Received: 25 June 2025 | Revised: 12 November 2025 | Accepted: 21 November 2025 | Published online: 31 December 2025

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 00(00) 1–5
DOI: 10.47852/bonviewJCCE52026581

BON VIEW PUBLISHING

Transforming Data Representation: A Comparative Analysis of Tabular and Image-Based Approaches with XAI

Malik AL-Essa[1,*], Mohammad Alsharo[2], Yazan Alnsour[3], Wasim A. Ali[4] and Omar Almomani[5]

[1] *Department of Computer Science, The University of Jordan, Jordan*

[2] *Department of Information Systems, Al al-Bayt University, Jordan*

[3] *Department of Management Information Systems, Prince Mohammad Bin Fahd University, Saudi Arabia*

[4] *Department of Electrical and Information Engineering, Politecnico di Bari, Italy*

[5] *Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Jordan*

Abstract: Cyber-attacks are increasingly becoming a major concern for individuals and organizations alike. Meanwhile, attackers are employing advanced techniques taking advantage of the growing power of Artificial Intelligence (AI) to develop highly sophisticated attacks at an accelerated pace. Consequently, developing effective tools to detect cyber-attacks and protect digital assets has become of utmost importance to practice and research. Numerous AI-based techniques, mainly Machine Learning (ML) and Deep Learning (DL), have been investigated in the literature to protect digital assets from cyber threats. Particularly, DL has received significant attention in recent cybersecurity research due to its powerful capabilities in managing huge amounts of data and detecting malicious cyber threats. The majority of the proposed threat-detecting techniques in the literature focus on using Deep Neural Networks (DNNs). However, some research in the literature utilized the capabilities of Convolutional Neural Networks (CNNs) in detecting cyber threats by converting the cybersecurity data from tabular data into images. This paper aims to investigate the efficiency of both approaches in detecting cyber threats with the help of eXplainable AI (XAI). Our findings indicate that on NSL-KDD, the DL model trained on tabular features achieved WeightedF1 = 0.74, MacroF1 = 0.56, and Overall Accuracy (OA) = 0.78, compared with 0.73, 0.54, and 0.76 for the image-based model. On CICMaldoid20, the tabular model achieved 0.79 (WeightedF1), 0.80 (MacroF1), and 0.80 (OA), compared with 0.76, 0.75, and 0.77 for the image-based counterpart. These results suggest that while image transformations can be beneficial in specific classes, tabular models consistently deliver stronger overall performance.

Keywords: DeepInsight, Deep Learning, malware detection, intrusion detection, XAI

1. Introduction

Recently, one of the major applications of AI, specifically Deep Learning (DL) as a subdomain of AI, is to resolve cybersecurity threats such as intrusion and malware detection by leveraging complex neural network architectures that can automatically learn hierarchical features from large datasets, where DL achieves advanced classification capabilities [1–4]. Intrusion detection systems (IDS) comprise two fundamental categories: The first type is signature-based intrusion detection systems (SIDS) [5]. They spot malicious traffic by comparing what's happening on the network to a library of known attack "signatures." If current traffic matches one of those stored patterns, the system raises an alert.

The second type is the anomaly-based intrusion detection system (AIDS). It uses ML or DL to learn what "normal" network traffic looks like, then watches for behavior that doesn't fit. After the models are trained to tell benign from malicious patterns, any significant deviation from the learned baseline is flagged as an anomaly [6, 7]. In the literature, researchers investigated various ML and DL approaches to detect cyber threats. Most data related to cybersecurity problems are tabular, so most of the proposed approaches are based on Deep Neural Networks (DNNs). Recently, some techniques have been proposed using Convolutional Neural Networks (CNNs) to detect both intrusions and malware apps in cybersecurity problems, such as converting the data from the tabular form into images to use the powerful capabilities of CNNs [8, 9]. For instance, DeepInsight is a well-known technique for converting tabular data into images and can convert both network traffic data and malware apps into images [10]. Furthermore, eXplainable AI (XAI) techniques can be used to investigate to what extent converting cybersecurity data into images can help security practitioners understand cyber threats.

Prior CNN-based approaches span both intrusion detection and Android malware analysis. For network intrusion detection, Ding and Zhai [11] trained a CNN directly on NSL-KDD features and reported competitive performance on that benchmark. Subsequent studies proposed CNN variants operating on image-like encodings of traffic, such as SeNet-I, which serializes flows into three-channel images before CNN classification, and channel-attention CNNs tailored to IDS tasks and evaluated on NSL-KDD/CSE-CIC datasets [12, 13]. For Android malware, previous studies converted APK/DEX artifacts into grayscale or RGB images and then applied CNNs or transfer learning, including broad CNN comparisons (e.g., EfficientNet family), image-based detection pipelines, and models trained on CICMalDroid20 (sometimes using ensembles) [14, 15].

***Corresponding author:** Malik AL-Essa, Department of Computer Science, The University of Jordan, Jordan. Email: m.alessa@ju.edu.jo

Unlike prior studies, which typically evaluated a *single* representation (either tabular features or image conversions) in isolation, our work offers a controlled, side-by-side comparison of *tabular* vs. *image-based* representations using the same datasets (NSL-KDD and CICMalDroid20 datasets), matched preprocessing, and aligned metrics (WeightedF1, MacroF1, and Overall Accuracy). Moreover, we integrate *XAI* analyses for both modalities (e.g., feature attribution for tabular models and saliency/activation inspection for image-based models) to explain *why* models succeed or fail on specific classes. This joint, modality-aware evaluation clarifies when image transformations help and when tabular features remain superior overall, which is an evident gap not addressed by prior CNN-only pipelines.

The main contribution of this paper is to measure how the impact of transforming cybersecurity datasets into images affects both the detection of cyber threats and the understanding of the decisions generated by the DL models. We use XAI techniques to explain the decisions generated by the DL models trained on both datasets, i.e. tabular datasets and transformed datasets into images.

The paper is structured as follows: Section 2 reviews related works. The proposed methodology is explained in Section 3, while the evaluation results are discussed in Section 4. Finally, Section 5 draws conclusions, refocuses on the purpose of the research, and illustrates future developments.

2. Literature Review

IDS play a key role in protecting data and network infrastructure from cyber threats. An *intrusion* is any unauthorized activity that can harm an information system by undermining the confidentiality, integrity, or availability of its data [6]. SIDS perform well at detecting *known* attacks, since they match traffic against a database of stored signatures. However, they struggle with *unknown* or zero-day attacks because no matching signature exists yet.

To address this gap, researchers have explored a wide range of ML and DL approaches. For example, Newaz et al. [16] evaluate several ML models, k-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), and Decision Trees (DT), to detect attacks on personal medical devices. With the rapid progress of DL, many IDS now rely on neural architectures: a CNN-based IDS that converts traffic into images is proposed by Lin et al. [17]; ResNet variants are used for network-traffic classification in references [18–20]; an unsupervised approach based on autoencoders is presented in research by Ghorbani and Fakhrahmad [21]; and an IDS using a Restricted Boltzmann Machine (RBM) targets large-scale smart-grid cybersecurity in research by Alam et al. [22]. Beyond classifiers, Abualhaj et al. [23] combine AI, honeypots, and an Intrusion Prevention System (IPS) to detect attacks, alert defenders, and limit attacker sessions. Traditional feature-selection + ML pipelines remain effective as well [24]. Finally, DeepInsight [25] transforms network traffic into images for DL-based detection. In our study, we extend this line by performing multi-class detection, whereas Tran et al. [25] focused on binary classification.

To clarify how our work complements and extends prior studies, Table 1 summarizes representative image- and tabular-based approaches in the literature, listing the dataset(s) used, the data representation, and the model family. Unlike previous studies that typically evaluate a single representation or a single dataset in isolation, our study presents a controlled, side-by-side comparison of **tabular** versus **image-based** representations on the same benchmarks (NSL-KDD and CICMalDroid20 datasets).

Table 1
Representative prior works and proposed method

Study	Dataset(s)	Representation	Model type
[11]	NSL-KDD	Tabular	CNN
[12]	NSL-KDD / CIC	Serialized RGB image	CNN
[13]	NSL-KDD, CIC-IDS2017	Tabular / image variants	CNN, GRU, CBAM
[25]	Various (network)	DeepInsight image encoding	CNN
[14]	Android datasets	Bytecode / RGB image	CNN, transfer learning
This work	NSL–KDD, CICMalDroid20/ CIC-IDs	**Tabular vs. image** (same data)	DL (tabular & CNN), XAI analyses

3. Methodology

The proposed methodology consists of two main parts: The first one is based on learning a DL model using a tabular dataset while the second one is based on converting the tabular data into images. An XAI approach is used to compare between the two methods. We evaluate two ways of feeding the same cybersecurity data into deep models. In the **tabular pipeline**, each sample is a fixed-length vector of engineered features describing either a network connection (NSL-KDD) or an Android application (CICMalDroid20). Categorical attributes are encoded, and numeric attributes are scaled; the resulting vectors are passed directly to a deep neural network. In the **image pipeline**, we map the same features for each sample into a 2D layout using a reproducible rule (DeepInsight), scale values to an image intensity range, and produce a single, or three-channel image that is resized and normalized for a CNN. In both pipelines, the model outputs one of the dataset's target classes for each sample (e.g., Normal/attack families for NSL-KDD; Benign/Banking/Adware/SMS malware/Riskware for CICMalDroid20). DeepInsight converts the cybersecurity dataset samples into images.[1] The methodology of this work is defined in four-stepped for cyber-threat detection:

[S1] Convert each sample's features into a 2D image using DeepInsight (image inputs).

[S2] Train a deep model on the **tabular inputs**.

[S3] Train a CNN on the **image inputs**.

[S4] Apply XAI to the tabular model (e.g., SHapley Additive exPlanations (SHAP)).

[S5] Apply XAI to the image model (e.g., saliency).

[S6] Compare the explanations and classification results across the two pipelines.

After training two different DL models using the original dataset and the transformed one as in steps S2–S23, XAI techniques are used to explain the decisions generated by both DL models to understand and to quantify the effect of how transforming tabular data can help detect cyber threats. Post-hoc explanations are used in this work.

3.1. DeepInsight

DeepInsight is a technique that allows using CNNs on data that is not originally in image format [10]. It achieves this by converting the data into a well-organized image representation. Furthermore, DeepInsight allows CNNs to automatically learn important features directly from non-image data, achieving good performance in various tasks. DeepInsight utilized t-distributed stochastic neighbor

[1] https://alok-ai-lab.github.io/DeepInsight/.

embedding (t-SNE) (along with other techniques) to transform tabular data into images [26]. t-SNE is primarily used within DeepInsight for dimensionality reduction. Malware samples often contain hundreds of interrelated features, as well as traffic network data. DeepInsight excels at converting these samples into images, effectively reducing the data dimensionality while preserving the crucial relationships among features. We choose t-SNE as the dimensionality-reduction step in our DeepInsight pipeline because our objective is to produce a 2-D feature layout that emphasizes local feature relationships that a CNN can exploit. t-SNE is specifically designed to preserve local neighborhoods in the high-dimensional space, which tends to place strongly related features close together on the image canvas in a more efficient way than PCA or UMAP; this local clustering behavior is well suited for convolutional feature learning where nearby pixels jointly form meaningful patterns. In the cybersecurity context (malware and network feature sets), many important signals are non-linear and locally correlated (e.g., groups of related API calls, protocol fields, or behavioral indicators). Thus, a method that highlights local affinities helps produce spatial patterns that CNNs can use to distinguish classes. We generated the 2D feature layout using t-SNE with perplexity = 30, learning rate = 'auto', number of iterations (n_iter) = 1000, and random seed (random_state) = 42.

3.2. XAI

Various XAI methods are used in this work: Gradient explanations [27] for an input sample $\boldsymbol{s}$ is $E_{grad}(\boldsymbol{s}) = \frac{\partial M_\Delta}{\partial s}$. The gradients indicate how much each input dimension would change the predictions of the model $M_\Delta(\boldsymbol{s})$ within a nearby region around the input. Gradient $\odot$ Input refers to the element-wise product of the gradients and the input [28]. Gradient $\odot$ Input can be explained as $\boldsymbol{s}\odot = \frac{\partial M_\Delta}{\partial s}$ which can address gradient saturation and reduce visual diffusion. Grad-CAM (Gradient-Weighted Class Activation Mapping) builds on CAM [29] but works with a wide range of CNN architectures, not just those ending in global average pooling. It produces a coarse heatmap that shows which parts of an image mostly influenced the model's prediction for a chosen class. Concretely, Grad-CAM uses the gradients of the target class score with respect to the last convolutional layer to weight that layer's feature maps, then sums them to get a class-specific localization map. A ReLU is applied at the end to zero out negative values so the heatmap emphasizes only the regions that support the target class. Without this ReLU step, the map can highlight areas that argue against the class as well, making the visualization less focused and less useful.

The SHAP technique is used in this work to explain the decisions generated by DL models trained using tabular data [30]. SHAP as a local XAI technique is based on a game theory approach to explain the decisions generated by a DL model. SHAP computes feature importance in DL models by averaging how much each feature value influences the output across all potential predictions. SHAP attributes each prediction to input features by averaging their marginal contributions across feature subsets. In practice, we applied SHAP to the trained **tabular model** as follows: (i) fit the model on the training split; (ii) select a small, stratified background sample from the training data to estimate conditional expectations; (iii) compute per-sample SHAP values on the test split; (iv) summarize global importance with the mean absolute SHAP value per feature; and (v) produce class-conditioned summaries by aggregating SHAP values over test samples of the same class. We report feature importances in the same feature space used by the classifier and use these summaries to interpret per-class performance patterns reported in Section 4.

3.3. SHAP (KernelExplainer) implementation details

For tabular models we used the SHAP library (shap v0.41.0) and specifically the KernelExplainer implementation to compute approximate Shapley values. KernelExplainer is a model-agnostic that approximates each feature's Shapley contribution by fitting a weighted linear model to many randomized perturbations of the input. We selected this because (a) it supports any classifier or preprocessing pipeline (including those with non-differentiable steps), (b) it provides a principled Shapley-based attribution that is comparable across model families, and (c) it produces reliable local explanations that can be aggregated to yield global feature importance.

Since KernelExplainer is computationally intensive for high-dimensional inputs, we applied the following reproducible practical choices and mitigations:

1) **Environment and versions:** Python 3.9, shap==0.41.0, numpy/pandas/scikit-learn.
2) **Preprocessing:** Inputs passed to the explainer were already preprocessed exactly as during model training (categorical encoding, scaling). We persist in and reuse the exact encoder/scaler objects to guarantee identical transforms.
3) **Background (reference) dataset:** We built a fixed background set of at most 100 samples selected by stratified sampling from the training set to preserve class proportions. The same background set is reused for all KernelExplainer calls to ensure reproducibility.
4) **KernelExplainer parameters:** We used 'nsamples=1000' (number of Monte Carlo perturbation samples per explanation call) and a fixed random seed ('random_state=42'). For multi-class outputs, we computed SHAP values per class by wrapping the model to return the predicted probability for a single target class and calling KernelExplainer separately for each class.
5) **Outputs and aggregation:** For each sample, we saved per-feature SHAP values for all classes. Global importance rankings were produced by averaging absolute SHAP values across the selected test subset (or full test set when computed).

4. Empirical Evaluation and Results

4.1. Datasets

We use two different representations of the raw datasets depending on the modeling pipeline: (1) a *tabular* representation that preserves the original features reported by each dataset, and (2) an *image-based* representation in which the same information is converted into a 2D image that can be processed by a convolutional neural network as reported in Table 2. **Each dataset contains:**

1) **NSL–KDD:** Each record corresponds to a network connection and is described by 41 features (3 categorical, 37 numeric, and 1 label). Labels fall into five high-level classes: *Normal*, *DoS*, *U2R*, *R2L*, and *Probe*. We follow the standard split: KDDTrain+20Percent for training and KDDTest+ for testing. We standardize the **37** numeric fields and one-hot encode the three categorical fields (**protocol_type**, **service**, **flag**), yielding **118** numeric features (label handled separately).
2) **CICMalDroid20:** Each sample corresponds to one Android application and is represented by 471 features that capture static and dynamic characteristics (as reported in the dataset documentation). Samples are labeled as one of five classes: *Benign*, *Banking*, *Adware*, *SMS malware*, or *Riskware*. We use a stratified 70%/30% train/test split.

Table 2
Description of the datasets

Dataset	# Training samples	# Testing samples	# Labels	# Features
NSL-KDD	25192	22544	5	41
CICMalDroid20	8118	3480	5	471

Inputs (what the model receives):

1) **Tabular pipeline.** Each sample is one-row feature vector. Categorical attributes are encoded (e.g., one-hot or integer encoding) and numeric attributes are normalized (min–max scaling). After these steps, each record becomes a fixed-length numeric vector (length 118 for NSL–KDD; length 471 for CICMalDroid20) that is fed directly into the tabular DL model.
2) **Image pipeline.** To build image inputs, we convert the same per-sample feature set into a 2D array and then into an image suitable for CNNs. Concretely, features are (optionally) ordered by a reproducible rule, scaled to the image intensity range, mapped into a 2D grid, and saved as a single- or three-channel image. Images are then resized to the CNN input size and normalized using the same scheme applied during CNN training.

4.2. Implementation details

The search space of the hyper-parameter optimized the hyperparameters of the DNN, as outlined in Table 3, powered by Hyperopt's tree-structured Parzen estimator (TPE) for hyperparameter tuning. This process utilized 20% of the training set as a validation set. Validation loss is utilized to determine the optimal configuration of the chosen parameters. Table 4 presents the layers chosen for each neural network architecture, while Table 5 reports the detailed architecture for the used CNN. All hidden layers in both architectures use ReLU to introduce nonlinearity, speed up training, and mitigate vanishing gradients, while keeping computations simple and sparse. The final layer applies softmax, which normalizes the model's logits into a probability distribution over classes, enabling straightforward interpretation, cross-entropy training, and thresholding for decisions. This pairing—ReLU in hidden layers and softmax at the output—provides stable optimization, fast convergence, and clear, probabilistic outputs suitable for multi-class classification. In addition, we employed the Adam optimization algorithm for gradient-based updates and initialized the weights using the Xavier method. The training was limited to a maximum of 150 epochs, with an early stopping (patience = 10 epochs) mechanism to select the best-performing models by minimizing validation loss for both DNN architectures. Table 6 summarizes the hyperparameters optimized through Hyperopt, which were subsequently applied to train the DL models.

Table 3
Hyper-parameter search space

Hyper-parameter	Values
Learning rate	[0.0001, 0.001]
Mini-batch size	{32, 64, 128, 256, 512}
Drop-out	[0,1]
Number of neurons per hidden layer	{32, 64, 128, 256, 512,1024 }

Table 4
Neural network architectures used for tabular data

Architecture	Layer
DNN	- Fully connected layers (three layers) - Drop-out layer (one layer) - Batch normalization layer (one layer)
CNN	- 2D CNNs layers - One drop-out layer - Two MAXPooling2D layers - One flatten-layer

Table 5
CNN architecture for image-based experiments

Layer type	Parameters
Input	28 × 28 × 3 image (transformed from tabular data)
Conv2D-1	Filters = *neurons1* (32, 64, 128, or 256 via Hyperopt), kernel = 2 × 2, stride = 1, activation = ReLU, kernel initializer = Glorot uniform
Dropout-1	Rate = *dropout* (optimized via Hyperopt)
MaxPooling-1	Pool size = 2 × 2, stride = 2
Conv2D-2	Filters = *neurons2* (32, 64, 128, or 256), kernel = 2 × 2, stride = 1, activation = ReLU, kernel initializer = Glorot uniform
MaxPooling-2	Pool size = 2 × 2, stride = 2
Flatten	–
Dense-1	Units = *neurons3* (32, 64, 128, or 256), activation = ReLU
Dense-Output	Units = number of classes (5 for NSL-KDD, 5 for CICMalDroid20), activation = Softmax, kernel initializer = Glorot uniform

Table 6
Optimized hyperparameters

Dataset	Learning rate	Batch size	Drop out layer	Neurons layer1	Neurons layer2
NSL-KDD	0.0006	128	0.1026	1024	128
NSL-KDD*	0.0005	256	0.1120	256	64
CICMalDroid20	0.0007	512	0.3041	1024	1024
CICMalDroid20*	0.0007	128	0.03	256	64

Note: The parameters selected via Hyperopt for each dataset. The dataset with * is the image-based one.

4.3. Empirical evaluation and discussion

We evaluated the proposed method in this work using cybersecurity datasets (i.e. CICMalDroid20 and NSL-KDD). The experiments in this work aim to investigate the effectiveness of converting malware apps and network traffic into images in order to use CNNs as a classifier to detect malicious apps and traffic.

4.3.1. Performance metrics

We evaluated the classifications on the testing sets using standard multi-class classification metrics. Specifically, WeightedF1, MacroF1, and Overall Accuracy (OA) in this work for evaluating the conducted experiments. To focus on assessing the number of true positives, OA is used in this work, while we used F1-based scores to evaluate false negatives, false positives, and the true positives. Moreover, since F1 can be measured per class, we consider WeightedF1 and MacroF1 to aggregate F1 values measured on all the classes in a single value by computing the weighted mean and the simple mean, respectively. In balanced domains, WeightedF1 and MacroF1 are expected to yield similar values. However, in imbalanced domains WeightedF1 can provide a misleading evaluation of the performance of rare classes due to the dominance of majority classes in the metric. Since our study involves both balanced datasets (CICMalDroid20 dataset) and imbalanced datasets (NSL-KDD dataset), we assessed the accuracy of the proposed evaluation across all these metrics during experimentation.

4.3.2. Results and discussions

The empirical evaluation assessed whether transforming malware apps and network traffic traces can help cybersecurity practitioners

analyze cyber threats. This analysis is based on two main objectives: The first one is the classification accuracy of the model. The second one is an XAI approach to explain the decisions generated by the model and to test whether transforming tabular data into images can help understand the decisions generated by the DL model. Results are from a **single training run per model:** the final configuration was chosen via **Hyperopt** on the validation split and then evaluated once on the test set. **Because we did not repeat training across random seeds, our results do not capture training stochasticity;** small differences should be interpreted cautiously.

Table 7
Global performance metrics for tabular and image-based models

	NSL-KDD			CICMalDroid20		
Model	**WeightedF1**	**MacroF1**	**OA**	**WeightedF1**	**MacroF1**	**OA**
MΔ	**0.74**	**0.56**	**0.78**	**0.79**	**0.80**	**0.80**
$M_{\Delta'}$	0.73	0.54	0.76	0.78	0.75	0.77

Table 8
Per-class F1 scores for NSL-KDD and CICMalDroid20.

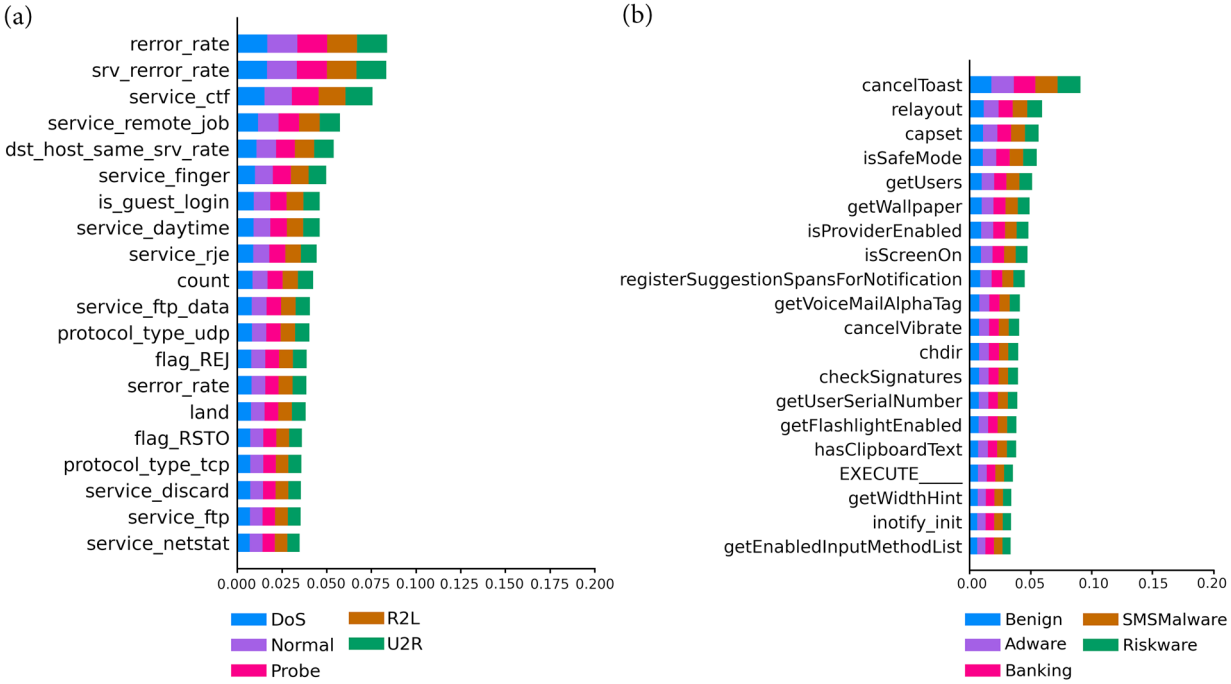
			F1 score	
Dataset	**Label**	**Support**	**Tabular data**	**Transformed images**
NSL-KDD	Benign	9711	**0.80**	0.79
	DoS	7458	**0.89**	0.88
	Probe	2421	**0.77**	0.71
	R2L	2754	**0.18**	0.17
	U2R	200	**0.17**	0.14
CICMalDroid20	Benign	539	**0.89**	0.80
	Adware	376	**0.78**	0.66
	Banking	630	**0.78**	0.65
	SMSMalware	1171	0.83	**0.88**
	Riskware	764	0.72	**0.76**

Table 7 reports the global performance metrics of both models. For NSL-KDD, the tabular-based model (M_{Δ}) achieved a WeightedF1 of 0.74, Macro-F1 of 0.56, and OA of 0.78, compared to 0.73, 0.54, and 0.76 for the image-based model ($M_{\Delta'}$). For CICMalDroid20, M_{Δ} obtained 0.79 (WeightedF1), 0.80 (MacroF1), and 0.80 (OA), versus 0.78, 0.75, and 0.77 for $M_{\Delta'}$. These results confirm that tabular data yields consistently higher global performance across both datasets.

However, a closer look at the per-class F1 scores in Table 8 reveals that image transformation can still provide benefits in specific scenarios. For example, in CICMalDroid20, the image-based model outperformed the tabular-based one for SMSMalware (0.88 vs. 0.83) and Riskware (0.76 vs. 0.72). In contrast, for the majority of classes, including Benign, Adware, and Banking, the tabular model was clearly superior. Similarly, in NSL-KDD, the tabular model achieved better F1 scores for all attack classes, including the challenging minority classes U2R (0.17 vs. 0.14) and R2L (0.18 vs. 0.17). In summary, while tabular representations consistently deliver higher overall performance in terms of WeightedF1, MacroF1, and OA, the image-based approach demonstrates potential advantages for certain balanced malware classes such as SMSMalware and Riskware. This suggests that hybrid approaches combining both representations could be a promising direction for future work.

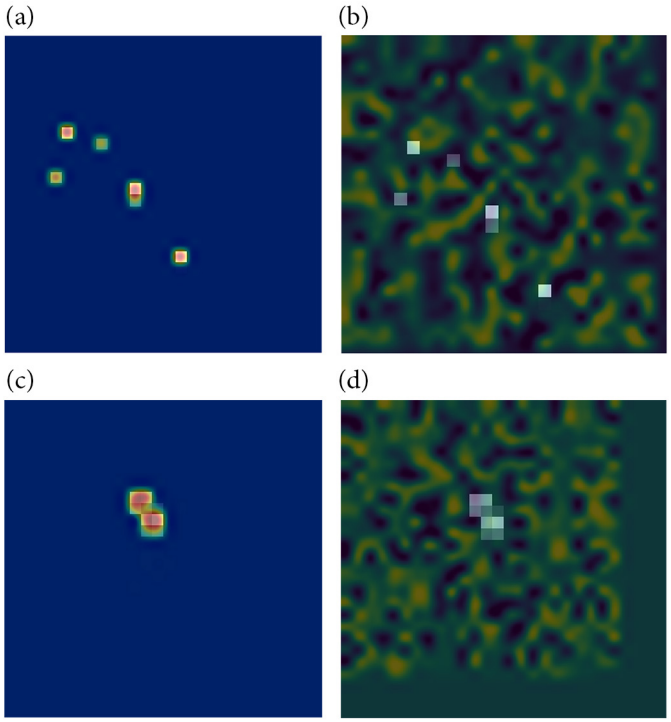
An XAI approach is used to explain the decisions of both models. The SHAP technique is used to explain the decisions generated by M_{Δ}(i.e. the model trained on tabular data), whereas Gradient ⊙ Input and Grad-CAM are used to explain the decisions generated by $M_{\Delta'}$. Figure 1 demonstrates the global feature ranking computed with SHAP. As illustrated in this figure, security practitioners can understand what features affect the decisions generated by the DL model. Figure 1a

Figure 1
Global SHAP feature importance for the models trained on tabular data. (a) NSL-KDD: top features contributing to each attack class. (b) CICMalDroid20: top features across malware categories



Note: Bar length shows the mean absolute SHAP value, where longer bars indicate stronger feature contribution to classification.

Figure 2
CNN explanation results for image-based models. (a, c) Grad-CAM heatmaps over DeepInsight feature images (warmer = stronger positive influence on the predicted class). (b, d) Gradient × input saliency maps (brighter = larger per-pixel contribution)



Note: How to read: coherent, contiguous hotspots centered on feature clusters known to be class-indicative (e.g., SMS/telephony for SMSMalware; protocol/flag clusters for NSL-KDD attacks) indicate a meaningful explanation; scattered or peripheral activations over low-variance regions suggest a weak/vague signal. The top row shows NSL-KDD samples; the bottom row shows CICMalDroid20 samples.

shows the features ranking for the NSL-KDD dataset, which indicates that 'srv_rerror_rate' contributes more to the detection of 'R2L' attack while 'service_ctf' contributes more to detecting the 'Probe' attack. Moreover, Figure 1b shows the global feature ranking for the CICMalDroid20 dataset. Feature 'cancelToast' has the main effect on the model prediction whereas the other features have a lower effect. Meanwhile, Figure 2 demonstrates the explanations generated for two random samples selected from the training dataset for both NSL-KDD and CICMalDroid20 datasets. Figure 2a and b illustrates the explanations generated by the Grad-CAM technique, while Figure 2c and d illustrates the explanations generated by Gradient⊙Input technique. The explanations generated for transformed images can illustrate where the model focuses for prediction; however, the explanation is still vague compared to the explanations generated for tabular data. The explanations generated for the models trained on tabular data can help security practitioners more in understanding the attacks and how the model can classify them, whereas, in transformed images, the explanations are not clear and give no additional information regarding the decisions generated by the model. Furthermore, transforming tabular data into images increases the complexity required to train a model.

We did not run a multi-rater interpretability study in this version; a quantitative assessment with reviewer counts and inter-rater agreement is deferred to future work. For a true R2L sample, **Grad-CAM** produces multiple small hotspots spread across the image, including regions corresponding to mixed protocol/service features, rather than a coherent focus over the login/authentication cluster. In contrast, **tabular SHAP** concentrates importance on features aligned with R2L behavior (e.g., login failure counts and access attempts). The image saliency is thus less **localized and less class-specific**, which we describe as "vague" in this context.

5. Conclusion

This study examined the effectiveness of transforming tabular cybersecurity data into images to enhance cyber-threat detection, with XAI applied to interpret DL model decisions. Overall, our results suggest that tabular data representations generally yield better performance and more interpretable results for practitioners. However, the evaluation also revealed that image-based transformations can provide benefits in specific scenarios, such as improving detection accuracy for certain malware categories in the CICMalDroid20 dataset. Furthermore, while tabular-based models allowed security practitioners to better understand classification outcomes, XAI applied to image-transformed models often produced vague explanations, limiting their practical value. Moreover, the image pipeline was substantially heavier: training took ≈2–4× longer and used ≈2–3× more peak GPU memory than the tabular DNN with matched data splits and early stopping. The tabular model also achieved lower inference latency and can run comfortably on CPU, reinforcing its practicality for constrained environments. Future work will expand this evaluation by incorporating additional cybersecurity datasets and experimenting with diverse CNN architectures to further explore the conditions under which image transformation may enhance cyber-threat detection.

Acknowledgement

The authors thank the reviewers for their insightful remarks and constructive comments.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Malik AL-Essa: Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Data curation, Writing – original draft, Writing – review & editing. **Mohammad Alsharo:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – review & editing. **Yazan Alnsour:** Conceptualization, Methodology, Software, Validation, Resources, Data curation, Writing – review & editing. **Wasim A. Ali:** Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Data curation, Writing – review & editing. **Omar Almomani:** Writing – review & editing, Project administration.

References

[1] Prümmer, J., van Steen, T., & van den Berg, B. (2024). A systematic review of current cybersecurity training methods. *Computers & Security*, *136*, 103585. https://doi.org/10.1016/j.cose.2023.103585

[2] Salem, A. H., Azzam, S. M., Emam, O. E., & Abohany, A. A. (2024). Advancing cybersecurity: A comprehensive review of AI-driven detection techniques. *Journal of Big Data*, *11*(1), 105. https://doi.org/10.1186/s40537-024-00957-y

[3] AL-Essa, M., & Appice, A. (2021). Dealing with imbalanced data in multi-class network intrusion detection systems using XGBoost. In *Communications in Computer and Information Science*, 5–21. https://doi.org/10.1007/978-3-030-93733-1_1

[4] Jacob, S. L., & Sultana Habibullah, P. (2024). A systematic analysis and review on intrusion detection systems using machine learning and deep learning algorithms. *Journal of Computational and Cognitive Engineering*, *4*(2), 108–120. https://doi.org/10.47852/bonviewJCCE42023249

[5] Einy, S., Oz, C., & Navaei, Y. D. (2021). The anomaly- and signature-based IDS for network security using hybrid inference systems. *Mathematical Problems in Engineering*, *2021*(1), 6639714. https://doi.org/10.1155/2021/6639714

[6] Hnamte, V., & Hussain, J. (2021). An extensive survey on intrusion detection systems: Datasets and challenges for modern scenario. In *2021 3rd International Conference on Electrical, Control and Instrumentation Engineering*, 1–10. https://doi.org/10.1109/ICECIE52348.2021.9664737

[7] Almomani, O., Alsaaidah, A., Abu-Shareha, A. A., Alzaqebah, A., Amin Almaiah, M., & Shambour, Q. (2025). Enhance URL defacement attack detection using particle swarm optimization and machine learning. *Journal of Computational and Cognitive Engineering*, *4*(3), 296–308. https://doi.org/10.47852/bonviewJCCE52024668

[8] Mohammadpour, L., Ling, T. C., Liew, C. S., & Aryanfar, A. (2022). A survey of CNN-based network intrusion detection. *Applied Sciences*, *12*(16), 8162. https://doi.org/10.3390/app12168162

[9] Ali, R., Ali, A., Iqbal, F., Hussain, M., & Ullah, F. (2022). Deep learning methods for malware and intrusion detection: A systematic literature review. *Security and Communication Networks*, *2022*(1), 1–31. https://doi.org/10.1155/2022/2959222

[10] Sharma, A., Vans, E., Shigemizu, D., Boroevich, K. A., & Tsunoda, T. (2019). DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, *9*(1), 11399. https://doi.org/10.1038/s41598-019-47765-6

[11] Ding, Y., & Zhai, Y. (2018). Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, 81–85. https://doi.org/10.1145/3297156.3297230

[12] Farrukh, Y. A., Wali, S., Khan, I., & Bastian, N. D. (2023). SeNet-I: An approach for detecting network intrusions through serialized network traffic images. *Engineering Applications of Artificial Intelligence*, *126*, 107169. https://doi.org/10.1016/j.engappai.2023.107169

[13] Cao, B., Li, C., Song, Y., Qin, Y., & Chen, C. (2022). Network intrusion detection model based on CNN and GRU. *Applied Sciences*, *12*(9), 4184. https://doi.org/10.3390/app12094184

[14] Huang, T. H.-D., & Kao, H.-Y. (2018). R2-D2: ColoR-inspired convolutional neural network (CNN)-based AndroiD malware detections. In *2018 IEEE International Conference on Big Data*, 2633–2642. https://doi.org/10.1109/BigData.2018.8622324

[15] Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., & Pham, T. D. (2022). EfficientNet convolutional neural networks-based Android malware detection. *Computers & Security*, *115*, 102622. https://doi.org/10.1016/j.cose.2022.102622

[16] Newaz, A. I., Sikder, A. K., Babun, L., & Uluagac, A. S. (2020). HEKA: A novel intrusion detection system for attacks to personal medical devices. In *2020 IEEE Conference on Communications and Network Security*, 1–9. https://doi.org/10.1109/CNS48642.2020.9162311

[17] Lin, S. Z., Shi, Y., & Xue, Z. (2018). Character-level intrusion detection based on convolutional neural networks. In *2018 International Joint Conference on Neural Networks*, 1–8. https://doi.org/10.1109/IJCNN.2018.8488987

[18] Shafiq, M., & Gu, Z. (2022). Deep residual learning for image recognition: A survey. *Applied Sciences*, *12*(18), 8972. https://doi.org/10.3390/app12188972

[19] Lim, H.-K., Kim, J.-B., Heo, J.-S., Kim, K., Hong, Y.-G., & Han, Y.-H. (2019). Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication*, 046–051. https://doi.org/10.1109/ICAIIC.2019.8669045

[20] Xue, J., Chen, Y., Li, O., & Li, F. (2020). Classification and identification of unknown network protocols based on CNN and T-SNE. *Journal of Physics: Conference Series*, *1617*(1), 012071. https://doi.org/10.1088/1742-6596/1617/1/012071

[21] Ghorbani, A., & Fakhrahmad, S. M. (2022). A deep learning approach to network intrusion detection using a proposed supervised sparse auto-encoder and SVM. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, *46*(3), 829–846. https://doi.org/10.1007/s40998-022-00498-1

[22] Alam, K., Imran, M. Al, Mahmud, U., & Fathah, A. Al. (2024). Cyber attacks detection and mitigation using machine learning in smart grid systems. *Journal of Science and Engineering Research*, *1*(01), 38–55. https://doi.org/10.70008/jeser.v1i01.43

[23] Abualhaj, M. M., Al-Khatib, S. N., Al-Allawee, A., Munther, A., & Anbar, M. (2024). Enhancing network intrusion detection systems through dimensionality reduction. In *International Conference on Soft Computing and Data Mining*, 244–253. https://doi.org/10.1007/978-3-031-66965-1_24

[24] Abualhija, M., Al-Shaf'i, N., Turab, N. M., & Hussein, A. (2023). Encountering social engineering activities with a novel honeypot mechanism. *International Journal of Electrical and Computer Engineering*, *13*(6), 7056. https://doi.org/10.11591/ijece.v13i6.pp7056-7064

[25] Tran, T. P., Nguyen, V. C., Vu, L., & Nguyen, Q. U. (2021). DeepInsight-convolutional neural network for intrusion detection systems. In *2021 8th NAFOSTED Conference on Information and Computer Science*, 120–125. https://doi.org/10.1109/NICS54270.2021.9701572

[26] van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(2008), 2579–2605.

[27] Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations Workshop Track Proceedings*, 1–8.

[28] Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2016). Not just a black box: Learning important features through

propagating activation differences. In *Proceedings of the 33rd International Conference on Machine Learning*, 1–6.

[29] Wang, S., & Zhang, Y. (2023). Grad-CAM: Understanding AI models. *Computers, Materials & Continua*, *76*(2), 1321–1324. https://doi.org/10.32604/cmc.2023.041419

[30] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777.

How to Cite: AL-Essa, M., Alsharo, M., Alnsour, Y., Ali, W. A., & Almomani, O. (2025). Transforming Data Representation: A Comparative Analysis of Tabular and Image-Based Approaches with XAI. *Journal of Computational and Cognitive Engineering*. https://doi.org/10.47852/bonviewJCCE52026581