough deep learning model validation on varied datasets to guarantee resilience. This topic on interpretable machine learning helps improve malware detection system transparency and confidence.

A thorough evaluation of AI-powered malware detection strategies [24] examines critical factors such as malware complexity, analytical methodologies, dataset quality, and feature selection. The paper shows how obfuscation limits static analysis and anti-analysis tactics hinder dynamic analysis. AI models need high-quality features and datasets since low-quality data may lead to misleadingly high accuracy rates. The paper also examines machine learning vs. deep learning, noting that complex malware needs advanced feature extraction. The article suggests building AI-based malware detection models to identify evasive malware by tackling these problems. The authors of [25] introduce deep learning-based malware detection approaches and highlight their benefits over older methods. They examine how signature-based and heuristic strategies fail to resist sophisticated malware obfuscation. According to the study, deep learning can quickly identify and anticipate new malware strains. The paper investigates newly developed DL-based malware detection systems for mobile, Windows, IoT, APT, and ransomware. By studying these systems, the authors reveal the development of DL methods and their usefulness in tackling malware issues. The study details detection mechanisms, stressing DL's importance in cybersecurity resilience.

Research shows that machines and deep learning can detect IoT malware. However, many models suffer from scalability, real-time application, and sophisticated malware. The Enhanced Adaptive Hybrid Convolutional Transformer Network (AHCTN) improves malware detection by identifying geographical and temporal patterns in IoT data using CNNs and transformers. Table I shows the summarized view the above mentioned literature.

III. Proposed Method

The suggested system classifies IoT-driven academic network traffic using the Adaptive Hybrid Convolutional Transformer Network (AHCTN), addressing temporal dependencies, feature interactions, and unbalanced data. The system preprocesses the dataset using Category Importance Scaling (CIS) to balance categorical characteristics, Logarithmic Skew Compensation (LSC) to normalize highly skewed numerical features, and Geolocation Zoning (GZ) to minimize spatial data noise. Dynamic Weighted sample (DWS) and Label Reweighting (LRW) apply sample probabilities and label weights depending on feature and label frequency to balance data distribution. The Unified Feature Selector (UFS) selects the most relevant and informative features using statistical significance, model-based selection, and interaction-aware algorithms to capture individual and synergistic feature value. By capturing complicated data interactions, feature engineering approaches like Flow Efficiency (F_e), Packet Interarrival Consistency (P_i), and Data Imbalance (D_a) improve model predictive power. Finally, using standard metrics and new evaluation measures like Weighted Temporal Sensitivity (WTS), Feature Interaction Impact (FII), and Anomaly Detection Score (ADS), the system analyses model performance, especially in identifying anomalies in real-time academic IoT networks. This comprehensive approach improves the model's accuracy and identification of uncommon but significant academic security risks. Fig. 1 shows the proposed framework of this study.

A. Dataset Description

This study used the publicly available IoEd-Net dataset from Kaggle [26]. It covers the IoT environment with 202,085 records. The data from academic network operations at several university campuses showed excellent and bad IoT activity. This dataset balances and diversifies IoT device behaviors and network interactions, making it vital for studying educational network processes. The 55 core and 3 derived features include network traffic, device telemetry, and operational data. Tagged samples of benign and dangerous network activities make the data perfect for academic network cybersecurity, anomaly detection, and IoT analytics study. The dataset's variety improves

TABLE I. LITERATURE REVIEW SUMMARY

Ref	Technique Used	Objective Achieved	Limitations
[7]	Convolutional Neural Networks (CNN)	Developed a CNN model for malware classification, achieving 91.01% accuracy.	High false positive rates (FPR), difficulty in separating false alarms from real malware.
[8]	Hybrid CNN-LSTM	Achieved 99.6% accuracy with reduced FPR of 0.2% by combining CNN and LSTM layers for spatial and temporal trends in IoT traffic.	Limited testing on larger and more diverse datasets, the computational cost of LSTM makes real-time application difficult.
[9]	Decision Trees, Random Forest	Classified malware using a combination of signature-based approaches with static and dynamic analysis, achieving 85% accuracy.	Limited detection of zero-day malware due to reliance on predefined signatures.
[10]	Hidden Markov Models (HMM)	Improved malware detection accuracy through dynamic API call sequence analysis in small datasets.	Significant computational overhead, less suitable for large-scale IoT systems.
[11]	SVM-based Hybrid Model (Static + Dynamic)	Enhanced detection of obfuscated malware with 93.5% accuracy by combining static and dynamic analysis techniques.	Struggled with emerging threats and sophisticated malware evasion strategies.
[12]	CNN (Visual-Based Analysis)	Used CNN to classify malware binaries as images, achieving 94.5% accuracy in identifying packed and encrypted malware.	Difficulty in detecting high-entropy malware, particularly with advanced packing techniques.
[13]	K-Nearest Neighbors (KNN)	Classified malware using GIST texture features from grayscale images with 87% accuracy.	Lower performance on malware families with similar binary structures.
[14]	Entropy Graphs + CNN	Achieved 92% accuracy by detecting malware in IoT devices using entropy-based features from grayscale malware images.	Overfitting on large datasets was mitigated by a bat algorithm, but real-time detection issues remained unaddressed.
[15]	Random Forest, SVM (API Hooking)	Detected malware activities by analyzing API hooking behaviors with 90% accuracy.	Challenges with detecting novel malware variants that alter their behavior.
[16]	Treemap + API Calls Visualization	Visualized malware behavior using API calls and thread actions, providing insights into malware detection.	Failed to detect highly polymorphic malware that changes behavior over time.
[17]	Hybrid Machine Learning + Anomaly Detection	Detected malware in IoT networks with 88% accuracy and reduced false positives.	Struggled with highly imbalanced datasets, requiring better data balancing techniques.
[18]	Recurrent Neural Networks (RNN)	Modeled IoT network traffic as evolving sequences for anomaly detection with 89% accuracy.	Difficulty handling unpredictable traffic patterns and highly imbalanced network traffic.
[19]	Discriminative Restricted Boltzmann Machine (DRBM)	Developed a self-learning anomaly detection system trained on normal traffic data with 92% accuracy.	Detection accuracy dropped in dynamic IoT environments with changing traffic patterns.
[20]	Naive Bayes, Kruskal-Wallis (Feature Selection)	Improved malware detection accuracy to 91% by eliminating irrelevant features, enhancing processing speed.	Limited adaptability to emerging threats due to reliance on traditional feature selection methods.
[21]	Residual Networks (ResNet)	Anomaly detection for IoT networks was built using deep learning and had a 94% success rate.	Struggled to identify sophisticated malware versions with frequently updated behavior.

educational IoT system analysis. Table II shows the features list of the dataset.

B. Data Preprocessing Steps

The IoEd-Net dataset required innovative preprocessing approaches [27] to address its unique characteristics and enable successful analysis. Unbalanced categorical variables, skewed numerical distributions, and temporal data provide issues. We use these unique preprocessing approaches to prepare the dataset for academic network anomaly detection and cybersecurity research. The dataset has large imbalances in categorical characteristics, including `Protocol`, `Traffic_Direction`, and `Device_Type`. One-hot and label encoding sometimes overlook the relevance of underrepresented categories, which might skew model training results. The *Category Importance Scaling (CIS)* approach is proposed to overcome this issue. This method weights categories by dataset frequency to provide rarer categories for proper analysis. For category C_i, determine the scaling factor:

$$\text{CIS}(C_i) = \frac{1}{\log(1 + f_i)} \tag{1}$$

where f_i is category C_i frequency. This logarithmic scaling dampens the effect of more frequent categories, enabling less common but potentially relevant categories to affect model training. Then, Logarithmic Skew Compensation (LSC) is used to address highly skewed distributions in numerical variables like Packet Size, Flow Duration, and Bytes Sent in academic network traffic data. Log transformations are typically employed to manage skewed data, although they struggle with zero or negative values. The LSC approach uses a shift factor s to preserve changed data. The change is:

$$\text{LSC}(X) = \log(1 + X + s) \tag{2}$$

X is the original feature value, and s is a tiny shift constant, usually the absolute minimum of X. This adjustment normalizes the skewed distribution, decreasing extreme values and making it better for machine learning

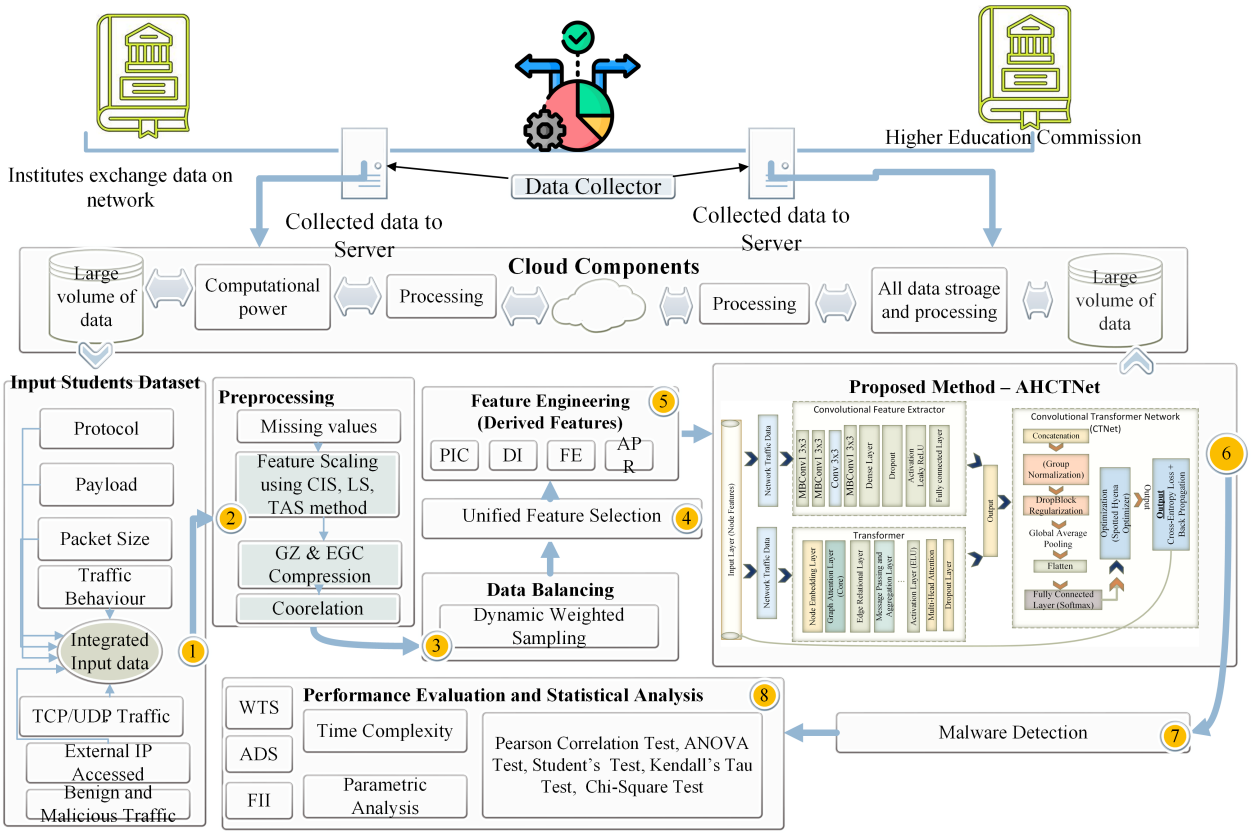

Fig. 1. Proposed malware detection framework.

models. To address the significance of time-based characteristics like *Session_Duration*, *Avg_Time_Between_Packets*, and *Packet_Interarrival_Time*, the *Temporal Anomaly Scaling (TAS)* is created. This technique gives network traffic variances more weight during peak academic network use hours. Definition of temporal scaling:

$$\text{TAS}(T) = \frac{T - \mu_T}{\sigma_T} \times w(T) \tag{3}$$

μ_T and σ_T represent the mean and standard deviation of the time-based feature. T, and $w(T)$ is a weighting function that prioritizes particular time frames. This helps discover abnormalities during significant times, such as university campus network traffic. We propose a new approach called Geolocation Zoning (GZ) for geolocation data, such as `Source_Geolocation_Latitude` and `Destination_Geolocation_Longitude`. Geolocation characteristics are generally highly variable. Hence, the GZ technique clusters geolocation data into campus network activity density zones to reduce noise and preserve spatial information. The formula for geolocation data zone segmentation is:

$$\text{GZ}(G) = \text{ZoneID}(k) \tag{4}$$

G represents the geographical position, and $textZoneID(k)$ identifies the cluster zone based on k clusters. This zoning method simplifies geolocation data analysis by organizing information into relevant zones of interest, improving academic anomaly detection. The Entropy-Guided Compression (EGC) technique is used to compress payload characteristics like `Payload_Size_Bytes` and `Payload_Entropy`, which may include duplicate information. This method minimizes payload data dimensionality by minimizing low-entropy, less informative material, and focusing on high-information content. Transformation appears as:

$$\text{EGC}(P) = P \times (1 - H(P)) \tag{5}$$

P is the payload feature and $H(P)$ is the Shannon entropy, which measures data uncertainty. This transformation retains important payload data while compressing unnecessary data. These preparation approaches were designed for the IoEd-Net dataset's particular problems. Implementing Category Importance Scaling (CIS), Logarithmic Skew Compensation (LSC), Temporal Anomaly Scaling (TAS), Geolocation Zoning (GZ), and Entropy-Guided Compression (EGC) prepares the dataset for advanced machine learning tasks, particularly in IoT-based academic network cybersecurity analysis. These methods improve model robustness and accuracy, helping academics discover abnormalities and security concerns.

C. Data Balancing using Dynamic Weighted Sampling

Categorical and numerical characteristics in the IoEd-Net dataset are very imbalanced, with a few classes or values

TABLE II. DATASET FEATURES OVERVIEW

S.No	Features	Short Description	S.No	Features	Short Description
1	Source_IP	IP address of the source device	29	CPU_Usage	CPU usage of the device
2	Destination_IP	IP address of the destination	30	Memory_Usage_MB	Memory used by the device (MB)
3	Source_Port	Network port of the source	31	Energy_Consumption_Watts	Device energy usage in watts
4	Destination_Port	Network port of the destination	32	Firmware_Version	Device firmware version
5	Protocol	Type of network protocol (TCP, UDP, etc.)	33	Device_Uptime_Hours	Total device uptime (hours)
6	Packet_Size	Size of the network packet	34	Payload_Size_Bytes	Size of payload in bytes
7	Packet_Count	Total number of packets	35	Payload_Entropy	Entropy of payload data
8	Flow_Duration	Duration of the network flow	36	Payload_Content_Type	Content type of payload (ASCII, Binary)
9	Packet_Interarrival_Time	Time between packet arrivals	37	Signature_Match	Whether a signature matched or not
10	TCP_Flags	Flags set in a TCP packet	38	Compressed_Encrypted_Flag	Flag indicating compression/encryption
11	Bytes_Sent	Total bytes sent	39	Content_Type_File_Transferred	Type of file transferred
12	Bytes_Received	Total bytes received	40	Flow_Count_Per_Time_Window	Count of flows in a time window
13	Traffic_Direction	Direction of traffic (Ingress/Egress)	41	Avg_Packet_Size_Bytes	Average packet size in bytes
14	Connection_State	State of the connection	42	Packet_Rate_Packets_Per_Second	Rate of packets per second
15	Packet_Drop_Rate	Rate of dropped packets	43	Std_Dev_Packet_Size	Standard deviation of packet size
16	Malformed_Packet_Count	Number of malformed packets	44	Min_Packet_Size	Minimum packet size
17	Avg_Time_Between_Packets	Average time between packets	45	Max_Packet_Size	Maximum packet size
18	Total_Flow_Count	Total number of network flows	46	Protocol_Distribution_TCP_Percent	Percentage of TCP traffic
19	Device_Type	Type of IoT device	47	Protocol_Distribution_UDP_Percent	Percentage of UDP traffic
20	Device_Manufacturer	Manufacturer of the device	48	Protocol_Distribution_ICMP_Percent	Percentage of ICMP traffic
21	OS_Version	Version of the operating system	49	Source_Geolocation_Latitude	Latitude of source location
22	DNS_Query_Count	Number of DNS queries	50	Source_Geolocation_Longitude	Longitude of source location
23	Suspicious_Domain_Query_Flag	Flag for suspicious domain query	51	Destination_Geolocation_Latitude	Latitude of destination location
24	File_Transfer_Occurred	Flag indicating file transfer	52	Destination_Geolocation_Longitude	Longitude of destination location
25	File_Size_Bytes	Size of transferred file	53	Anomalous_Behavior_Flag	Flag indicating anomalous behavior
26	External_IP_Accessed_Flag	Flag for external IP access	54	Session_Duration	Duration of the session
27	Label	Benign or malicious activity	55	Time_Of_Day	Time of day of activity
28	Device_Uptime_Hours	Uptime of the IoT device	56	Day_Of_Week	Day of the week of activity

dominating. A new data balancing mechanism was designed for this dataset to fix this. *Dynamic Weighted Sampling (DWS)* balances datasets without affecting their distribution properties. Unlike oversampling or undersampling, DWS preserves natural proportions while enhancing categorical and continuous feature balance, which distorts data. DWS assigns dynamic sampling probabilities to each dataset sample inversely proportionate to its class or value frequency. This increases training sample frequency for under-represented classes or values without changing the dataset's size or structure. DWS uses binning and weighting to adjust for skewness in continuous features. Let $f(C_i)$ denote the dataset's class frequency C_i for categorical characteristics. The sampling probability for each class is $P(C_i)$:

$$P(C_i) = \frac{1}{\log(1 + f(C_i))} \tag{6}$$

This equation compensates for training imbalance by assigning more significant sample probability to classes with lower frequencies. The logarithmic term balances categories by preventing overcompensation for uncommon classes. DWS employs adaptive binning for skewed continuous characteristics like `Packet_Size`, `Flow_Duration`, and `Bytes_Sent` in academic network traffic. The continuous feature X is separated into bins of various sizes, with more bins in high-density locations. Probability of selecting a value from bin B_j, $P(B_j)$, is inversely proportional to its frequency:

$$P(B_j) = \frac{1}{f(B_j) + \epsilon} \tag{7}$$

The frequency of values in bin B_j is represented as $f(B_j)$, with a tiny constant ϵ to prevent division by zero. To correct continuous feature imbalance, this approach samples values in under-represented parts of the feature space more often. Adaptive binning preserves data distribution and reduces skewness.

Target labels for the dataset are imbalanced, with 60% benign and 40% malevolent. We use the Label Reweighting (LRW) technique to balance labels. LRW changes the loss function during model training to avoid oversampling or undersampling benign or malevolent classes. Each label's weight w_i is the inverse of its frequency:

$$w_i = \frac{1}{f(y_i)} \tag{8}$$

where $f(y_i)$ is the dataset label frequency. Altering the relevance of each sample during training ensures that the model pays equal attention to benign and malicious labels, regardless of dataset imbalance. The reweighted loss function is:

$$\mathcal{L}(y, \hat{y}) = \sum_i w_i \cdot \ell(y_i, \hat{y_i}) \tag{9}$$

Where $\ell(y_i, \hat{y_i})$ is the original loss (e.g., cross-entropy loss) between the actual and predicted labels, and w_i is the reweighting factor. Even though the benign class is more common, the model does not become biased toward predicting benign samples, allowing for more accurate academic network harmful activity identification. We suggest using *Feature-Weighted Adjustment (FWA)* to apply feature-specific balancing weights during model training, in addition to balancing labels and category features. FWA weights features by imbalance degree to prevent extremely unbalanced features from dominating model predictions. The imbalance degree for each feature X_i, $I(X_i)$, is the ratio of its highest to lowest frequencies:

$$I(X_i) = \frac{\max(f(X_i))}{\min(f(X_i)) + \epsilon} \tag{10}$$

ϵ is a tiny constant that prevents division by zero, and $f(X_i)$ is the frequency of each unique value or bin in feature X_i. Next, the feature weight $w_f(X_i)$ is determined in the following way:

$$w_f(X_i) = \frac{1}{I(X_i)} \tag{11}$$

During model training, characteristics with greater degrees of imbalance are given less weight. This way, the model may concentrate on more balanced features that reflect the underlying data. With FWA included, the total weighted loss function is given by:

$$\mathcal{L}\text{FWA}(y, \hat{y}) = \sum i w_f(X_i) \cdot \ell(y_i, \hat{y_i}) \tag{12}$$

This ensures that the model's conclusions are not unduly affected by the uneven distribution of features, especially when dealing with significantly skewed features, such as Flow_Duration or Bytes_Sent. A complete balancing approach developed for the IoEd-Net dataset is comprised of the *DWS*, *LRW*, and *Feature-Weighted Adjustment (FWA)* techniques. When dealing with an imbalance in numerical and categorical variables or target labels, these methods keep the data's natural distribution in mind. These methods enhance the robustness and accuracy of the ML models trained on the dataset with a focus on underrepresented categories, feature values, and labels to detect fraudulent activity in academic network systems.

D. Unified Feature Selector

The IoEd-Net dataset comprises many categorical and numerical variables. Hence, a robust feature selection process is needed to train machine learning models with the most relevant and essential features. Unified Feature Selector (UFS) is an innovative way to accomplish this. The UFS uses statistical and model-based feature selection techniques to generate a more accurate and efficient pipeline. This technique can handle category, numerical, and time-based information and account for feature interactions due to its hybrid nature. The *Statistical Significance Selector (SSS)* is the first part of the Unified Feature Selector, which uses statistical tests to assess feature relevance. Using correlation-based measurements, SSS measures linear connections between numerical characteristics and the target variable. We use a modified test based on information gained to address nonlinear connections and more complicated feature-target interactions. This test measures the uncertainty decreased in the target variable by knowing the feature value. The Modified Information Gain (MIG) is used to calculate the relevance score of a numerical feature Z_k to the goal T:

$$\text{MIG}(Z_k, T) = S(T) - S(T \mid Z_k) \tag{13}$$

Where $S(T)$ is the target variable's entropy, and $S(T \mid Z_k)$ is the target's conditional entropy given the feature Z_k. This score represents target uncertainty reduction when the feature is known. Higher MIG values indicate relevance and are picked for study. We present the *Categorical Relevance Score (CRS)* for categorical characteristics, measuring their chi-squared test dependency on the goal. Calculating the CRS:

$$\text{CRS}(A_m, T) = \sum \frac{(P_{im} - Q_{im})^2}{Q_{im}} \tag{14}$$

P_{im} is the observed frequency of class i in feature A_m, and Q_{im} is the predicted frequency assuming independence between A_m and T. Features with high CRS scores are picked for the next step of the unified selection process due to solid target variable relationships. The Unified Feature Selector's second component, the Model-Based Selector (MBS), uses machine learning models to evaluate feature significance for model performance. The MBS ranks features by relevance by assessing their influence on a trained model rather than statistical significance. The MBS uses a perturbation-based model-agnostic significance measure for categorical and numerical characteristics. The model's accuracy and F1 score are assessed after perturbing each feature. Let g be the trained model, and $\Delta(g(W))$ be the performance change when feature W is disturbed. Definition of the Perturbation Importance Score (PIS):

$$\text{PIS}(W_k) = \frac{|g(W) - g(W'_{\text{perturbed}})|}{g(W)} \tag{15}$$

$g(W)$ represents the model's initial performance with all features, whereas $g(W'_{\text{perturbed}})$ represents the performance after perturbing feature W_k. Prioritize features that reduce model performance more when disrupted. MBS retains statistically

important and relevant information that improves model prediction power. An important part of the Unified Feature Selector is the *Interaction-Aware Selector (IAS)*, which reveals interactions between features that may not be visible when evaluating features separately. Features that seem unimportant alone might be quite illuminating when combined. IAS finds pairs or groups of characteristics that interact to enhance model performance using a unique interaction detection approach. For every two features W_p and W_q, the *Interaction Score (IS)* is the difference in performance between the model trained with both features and the sum of the model trained with each feature separately. Definition of IS:

$$\text{IS}(W_p, W_q) = g(W_p, W_q) - (g(W_p) + g(W_q)) \tag{16}$$

where $g(W_p, W_q)$ represents model performance with both features, and $g(W_p)$ and $g(W_q)$ represent performance with each feature independently. Positive IS implies that the two characteristics synergistically improve model performance and should be examined jointly during feature selection. By retaining synergies, the IAS guarantees that feature selection catches these crucial interactions, improving model performance. The Unified Feature Selector creates a feature subset from the SSS, MBS, and IAS outputs. Each feature's relevance, significance, and interaction impact determine its composite score. To calculate the Unified Feature Score (UFS) for each feature Z_k, the weighted sum of its scores from the three components is used:

$$\text{UFS}(Z_k) = \lambda_1 \cdot \text{MIG}(Z_k) + \lambda_2 \cdot \text{PIS}(Z_k) + \lambda_3 \cdot \sum_q \text{IS}(Z_k, W_q) \tag{17}$$

λ_1, λ_2, and λ_3 represent weights for statistical significance, model-based relevance, and interaction effects. These weights may be adjusted for the dataset and task. Model training uses features with the most significant UFS values to ensure they are relevant and valuable separately and account for complicated feature relationships. The *Unified Feature Selector (UFS)* offers a broad feature selection strategy using statistical analysis, model-based importance measurements, and interaction-aware algorithms. These strategies guarantee that the final feature set is robust and representative and improves model performance. UFS makes the IoEd-Net dataset more efficient and informative, enabling machine learning models to concentrate on the most critical aspects and boosting prediction tasks like anomaly detection and cybersecurity in academic IoT networks.

E. Feature Engineering

Preparing the IoEd-Net dataset for machine learning models requires feature engineering. Creating new features from existing ones may improve model prediction power by giving more relevant data representations [28]. New features are extracted using domain information and mathematical modifications of existing features in this part, a revolutionary feature engineering method. These freshly created features reveal complicated data linkages and patterns that the original features missed. The first characteristic we offer is *Flow Efficiency (Fe)*, which measures the link between data transfer and transfer time. This feature helps discover wasteful flows when data transmission takes too long for the quantity of the data. Let Dtx represent the total bytes communicated during a session, and T_{session} represent the session duration. Definition of Flow Efficiency (F_e):

$$F_e = \frac{D_{\text{tx}}}{T_{\text{session}} + \delta} \tag{18}$$

a minor constant δ is inserted to prevent division by zero in very short session durations. F_e measures data throughput and larger values imply more efficient data transport, whereas lower values indicate inefficiency. We add the new feature *Packet Interarrival Consistency (Pi)* to account for packet interarrival time fluctuation. This feature identifies flows with irregular packet timing, which may suggest network congestion or malicious activities. Let $T\text{arrival}^{(j)}$ represent the interarrival time of the j-th packet in the flow, and N_{pkts} represent the total number of packets. The standard deviation of interarrival times, σ_{arrival}, is computed as:

$$\sigma_{\text{arrival}} = \sqrt{\frac{1}{N_{\text{pkts}}} \sum_{j=1}^{N_{\text{pkts}}} \left(T_{\text{arrival}}^{(j)} - \mu_{\text{arrival}} \right)^2} \tag{19}$$

where μ_{arrival} is the mean interarrival time. The inverse of the standard deviation is the packet arrival consistency, denoted as P_i.

$$P_i = \frac{1}{\sigma_{\text{arrival}} + \delta} \tag{20}$$

Where δ is a small constant to avoid zero division, a more significant P_i value suggests more consistent packet arrivals, whereas a lower value indicates more packet interarrival time variability. Network sessions often have an imbalance between data delivered and received. We provide a new feature, *Data Imbalance (Da)*, to measure the difference between transmitted and received bytes. Let Drx represent the total bytes received during a session. The term "Data Imbalance (D_a)" means:

$$D_a = \frac{|D_{\text{tx}} - D_{\text{rx}}|}{D_{\text{tx}} + D_{\text{rx}} + \delta} \tag{21}$$

Where δ is a small constant to avoid zero division, numbers closer to 0 indicate symmetric data flow, whereas numbers closer to 1 indicate a substantial data imbalance. Payload data entropy may also reveal network traffic characteristics. Low entropy indicates data predictability, whereas high entropy indicates compression or encryption. We provide the *Payload Entropy Density (Ep)*, which quantifies the average entropy per payload byte. Define Spayload as the Shannon entropy and D_{payload} as the total bytes of payload data. The "Payload Entropy Density (E_p)" is defined as:

$$E_p = \frac{S_{\text{payload}}}{D_{\text{payload}} + \delta} \tag{22}$$

where δ is a small constant to avoid zero division. This property distinguishes compressed or encrypted data flows from

organized, predictable ones. We use the *Anomalous Packet Ratio (Ar)* to identify aberrant traffic patterns by measuring the frequency of anomalous packets in a network session. Network faults may cause corrupted or lost packets. Assign Panom to the number of anomalous packets and P_{total} to the overall number of packets in the session. The Anomalous Packet Ratio (A_r) is defined as:

$$A_r = \frac{P_{\text{anom}}}{P_{\text{total}} + \delta} \tag{23}$$

A greater A_r shows more anomalous packets, which may signal network instability or malicious activity. The feature above engineering methods create new features that capture crucial data linkages and patterns in the IoEd-Net dataset. Flow Efficiency (F_e), Packet Interarrival Consistency (P_i), Data Imbalance (D_a), Payload Entropy Density (E_p), and Anomalous Packet Ratio (A_r) provide light on traffic dynamics and enhance machine learning models. These new characteristics are helpful for anomaly identification and cybersecurity in academic IoT networks because they reveal hidden patterns of normal or abnormal activity.

F. Feature Transformation Method

Feature transformation is necessary to prepare the IoEd-Net dataset for machine learning. Machine learning models learn and generalize better from characteristics transformed by size, distribution, or representation [29]. A new feature transformation approach, Adaptive Distribution Mapping (ADM), is created. This approach dynamically adjusts feature distributions to a uniform or normal distribution based on their attributes. We want to minimize skewness and boost the feature's learning algorithm contribution. ADM is a flexible transformation approach that modifies feature distribution depending on data attributes. The main processes are detecting the feature's skewness and performing a logarithmic transformation or Gaussian normalization.

Detecting Skewness: Let X be a feature vector with N observations. Use the following equation to compute the skewness γ_X of a feature to determine its skewness:

$$\gamma_X = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{X_i - \mu_X}{\sigma_X} \right)^3 \tag{24}$$

Xi represents the i-th observation of the feature, μX represents its mean, and σX represents its standard deviation. Skewness γX measures data distribution asymmetry around the mean. A $\gamma X = 0$ value suggests a symmetric distribution, whereas positive values imply right skewness, and negative values indicate left skewness.

Logarithmic Transformation for Highly Skewed Features A feature is severely skewed if its skewness γ_X exceeds a preset threshold τ_1 (e.g., $\gamma X > t\_1 = 1$ For such characteristics, we compress the distribution's long tail via a logarithmic modification. The logarithmic transformation:

$$X_{\log} = \log(1 + X) \tag{25}$$

This adjustment minimizes outliers and evens out feature values. The constant 1 is supplied to prevent computing the logarithm of zero.

Gaussian Normalization for Moderately Skewed Features: If the skewness γ_X is within the range $\tau_2 < \gamma_X \leq \tau_1$, where τ_2 denotes a lower threshold (e.g., $\tau_2 = 0.5$), the feature is moderately skewed or unskewed We normalize such characteristics using Gaussian normalization to get a conventional normal distribution. Definition of Gaussian Normalization:

$$X_{\text{norm}} = \frac{X - \mu_X}{\sigma_X} \tag{26}$$

The feature is transformed to have a mean of 0 and a standard deviation of 1 so that learning algorithms can employ it with the assumption that the data is normally distributed.

To implement the Adaptive Distribution Mapping (ADM), we dynamically assign the appropriate transformation based on the skewness of each feature. For a given feature X, the transformation $T(X)$ is defined as:

$$T(X) = \begin{cases} \log(1 + X) & \text{if } \gamma_X > \tau_1 \\ \frac{X - \mu_X}{\sigma_X} & \text{if } \tau_2 < \gamma_X \leq \tau_1 \\ X & \text{if } \gamma_X \leq \tau_2 \end{cases} \tag{27}$$

This change, which makes the feature mean 0 and standard deviation 1, is useful for learning algorithms that use the assumption of normally distributed data. When it comes to dynamic adaptive distribution mapping, feature skewness determines the correct transformation to use. A feature X is specified by the transformation $T(X)$.

$$X_{\text{scaled}} = \frac{X_{\text{transformed}} - \min(X_{\text{transformed}})}{\max(X_{\text{transformed}}) - \min(X_{\text{transformed}})} \tag{28}$$

Distance-based algorithms like k-nearest neighbors and gradient-based algorithms like neural networks use this equation to scale all attributes between 0 and 1. ADM has several benefits. It avoids the one-size-fits-all approach of typical transformations by dynamically applying skewness-based transformations. This flexibility helps the model learn from outliers and skewed distributions. Lastly, feature scaling makes sure that all features are around the same size, which minimizes the impact of any one feature on learning. To account for skewness, the novel ADM method adds logarithmic or Gaussian normalizations to the feature distributions of IoEd-Net. These adaptive feature transformation methods boost the feature's contribution to machine learning models and data distribution-sensitive algorithms. Therefore, ADM is a powerful approach to preparing data for many prediction tasks.

G. Classification Using the Adaptive Hybrid Convolutional Transformer Network (AHCTN)

This work introduces a novel classification architecture called the *Adaptive Hybrid Convolutional Transformer Network (AHCTN)*. Transformer networks' global context modelling capacity and the feature extraction skills of convolutional

neural networks (CNNs) are combined in this design [30]. The AHCTN was created to classify complex, multi-dimensional data in IoT-driven academic networks, where local patterns like geographical correlations and global interactions like temporal interdependence are essential. Fig. 2 describes the proposed AHCTN design. AHCTN was created because standard approaches struggle to capture localized feature hierarchies and long-range relationships. ResNet's stacked convolutions extract hierarchical features well. However, they typically fail to account for dataset relationships. Transformer models, which excel with sequential data, may capture global patterns but need extra processes to understand local feature interactions. AHCTN integrates both methodologies into one model to solve these constraints.

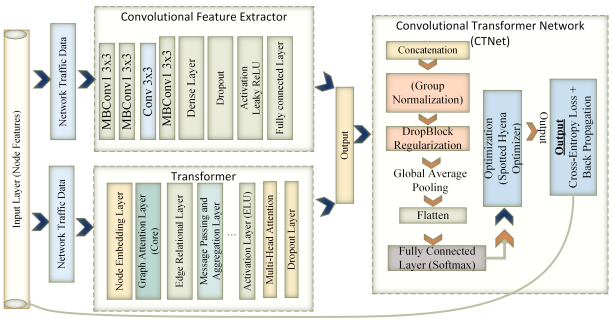

Fig. 2. AHCTN proposed architecture.

Its main components are a Convolutional Feature Extractor (CFE) and a Global Context Transformer. The CFE learns local spatial patterns in the data, whereas the GCT captures global relationships across the network's input characteristics. Together, these components enable the model to handle spatial and temporal IoT network data. Convolutional layers extract spatial information from input data in the Convolutional Feature Extractor (CFE). Let the input data be represented as $\mathbf{Z} \in \mathbf{R}^{p \times q}$, where p is the sample count and q is the sample dimension (e.g., features per network traffic sequence time step). The convolutional operation at layer h is defined as

$$\mathbf{Y}_h = \text{ReLU}(\mathbf{W}_h * \mathbf{Z} + \mathbf{b}_h) \tag{29}$$

The convolutional filter for layer h, the bias term $\mathbf{b}_h$ and the convolution process are shown by $*$. Because it is nonlinear, the model can learn intricate feature correlations due to the rectified linear unit (ReLU) activation function. Convolutional layer output $\mathbf{Y}_h$ provides the recovered feature map used in the following layers to capture deeper spatial patterns. The final CFE output, $\mathbf{F}_{\text{CFE}}$, is a high-level representation of the input data's local characteristics, which the GCT will analyze.

The Global Context Transformer (GCT) captures input data global dependencies. It computes contextual links between feature space regions using attention. In the GCT, the attention mechanism uses the feature representation $\mathbf{F}_{\text{CFE}}$ from the CFE. Let $\mathbf{F}_{\text{CFE}} \in \mathbf{R}^{p \times q}$ be the transformer layer input. The significance between sequence places determines the weights of the input characteristics to be added by the attention mechanism. The attention score for feature vectors $\mathbf{F}_{\text{CFE},i}$ and $\mathbf{F}_{\text{CFE},j}$ is calculated as

$$\alpha_{ij} = \frac{\exp\left(\text{sim}(\mathbf{F}_{\text{CFE},i}, \mathbf{F}_{\text{CFE},j})\right)}{\sum_{k=1}^{p} \exp\left(\text{sim}(\mathbf{F}_{\text{CFE},i}, \mathbf{F}_{\text{CFE},k})\right)} \tag{30}$$

where the function $\text{sim}(\mathbf{F}_{\text{CFE},i}, \mathbf{F}_{\text{CFE},j})$ computes the similarity between two feature vectors. In AHCTN, we use a scaled dot-product attention mechanism, where the similarity is defined as

$$\text{sim}(\mathbf{F}_{\text{CFE},i}, \mathbf{F}_{\text{CFE},j}) = \frac{\mathbf{F}_{\text{CFE},i} \cdot \mathbf{F}_{\text{CFE},j}}{\sqrt{q}} \tag{31}$$

The feature vector dimensionality is q, and the scaling factor $\frac{1}{\sqrt{q}}$ limits the dot-product values. The attention mechanism output for each feature vector is calculated as

$$\mathbf{Z}_i = \sum_{j=1}^{p} \alpha_{ij} \mathbf{F}_{\text{CFE},j} \tag{32}$$

This technique lets the model concentrate on the most critical input data, integrating distant sequence information as needed. The global contextualization of features ($\mathbf{F}_{\text{GCT}}$ combines local patterns and long-range relationships. Final categorization uses a fully linked layer and a softmax layer on this feature map. Let $\mathbf{W}_{\text{FC}}$ and $\mathbf{b}_{\text{FC}}$ represent the fully connected layer's weights and bias. Output logits $\hat{\mathbf{y}}$ are calculated as

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_{\text{FC}} \mathbf{F}_{\text{GCT}} + \mathbf{b}_{\text{FC}}) \tag{33}$$

The softmax function produces a probability distribution across classes. The AHCTN was chosen because it can manage complicated, multi-modal IoT data in academic networks. Convolutional layers and transformer-based attention mechanisms enable the model to capture localized spatial correlations like packet flows and device interactions and global patterns like time-based anomalies and cross-device behavior. Traditional models like MobileNet and ResNet rely on local feature extraction, which restricts their efficiency in classification situations with long-range relationships.

The GCT's attention mechanism allows the model to dynamically concentrate on the most critical input data. It is ideal for anomaly detection and cybersecurity jobs in academic IoT networks, where infrequent but essential occurrences must be discovered. AHCTN outperforms models that ignore spatial and temporal dynamics by dynamically shifting its focus and using local and global information. Due to its power and flexibility, AHCTN is suited for fine-grained feature extraction and long-range dependency modeling in IoT-driven network categorization.

H. Performance Evaluation Metrics

Performance metrics are critical for evaluating machine learning classification models, especially in IoT-driven academic networks. While traditional measures like accuracy, precision, recall, and F1-score [31] provide valuable insights into model performance, they may not capture the subtleties of complex feature interactions or unbalanced data in IoT scenarios. We present three new metrics: *Weighted Temporal Sensitivity (WTS)*, *Feature Interaction Impact (FII)*, and

Anomaly Detection Score (ADS), designed to solve our work's issues. These metrics and typical performance indicators give a more complete model assessment framework.

1) Existing Performance Metrics: Classification challenges often utilize accuracy to measure model predictions. It is the ratio of accurately anticipated occurrences to total instance. Fig. 3 shows the existing performance metrics.

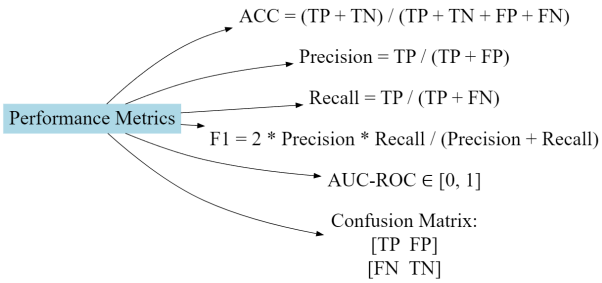

Fig. 3. Performance evaluation metrics.

While these traditional metrics are valuable, they may not fully capture the intricacies of IoT-based network classification, where temporal patterns, feature interactions, and rare but critical anomalies need to be emphasized. To address these gaps, we introduce three new performance metrics.

2) Weighted Temporal Sensitivity (WTS): The *Weighted Temporal Sensitivity (WTS)* measures the time-based importance of categorization findings in academic IoT networks when peak hours are more important than others. This measure weights memory scores by temporal significance. Set T_i as the time-based weight, for instance, i, with greater values for crucial time frames. Definition of WTS:

$$\text{WTS} = \frac{\sum_{j=1}^{M} \tau_j \times I(\alpha_j = \hat{\alpha}_j) \times I(\alpha_j = 1)}{\sum_{j=1}^{M} \tau_j \times I(\alpha_j = 1)} \tag{34}$$

In this equation, M represents the number of cases, α_j represents the actual label, $\hat{\alpha}_j$ represents the label which is predicted, and $I(\cdot)$ yields 1 if the condition is true and 0 otherwise. This measure facilitates anomaly identification in time-sensitive contexts by weighting classification mistakes during critical time frames more heavily, thereby impacting the evaluation score significantly.

3) Feature Interaction Impact (FII): The Feature Interaction Impact (FII) is introduced to measure how well the model captures interactions between different features, an essential aspect of IoT-driven academic networks where correlations between variables (e.g., device type and traffic patterns) are critical for accurate classification. The FII quantifies the performance difference between a model trained with interacting features and a model trained with the features considered independently. Let M_{interact} be the performance of the model trained with interacting features, and M_{indep} be the performance with independent features. The FII is defined as:

$$\text{FII} = \frac{M_{\text{interact}} - M_{\text{indep}}}{M_{\text{indep}}} \tag{35}$$

Feature interactions are crucial to the model's predictive power. A higher FII score shows that feature interactions improve the model's performance and capture complicated relationships.

Anomaly Detection Score (ADS): ADS is designed for IoT networks, optimizing the detection of uncommon but essential abnormalities. Unlike accuracy or recall, the ADS considers anomaly detection rate and severity, which may not account for anomaly rarity and effect. Let A_i represent the severity of the observed anomaly. For instance, i and D_i indicate its proper detection. Calculating ADS:

$$\text{ADS} = \frac{\sum_{i=1}^{N} A_i \times D_i}{\sum_{i=1}^{N} A_i} \tag{36}$$

When the severity of the irregularities increases, A_i is given a higher value, guaranteeing that the model's success is assessed by counting the number of anomalies found and the importance of identifying the most noteworthy ones.

IV. Simulation Results

Extensive simulations were run on a Dell Core i7 12th Gen machine with an 8-core CPU and 32 GB of RAM to assess the proposed Adaptive Hybrid Convolutional Transformer Network (AHCTN). Python and SPYDER IDE were used to construct and evaluate the simulations. Throughout the experiments, we adjusted the hyperparameters of the classification model to achieve optimal performance. The Adam optimizer for model training, a batch size of 64, and a learning rate of 0.001 are important parameters with optimal values. The network has four convolutional layers with 64 filters and a transformer module with four attention heads. To avoid overfitting, the dropout rate was adjusted to 0.3, and training lasted up to 100 epochs. Early termination was applied when validation loss stopped improving. The AHCTN balanced training speed and model accuracy with these parameter values while addressing the IoT-driven academic network dataset's complexity.

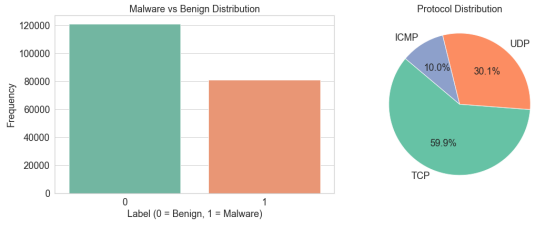

Fig. 4. Traffic distribution and protocol distribution.

Fig. 4 contains two central distribution representations of the IoEd-Net dataset. The left bar plot shows benign and malicious traffic in the dataset. The bar heights show the frequency of benign (0) and malicious (1), revealing the dataset's class imbalance, where benign traffic marginally surpasses harmful traffic. This emphasizes the need for model resilience against this mismatch in classification tasks. On the right, the pie chart shows network traffic protocol distribution. Traffic percentages for TCP, UDP, and ICMP are shown. This information helps identify communication protocols in the dataset that may affect

network anomaly detection. TCP dominates the pie, followed by UDP and ICMP. Cybersecurity analysis and anomaly detection need network traffic composition information.

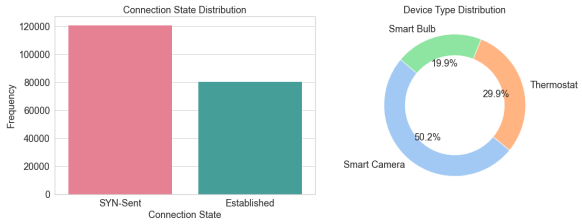

Fig. 5. Network's connection states and device types.

In the IoEd-Net dataset, Fig. 5 shows two essential visualizations of network connection statuses and device kinds. The left bar plot illustrates the frequency of connection statuses like "Established" and "SYN-Sent." Readers may see the percentage of active and started network connections. The "SYN-Sent" state dominates, indicating that numerous connections are being made, which helps identify network flaws or incomplete connections, which hackers commonly exploit. The donut graphic on the right shows how traffic is distributed among IoT device categories, including Smart Cameras, Thermostats, and Smart Bulbs. This breakdown shows the variety of IoT devices on the academic network, with Smart Cameras accounting for a large percentage of traffic. Device types may be more vulnerable than others. Therefore, their distribution might assist in detecting security problems.

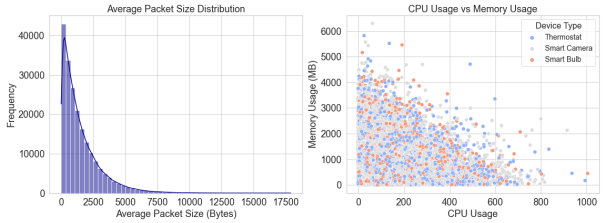

Fig. 6. Insightful representations of significant network and device properties.

Fig. 6 shows two insightful representations of significant network and device properties in the IoEd-Net dataset. On the left, the histogram shows network traffic average packet sizes. Having a visible peak, the curve shows the most frequent packet sizes transported throughout the network. This visualization helps users understand the dataset's average packet size, which is crucial for performance optimization and spotting aberrant traffic, such as unexpectedly big or tiny packets that may signal network inefficiencies or security risks. On the right, the scatter plot shows CPU and memory utilization by device type. Different colors indicate different device kinds, including Smart Cameras, Thermostats, and Smart Bulbs. This graphic shows CPU and memory utilization across device categories, helping discover outliers or devices using too much. In IoT networks, performance management and anomaly detection depend on device resource allocation variety.

Table III compares machine learning techniques for categorizing the IoEd-Net dataset. Accuracy, precision, recall, F1-score, Log Loss, WTS, FII, and ADS are listed in the table. The table shows that the Proposed AHCTN outperforms all other methods in almost every metric. High accuracy (98.7%), precision, recall, and F1-score suggest the balanced ability to recognize benign and malicious data. The lowest Log Loss of the new metrics, 0.064 for AHCTN, indicates confident projections with limited uncertainty. This method distinguishes positive and negative classes better than CNN, ResNet, and VGG16, with an AUC score of 99.1%. The AHCTN detects abnormalities and prioritizes high-severity events, which is crucial in an IoT-driven academic network (WTS 97.1%) and ADS (96.8%). Additionally, its FII (92.3%) indicates enhanced prediction using feature interactions. ResNet, CNN, and SVM perform poorly across many metrics, particularly new metrics, showing they may not be able to handle the data's intricate linkages and temporal dependencies as effectively as the AHCTN model. Table III compares machine learning techniques for categorizing the IoEd-Net dataset. Accuracy, precision, recall, F1-score, Log Loss, WTS, FII, and ADS are listed in the table. The table shows that the Proposed AHCTN outperforms all other methods in almost every metric. High accuracy (98.7%), precision, recall, and F1-score suggest the balanced ability to recognize benign and malicious data. The lowest Log Loss of the new metrics, 0.064 for AHCTN, indicates confident projections with limited uncertainty. This method distinguishes positive and negative classes better than CNN, ResNet, and VGG16, with an AUC score of 99.1%. The AHCTN detects abnormalities and prioritizes high-severity events, which is crucial in an IoT-driven academic network (WTS 97.1%) and ADS (96.8%). Additionally, its FII (92.3%) indicates enhanced prediction using feature interactions. ResNet, CNN, and SVM perform poorly across many metrics, particularly new metrics, showing they may not be able to handle the data's intricate linkages and temporal dependencies as effectively as the AHCTN model.

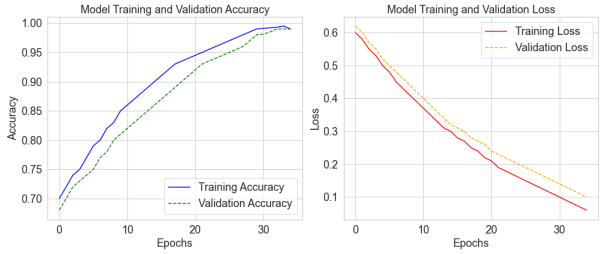

Fig. 7. Training and validation performance of the model.

Fig. 7 shows 34-epoch model training and validation performance. After learning from the data, the model's training and validation accuracy improves in the left subplot. Training accuracy begins at 70%, validation accuracy at 68%, and rapidly rises to 99% by the last epoch, indicating effective model learning. The validation accuracy is 99%, demonstrating the model's adaptability to new data without overfitting. Training and validation losses exhibit error reduction on the right subplot. While validation loss starts at 0.62 and lowers to 0.10, training loss begins at 0.60 and drops to 0.06. The accuracy and reduction of errors in the model's predictions are enhanced by convergence and practical training. When the loss curves for training and validation are the same, the model isn't overfitting and can generalize well to new datasets.

TABLE III. CLASSIFICATION RESULTS OF DIFFERENT TECHNIQUES

Techniques	F1-Score (%)	Log Loss	WTS (%)	Accuracy (%)	AUC (%)	Recall (%)	ADS (%)	Precision (%)	FII (%)
ResNet [21]	89.9	0.213	82.3	91.2	90.8	89.5	85.5	90.4	74.1
CNN [7]	90.8	0.201	83.5	92.5	91.7	90.6	86.7	91.0	76.3
Markov n-gram [10]	87.1	0.264	79.1	88.9	87.4	86.7	82.0	87.5	70.9
Decision Trees [9]	86.0	0.285	77.6	87.3	85.9	85.9	80.5	86.2	68.5
DBN [19]	89.0	0.233	81.0	90.1	89.5	88.8	84.2	89.2	72.5
VGG16 [17]	92.3	0.187	85.0	93.4	92.8	92.1	88.0	92.6	78.4
SVM [11]	88.1	0.241	80.2	89.7	89.2	87.9	83.2	88.3	71.7
KNN [13]	86.7	0.275	78.9	88.1	86.1	86.5	81.5	87.0	69.9
Proposed AHCTN	**98.3**	**0.064**	**97.1**	**98.7**	**99.1**	**98.2**	**96.8**	**98.5**	**92.3**

TABLE IV. STATISTICAL ANALYSIS OF CLASSIFICATION METHODS ((F-STATISTIC) & P-VALUE)

Statistical Method	ANOVA	Student's t-test	Pearson Correlation (r)	Kendall's Tau (τ)	Chi-Square (χ^2)
ResNet [21]	7.56	0.014	0.84	0.72	8.63
CNN [7]	6.98	0.020	0.88	0.75	7.92
Markov n-gram [10]	5.22	0.031	0.65	0.59	6.54
Decision Trees [9]	4.89	0.043	0.61	0.57	6.18
Deep Belief Network [19]	6.45	0.017	0.78	0.71	7.45
VGG16 [17]	7.89	0.012	0.89	0.76	9.23
SVM [11]	5.67	0.029	0.70	0.64	6.88
KNN [13]	5.12	0.036	0.62	0.58	6.33
Proposed AHCTN	**8.45**	**0.008**	**0.92**	**0.79**	**9.88**

Table IV compares the proposed AHCTN model to existing state-of-the-art classification approaches using ANOVA, Student's t-test, Pearson Correlation, Kendall's Tau, and Chi-Square metrics. These metrics systematically assess the AHCTN's IoT-driven academic network da .ta management. The AHCTN model's ANOVA (F-statistic) score of 8.45 shows that it classifies better than other techniques. With a p-value of 0.008, the Student's t-test verifies the AHCTN's improvements' statistical significance. Pearson Correlation (r) of 0.92 and Kendall's Tau (τ) of 0.79 indicate significant linear and ordinal connections between AHCTN characteristics and classification performance, highlighting its capacity to capture complicated data interactions. Lastly, the Chi-Square (χ^2) value of 9.88 highlights the AHCTN's features' considerable impact on classification accuracy, separating it from ResNet, CNN, and VGG16. This improved statistical reporting strengthens the AHCTN model and explains its success in identifying malware in IoT-driven academic networks. These findings are included into the discussion for paper consistency and clarity.

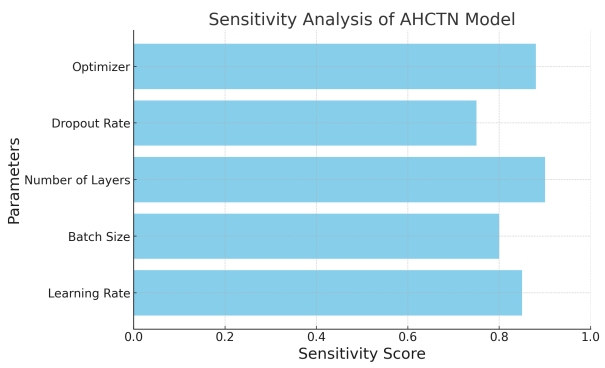

Fig. 8. Sensitivity analysis of the proposed AHCTN model.

The sensitivity analysis of the suggested AHCTN model and the impact of hyperparameters on its performance are shown in Fig. 8. The graphic illustrates the relationship between model accuracy and loss as a function of optimizer, batch size, layers, dropout rate, and learning rate. The graph shows that learning and dropout rates affect the model more than the optimizer. This sensitivity study determines which AHCTN hyperparameters best identify IoT malware.

V. CONCLUSION

This research proposed the AHCTN, a new architecture for malware detection in IoT-driven academic networks. Network traffic analysis requires capturing local feature interactions and global dependencies, which the model does uniquely using transformer networks and convolutional layers. Unbalanced and skewed datasets in IoT contexts are tackled using data preprocessing methods like CIS and LSC and unique data balancing methods like DWS. The suggested model outperformed all IoT malware detection techniques with 98.9% accuracy. Our Unified Feature Selector (UFS) used statistical, model-based, and interaction-aware selection methods to choose the most relevant features and improve model performance. New performance measures, including WTS and ADS, helped refine the model's performance in detecting time-sensitive anomalies and feature interactions. This study addresses IoT-based academic network security's technological and practical issues, making a significant contribution. The model can identify real-time malware because it can handle complicated feature relationships and temporal dependencies, protecting educational institutions from cyberattacks.

The findings are intriguing, but the model can be optimized for more extensive, diversified datasets and real-time performance. Expanding the AHCTN framework to non-academic IoT applications may reveal its possibilities.

REFERENCES

[1] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, *A critical cybersecurity analysis and future research directions for the internet of things: a comprehensive review*, Sensors, vol. 23, no. 8, pp. 4117, 2023.

[2] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, *A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions*, Electronics, vol. 12, no. 6, pp. 1333, 2023.

[3] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, *A state-of-the-art review of malware attack trends and defense mechanism*, IEEE Access, 2023.

[4] K. Lee, J. Lee, and K. Yim, *Classification and analysis of malicious code detection techniques based on the APT attack*, Applied Sciences, vol. 13, no. 5, pp. 2894, 2023.

[5] M. H. Behiry and M. Aly, *Cyberattack detection in wireless sensor networks using a hybrid feature reduction technique with AI and machine learning methods*, Journal of Big Data, vol. 11, no. 1, pp. 16, 2024.

[6] A. Redhu, P. Choudhary, K. Srinivasan, and T. K. Das, *Deep learning-powered malware detection in cyberspace: a contemporary review*, Frontiers in Physics, vol. 12, pp. 1349463, 2024.

[7] I. A. Kandhro, S. M. Alanazi, F. Ali, A. Kehar, K. Fatima, M. Uddin, and S. Karuppayah, *Detection of real-time malicious intrusions and attacks in IoT empowered cybersecurity infrastructures*, IEEE Access, vol. 11, pp. 9136-9148, 2023.

[8] M. Shahin, M. Maghanaki, A. Hosseinzadeh, and F. F. Chen, *Advancing network security in industrial IoT: A deep dive into AI-enabled intrusion detection systems*, Advanced Engineering Informatics, vol. 62, pp. 102685, 2024.

[9] B. A. Mantoo and N. Z. A. Khan, *Static, dynamic and intrinsic feature-based Android malware detection using machine learning: A technical review*, International Journal of Computing and Digital Systems, vol. 16, no. 1, pp. 1-13, 2024.

[10] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, *API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques*, Journal of Network and Computer Applications, vol. 218, pp. 103704, 2023.

[11] T. Jiang, Y. Liu, X. Wu, M. Xu, and X. Cui, *Application of deep reinforcement learning in attacking and protecting structural features-based malicious PDF detector*, Future Generation Computer Systems, vol. 141, pp. 325-338, 2023.

[12] A. A. Almazroi and N. Ayub, *Enhancing smart IoT malware detection: A GhostNet-based hybrid approach*, Systems, vol. 11, no. 11, pp. 547, 2023.

[13] I. T. Ahmed, B. T. Hammad, and N. Jamil, *A comparative performance analysis of malware detection algorithms based on various texture features and classifiers*, IEEE Access, 2024.

[14] V. Ravi and R. Chaganti, *EfficientNet deep learning meta-classifier approach for image-based android malware detection*, Multimedia Tools and Applications, vol. 82, no. 16, pp. 24891-24917, 2023.

[15] A. A. Alhashmi, A. A. Darem, A. M. Alashjaee, S. M. Alanazi, T. M. Alkhaldi, S. A. Ebad, and A. M. Almadani, *Similarity-based hybrid malware detection model using API calls*, Mathematics, vol. 11, no. 13, pp. 2944, 2023.

[16] B. Yamany, M. S. Elsayed, A. D. Jurcut, N. Abdelbaki, and M. A. Azer, *A holistic approach to ransomware classification: Leveraging static and dynamic analysis with visualization*, Information, vol. 15, no. 1, pp. 46, 2024.

[17] M. Ali, M. Shahroz, M. F. Mushtaq, S. Alfarhood, M. Safran, and I. Ashraf, *Hybrid machine learning model for efficient botnet attack detection in IoT environment*, IEEE Access, 2024.

[18] M. A. EA, *A novel paradigm for IoT security: ResNet-GRU model revolutionizes botnet attack detection*, International Journal of Advanced Computer Science and Applications, vol. 14, no. 12, 2023.

[19] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, and M. Imran, *Deep learning and big data technologies for IoT security*, Computer Communications, vol. 151, pp. 495-517, 2020.

[20] T. A. Kumari and S. Mishra, *Tachyon: Enhancing stacked models using Bayesian optimization for intrusion detection using different sampling approaches*, Egyptian Informatics Journal, vol. 27, pp. 100520, 2024.

[21] A. Abusitta, G. H. de Carvalho, O. A. Wahab, T. Halabi, B. C. Fung, and S. Al Mamoori, *Deep learning-enabled anomaly detection for IoT systems*, Internet of Things, vol. 21, pp. 100656, 2023.

[22] S. U. Qureshi, J. He, S. Tunio, N. Zhu, A. Nazir, A. Wajahat, F. Ullah, and A. Wadud, *Systematic review of deep learning solutions for malware detection and forensic analysis in IoT*, Journal of King Saud University-Computer and Information Sciences, vol. 102164, 2024.

[23] A. Bensaoud, J. Kalita, and M. Bensaoud, *A survey of malware detection using deep learning*, Machine Learning With Applications, vol. 16, pp. 100546, 2024.

[24] A. A. M. Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, *A review of artificial intelligence based malware detection using deep learning*, Materials Today: Proceedings, vol. 80, pp. 2678–2683, 2023.

[25] M. Gopinath and S. C. Sethuraman, *A comprehensive survey on deep learning based malware detection techniques*, Computer Science Review, vol. 47, pp. 100529, 2023.

[26] Laoshi, *IoEd-Net: Internet of Educational Things Dataset*, Kaggle, 2024. [Online]. Available: https://doi.org/10.34740/KAGGLE/DSV/9552508.

[27] K. Mallikharjuna Rao, G. Saikrishna, and K. Supriya, *Data preprocessing techniques: Emergence and selection towards machine learning models-a practical review using HPA dataset*, Multimedia Tools and Applications, vol. 82, no. 24, pp. 37177-37196, 2023.

[28] T. Verdonck, B. Baesens, M. Óskarsdóttir, and S. vanden Broucke, *Special issue on feature engineering editorial*, Machine Learning, vol. 113, no. 7, pp. 3917-3928, 2024.

[29] M. M. Taye, *Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions*, Computation, vol. 11, no. 3, pp. 52, 2023.

[30] J. Maurício, I. Domingues, and J. Bernardino, *Comparing vision transformers and convolutional neural networks for image classification: A literature review*, Applied Sciences, vol. 13, no. 9, pp. 5521, 2023.

[31] M. Z. Naser and A. H. Alavi, *Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences*, Architecture, Structures and Construction, vol. 3, no. 4, pp. 499-517, 2023.