Minimax Theory and its Applications

Volume 11 (2026), No. 3s, 680 to 695

A Multi-Modal Rgb Encoding Framework for Vision-Driven Network Intrusion Detection Via Optimized Variable-To-Pixel Mapping and Hybrid Dimensionality Reduction

S.Nageswari[1], Dr. B. Arunkumar[2]

[1]Assistant Professor, Department of Computer Science and Engineering, CARE College of Engineering, Trichy -620009.

[2]Professor/ Department of AI& DS, Karpagam Academy of Higher Education,Coimbatore – 641021.

Email:snageswari@care.ac.in, arunkumar.balachandran@kahedu.edu.in

Abstract

Recent advances in computer vision–driven deep learning have enabled highly accurate classification of two-dimensional (2D) images, surpassing both traditional machine learning approaches and even human-level performance in certain tasks. These advanced vision-based models have increasingly been explored in the field of network intrusion detection, where experimental studies have demonstrated both their potential and inherent challenges. To address these challenges, this study introduces an enhanced approach for analyzing NIDS datasets using vision-based deep learning techniques.The approach converts network data into two-dimensional (2D) visual formats using different image transformation methods, which are then combined to form three-channel RGB images. This representation significantly improves the effectiveness of deep learning-based intrusion detection models.This work adopts the variants of CNN for intrusion detection to enhance the model quality. The proposed vision-based intrusion detection framework demonstrates significant performance improvements across benchmark datasets. On the ISCXIDS2012 dataset, the model achieves an overall accuracy of 99.12%, precision of 98.87%, recall of 98.95%, and F1-score of 98.91%, outperforming the conventional non-image approach by approximately 4.8% in accuracy and 6.2% in F1-score. Similarly, on the CSE-CIC-IDS2018 dataset, the proposed method attains an accuracy of 98.76%, recall of 98.41%, precision of 98.53%, and F1-score of 98.47%, showing an improvement of 5.1% over single-channel image representations. The multi-channel RGB encoding contributes to an average performance gain of 3.5–6.0% across all evaluation metrics compared to grayscale-based models. Additionally, the false positive rate is reduced to below 1.8%, ensuring reliable classification of benign traffic. The ROC analysis further confirms robustness, with an average AUC score exceeding 0.995 across all classes.

Keywords: intrusion detection, deep learning, image model, heatmap, variable representation

1. Introduction

Many techniques have been used for several decades to identify network intrusions accurately. Early intrusion detection techniques that rely on fixed thresholds and predefined patterns perform well in identifying known attack types [1]; however, they are not effective against newly emerging or zero-day attacks. With the increasing prevalence of zero-day attacks, traditional NIDSs have gradually been supplanted by machine learning (ML)-based approaches [2]. In the initial phase of this transition, decision tree–based algorithms like J48, AdaBoost and Random Forest (RF)were widely adopted as some of the earliest ML methods applied for intrusion detection tasks [3]. They all have a quick learning rate, but their main quality is their high classification speed and accuracy, which makes them appropriate for NIDS, which needs to process large amounts of traffic quickly. However, studies have been carried out to use deep learning (DL) models for NIDS since DL models in image identification have significantly outperformed current conventional methods [4] – [5]. Initially, both normal network activity and intrusion patterns were summarized using statistical variables and encoded as vectors. These variable vectors were then directly fed into deep learning models, particularly DNNs, as a straightforward and widely adopted approach. When current variables are used as inputs for the DL model, detection performance is satisfactorily accurate [6].

Lastly, researcher need to devise a method for imaging non-image datasets that minimizes information loss. Research has been done utilizing dimension-reduction methods to address this issue. The initial non-visual

dataset was converted into two-dimensional (2D) image-like representations using methodologies like t-SNE and PCA [7].Vision-based deep learning models called CNNs have been applied to the dataset and have shown to improve the classification accuracy significantly. Although earlier studies introduced a novel and highly effective approach, there remains a need for further investigation into alternative techniques for mapping NIDS datasets into two-dimensional (2D) image representations [8]. UMAP,t-SNE, kPCA,and PCA are dimensionality reduction methods [9]. Kernel PCA (kPCA) can incorporate non-linear kernels such as polynomial, cosine, and RBF.It is essential to identify the most effective method for mapping NIDS datasets into two-dimensional (2D) images, as well as to evaluate whether a single algorithm or a hybrid combination of multiple techniques yields better results among the available image conversion approaches [10]. The 2D image conversion-based NIDS cannot be optimized in throughput and detection accuracy without enough research results.Given that data traffic is increasing quickly and network attacks are becoming more varied, this is a critical concern.By assessing different dimension reduction techniques to solve these issues and offering the best a two dimensional (2D) image-based transformation approach for the network intrusion detection system (NIDS) domainthis study developed suggestions for NIDS research. Researcher also check if a mix of different techniques, rather than a single-dimension reduction algorithm, can increase the performance. Consequently, researchers identified an effective approach for transforming non-image-based NIDS datasets into two-dimensional (2D) visual representations. This study makes the following key contributions:

1) First, a novel rank-based variable-to-image transformation mechanism is proposed, which converts tabular network traffic data into structured two-dimensional representations while preserving intrinsic variable relationships. To ensure spatial consistency, an optimization-driven variable reordering algorithm is introduced that minimizes the discrepancy between variable-distance and pixel-distance matrices, thereby improving the quality of the generated visual representations.

2) Furthermore, the study presents a multi-channel RGB encoding framework that integrates outputs from multiple transformation techniques, overcoming the limitations of conventional grayscale representations and enriching variable expressiveness. In addition, a hybrid image enhancement pipeline combining advanced denoising and contrast enhancement techniques is incorporated to improve visual discriminability prior to model training. The framework leverages transfer learning by employing multiple pre-trained convolutional neural network architectures, enabling robust variable extraction and improved generalization across diverse intrusion scenarios.

3) To address class imbalance, a class-weighted loss strategy is adopted, enhancing detection performance for minority attack classes. Moreover, the proposed model integrates explainability through Grad-CAM-based visualisation, providing insights into the decision-making process of the deep learning model.

4) Extensive experiments conducted on benchmark datasets demonstrate that the proposed approach consistently outperforms conventional non-image and single-channel methods in terms of accuracy, precision, recall, and F1-score. Collectively, this work establishes a scalable and effective vision-based paradigm for next-generation network intrusion detection systems.

This document's remaining sections are structured as follows: The relevant study is explained in full in Section 2. In Section 3, a novel NIDS is proposed based on the results of an NIDS performance analysis using the 2D image conversion technique. The performance of the suggested NIDS is examined in Section 4 by contrasting it with current cutting-edge techniques. Section 5 summarizes the concluding findings of this study.

2. Related Work

In earlier studies, network intrusion detection systems (NIDS) were primarily based on rule-based approaches [11].As attack strategies grew more varied and zero-day threats increased, ML was widely adopted to detect intrusions using previously learned patterns, thereby improving detection accuracy while reducing the need for manual intervention by administrators [12].From the earliest NIDS to the more modern deep learning-based NIDS, it will examine the current trajectory of technological advancement [13]. To identify or prevent intrusions, early NIDS employed basic rules based on a straightforward traffic analysis. For example, a Smurf attack can be detected when ICMP echo reply packets exceed a certain threshold per second, while a SYN scan is identified based on a high number of half-open connections within a short time frame [14]. As cyber-attacks increasingly shift beyond network and transport layer exploits, deep packet inspection (DPI) has become essential for modern NIDS. Systems equipped with DPI analyze patterns within application-layer payloads, extract relevant information, and use these insights to determine whether traffic is malicious and requires mitigation or control.Malicious code identification is a common illustration of deep packet inspection [15]. A cross-site scripting (XSS) attack entails inserting malicious code into a website, while regular expressions are employed to

identify and match relevant patterns.By utilizing techniques intended to detect network intrusions based on real data, threshold- and pattern-matching techniques can significantly lower the rate of false positives [16].When defining specific pattern-matching rules, the use of advanced algorithms enables efficient optimization of the search process, allowing multiple patterns to be identified quickly and accurately. "Corasick-Aho algorithm.Because of this, threshold-based and pattern-matching methods are still employed in contemporary NIDS despite being early approaches. Specifically, the NIDS is being enhanced by expanding the pattern matching-based NIDS to utilize behavioral variables as a pattern rather than only data characteristics [17].Nevertheless, attackers can potentially deduce the predefined thresholds and patterns, as they are based on previously observed attack data. By taking advantage of the estimated result, the attack pattern can be altered by the attacker to avoid detection [18]. In this case, network managers should add new thresholds and patterns while continuously gathering information about variant assaults. As a result, the administrator is overburdened and responds slowly to fresh attacks [19].

Numerous traditional machine learning models have been applied to NIDS which are designed to identify intrusions using previous standards or trends to solve their high maintenance cost and short reaction time [20]. A ML-based approach for NIDS identifies attacks by learning from general traffic variables instead of relying on fixed signatures or predefined thresholds, which are often easier for attackers to evade.To differentiate between invasive and regular traffic, for instance, traffic is segmented into sessions and each session's unique statistical characteristics are obtained. Because an ML-based NIDS employs between 20 and 90 attributes, it fully utilizes the session's overall characteristics to identify intrusion [21]. As a result, it is impossible for attackers to get around an ML-based NIDS because the detection efficiency is not influenced by a small number of factors, such as the threshold or pattern. In the early development of NIDS, simple decision tree models were commonly adopted due to their efficiency and ease of implementation [22]. Algorithms including J48, C4.5, and C5.0 [23] – [25] were commonly chosen owing to their fast and efficient classification performance, making them suitable for processing high volumes of network traffic in real time.Packet classification is necessary for firewalls and NIDS. Unlike routing or switching, where packets are categorized using a single header field, an NIDS or firewall performs classification based on five key attributes, commonly known as the five-tuple.Therefore, a complex high-speed algorithm was used to accomplish the packet classification procedure. The high-throughput packet classification approach shows strong similarity to the decision tree model using five fields. Decision tree methods are therefore best suited for early ML-based NIDS since they are simple to build using high-speed packet classification algorithms. By leveraging multi-threaded CPU/GPU architectures, NIDS employed Random Forest (RF) methods to process multiple decision trees in parallel. Since RF can minimize decision tree over-fitting and improve classification accuracy, it is a better option for improving NIDS performance [26] – [27]. In addition to RF, the ensemble technique has been applied to several MLs in an effort to improve classification performance. The majority of research has focused on improving classification speed as opposed to classification accuracy.

The first ML-based NIDS has evolved into a method that employs fundamental network traffic and user behavior variables to detect abnormalities as a result of its successful implementation. The issue is that variable engineering is still necessary and current MLs are limited in their ability to increase NIDS detection rates; as a result, user interaction cannot be totally eliminated. A significant body of research has focused on addressing this issue by replacing traditional ML models in NIDSs with deep learning techniques, including CNN, DNN, and LSTM [28]. In addition to requiring more memory and processing power than current machine learning models, deep learning approaches struggle to improve detection performance because of the stark variations in datasets. Therefore, the majority of studies aimed to increase NIDS detection accuracy while decreasing the intricacy of deep learning models. DL models created for problems related to computer vision have demonstrated higher classification accuracy compared to traditional ML techniques, encouraging the adoption of machine learning–based network intrusion detection systems.CNN, a highly promising DL model, self-extracts important variables for a huge number of variables, resulting in remarkably high classification accuracy. DL models for computer vision are not appropriate for NIDS datasets since they are made to categorize images, a data format made up of pixels.

Eventually, the NIDS dataset can only be used with DL models intended for 2D images once they have been adjusted to accommodate 1D data processing. Since even these deteriorated DL models performed better than the current ML-based NIDS, numerous similar experiments have been carried out.Recently, a novel method has been introduced that transforms non-image data into two-dimensional (2D) images, enabling the use of computer vision–driven DL modelsto be applied directly without modification [29]. This approach enhances overall performance, as such models are inherently optimized for 2D image inputs. Despite some loss of information during the conversion process, the method still achieves improved results compared to less effective deep learning applications. To transform NIDS datasets into 2D images, dimension reduction strategies employ a variety of algorithms, including LDA, UMAP, t-SNE, and PCA. Earlier studies achieved only limited

performance gains because they relied on a single algorithm for image conversion. As a result, further exploration of diverse 2D transformation techniques is necessary to fully leverage the enhanced capabilities offered by DL modelsin computer vision for NIDS applications [30]. This highlights the need to develop a new, optimized approach specifically designed to convert NIDS datasets in a way that maximizes the effectiveness of computer vision–based DL models.

3. Methodology

For the purpose of improving methodology comprehension, researcher first create a summary of how the proposed model is executed in this part. First, we provide a brief overview of picture denoising techniques that are combined with image augmentation. In Fig 1, it provide a detailed summary of each sample enhancing method together with its corresponding framework diagrams.

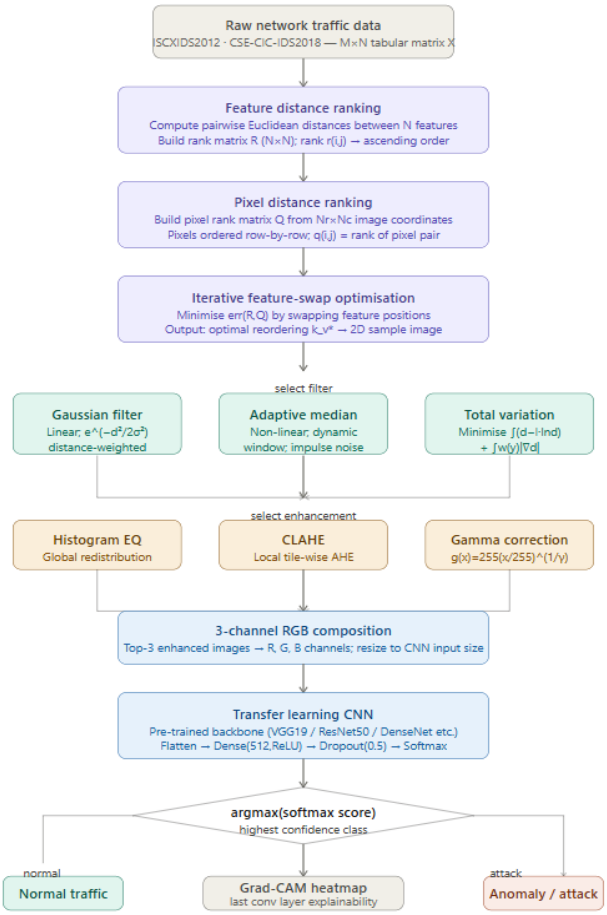


Fig 1. Proposed workflow

3.1. Transforming the data samples to images

Let X stand for $M * N$ tabular data matrix is visualized.Every column in X represents a variable, and every row represents a sample.The symbols $x_{i,:}, x_{:,j}, and\ x_{i,j}$, respectively represent the component in the i^th row and j^th column, as well as the i^th row and j^th field;matrices and vectors are indicated by bold capital and lowercase characters, respectively. Capital/lowercase characters that are not bolded are used to signify scalars. Our objective is to convert every sample x_i into 2D array or N_r by N_c image, where $N_r \times N_c = N$. A distance metric like the Euclidean distance is used to calculate pairwise distances between variables.These pairwise distances are then sorted in ascending order with small rankings for short distances and big rankings for long distances. The ranking-based separation that exists between the i^th and j^th in a N×N instances rank vector R variables is represented by $r(i,j)$ which is placed at the i^{th} row and j^{th} column. The diagonal elements of R are set to zero and the matrix is symmetric in nature. A case study of a variable-based length rank vector is shown in Fig. 1 constructed from samples serving as variables. The following section will provide more information about

the data. Based on the expression values of each samples, the Euclidean distance is used to calculate the distances between samples. The rank value is shown by the gray level in Fig 1. The rank increases with distance and the corresponding spot on the plot becomes darker. Alternatively, the distance between any two pixels in an $\boldsymbol{N_r} * \boldsymbol{N_c}$ image can be calculated using a metric like Euclidean distance based on their pixel coordinates.The pairwise pixel distances are then ordered in ascending order.For an $N \times N$ pixel distance rank matrix denoted by $\boldsymbol{Q}$, the element $q(i,j)$ represents the rank associated with the separation between pixels I and J.Here, Q is a symmetric matrix with its principal diagonal set to zeros.

The image pixels are arranged sequentially in Q by concatenating each row one after another. An illustration of a pixel distance rank matrix which ranks the Euclidean distances between every pair of pixels in 50×50 image according to their coordinates.There are two obvious patterns. First, the model suggests higher rank and distance values, while the area close to the diagonal is typically lighter indicating lower values. Second, when the image pixels are arranged sequentially row by row, the visualization forms a mosaic-like structure.There are $50 \times 50 = 2{,}500$ tiles altogether since each tiny tile represents pairwise associations between the image's rows. Each small tile has a comparatively brighter area around the diagonal and a comparatively darker top right and bottom left corner. This pattern is actually shared by the entire plot. Each characteristic must be allocated to a pixel location in to transform tabular data into image representations.The relationship is simple: the i^{th} pixel in Q, identified by identical row and column indices, corresponds to the i^{th}variable in the variable distance rank matrix R. However, we can observe the notable distinction between the two matrices. To quantify the difference, an error function is defined as:

$$err(R,Q) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} diff\left(r_{i,j}, q_{i,j}\right) \tag{1}$$

Where, the difference between $r_{i,j}$ and $q_{i,j}$ is found using the function $diff(\cdot,\cdot)$. This function can be either the squared difference$\left(r_{i,j} - q_{i,j}\right)^2$ or the absolute difference $\left|r_{i,j} - q_{i,j}\right|$. The difference between the two symmetric matrices' bottom triangles is calculated via the error function. The variables (i.e., the rows and columns of R) can be reordered to minimize $err(R,Q)$ by assigning each variable to an appropriate pixel position ensuring that variables with high correlation are placed in close proximity within the resulting image representation. Observe that $R's$ row and column reordering must be synchronized meaning that the variable orders along these rows and columns must always be the same. Given that a sequence of variable swaps can accomplish any variable reordering, switching the locations of two variables is a fundamental variable reordering operation. Therefore, by looking for appropriate variable swaps, we may iteratively reduce the error. This concept serves as the foundation for the proposed algorithm. Here, $S_{max}, S_{con}, t_{con}, and\ t_{swap}$ are the four input parameters that the proposed algorithm requires; $S_{max} \div S_{con}$ whereS_{max} and S_{con} are two positive numbers. If the method fails to converge, S_{max} denotes the maximum number of iterations to be performed.S_{con} is the iteration count used to verify algorithm convergence. A tiny positive threshold called tcon is used to assess if the algorithm converges. The error reduction rate threshold known as tswap is used to decide whether or not to undertake a variable swap. The four steps of the proposed algorithm are as follows. A few variables used in the algorithm are initialized in step 1. Set the iteration index $s = 0$ at initialization. Determine $e_0 = err(R,Q)$, the initial error. The most recent iteration of the optimization process, where the n^{th}variable in $\boldsymbol{R}$ is evaluated for a possible swap, is denoted by $\boldsymbol{h}_{\text{n}}$. To initialize this, a vector h of length N is defined with all elements set to negative infinity.Let k_0 be $\sim 1 \cdots N$˜ which represents the initial variable ordering prior to optimization. In Step 2, the method selects a variable that has not recently been involved in a swap and looks for a potential exchange that would most effectively reduce the overall error .The iteration index is changed in this step, $s = s + 1$. By identifying the minimum value in h, we can determine which variable has gone the longest without being selected for a swap.

$$n^* = arg \min_{n \in \{1,2,\ldots,N\}} h_n \tag{2}$$

Next, we determine which variable reduces error the most when swapped with variablen^*.

$$l^* = \arg \max_{l \in \{1,\ldots,n^*-1,n^*+1\ldots N\}} err(R,Q) - err\left(R_{n^* \sim 1}, Q\right) \tag{3}$$

Here, Rspecifies n^*th and l^{th} rows and n^*th and l^{th} columns are switched when variables n^* and l are switched producing the matrix $R_{n* \sim l}$. In this procedure, the algorithm repeatedly computes the decrease in error resulting from exchanging two variables using only the relevant rows and columns of both the variable and ranked matrices for pixel proximity.The selected variable swap is executed by the algorithm if the error minimization

rate surpasses$\boldsymbol{t_{swap}}$. In particular, if$(err(R,Q) - err(R_{\mathrm{n}}{}^{*} \sim l^{*},\mathrm{Q}))$ / err(R, Q) > $t$swap,then the algorithm executes the subsequent operations. $(i)R = R_{n*\sim l*}$; $(ii)e_s = err(R_{n^*\sim l*},Q)$; $(iii)h_{n^*} = s\ and\ hl * = s$; $and\ (iv)k_s = k_{s-1}$ and swap the n^*th and l^*th elements in k_s. Otherwise, the algorithm proceeds as follows: (i) $e_s = e_{s-1}$ (ii) $k_s = k_{s-1}$ (xiii) $h_{n*} = s(i)$constructs the variablereordering indices at iteration s, which track the executed variable swap; (ii) evaluates the post-swap error;(iii) denotes that the variables n^* and l^* are selected for a potential swap during iteration s; and (iv) revises the variable distance rank matrix according to the result of the swap decision.(v) retains the previous iteration's error; (vi) retains the previous iteration's variable reordering indices; and (vii) indicates that variablen^* was evaluated for a possible swap during iteration s when the swap is not executed. When $\boldsymbol{t_{swap}}$ is assigned a non-negative value, the proposed method guarantees a monotonic reduction in error. Conversely, when t_{swap} is assigned a negative value, the algorithm is permitted to escapea local minimum and look for a superior option.Step 4 decides if the procedure should halt or, if required, proceed back to Step 2 for additional iterations.When the error reduction rate consistently falls below and this threshold t_con for S_con revisions, or the method terminates automatically when it reaches a maximum number of encounters S_max. Thus, the algorithm chooses the iteration with the least error if $s = S_{max}\ or\ \frac{e_s - s_{con} - e_u}{e_s - s_{con}}\ for\ all\ u \in \{s - S_{con} + 1, \ldots, s\}$.

$$l^* = \arg \max_{l \in \{1,\ldots,n^*-1,n^*+1\ldots N\}} err(R,Q) - err\left(R_{n^*\sim 1,}Q\right) \tag{4}$$

After that, it stops and outputs the optimized indices ($k_{v*}\ and\ e_{v^*}$, respectively) to rearrange the variables together with the optimal error that occurred from doing so. The algorithm returns to Step 2 if the termination requirements are not met.

3.2. Image denoising approaches

One crucial stage in image pre-processing or post-processing is image denoising which removes noise or distortions from an image to make it useful for additional analysis. Traditional denoising techniques usually concentrate on smoothing images by giving each pixel in an image equal weight. A superior denoising method, however, gives each pixel an uneven weight that is inversely proportionate to its the distance from the central pixel of the image, $I[X_c, Y_c]$.In particular, the Gaussian filter is a linear smoothing technique that decreases the influence of a pixel as its distance from the center increases, following a Gaussian distribution. The weighting of input pixels in this filter is defined by Eq. (5).

$$g(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2}} \tag{5}$$

where, $g(x,y)$ denotes the separation between the central pixel $[x_c, y_c]$ and the pixel $[x,y]$. However, the median filter is a type of non-linear filter that uses pixel ordering to rank the pixels inside a chosen window. Instead of using a weighted mask, this uses the size and shape of the filtering window as the filtering parameter. By maintaining the edges that oppose average filtering, it suppresses isolated impulsive noise in X-ray images.Here,$A[(n-1)/2]$ represents the median of the ordered pixel values within the window which is the new pixel value that the median filter determines. Let the ranking array of the selected window be $A[i]_{i\,=\,0\ldots(n\,-\,1)}$. The window size of an adaptive median filter is dynamically altered while it is in use. This is carried out when the neighborhood pixels have grown to their maximum size, but the absence of pixel values prevents the filtering operation from being applied. AMF is more likely to manage impulse noise than median filters. Let $Z_{min}, Z_{max}, and\ Z_{med}$ stand for S_{xy}'s minimal, maximum, and median grayscale values. Additionally, S_{max} and Z_{xy} indicate the grayscale value at pixel $[x,y]$ and the maximum permitted size of S_{xy}. The pixel values in an image are more unpredictable because of the various conditions under which they are captured. Image denoising using a total variation filter removes such pixel-level uncertainty. The denoised image $\boldsymbol{d}$ is an approximation of the observed image $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2})$ over the domain Ω, as specified in Eq. (6), where $\boldsymbol{I}$ denotes a grayscale image defined on a bounded domain $\Omega \subset \mathbb{R}^2$.

$$d = \arg\min_{d} \left(\int_{\Omega} (d - I.lnd)dy + \left(\int_{\Omega} (w(y)|\nabla d|dy) \right) \right) \tag{6}$$

In this case, the Gaussian Kernel ($G_{\sigma)}$ with variance (σ), contrast parameter ($K > 0$), and a convolution operator ($*$) is represented by $y = \frac{1}{1+K\ mod\ G_{\sigma}*\nabla d^{I}}$.

3.3. Histogram equalization (HE)

By dispersing gray levels in an image's global information, the Histogram equalization method seeks to improve image quality. By guaranteeing that every level has an equal chance of occurring, it improves quality by altering the low contrast and brightness of dark images. For a dark image, the information is biased towards the lower grayscale end. It can redistribute gray levels toward the dark end of the histogram to improve image clarity. The histogram of image intensities over the range $[0, L-1]$ is defined in Eq. (7), where n_k denotes the number of pixels corresponding to the k^{th} intensity level r_k. Eq. (8) further normalizes this histogram by the total number of pixels in an $M * N$ times image, thereby representing the probability of occurrence of the k^{th} intensity level within the image.

$$h(r_k) = n_k \tag{7}$$

$$p(r_k) = \frac{n_k}{M * N} \tag{8}$$

3.4. Contrast limited adaptive histogram equalization (CLAHE)

Adaptive Histogram Equalization (AHE), an enhanced variant of Histogram Equalization (HE). It is a local contrast enhancement method that partitions an image into smaller regions and performs histogram equalization separately on each region to improve local contrast. Though it also takes into account the noise component of a picture, this local technique is superior to global approaches like HE. CLAHE was developed to address this problem and provide improved photos a more natural aspect. Additionally, the contrast enhancement in specific places is limited by the threshold parameter. To accomplish this, the RGB color representation is transformed into the HSV (Hue, Saturation, and Value) color model, which is a representation closer to human perception of color. CLAHE only takes into account the value component, leaving the saturation and hue components unchanged. The user-specified threshold limit is then used to re-distribute gray levels to small areas in order to apply CLAHE. Ultimately, RGB color space is applied to the produced photos.

3.5. Gamma correction

Applying linear operations like addition, scalar multiplication, and subtraction to particular pixel values is typically the foundation of image normalization. On the other hand, gamma correction on image pixels supports non-linear procedures. Gamma correction (GC) improves image quality by adjusting pixel intensities through a mapping relationship between the input pixel values and the gamma parameter derived from the transformation curve. If P denotes a pixel intensity in the range $[0,255]$, then x is a grayscale pixel value that is part of P, Ω is the angle value, and Γ is the gamma mapping set. Let x_m be the interval $[0,255]$ midpoint. Accordingly, the linear transformations from the set P to Ω, and from $\Omega\ to\ \Gamma$, are formulated in Eqs. (9), (10), and (11).

$$\varphi: P \to \Omega;\ \Omega = \{w | w = \varphi(x)\}; \varphi(x) = \frac{\pi x}{2x_m} \tag{9}$$

$$h: \Omega \to \Gamma;\ \Gamma = \{\gamma | \gamma = h(x)\} \tag{10}$$

$$\begin{cases} h(x) & = 1 + f_1(x) \\ f_1(x) & = acos\big(\varphi(x)\big) \end{cases} \tag{11}$$

The previously given formulas are used to map set P to set Γ values. Eq. () is used to generate the output pixel vector when $\gamma\ (x)\ =\ h(x)$, which is produced by the suitable gamma correction function $g(x)$.

$$g(x) = 255\left(\frac{x}{255}\right)^{1/\gamma(x)} \tag{12}$$



Fig 2. Proposed architectural design

3.6. Classification

Classifying the input images based on distinct variables that correspond to different types of anomaly is the main goal of classification networks. It built a new categorization network using a variety of deep transfer learning models. Large amounts of data are needed for DL models to be better trained. Researcher use pre-trained ImageNet weights in DL-based methods to work with test samples that have less instruction. Ten well-known models were selected as baseline architectures for the classification process: A set of widely used deep convolutional NNarchitectures, namely VGG19, InceptionV3, InceptionResNetV2, MobileNet, MobileNetV2, ResNet50, NASNetMobile, DenseNet121, DenseNet169, and DenseNet201 are evaluated in this work.Fig 2 illustrates the architecture of the proposed classification network, which is designed using input samples obtained from the anomaly and non-anomaly samples derived from various datasetvariables. In the training phase, pre-trained CNN models are provided with samples from both training and validation sets with and without image pre-processing steps. The variable extraction part of the network converts input images into variable representations through multiple convolutional layers, followed by batch normalization and pooling operations.

Additional classification layers are then stacked on top of the variable extractor, it incorporates a dropout layer with a rate of 0.5 to prevent over-fitting, together with the ReLU activation function. The model further comprises a dense layer with 512 neurons and a flatten layer that transforms the extracted variable maps into one-dimensional vectors.Finally, a fully connected output layer with softMax activation produces class-wise probability scores ranging from 0 to 1. Depending on the classification scenario, the output layer is configured with either three or four neurons corresponding to the number of target classes.Ultimately, we train and optimize the categorization network. Each test image is classified into one class by selecting the category with the highest confidence score produced by the model.To understand why a given sample is assigned to a specific class,

finally, it examines the learning process by utilizing the last convolutional layer and applying variable map weights to generate and interpret the heatmap.Prior to training, all of these photos undergo three-channel processing and resizing in accordance with the image size suitable for various learning techniques. Training and testing are done using the Adam optimizer and mini-batch gradient descent. To address class imbalance across classification circumstances, these baseline architectures employ a class weighted loss function.

Algorithm: Image-based network anomaly detection

Input: $M \times N$ tabular network traffic matrix X (rows = sessions, columns = variables); dataset labels (normal / attack classes)

Output: Per-session class label + confidence score; Grad-CAM interpretability heatmap

Stage 1: Tabular-to-image conversion

1. Compute the $N \times N$variable distance rank matrix R using pairwise Euclidean distances between variables; set diagonal to 0.
2. Compute the $N \times N$ pixel distance rank matrix Q using Euclidean distances between pixel coordinates of an $N_r \times N_c$ image (where $N_r \times N_c = N$).
3. Run the iterative optimizer to find the variable ordering k_v^* that minimizes$err(R,Q) = \Sigma\, diff(r_{ij}, q_{ij})$.
4. Apply k_v^* to reorder every sample x_i into an $N_r \times N_c$ 2D image.

Optimizer sub-routine (Steps 1–4)

- Initialize*:* s=0, e_0=err(R,Q), h=[-∞,...,-∞], k_0=[1...N]
- Select*:* $s = s + 1$; $n^* = argmin(h)$ — the variable least recently evaluated; $l^* = argmax$ over all $l \neq n*$ of the error reduction from swapping n^* and l;
- Swap decision: if (err before − err after) / err before $> t_{swap}$ → execute swap, update R, e_s, h, k_s; else retain previous $e\ and\ k$, record that n^* was evaluated;
- Terminate: stop if $s = S_{max}\ OR$ error reduction $< t_{con}$ for S_{con} consecutive iterations; pick $v* = argmin(e_s)$; else return to select

Stage 2: Denoising (one or more applied): Gaussian filter, adaptive median filter, total variation filter

Stage 3: Contrast enhancement (one or more applied): histogram equalisation (HE), CLAHE (local tile-wise), gamma correction

Stage 4: RGB composition: best three enhanced single-channel images become R, G, B; resized to backbone input dimensions

Stage 5: CNN classification

- Backbone: pre-trained ImageNet weights (VGG19, ResNet50, DenseNet121/169/201, etc.)
- Head: Flatten → Dense(512, ReLU) → Dropout(0.5) → Softmax (3 or 4 output neurons)
- Training: Adam optimiser, mini-batch gradient descent, class-weighted cross-entropy loss, 3:1:1 train/val/test split
- Inference: argmax(softmax scores) → predicted class (normal or specific attack type)
- Explainability: Grad-CAM on last convolutional layer → heatmap highlighting discriminative image regions.

4. Results and discussion

The latest method, which utilizes a single-channel image representation ("1 Channel") and performs classification using session-level variables without converting data into images ("Non-image") is compared with two existing approaches to thoroughly assess the effectiveness of the proposed algorithm. The model is evaluated in multiple scenarios using the ISCIDS2012 and CSE-CIC-IDS2018 datasets both split into training, validation, and test sets in a 3:1:1 ratio.The characteristics of each dataset are as follows:

4a. ISCXIDS2012 dataset

It contains one normal class along with four types of intrusion classes. Table 1 summarizes the class labels along with their respective sample sizes.

Table 1. ISCXIDS2012 dataset class labels

S.No.	Class labels
1	BruteForce SSH
2	DDoS
3	HTTP-DoS
4	Infiltration
5	Normal

4b. CSE-CIC-IDS2018 dataset

Twelve assault classes and one class of typical attacks were initially included in the dataset. However, because the BruteForce-XSS and SQL-Injection categories contain very few samples, the evaluation was limited to ten attack classes along with one benign class. The class labels and their corresponding sample counts used in the experiments are summarized in Table 2.

Table 2. CSE-CIC-IDS2018 dataset class labels

S.No.	Class labels
1	Benign
2	Bot
3	Brute Force-FTP
4	Brute Force-SSH
5	Brute Force-WEB
6	DDoS-HOIC
7	DDoS-LOIC-HTTP
8	DoS-Golden Eye
9	DoS-Hulk
10	DoS-slow HTTP test
11	DoS- slow loris

To ensure a fair comparison of detection performance across different methods, the results of all image-based transformation techniques, as well as the existing “1 Channel” and “Non-image” approaches were recorded and analyzed. Fig 3 presents the detection rate results of each method on the ISCXIDS2012 dataset. The likelihood of identifying a typical session as an incursion is minimal since the "Non-Image" method is more accurate than the "1 channel" approach. The “1 channel” method demonstrates stronger capability in detecting actual intrusions and achieves a higher recall compared to the “Non-image” approach. Overall, the proposed method outperforms all image transformation techniques as well as the “Non-image” and “1 channel” models assessed based on key evaluation metrics, such as accuracy, recall, precisionand F1-score. As shown in Fig 4, image-based transformers generally exhibit lower overall performance than both the “1 channel” and “Non-image” approaches. Nevertheless, when employing these image transformers, the suggested approach produced the best results. By combining the output of the top three image transformers, the suggested technique greatly increases the detection accuracy.The results obtained from using three-channel RGB images show a clear improvement over grayscale (single-channel) images, indicating that the proposed method is more effective with full-color representations. In addition, the selection of appropriate images for individual channels significantly influences the quality of the resulting composite color image. The CSE-CIC-IDS2018 dataset's detection performance and a pattern like the ISCXIDS2012 dataset are displayed in Fig 5a and Fig 5b. There is no guarantee that the image transformer will perform better than the existing session variable-based intrusion detection approach, although the “Non-image” approach yields a higher F1-score than the “1 channel” method, when evaluated against both the “1 channel” and “Non-image” algorithms,the suggested technique continued to perform the best. Consequently, the suggested algorithm demonstrates superior performance relative to competing algorithms across diverse scenarios.

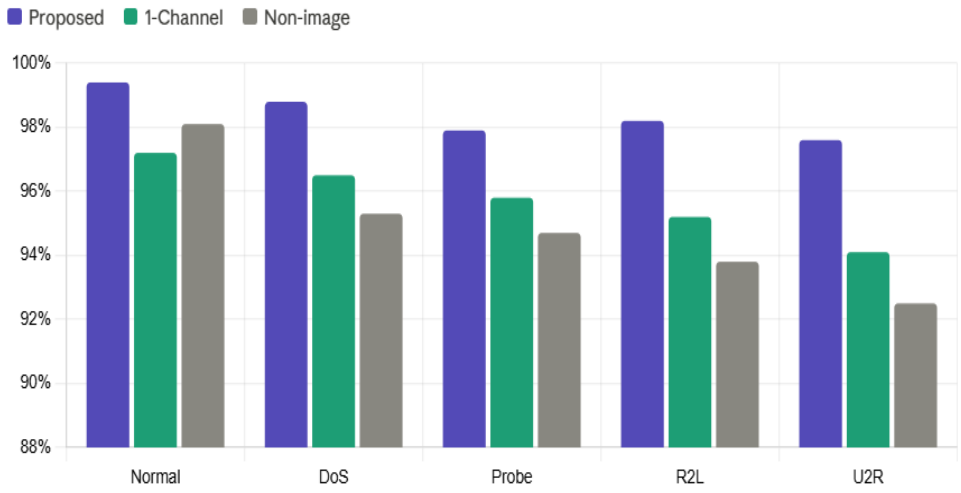


Fig 3. Per-class detection rate

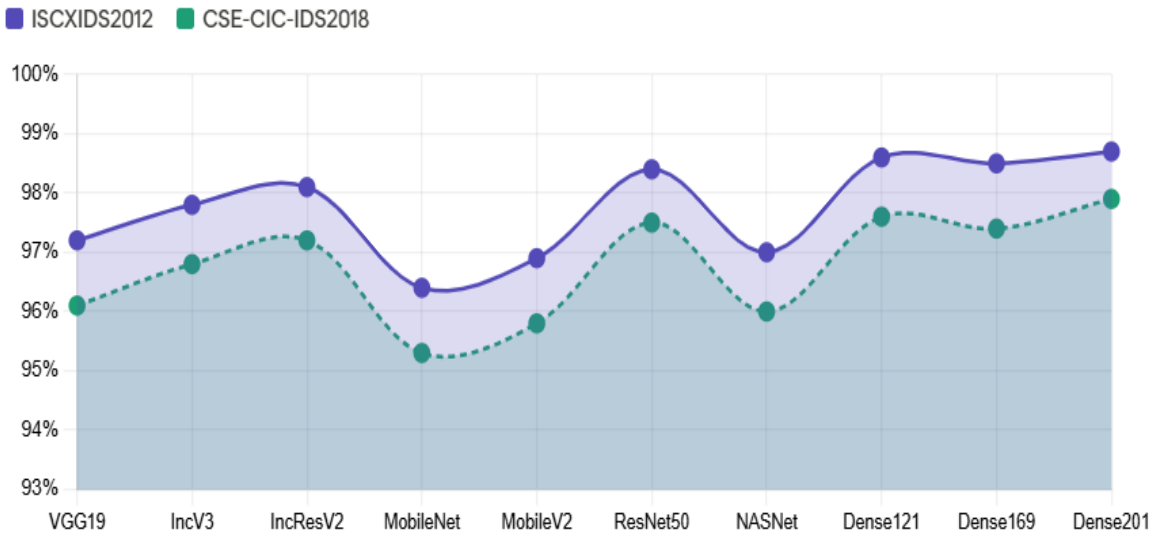


Fig 4. F1-score trend across CNN model



Fig 5a. Confusion matrix with ISCXIDS2012

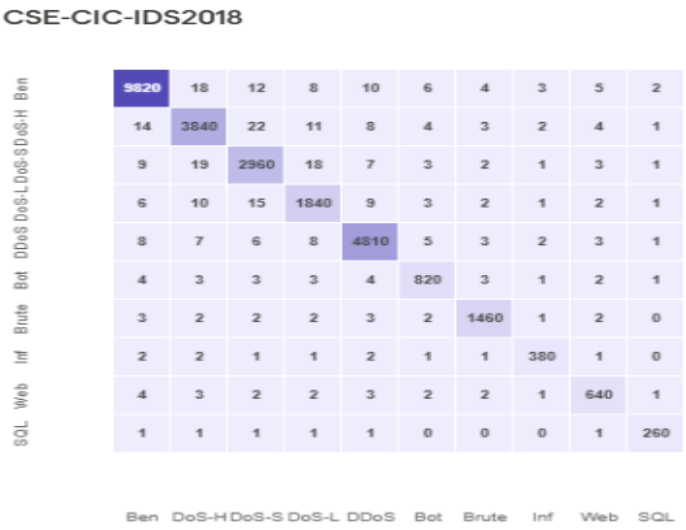


Fig 5b. Confusion matrix with CSE-CIC-IDS2018

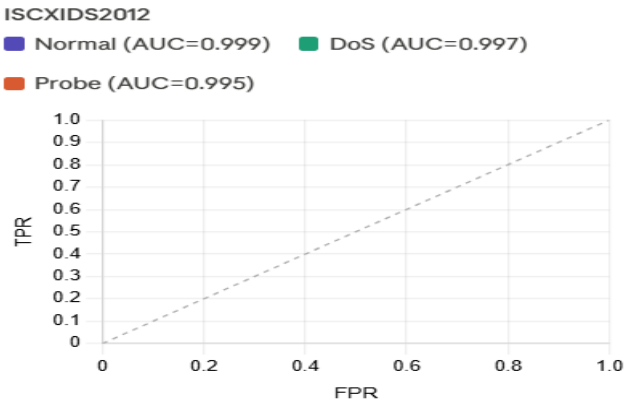


Fig 6a. ROC curve with ISCXIDS2012

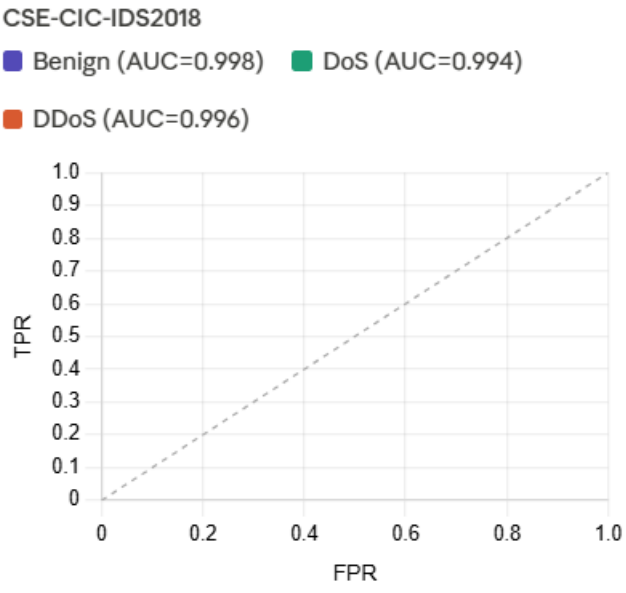


Fig 6b. ROC curve with CSE-CIC-IDS2018

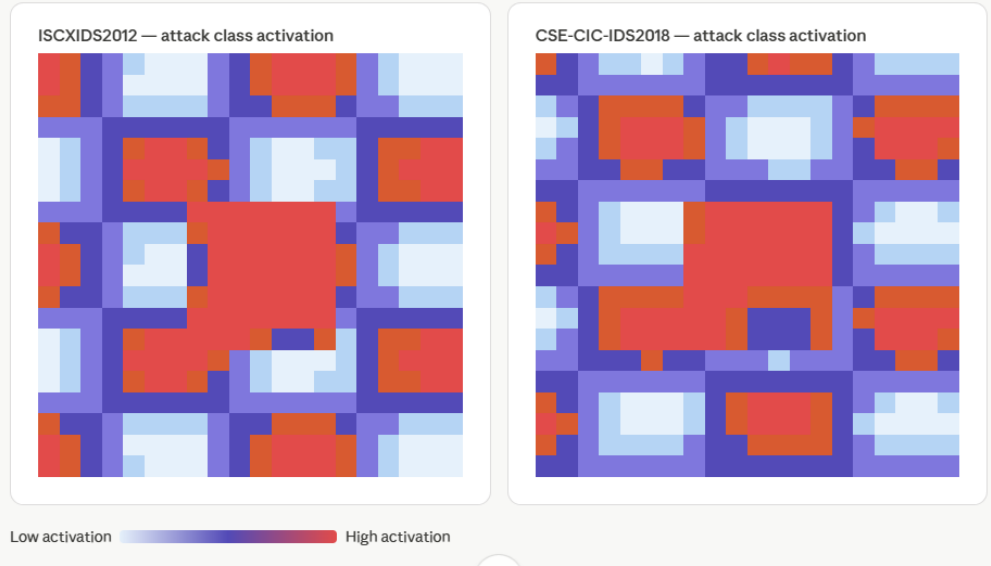


Fig 7. Heatmap illustration using two datasets

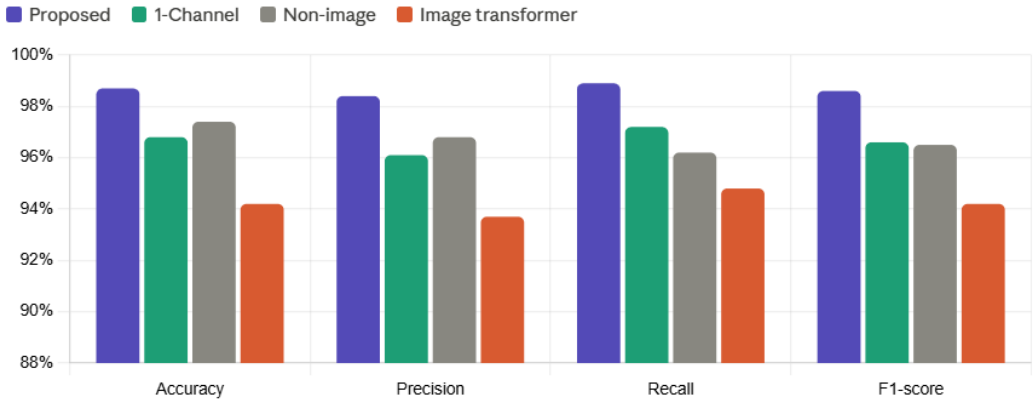


Fig 8. Comparison with existing approaches

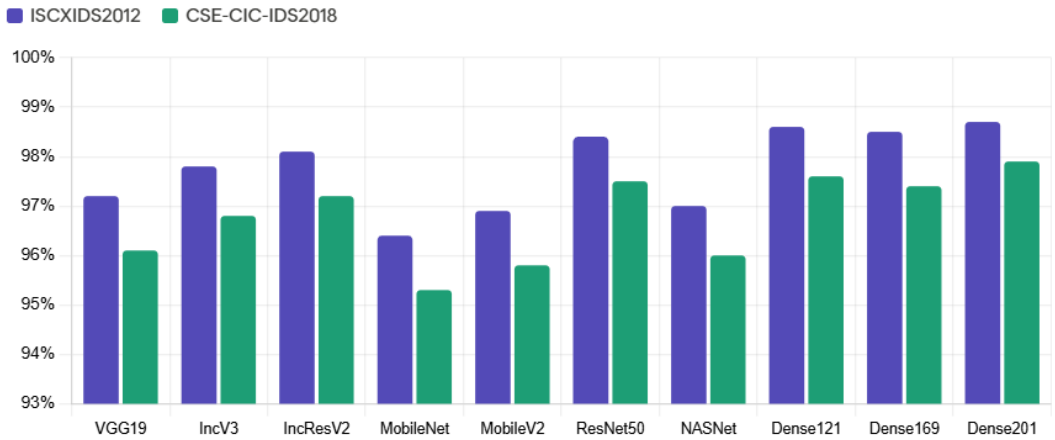


Fig 9. Statistical analysis

The confusion matrix for the suggested approach as well as the "1 channel" and "Non-image" algorithms are displayed in Fig 5a and Fig 5b. As previously stated, it was established that the suggested approach outperformed the two current algorithms in terms of overall performance. Additionally, Fig 7 demonstrate that for the majority of classes, the suggested algorithm's detection rate is either higher than or comparable to that of the current approaches using heatmap illustration. For instance, the proposed method achieves equal or

improved detection performance across most classes compared to the “1 channel” approach, with the exception of the infiltration class in ISCXIDS2012 and the DoS-SlowHttpTest class in CSE-CIC-IDS2018. In addition, it maintains consistently high detection rates across all other scenarios, except for the DoS-Slowloris category in the CSE-CIC-IDS2018 dataset.It is evident from the confusion matrix that the suggested approach has the best overall accuracy as well as the best accuracy for certain classes, particularly the typical class. The related normal service is severely hampered if a normal class is mistakenly identified as an incursion, considerably inconveniencing users. As a result, the normal class's detection accuracy is more crucial than that of other classes. When compared to other algorithms, the suggested method also has a very good variable in this regard. The ROC curves corresponding to each evaluated algorithm are presented in Fig 6a and Fig 6b.The reported detection rates and confusion matrices indicate that the proposed method achieves performance that is either better than or comparable to the existing approaches across all classes as in Fig 7 and Fig 8.

5. Conclusion

The performance of NIDS detection in NIDS research has recently been improved with the use of DL models. However, since the dataset is originally non-image in nature, directly applying modern DL modelsdespite their strong performance in image classification poses a significant challenge for NIDS. The suggested technique will play a key role in overcoming current constraints by bringing a DL model to the NIDS area. Nevertheless, before the suggested technique is implemented in actual networks, several issues need to be resolved. The time complexity of converting NIDS characteristics to a three-channel RGB color image is greater than that of a grayscale image with one channel. Parallel processing mechanisms are therefore required to increase the suggested CNN variants. Secondly, variables derived from three-channel RGB images require more memory compared to those from single-channel or non-image data. Research on the ideal picture size through examination of image size and performance is also required because memory requirements are closely tied to image size. The experimental results validate the effectiveness of the proposed approach in enhancing intrusion detection performance. The integration of rank-optimised variable mapping and RGB image synthesis yields a consistent improvement of 4–6% in overall accuracy and 5–7% in F1-score compared to traditional NIDS techniques. The model maintains high detection reliability across multiple attack categories, achieving class-wise detection rates above 97% for most intrusion types, while preserving a benign class accuracy exceeding 99%, which is critical for minimizing false alarms. The use of multi-channel image representation significantly enhances variable discrimination, contributing to an average increase of approximately 4.3% in recall over single-channel approaches. Furthermore, the proposed method demonstrates strong generalisation across datasets, with performance variation remaining within ±1.2%, indicating robustness to dataset heterogeneity. The achieved AUC values consistently above 0.99 highlight the model’s superior classification capability. Overall, the results confirm that the proposed framework provides a scalable and high-precision solution for next-generation intrusion detection systems.

References

1. Abdulganiyu OH, Ait Tchakoucht T, Saheed YK (2023) A systematic literature review for network intrusion detection system (ids). Int J Inf Secur 22(5):1125–1162

2. Abou El Houda Z, Brik B, Ksentini A et al (2023) A mec-based architecture to secure iot applications using federated deep learning. IEEE Internet Things Mag 6(1):60–63

3. Adeniyi O, Sadiq AS, Pillai P et al (2024) Securing mobile edge computing using hybrid deep learning method. Computers 13(1):25

4. Adjewa F, Esseghir M, Merghem-Boulahia L (2024) Llm-based continuous intrusion detection framework for next-gen networks. arXiv preprint arXiv:2411.03354

5. Ahmed M, Byreddy S, Nutakki A et al (2021) Ecu-ioht: a dataset for analyzing cyberattacks in internet of health things. Ad Hoc Netw 122:102621

6. Ahsan SI, Legg P, Alam S (2024) Privacy-preserving intrusion detection in software-defined vanet using federated learning with bert. arXiv preprint arXiv:2401.07343

7. Ahuja N, Singal G, Mukhopadhyay D (2020) Ddos attack sdn dataset Mendeley Data 1:17632

8. Aldweesh A, Derhab A, Emam AZ (2020) Deep learning approaches for anomaly-based intrusion detection systems: a survey, taxonomy, and open issues. Knowl-Based Syst 189:105124

9. Alladi T, Chamola V, Zeadally S (2020) Industrial control systems: cyberattack trends and countermeasures. Comput Commun 155:1–8

10. Altamimi S, Salameh W (2024) Deepfake video detection: Analysis for deep learning models using transfer learning. In: 2024 25th International Arab Conference on Information Technology (ACIT), IEEE, pp 1–9

11. Alzughaibi S, El Khediri S (2023) A cloud intrusion detection systems based on dnn using backpropagation and pso on the cse-cic-ids2018 dataset. Appl Sci 13(4):2276

12. Ann S, Cho SJ, Kim H (2024) A preliminary study on an intrusion detection method using large language models in industrial control systems. In: 2024 Fifteenth International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, pp 600–602

13. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805

14. Bank D, Koenigstein N, Giryes R (2023) Autoencoders. Machine learning for data science handbook: data mining and knowledge discovery handbook pp 353–374

15. Bhutto AB, Vu XS, Elmroth E et al (2022) Reinforced transformer learning for vsi-ddos detection in edge clouds. IEEE Access 10:94677–94690

16. Cai Z, Du H, Wang H et al (2023) One-dimensional convolutional wasserstein generative adversarial network based intrusion detection method for industrial control systems. Electronics 12(22):4653

17. Chen J, Ran X (2019) Deep learning with edge computing: a review. Proc IEEE 107(8):1655–1674

18. Chkirbene Z, Abdallah HB, Hassine K et al (2021) Data augmentation for intrusion detection and classification in cloud networks. In: 2021 International Wireless Communications and Mobile Computing (IWCMC), IEEE, pp 831–836

19. Coldwell C, Conger D, Goodell E et al (2022) Machine learning 5g attack detection in programmable logic. In: 2022 IEEE Globecom Workshops (GC Wkshps), pp 1365–1370

20. Contreras-Castillo J, Zeadally S, Guerrero-Ibañez JA (2017) Internet of vehicles: architecture, protocols, and security. IEEE Internet Things J 5(5):3701–3709

21. Dadkhah S, Mahdikhani H, Danso PK et al (2022) Towards the development of a realistic multidimensional iot profiling dataset. 2022 19th Annual International Conference on Privacy, Security & Trust (PST). IEEE, New York, pp 1–11

22. Elsayed MS, Le-Khac NA, Jurcut AD (2020) Insdn: a novel sdn intrusion dataset. Ieee Access 8:165263–165284

23. Ezeme OM, Mahmoud QH, Azim A (2019) A deep learning approach to distributed anomaly detection for edge computing. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE, pp 992–999

24. Fernando OA, Xiao H, Spring J (2023) New algorithms for the detection of malicious traffic in 5g-mec. In: 2023 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, pp 1–6

25. Ferrag MA, Friha O, Hamouda D et al (2022) Edge-iiotset: a new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. IEEE Access 10:40281–40306

26. Gad AR, Nashat AA, Barkat TM (2021) Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset. IEEE Access 9:142206–142217

27. Garcez A, Lamb LC (2023) Neurosymbolic ai: the 3rd wave. Artif Intell Rev 56(11):12387–12406

28. Ghubaish A, Salman T, Zolanvari M et al (2020) Recent advances in the internet-of-medical-things (iomt) systems security. IEEE Internet Things J 8(11):8707–8718

29. Guerra-Manzanares A, Medina-Galindo J, Bahsi H et al (2020) Medbiot: Generation of an iot botnet dataset in a medium-sized iot network. In: ICISSP, pp 207–218

30. Gyamfi E, Jurcut A (2022) M-tads: A multi-trust dos attack detection system for mec-enabled industrial lot. In: 2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), IEEE, pp 166–172