PAC-X: Fuzzy Explainable AI for Multi-Class Malware Detection

Mohd Saqib, *Student Member, IEEE,* Benjamin C. M. Fung, *Senior Member, IEEE,* Philippe Charland

***Abstract*—Researchers often approach malware detection as a binary classification problem. However, evidence indicates that malware can belong to multiple families simultaneously, and malicious files frequently exhibit numerous benign features. Attackers exploit this by embedding malicious intent within benign features, making malware detection a problem better suited for fuzzy systems. Furthermore, providing explainability for such fuzzy classification remains a significant challenge, requiring specialized Explainable AI (XAI) frameworks. Existing XAI approaches offer insights into model decisions but are vulnerable to adversarial attacks that manipulate features to mislead models. To address these issues, we propose PAC-X, a novel XAI framework for malware detection. PAC-X integrates the Conditional Attention Neural Network (CAN-Net) to deliver comprehensive multi-fuzzy-class explainability and employs Contextual Fuzzy Clustering (CFC) to extract contextual insights from training data. This framework is resilient to adversarial manipulations, maintaining reliable and interpretable explanations even under adversarial conditions designed to mislead the model. Through extensive evaluations on diverse malware datasets, PAC-X demonstrates superior explainability and robustness compared to state-of-the-art XAI methods. It provides a critical advancement in cybersecurity by addressing the complexities of evasive malware detection and enabling a deeper interpretation of multi-class malware characteristics.**

***Index Terms*—Explainable AI, Malware analysis, Fuzzy system, Fuzzy clustering, Adversarial attack.**

I. Introduction

Malware often demonstrates characteristics of multiple families, embedding behaviors across categories such as ransomware, spyware, or benign software [1]. Although conventional Machine Learning (ML) and Deep Learning (DL) models can manage multi-class classification, they assume clear decision boundaries, which do not align with the inherent uncertainty in malware analysis. Attackers frequently employ obfuscation and polymorphism to blur these boundaries, introducing ambiguity into the feature space [2]. This ambiguity challenges crisp classification, as a sample may partially align with several classes. Fuzzy classification, by contrast, allows the model to assign degrees of membership across multiple classes, effectively capturing such uncertainties. This approach reflects the probabilistic nature of malware traits and provides a more robust framework for analyzing adversarial manipulations.

Various Explainable AI (XAI) methodologies, including SHapley Additive exPlanations (SHAP) [3] and Local Interpretable Model-Agnostic Explanations (LIME) [4], offer feature-based interpretations but primarily focus on local, instance-specific attributions. These methods often lack the depth necessary for critical analysis as they fail to provide global insights across the data distribution and are limited to pointwise perturbations [5]. Moreover, their reliance on synthetic perturbations can lead to explanations that are detached from the true data manifold, making them vulnerable to adversarial manipulation [5]. In security-critical applications such as malware detection, where class overlaps and adversarial behaviors are prevalent, this shallow focus on local approximations without robust global context limits their effectiveness. Guo et al. [6] introduce the Local Explanation Method using Nonlinear Approximation (LEMNA), which provides explainability for cybersecurity applications but does not offer comprehensive multiclass explainability. However, the proposed PAC-X algorithm introduces a structured approach to explanations, by employing three key concepts: Prospect, Aspect, and Context, akin to how humans naturally discuss and analyze delineating phenomena or entities in the empirical world. Below are formal definitions of these terms used in this proposed model, in the context of an explainable classification model for malware detection.

Prospect in PAC-X denotes the potential classifications the model can make, such as benign, ransomware, spyware, etc. For example, if the model classifies a file as 'ransomware', it indicates a high likelihood that the file poses a malicious threat of that nature. The term 'prospect' thus relates to the potential implications and nature of the file based on its classification.

Aspect refers to the distinct features the model utilizes for classification, such as strings, API calls, and import/export data. For instance, specific API call patterns might be identified as indicative of spyware, and PAC-X would highlight these patterns as crucial factors in the classification process.

Context relates to the data clustering derived from the training process, where clusters are formed based on shared characteristics among data points. A cluster of files using

Mohd Saqib developed the model, designed and completed the initial writing of this article. He is with the School of Information Studies, McGill University, Montreal, Quebec, Canada. mohd.saqib@mail.mcgill.ca (0000-0003-2125-2162).

Benjamin C. M. Fung provided guidance throughout the project from model design to manuscript preparation. He is with the School of Information Studies, McGill University, Montreal, Quebec, Canada. ben.fung@mcgill.ca (0000-0001-8423-2906).

Philippe Charland provided guidance throughout the manuscript preparation process and feedback at every stage of the project. He is in the Mission Critical Cyber Security Section, Defence R&D Canada, Quebec, Canada, philippe.charland@drdc-rddc.gc.ca (0000-0003-4051-9942).

Manuscript received MM DD, YYYY; revised MM DD, YYYY.

similar import/export commands and predominantly labeled as 'benign' would provide a context for evaluating new files. Should a new file align with this cluster but be identified as potential malware, PAC-X would use this discrepancy to contextualize its classification decision.

The major contributions of this research include:

- We introduce PAC-X, an explainable AI framework for fuzzy multi-label malware detection. It captures class overlaps and uncertainties, offering comprehensive explanations across all relevant classifications, even under adversarial conditions.
- We develop CAN-Net, a neural network with class-specific attention mechanisms and a customized loss function that highlights the relative importance of features for each malware class.
- We propose CFC, a clustering method that incorporates contextual information from training data to model class overlaps and ambiguities, enhancing interpretability.
- PAC-X achieves superior detection accuracy and explainability compared to LIME, LEMNA, and SHAP. It excels in consistency (uniformity within classes) and robustness (distinction between classes) across multiple datasets.

These contributions highlight PAC-X's ability to address the complexities of fuzzy classification tasks, setting new standards for explainability and transparency in AI-driven malware analysis.

The rest of the manuscript is organized as follows: In-depth background information is provided in the next section, which discusses the challenges in malware analysis, the role of explainability, and the limitations of current frameworks. Section 3 defines the problem and sets the stage for developing our model, as elaborated in Section 4. Section 5 presents the results and discussion, leading to the concluding remarks in Section 6.

II. Background

A. Challenges in malware detection

For Windows-based malware, Saxe et al. [7] and Baldangombo et al. [8] demonstrated that ANNs achieve superior accuracy in processing PE files but remain black-box models, offering limited interpretability of their decision-making processes. Similarly, for Android-based malware detection, Samaneh et al. [9] proposed a semi-supervised learning technique called the pseudo-label stacked auto-encoder (PLSAE). In another study, Kiraz et al. [10] showed that CNN-based techniques produce significant and successful outcomes when applied to Android malware detection using image representations. This is because ANNs capture complex, non-linear relationships through distributed representations across layers, making it difficult to trace specific input features to final decisions, especially in critical applications like malware detection [11].

While Decision Trees (DT) and Random Forests (RF) offer some explainability, their detection accuracy is often low on small or low-dimensional malware datasets, as seen in [12] (32.2%). In high-dimensional settings, DTs may achieve higher accuracy, for example [13] (97.2%), but the resulting tree structures become too complex for practical interpretation by malware analysts. Moreover, DT-based methods typically explain only the winning class without providing insights into other potential classes or incorporating contextual information from the feature-value distribution in the training data, unlike the structured explanations offered by PAC-X.

In addition, Raff et al.[14] adopted a unique approach by utilizing PE byte sequences as features, while Mourtaji et al.[15] transformed malware binaries into grayscale images to apply Convolutional Neural Networks (CNNs). However, despite these innovative methods, Mourtaji et al. [15] did not provide meaningful explanations or interpretations, as simply highlighting pixels in an image does not convey actionable insights about malicious behavior. In contrast, PAC-X reveals the reasoning behind the traits of each class, offering more interpretable and actionable explanations.

Malware's inherent goal is to evade detection, employing tactics such as obfuscation, encryption, and polymorphism [2]. ML/DL-based models trained on features extracted through static and dynamic analysis often face challenges because these features may not fully capture true malicious behaviors. As a result, traditional explanatory frameworks struggle to provide reliable interpretations. For example, in the study [16], the authors proposed an association rule-based malware classification method using common subsequences of API calls to capture dynamic behavioral patterns. While effective in identifying API transitions, this approach does not offer class-specific or context-aware explanations, may be affected by adversarial evasion tactics, and lacks robustness when comparing across other classes or to benign samples. In contrast, PAC-X provides detailed, multi-class, and fuzzy explanations, revealing both primary and secondary class traits with contextual insights that remain robust even under adversarial conditions. PAC-X addresses this gap by leveraging CFC and class-specific attention, enabling robust explanations that remain meaningful even when malware behavior is partially hidden or adversarially altered.

B. Review of Current XAI Frameworks

Attention-based methods are widely used to impart explainability across different data types. For malware detection, Bose et al. [17] analyzed weights and gradients in MalConv to highlight important raw bytes, emphasizing the role of header bytes. Similar strategies were adopted by [18] and [19] by combining CNNs and RNNs. Attention mechanisms have also been integrated into MLPs for feature identification [20] [21]. In addition to attention, gradient-based techniques [12] [22] [23] evaluate feature relevance by tracking gradients across network layers, identifying key system calls and events influencing decisions. Similarly, for Android-based malware, the study by Jo et al. [24] proposed a Vision Transformer (ViT)-based malware detection model and a malicious behavior extraction method using attention maps. However, attention-based methods usually provide global explanations and are less effective when attackers make local feature changes, as they may overlook fine-grained manipulations. Similarly, gradient-based explanations are sensitive to input

perturbations and may fail to provide stable insights under adversarial conditions. PAC-X addresses these limitations by offering class-specific, local, and robust fuzzy explanations that remain reliable even when features are adversarially altered.

Feature-based explanation methods such as LIME [4] and SHAP [3] have been widely used to provide local and global explanations. LIME has been applied in cybersecurity for model validation [25], [26], but it struggles with complex, non-linear boundaries. Guo et al. [6] addressed this limitation through LEMNA, which models feature dependencies using mixture regression but is less suited for multi-class explainability. SHAP has also been used to interpret malware models [27], [3], [28], [29], yet perturbation-based methods such as SHAP and LIME can generate unrealistic samples, especially when applied to PE-based malware. Some advanced algorithms have also been proposed to explain malicious PE files, such as GAGE [30] and CFGExplainer [31], but they demonstrate very low accuracy. Unlike these methods, PAC-X offers class-specific, multi-class fuzzy explanations that remain reliable under adversarial changes and are specifically designed for malware detection in PE files, addressing the limitations of prior approaches, without compromising detection power.

Another problem we identified is that not all methodologies guarantee efficacy, especially when perturbation techniques and synthetic data generation do not align with the specific characteristics of PE-based malware, which is the standard file format for executables, object code, and DLLs in Windows operating systems. For example, modifying features such as API call frequencies or byte patterns without considering the structural dependencies of PE files can lead to the generation of invalid or unrealistic samples that do not reflect actual malware behavior. The binary structure and low-level byte patterns in PE files make explanation methods particularly challenging. However, PAC-X specifically designed for interpretable malware detection in PE files, addressing the limitations of existing XAI methods.

III. Model Development

PAC-X is a multi-class, multi-level explainable algorithm. The construction of the PAC-X model involves two main components, which are as follows:

A. Conditional Attention Neural Network (CAN-NET)

The CAN-NET is designed to integrate class-specific attention mechanisms within a neural network framework to directly calculate feature contributions for each predicted class (see Figure 1). Unlike conventional attention layers, CAN-NET independently learns attention weights for each class, enabling precise, class-wise explainability.

Let $X = [x_1, x_2, ..., x_n]$ represent the input feature vector, where x_i is the i-th feature. The attention mechanism in CAN-NET assigns an attention weight vector $A_c = [a_{c1}, a_{c2}, ..., a_{cn}]$ for each class $c \in C$. These attention weights are learned parameters that directly indicate the importance of each feature x_i for predicting class c.

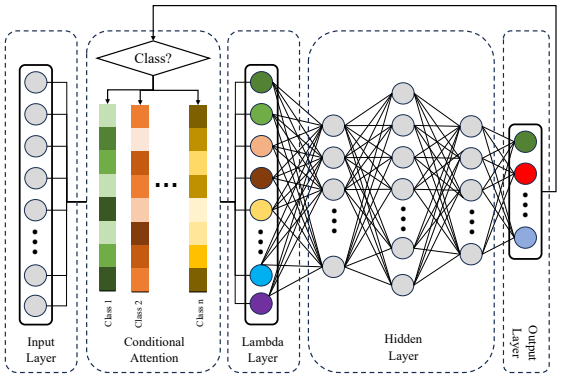


Fig. 1: CAN-Net architecture.

For each class c, the attention-weighted feature vector is computed as:

$$Z_c = \sum_{i=1}^{n} a_{ci} \cdot x_i \tag{1}$$

where a_{ci} is the attention weight of feature x_i for class c. This forms the logit score Z_c used for softmax prediction:

$$P(c|X) = \frac{\exp(Z_c)}{\sum_{j=1}^{|C|} \exp(Z_j)} \tag{2}$$

The model is trained using the standard cross-entropy loss:

$$L = -\sum_{c \in C} Y_c \log P(c|X) \tag{3}$$

where Y_c is the true label for class c.

During backpropagation, both the model weights and the attention weights are updated to minimize the loss. The attention weights A_c provide an interpretable mapping from features to the predicted class, directly quantifying each feature's contribution.

The unique conditional adjustment in CAN-NET focuses attention refinement on correctly predicted samples to stabilize learning and enhance feature consistency. For correctly predicted classes, the attention update follows:

$$A_c := A_c - \eta' \frac{\partial L}{\partial A_c} \quad \text{if } c = \hat{c} \tag{4}$$

where $\hat{c}$ is the predicted class, and η' is the attention learning rate. Incorrect predictions do not influence attention updates, allowing the network to selectively reinforce meaningful feature attributions.

Within the PAC-X framework, CAN-NET extracts two key components:

- **Prospect:** The softmax output, providing the class probability distribution.
- **Aspect:** The class-specific attention weights A_c corresponding to the predicted class, directly representing the feature contributions.

These outputs offer a transparent, local explanation of the prediction, with the attention mechanism serving as an intrinsic, not post-hoc, explainability layer.

In summary, CAN-NET provides a mathematically grounded mechanism to compute feature contributions through class-specific attention, satisfying the core requirement of PAC-X to deliver interpretable, multi-class fuzzy explanations.

B. Contextual Fuzzy Clustering (CFC)

CFC is an innovative approach incorporating contextual information into the fuzzy clustering paradigm. Traditional fuzzy clustering techniques, such as Fuzzy C-Means (FCM), allocate membership degrees to data points relative to each cluster, indicating the extent of their association with each cluster. CFC advances this concept by considering the context in which data points are situated, thereby providing a richer and more informative clustering output.

Unlike FCM, which solely relies on feature-based distances, CFC integrates both feature similarity and proximity to fuzzy set boundaries to determine membership degrees. This enables CFC to capture nuanced geometric relationships, particularly in scenarios where samples may fall near the edges of multiple fuzzy regions. The geometric proximity, based on fuzzy set structures, enhances local instance-level explainability, which traditional FCM cannot achieve. Moreover, while FCM provides global clustering results, CFC is specifically designed to support sample-specific, feature-wise explanations that remain robust even under adversarial modifications. This geometric integration makes CFC uniquely suitable for fuzzy explainability in malware detection tasks where feature overlaps and polymorphic behaviors are common.

CFC is applied in this study using a two-stage process: first, fuzzy sets are generated from each feature using K-Means clustering to identify initial cluster centers, and second, fuzzy set boundaries (triangular membership parameters) are calculated and stored. These fuzzy sets are later used to determine how closely each new test sample aligns with pre-defined fuzzy regions, enabling local, instance-level explanations.

Algorithm 1 Contextual Fuzzy Clustering (CFC)

Require: Dataset $\mathcal{X}$, number of clusters c, fuzziness parameter m, weighting coefficients α, β
Ensure: Membership degrees u_{ij}, centroids C_j
0: Initialize centroids C_j using K-Means for each feature
0: **repeat**
0: **for** each data point x_i and each cluster j **do**
0: Compute distance $\|x_i - C_j\| = \alpha \cdot d_{\text{feature}}(x_i, C_j) + \beta \cdot d_{\text{context}}(x_i, C_j)$
0: Update membership degree $u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - C_j\|}{\|x_i - C_k\|} \right)^{\frac{2}{m-1}}}$
0: **end for**
0: **for** each cluster j **do**
0: Update centroid $C_j = \frac{\sum_{i=1}^{n} u_{ij}^m \cdot x_i}{\sum_{i=1}^{n} u_{ij}^m}$
0: **end for**
0: **until** convergence criteria met
0: **return** Membership degrees u_{ij}, centroids C_j =0

The foundation of fuzzy clustering lies in the assignment of membership degrees u_{ij} to each data point i for each cluster j, signifying the level of belonging of each data point to each cluster. For a conventional FCM, the membership degree u_{ij} of the i-th data point to the j-th cluster is determined based on the distance d_{ij} between the data point and the cluster's centroid, typically employing the Euclidean distance. The membership degrees are normalized across all clusters for each data point to sum to 1. In this study, the optimal number of fuzzy sets (clusters) for each feature is determined using the Elbow Method and Silhouette Analysis, selecting the minimum of the two to prevent overfitting and ensure stable clustering.

Once the clusters are formed using K-Means, the fuzzy set parameters a, b, and c for each cluster are calculated as follows:

- b is the cluster centroid obtained from K-Means.
- a is the midpoint between the centroid of the current cluster and the centroid of the preceding (left) cluster.
- c is the midpoint between the centroid of the current cluster and the centroid of the following (right) cluster.
- For the first and last clusters, a and c are extended based on the minimum and maximum observed values of the feature, respectively.

This procedure ensures that the triangular fuzzy membership functions fully cover the feature space and accurately represent the separation between clusters.

In CFC, the membership degree calculation is enriched with contextual information. Let C_j denote the j-th fuzzy set (cluster), defined by its centroid and spread, and let x_i represent the i-th data point. The membership degree u_{ij} is then computed as follows:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - C_j\|}{\|x_i - C_k\|} \right)^{\frac{2}{m-1}}} \tag{5}$$

where:

- $\|x_i - C_j\|$ measures the distance between the data point x_i and the centroid of cluster C_j, encapsulating both feature space distance and contextual distance.
- c is the total number of clusters, optimally determined for each feature using Elbow and Silhouette methods.
- $m > 1$ is the fuzziness parameter. In this study, we set $m = 2$ following standard fuzzy clustering practice to balance cluster overlap and separation [32].

The distinctiveness of CFC lies in the calculation of distance $\|x_i - C_j\|$, which is articulated as a blend of feature space distance and contextual distance:

$$\|x_i - C_j\| = \alpha \cdot d_{\text{feature}}(x_i, C_j) + \beta \cdot d_{\text{context}}(x_i, C_j) \tag{6}$$

where:

- $d_{\text{feature}}(x_i, C_j)$ is the conventional feature-based distance. For instance, the Euclidean distance between x_i and the centroid of C_j.
- $d_{\text{context}}(x_i, C_j)$ is the contextual distance, which in this framework is strictly a geometric distance reflecting the proximity of the sample's feature value to the boundaries of the fuzzy set. It is calculated as:

$$d_{\text{context}}(x_i, C_j) = \min\left(|x_i - a|, |x_i - c|\right) \tag{7}$$

where a and c are the left and right boundaries of the fuzzy set C_j. This distance measures how close the feature value of the test sample is to the edges of the fuzzy set, not to its centroid. For multi-dimensional features, this distance is averaged across dimensions.

- α and β are weighting coefficients balancing the influence of feature-based and contextual distances. We empirically set $\alpha = 0.7$ and $\beta = 0.3$ to prioritize feature similarity while preserving edge proximity relevance.

The inclusion of this **edge proximity-based contextual distance** brings significant advantages. Unlike centroid-based methods, measuring the minimum distance to fuzzy set boundaries captures how close the test sample is to the decision boundary, which is often where class transitions or feature overlaps occur. This approach improves local, feature-specific explainability, especially in regions where membership is uncertain or changing. It also enhances the model's robustness by providing sensitivity to subtle feature variations near the edge, which is critical for identifying polymorphic and evasive malware behaviors.

During testing, each feature of the test sample is compared to the stored fuzzy sets based on this blended distance. The top three closest fuzzy sets for each feature are identified to generate local, feature-specific explanations. This mapping provides transparent and interpretable justifications for why the test sample is classified in a particular way.

The centroids are iteratively updated based on the current membership degrees:

$$C_j = \frac{\sum_{i=1}^{n} u_{ij}^m \cdot x_i}{\sum_{i=1}^{n} u_{ij}^m} \tag{8}$$

where n represents the number of data points.

The iterative updates continue until the changes in centroids or membership degrees between successive iterations fall below a predetermined threshold, indicating convergence (see Algorithm 1).

The output of CFC is a collection of fuzzy sets with well-defined feature boundaries, forming the contextual basis for PAC-X explanations. The number of fuzzy sets per feature is automatically determined, ensuring that the process is data-driven and not arbitrarily fixed.

CFC offers a comprehensive framework for discerning the complex structure of data by seamlessly integrating geometric contextual information into the clustering process. Through the mathematical formulation delineated above, CFC enriches the conventional fuzzy clustering paradigm, facilitating a more profound comprehension of feature-wise relationships within the data.

IV. Datasets and Implementation

A. Dataset details

We employed two distinct datasets to evaluate the proposed PAC-X framework: one consisting of Windows PE executable files and the other comprising Android APK files. Both datasets include a variety of malware families and benign samples, ensuring diversity and relevance to real-world malware detection.

TABLE I: Label Distribution in the PE and APK Datasets

PE Dataset		APK Dataset	
Label	**Count**	**Label**	**Count**
Benign	2,422	Benign	1,795
DownloadAdmin	584	Adware	1,253
GandCrab	454	Banking Malware	2,100
Emotet	304	SMS Malware	3,904
Bundlore	300	Riskware	2,546
Firseria	224	-	-
Gamarue	174	-	-
DownloadGuide	161	-	-
Bladabindi	146	-	-
Fareit	115	-	-
DarkKomet	106	-	-

The PE dataset contains a total of 4,990 executable files after pre-processing and removing duplicates, including malware samples collected from MalShare[1] and VirusShare[2], and benign samples sourced from SourceForge[3] and Download.com[4]. The malware samples cover multiple families, including ransomware, trojans, downloaders, and worms. Feature extraction was performed extensively. The dataset includes obfuscated and polymorphic samples, as commonly found in Windows-based malware, similar to datasets used in prior works [21], [14], [33].

The Android APK dataset, originally constructed by Samaneh et al. [9], consists of 11,598 samples. Features were dynamically collected using CopperDroid, a VMI-based dynamic analysis system capable of reconstructing both low-level OS behaviors and high-level Android-specific activities. This dataset includes system call frequencies, binders, and composite behaviors, with a total of 470 extracted features. The APK dataset spans a range of malware families, including Adware, Banking malware, SMS malware, and Riskware, as well as benign applications. The Android malware samples were sourced from VirusTotal[5], the Contagio security blog, AMD, MalDozer, and other widely referenced datasets.

B. Evaluation criteria

Explainability is evaluated by measuring the Maximum Mean Discrepancy (MMD) scores across distributions of feature importance. This analysis focuses on two key metrics:

Inconsistency (C): This metric assesses the dissimilarity of explanations within the same class. Explanations for randomly chosen files from each class are generated to gauge inconsistency. These files are segregated into groups based on even and odd indices, forming two separate data distributions. The MMD between these distributions is then calculated. A lower MMD score indicates lower inconsistency, which is desirable, as it suggests similar explanations within the same family of classes. However, in some cases, the model may exhibit high inconsistency within the same class due to the diversity of impurities present in the dataset. **Robustness**

[1]https://malshare.com
[2]https://virusshare.com
[3]http://sourceforge.net
[4]http://www.download.com
[5]https://www.virustotal.com/

(R): Robustness evaluates the distinctiveness of explanations among different classes. For this metric, explanations are generated for various sample sizes across all families, including benign samples. Subsequently, the MMD for each unique pairing is computed. An average robustness score for each class is derived by averaging these MMD values and further averaging them across different sample sizes. A robust model is indicated by higher robustness scores, reflecting clear differentiations in explanations across classes. However, consistency and robustness are often the trade-offs. Increasing robustness may reduce consistency within the same class. We introduce a combinative metric called **Explainability Power (EP)** to address this.

We conduct a comparative analysis with explanations derived from the state-of-the-art to assess the explainability provided by the PAC-X model. These methods can yield explanations at different scales, necessitating a normalization step before comparison. The MMD is utilized for this purpose, formulated as follows:

$$\text{MMD}(X,Y) = \left\| \frac{1}{n_X}\sum_{i=1}^{n_X}\phi(x_i) - \frac{1}{n_Y}\sum_{j=1}^{n_Y}\phi(y_j) \right\|_2^2 \quad (9)$$

In the above equation:

- X and Y represent the sets of feature importance scores from comparing two explanation methods.
- n_X and n_Y denote the number of observations in each set.
- x_i and y_j are each set's individual feature importance scores.
- $\phi(\cdot)$ is the kernel function that projects the data into a higher-dimensional feature space.
- The Euclidean norm $\|\cdot\|_2$ calculates the root of the sum of squares, thereby quantifying the distance between the two distributions.

Normalized scores are computed using the equation:

$$\text{Normalized MMD}_{ij} = \left| \frac{\text{MMD}_{ij} - \text{Min MMD}_j}{\text{Max MMD}_j - \text{Min MMD}_j} \right| \quad (10)$$

where MMD_{ij} refers to the discrepancy score between the i^{th} family's explanation and the j^{th} method's average, with Min MMD_j and Max MMD_j being the minimum and maximum MMD scores for the j^{th} method respectively. This normalization facilitates a direct comparison of the distinctiveness and consistency of explanations across different models and explanation methods.

If we want to calculate the EP of an XAI algorithm, it can be defined as:

$$EP = \frac{R}{C} \times 100$$

where R represents robustness and C represents consistency. A higher EP score indicates the model's higher level of explainability.

C. Implementation and parameter tuning

The core component of CAN-Net is the Conditional Attention Layer, which allows the model to focus on different parts of the input data based on the predicted class, providing more interpretable and precise explanations for each class. We have developed two distinct architectures for CAN-Net tailored to each dataset, as detailed in Table II. The hyperparameters for these architectures were optimized using the RandomizedSearchCV[6] algorithm.

TABLE II: Parameters of CAN-Net model for both datasets.

Parameter	APK data	PE data
Conditional attention layer (Shape: Number of features x Number of classes)		
Lambda layer (Shape: Number of features x 1)		
Dense layers and dropout		
Dense layers	(1024, 1024, 1024, 512, 512)	(512, 32, 16)
Activation	ReLU	ReLU
Dropout layers	(0.3, 0.2, 0.2, 0.1)	(0.3, 0.3)
Output layer		
Number of classes	5	16
Activation	Softmax	Softmax
Training parameters		
Optimizer	Adam	Adam
Loss function	Categorical cross-entropy	Categorical cross-entropy
Number of epochs	200	5
Split data (Train, Validation, Test)	(80%, 20%, 20%)	(80%, 20%, 20%)

Architecture for APK files: The architecture includes five dense layers with ReLU activation functions, interspersed with dropout layers to prevent overfitting. The final layer is a softmax output layer that produces the class probabilities. The model is compiled with the Adam optimizer and categorical cross-entropy loss function and trained for 200 epochs.

Architecture for PE files For the PE dataset, the CAN-Net model has a simpler architecture due to the different nature and complexity of the data. This model includes two dense layers with ReLU activation functions and dropout for regularization, followed by a softmax output layer. It is also compiled with the Adam optimizer and categorical cross-entropy loss function and trained for 5 epochs.

We implemented two additional models for each dataset, an ANN and Attention-based Neural Network (AttNN), using the same hyperparameters for performance comparison.

V. Results and Discussion

A. Explainability assessment and comparison

We implemented several state-of-the-art explainability algorithms on both datasets and evaluated their performance using multiple quantitative metrics (Table III and Table IV). The experiments were conducted across multiple runs, and the reported results represent the mean and standard deviation for each metric. To assess the statistical significance of performance differences, we conducted paired *t*-tests comparing PAC-X with other baseline explainers, namely LIME, SHAP, and LEMNA. The results show statistically significant improvements ($p < 0.05$) for PAC-X across most metrics and datasets, except for the robustness metric in the PE dataset's

[6]https://www.rdocumentation.org/packages/superml/versions/0.5.7/topics/RandomSearchCV

inconsistency scores ($p = 0.0844$). This minor deviation may be attributed to the limited variability within PE samples and the inherent similarity of structural patterns across certain malware families, which reduces sensitivity to perturbations.

For the Riskware family, SHAP performed better than all other algorithms, possibly because of its ability to capture the subtle dependencies among features that characterize Riskware. The consistency scores for PAC-X are lower than most other algorithms, indicating PAC-X's capability to provide consistent explanations for the same malware families. In terms of robustness, LIME outperformed other algorithms, including PAC-X. This is particularly evident for Benign and Adware families, where LIME exhibited significantly higher robustness. The performance of LIME can be attributed to its local surrogate model approach, which effectively captures the feature behavior for these classes. However, LIME did not perform well for SMS malware, indicating its limitations in handling certain malware types. PAC-X, on the other hand, demonstrated superior performance for most algorithms across all malware families. We calculated the EP for each algorithm and malware family (see Table III). AttNN performed exceptionally well for benign samples but showed significantly lower performance for other families. On average, PAC-X outperformed all state-of-the-art algorithms, demonstrating its effectiveness and robustness across diverse malware families.

B. Discriminative power analysis

Similar to the explainability assessment, we also conducted a discriminative performance evaluation to measure each model's classification capability. To ensure robustness and generalizability, the datasets were partitioned using multiple random seeds, creating diverse training and testing splits. We employed a k-fold cross-validation strategy, and the reported results in Table IV represent the average performance with standard deviation across folds. A paired *t*-test was performed to assess the statistical significance of differences in accuracy and F1-score between CAN-Net and other baseline models. For all metrics and both datasets, the results were statistically significant with $p < 0.05$, confirming the superiority of CAN-Net over the compared approaches. The performance metrics, presented in Table IV, show that CAN-Net outperforms the other models on both datasets. This indicates that CAN-Net is highly effective in identifying malware in APK files, and PE files dataset.

C. Scaling and computational complexity

We evaluated Mean Time to Detect (MTTD), which reflects the average time taken by the detector to correctly identify a threat. As shown in Table IV, CAN-Net, due to its use of multiple class-specific attention mechanisms, naturally incurs a higher detection time compared to other. However, the increase in MTTD is only marginal across both datasets and remains within an acceptable range for practical malware detection scenarios.

To further assess scalability, we varied the batch size during testing and observed that all detection models showed an approximately linear increase in detection time as the batch size increased (Figure 2). Additional experiments with varying batch configurations confirmed that other performance metrics such as robustness, inconsistency, and discriminative power remained largely stable, indicating that the scalability trade-off in CAN-Net does not compromise its detection reliability or explainability.

For completeness and to assess generalizability, we further computed the computational cost of PAC-X. PAC-X consists of two components: CAN-Net (a deep conditional attention network) and the Contextual Fuzzy Clustering (CFC) module. For an input with d features and C classes, the CAN-Net forward pass has asymptotic cost

$$O\Big(\sum_{\ell=1}^{L} h_{\ell-1}h_{\ell} \;+\; C \cdot d\Big),$$

where h_ℓ are the layer widths and $h_0 = d$. Training multiplies this cost by the number of training samples and epochs. The CFC preprocessing (K-Means initialization with fuzzy membership iteration) has complexity approximately

$$O(d \cdot N \cdot K_f \cdot I),$$

where N is the number of samples, K_f the average number of fuzzy sets per feature, and I the number of clustering iterations. During inference, the extra cost of mapping a sample to fuzzy sets is $O(d \cdot K_f)$.

Using the APK experiment settings from this paper ($d = 470$, $C = 5$, dense layers [1024,1024,1024,512,512]), a single forward pass requires on the order of 10^6–10^7 multiply–accumulate operations, while per-feature CFC mapping is an order of 10^3–10^4 distance computations per sample. Thus, the dense layers dominate runtime; the CFC is a non-trivial but one-time preprocessing cost that scales linearly with N and is highly parallelizable. These properties explain the empirical linear batch scaling observed in our MTTD experiments. Practical optimisations include reducing K_f, feature selection, model compression (pruning/quantization), and parallelising CFC across features.

D. Interactive explainability analysis

PAC-X provides an interactive graph that allows for the exploration of individual components through a click function. Figure 8 illustrates PAC-X's explainability for an APK file from the Android dataset, specifically from the 'Riskware' class. This visualization consists of three tracks: the inner track represents prospects (predicted labels and their probabilities, indicated by the area on the track), the mid-track represents aspects for a specific class/prospect, and the outermost track presents the context, including fuzzy sets to which the features belong based on their values. The center circle shows the actual label.

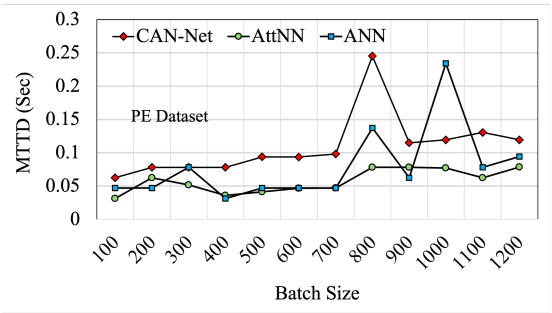
Case study: Riskware APK file. In the example shown in Figure 8a, the file belongs to the 'Riskware' class and was correctly predicted. However, the explainability feature of PAC-X reveals that some features also support other classes, with the second most probable class being 'Banking'. Using PAC-X's explainability, these features can be easily explored (see

TABLE III: Performance and comparison of PAC-X with state-of-the-art methods, considering Inconsistency, Robustness, and EP Score.

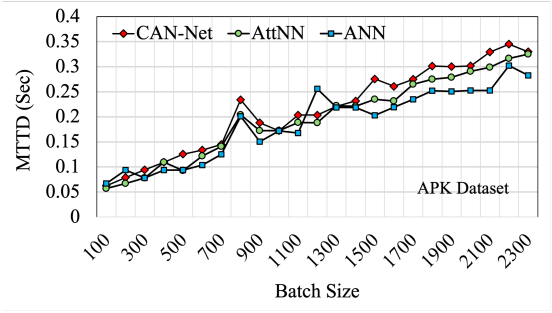
Metric	LIME	SHAP	LEMNA	PAC-X	p (LIME)	p (SHAP)	p (LEMNA)
APK files dataset							
Inconsistency	0.3020 ±3.57E-05	0.2324 ±0.0295	0.3133 ±2.35E-07	0.2333 ±0.1896	1.12E-05	9.92E-05	1.11E-05
Robustness	0.1545 ±0.0532	0.1081 ±0.0222	0.1147 ±0.0241	0.1233 ±0.0245	0.0061	0.0213	0.0231
EP Score	0.5097	0.4616	0.3808	0.5345			
PE files dataset							
Inconsistency	0.0945 ±0.1261	0.0256 ±0.0090	0.0951 ±0.2143	0.2091 ±0.2153	0.0241	0.0011	0.0844
Robustness	0.2426 ±0.1202	0.0352 ±0.01005	0.0438 ±0.0189	0.7665 ±0.1578	1.13E-09	4.07E-13	1.32E-12
EP Score	8.2137	1.5486	1.1030	23.2393			

TABLE IV: Model metrics (mean ± std), MTTD, and paired t-test p-values comparision.

Algorithm	Precision	Recall	f1 Score	Accuracy	MTTD	p (f1 Score)	p (Accuracy)	XAI
APK files dataset								
ANN	0.8759	0.8651	0.9073 ± 0.0078	0.9060 ± 0.0075	0.1865	0.0029	0.0017	No
AttNN	0.7599	0.8443	0.7957 ± 0.0026	0.8434 ± 0.0025	0.1979	2.39E−06	1.27E−05	Yes
ResNet18	0.8869	0.8859	0.8859	0.8859	–	-	-	No
DenseNet21	0.9071	0.9072	0.9071	0.9072	–	-	-	No
PLSAE [9]	0.9602	0.9600	0.9600	0.9601	0.2306	-	-	No
CNN [10]	0.8900	0.8900	0.8900	0.8900	–	-	-	No
ViT [24]	0.8753	0.8681	0.8701	0.8681	–	-	-	No
CAN-NET	0.9048	0.9041	0.9404 ± 0.0095	0.9402 ± 0.0096	0.2140	N/A	N/A	Yes
PE files dataset								
ANN	0.9948	0.9945	0.9921 ± 0.0023	0.9922 ± 0.0023	0.0791	0.0029	0.0015	No
AttNN	0.9403	0.9530	0.9386 ± 0.0186	0.9522 ± 0.0029	0.0575	1.14E−05	7.73E−06	Yes
CFGExplainer [31]	0.8100	0.7800	0.7900	0.8300	–	-	-	Yes
GAGE [30]	0.9000	0.8500	0.8700	0.8700	–	-	-	Yes
CAN-NET	0.9977	0.9977	0.9983 ± 0.0010	0.9983 ± 0.0010	0.1093	N/A	N/A	Yes



(a) PE files dataset



(b) APK files dataset

Fig. 2: MTTD analysis over batch size scaling.

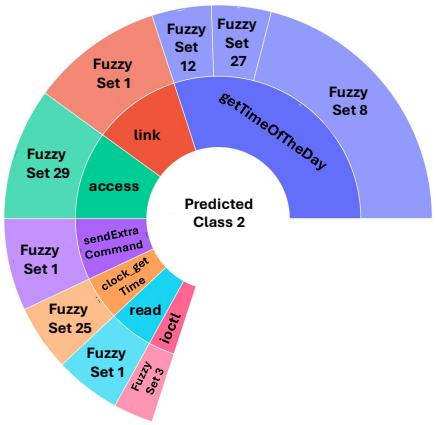


Fig. 3: Aspects of predicted class 2 and their contextual information.

Figure 3). Features such as *ioctl*, *gettimeofday*, *clock_gettime*, and *access* are the most supportive system calls used in 'Banking' related malicious APKs.

The *sigaction* system call is an aspect of the 'Riskware' class, which, according to its frequency, belongs to three fuzzy sets: 19, 73, and 23 (see Figure 4). These sets include a significant number of files from class 4 that belong to sets 23 and 73. However, set 19 also contains a notable number of files from the 'SMS malware' class.

Advantage of context. The context provided by PAC-X is particularly useful for understanding cases of misclassification or intentional adversarial manipulation to achieve incorrect predictions. For example, see Figure 5, which shows the explanation generated for a PE file from the Windows malware dataset. This file was classified as benign, but further exploration of its features reveals that the context includes

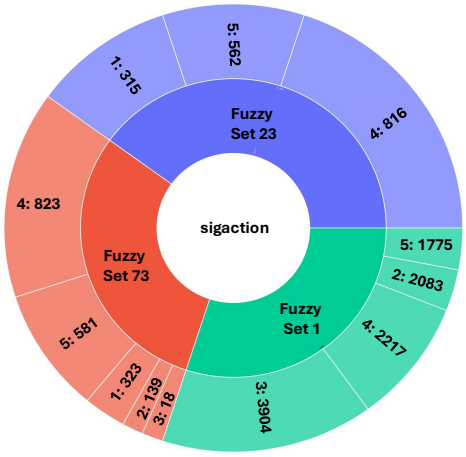


Fig. 4: Contexts of the feature $sigaction$, which supports the prediction for class 4, revealing that it belongs to three fuzzy sets based on the frequency of this call. These sets show the highest number of class 4 occurrences, with their frequencies displayed on the outermost track.

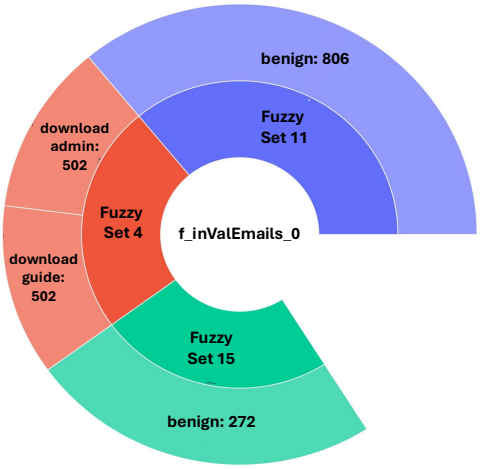


Fig. 5: Analysis of a file from the PE dataset. The initial prediction was $benign$, but upon examining the feature $f\_InValEmail\_0$ (a list of invalid emails), it was found to belong to three fuzzy sets: 4, 11, and 15. While sets 11 and 15 contain only benign samples, fuzzy set 4 includes two malware files, $downloadguide$ and $downloadadmin$. This indicates to malware analysts that the file requires further investigation.

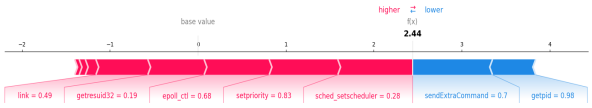


Fig. 6: SHAP explainability diagram

three fuzzy sets: 4, 11, and 15. Sets 11 and 15 primarily contain benign files, but set 4 includes files related to 'downloadadmin' and 'downloadguide' malware. This indicates that malware analysts should conduct further investigations. Upon examining the binary, invalid email IDs such as *@pdata*, *0@xdata*, and *0@idata* were found, which may be related to 'downloadadmin' and 'downloadguide' malware.

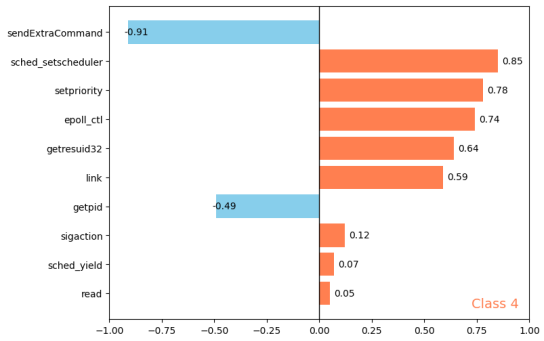


Fig. 7: LIME explainability diagram

Case study: Gandcrab malware. To demonstrate PAC-X's practical application and correctness, we conducted a case study on a Gandcrab malware sample[7]. This file belongs to the gandcrab malware family from Ransomware[8]. We focused only on the predicted class's top aspects (see Figure 8b).

The presence of _ReflectiveLoader@0 in the ExportsList is significant for malware analysis, particularly for Gandcrab ransomware. This export suggests the use of reflective DLL injection, a technique often employed by sophisticated malware to load a DLL from memory rather than from disk. In addition, the $longWord$ and $sentences$ features contain numerous references to $crypt$ and registry-related commands. Similar patterns were found in the import/export-related features, highlighted in red and pink.

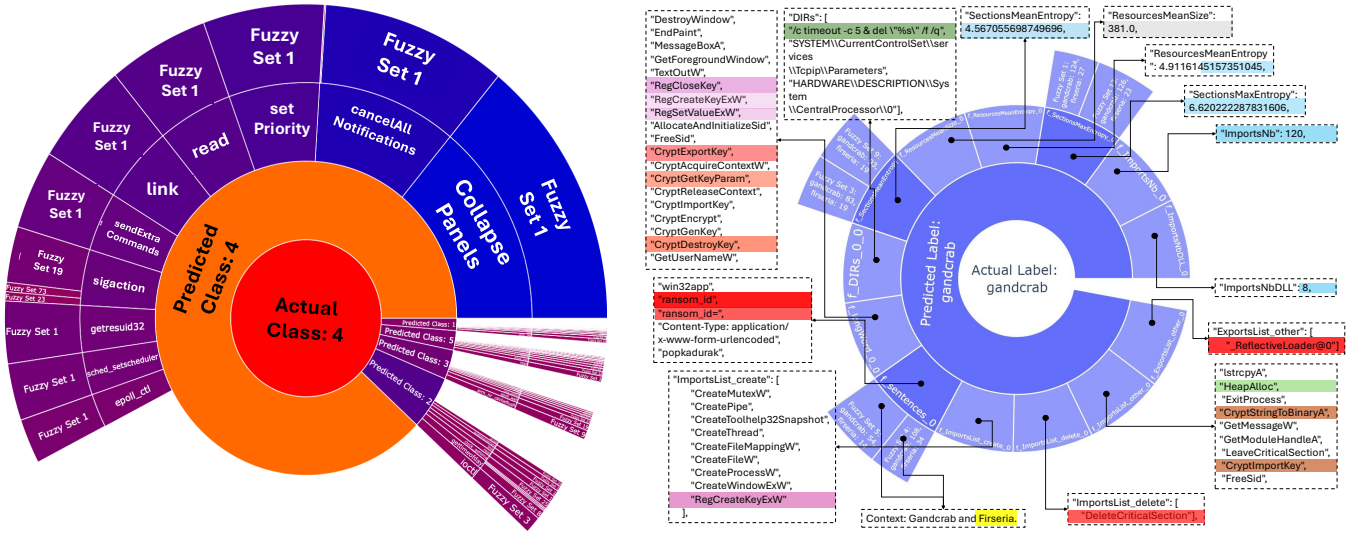
The DIRs list includes several directory paths and system registry paths that might be accessed or modified by a malware sample. The command highlighted in green deletes a specified file forcefully and quietly after a timeout of 5 seconds, a behavior consistent with ransomware that deletes its installer or evidence after execution. Additionally, the DIRs list contains registry paths related to network configuration parameters. Ransomware might manipulate these settings to control network behavior or ensure communication with its command and control (C2) servers.

The numeric features provide insights into the characteristics of the executable file associated with Gandcrab ransomware. High entropy in sections suggests encrypted or compressed data, common in ransomware for hiding its payload. These values are highlighted in light blue colour. These values indicate the presence of encrypted or compressed data, small resources for configuration or payload purposes, and numerous imports for system interactions typical of sophisticated malware like ransomware. We also explored the context of some features and found that most fuzzy sets contain $gandcrab$ and $firseria$ malware files. Since $firseria$ is also ransomware-related, it shares similar features with Gandcrab.

This analysis highlights the practical application of PAC-X in identifying and explaining malware behavior.

Empirical/Visual case study with traditional methods. The visual comparisons demonstrate that LIME explanations are limited to local feature importance for the predicted class,

[7]Name: origin_187__.file
[8]MD5: FFAE47A76B3CD5A0791856319852C84B



(a) PAC-X explainability diagram.

(b) Case study on Gandcrab malware.

Fig. 8: (a) PAC-X explainability diagram: Excluding the center, the diagram comprises several tracks. The innermost track presents *prospects*, which are the possible probabilities of different classes. The yellow track indicates that the predicted class is 4, while also showcasing the probabilities of other classes. The middle track displays aspects of the predicted class 4, with aspects of other classes available upon clicking the respective *prospect*. The outermost track presents the *context* of the aspect shown in the middle track. (b) Gandcrab case study: Critical ransomware and encryption-related features are highlighted in red, registry changes in pink, numerical features in light blue, and suspicious coding in light green (such as file deletion in DIRs feature and heap-related import libraries). Key contexts are highlighted in yellow.

offering no insight into other possible class contributions (Figure 7). SHAP improves by providing multi-class attributions but lacks the ability to explain the feature's position relative to decision boundaries or the broader contextual behavior within the data distribution (Figure 6). In contrast, PAC-X offers a more comprehensive explanation by combining class-specific feature attention with contextual fuzzy clustering (Figure 8a). This approach not only explains the primary class but also transparently reveals secondary class associations and the proximity of features to fuzzy decision regions. PAC-X uniquely captures multi-class, feature-wise, and context-aware insights, providing stable and robust explanations that remain interpretable even in adversarial scenarios—advantages not achievable by LIME and SHAP.

E. Explaining adversarial samples

We generated adversarial samples through obfuscation using Alcatraz, a binary obfuscator capable of obfuscating various PE files[9], ensuring that the actual functionality of the binaries remained intact. For the Android system call data, we enhanced the frequency of benign system calls to create adversarial behavior without altering the core functionality. Following this, we produced explainability assessments and evaluated them using the defined metrics, ultimately calculating the EP. Among the state-of-the-art methods, only SHAP focuses on explaining multiclass classifications. Therefore, we compared PAC-X's EP with SHAP's, and additionally conducted a qualitative analysis of explainability.

[9]https://github.com/weak1337/Alcatraz

1) Quantitative comparison: Upon evaluating the EP for both original and adversarial cases for both algorithms, it was observed that SHAP's EP decreased for adversarial examples. In contrast, PAC-X's EP generally increased across both datasets, with the exception of a few classes such as Riskware in the APK dataset and Gandcrab in the Windows dataset. While PAC-X explanations showed a reduction in EP for certain classes, on average, PAC-X enhanced the EP, whereas SHAP, on average, demonstrated a reduction.

Furthermore, we calculated the differences ($|\delta|$) between the EP for original and adversarial cases to observe the changes in explainability due to adversarial examples. Our findings revealed that in some instances, PAC-X maintained similar explanations even after adversarial attacks. However, most classes exhibited significant differences in EP from original to adversarial when compared to SHAP. This is because the obfuscation altered some of the malicious features, which were effectively detected by PAC-X's multi-conditional attention layer.

The detailed results are shown in Table V. For the APK files dataset, PAC-X demonstrated a substantial increase in EP for adversarial examples compared to SHAP. For instance, the Adware class showed an EP increase of 0.9021 for PAC-X, while SHAP had a decrease of 0.0978. Similarly, PAC-X exhibited an EP increase of 1.6126 for the Banking class, compared to SHAP's decrease of 0.4606. This trend was consistent across most classes, highlighting PAC-X's superior ability to handle adversarial attacks.

In the PE files dataset, PAC-X again outperformed SHAP in maintaining and even improving EP under adversarial con-

TABLE V: EP Scores for SHAP and PAC-X on original and adversarial samples. The $|\delta|$ column represents the absolute difference in EP between the original and adversarial samples.

Dataset	SHAP (EP)			PAC-X (EP)		
	Original	Adversarial	$\|\delta\|$	Original	Adversarial	$\|\delta\|$
APK Files	0.8646	0.9068	0.6349	0.3073	1.4081	1.1009
PE Files	0.3247	0.1418	0.2582	2.2841	2.8190	1.2535

ditions. PAC-X's EP increased by 2.2875 for the benign class, while SHAP's EP decreased by 0.0881. For the Darkkomet class, PAC-X showed an EP increase of 1.6194, whereas SHAP had a decrease of 0.1207. These results underscore PAC-X's robustness in providing reliable explanations even when faced with adversarial manipulation.

The quantitative analysis underscores PAC-X's robustness and resilience in providing meaningful explanations, even under adversarial conditions. PAC-X's ability to maintain or enhance EP in the face of adversarial manipulation demonstrates its superiority in generating reliable and interpretable explanations across various malware classes.

2) Qualitative comparison: The above metrics were determined based solely on the aspects of the $actualclass$, without considering contextual information. The contextual descriptions in PAC-X's explainability remain consistent regardless of adversarial manipulations. This inherent stability ensures that even when attackers attempt to evade detection by altering features, the contextual information serves as a reliable cross-reference. For instance, in the case of a malware sample that has been adversarially altered to evade detection, PAC-X can still provide insights into the original malicious context. This feature is invaluable for cybersecurity professionals who need to understand the underlying behavior and origin of the threat, even when faced with sophisticated evasion techniques.

PAC-X introduces a novel interactive explainability method, distinguishing itself from other state-of-the-art techniques. Unlike SHAP, which requires training to produce explanations for every explanation that is both time-consuming and computationally intensive—PAC-X provides immediate explanations once the model is trained. This efficiency becomes crucial in cybersecurity environments, where malware characteristics evolve rapidly due to obfuscation or adversarial manipulations. PAC-X leverages contextual information that remains stable even when malware definitions shift, ensuring reliable explanations without the need for retraining. This allows cybersecurity professionals to maintain situational awareness and respond promptly to emerging threats.

F. Limitations and future directions

Although PAC-X demonstrates strong performance and interpretability, several limitations remain that warrant further investigation. The framework's effectiveness depends on the quality and diversity of the training data, as poor or biased datasets may lead to misleading contextual relationships. Moreover, the CFC module involves a computationally intensive process to determine optimal cluster boundaries, which can increase training time.

Another challenge arises from data and concept drift, which are common in malware evolution. Since PAC-X relies on contextual patterns learned from historical samples, the emergence of new variants or evasion techniques may cause gradual degradation in accuracy or explainability. To address this, future work could explore online and incremental learning strategies, enabling CFC to update fuzzy sets dynamically as new samples appear. Additionally, the class-specific attention layers in CAN-Net can be fine-tuned through transfer or few-shot learning to adapt to unseen malware behaviors. These directions would enhance PAC-X's robustness and practicality for real-world, continuously evolving threat environments.

VI. Conclusion

In this paper, we introduced PAC-X, an innovative multi-label explainable model that provides comprehensive explainability across all classes with significant probabilities. Through the deployment of the CAN-Net, PAC-X extracts precise explanations for each class. Additionally, PAC-X employs CFC to map feature distributions within the training data, offering a nuanced understanding of the feature space. PAC-X demonstrates exceptional discriminative power, achieving an accuracy of 99.99% for the PE dataset and 96.26% for the APK dataset. It surpasses XAI benchmarks, including LIME, LEMNA, and SHAP, in terms of explainability. Our evaluation of consistency and robustness metrics confirmed that PAC-X provides more consistent and robust explanations than LIME and SHAP.

One of the key advantages of PAC-X is its resilience to adversarial manipulations. Even when malicious samples are altered to evade detection, PAC-X can still provide meaningful, context-aware explanations that reveal potential evasion tactics. Its interactive explainability allows users to dynamically explore model behavior across prospect, aspect, and context layers, offering a comprehensive view of decision-making. This multi-level structure helps malware analysts and cybersecurity experts not only identify influential features but also understand their interactions within the broader dataset. PAC-X ultimately enhances interpretability, transparency, and trust in AI-driven malware detection.

Acknowledgment

This research is supported by BlackBerry Limited (ALLRP 561035), Defence Research and Development Canada (contract no. W7701-217332), NSERC Alliance Grants (ALLRP 561035-20), NSERC Discovery Grants (RGPIN-2024-04087), NSERC CREATE Grants (CREATE-554764-2021), and Canada Research Chairs Program (CRC-2019-00041). For additional information, datasets, and source code related to this work, please visit our GitHub repository at https://github.com/McGill-DMaS/PACX

REFERENCES

[1] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings 12*. Springer, 2015, pp. 3–24.

[2] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[3] R. Alenezi and S. A. Ludwig, "Explainability of cybersecurity threats data using shap," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 01–10.

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[5] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 180–186. [Online]. Available: https://doi.org/10.1145/3375627.3375830

[6] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 364–379.

[7] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th international conference on malicious and unwanted software (MALWARE)*. IEEE, 2015, pp. 11–20.

[8] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," *arXiv preprint arXiv:1308.2831*, 2013.

[9] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, "Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder," *J. Netw. Syst. Manage.*, vol. 30, no. 1, jan 2022. [Online]. Available: https://doi.org/10.1007/s10922-021-09634-4

[10] Ö. Kiraz and İ. A. Doğru, "Visualising static features and classifying android malware using a convolutional neural network approach," *Applied Sciences*, vol. 14, no. 11, p. 4772, 2024.

[11] A. Abusnaina, A. Anwar, S. Alshamrani, A. Alabduljabbar, R. Jang, D. Nyang, and D. Mohaisen, "Systematically evaluating the robustness of ml-based iot malware detection systems," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022, pp. 308–320.

[12] P. Prasse, J. Brabec, J. Kohout, M. Kopp, L. Bajer, and T. Scheffer, "Learning explainable representations of malware behavior," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 53–68.

[13] T.-N. To, H. Do Hoang, P. T. Duy, and V.-H. Pham, "Maldex: An explainable malware detection system based on ensemble learning," in *2023 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*. IEEE, 2023, pp. 1–6.

[14] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole exe," *arXiv preprint arXiv:1710.09435*, 2017.

[15] Y. Mourtaji, M. Bouhorma, and D. Alghazzawi, "Intelligent framework for malware detection with convolutional neural network," in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, 2019, pp. 1–6.

[16] G. D'Angelo, M. Ficco, and F. Palmieri, "Association rule-based malware classification using common subsequences of api calls," *Applied Soft Computing*, vol. 105, p. 107234, 2021.

[17] S. Bose, T. Barao, and X. Liu, "Explaining ai for malware detection: Analysis of mechanisms of malconv," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[18] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 55–56. [Online]. Available: https://doi.org/10.1145/3128572.3140457

[19] Z. Pan, J. Sheldon, and P. Mishra, "Hardware-assisted malware detection using explainable machine learning," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*, 2020, pp. 663–666.

[20] B. Wu, S. Chen, C. Gao, L. Fan, Y. Liu, W. Wen, and M. R. Lyu, "Why an android app is classified as malware: Toward malware classification interpretation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–29, 2021.

[21] M. Q. Li, B. C. Fung, P. Charland, and S. H. Ding, "I-mad: Interpretable malware detector using galaxy transformer," *Computers & Security*, vol. 108, p. 102371, 2021.

[22] M. Melis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli, "Explaining black-box android malware detection," in *2018 26th european signal processing conference (EUSIPCO)*. IEEE, 2018, pp. 524–528.

[23] L. Pirch, A. Warnecke, C. Wressnegger, and K. Rieck, "Tagvet: Vetting malware tags using explainable machine learning," in *Proceedings of the 14th European Workshop on Systems Security*, 2021, pp. 34–40.

[24] J. Jo, J. Cho, and J. Moon, "A malware detection and extraction method for the related information using the vit attention mechanism on android operating system," *Applied Sciences*, vol. 13, no. 11, p. 6839, 2023.

[25] I. A. Khan, N. Moustafa, D. Pi, K. M. Sallam, A. Y. Zomaya, and B. Li, "A new explainable deep learning framework for cyber threat discovery in industrial iot networks," *IEEE Internet of Things Journal*, 2021.

[26] M. Kinkead, S. Millar, N. McLaughlin, and P. O'Kane, "Towards explainable cnns for android malware detection," *Procedia Computer Science*, vol. 184, pp. 959–965, 2021.

[27] M. M. Alani and A. I. Awad, "Paired: An explainable lightweight android malware detection system," *IEEE Access*, vol. 10, pp. 73 214–73 228, 2022.

[28] N. Aslam, I. U. Khan, S. Mirza, A. AlOwayed, F. M. Anis, R. M. Aljuaid, and R. Baageel, "Interpretable machine learning models for malicious domains detection using explainable artificial intelligence (xai)," *Sustainability*, vol. 14, no. 12, p. 7375, 2022.

[29] Z. Lu and V. L. Thing, ""how does it detect a malicious app?" explaining the predictions of ai-based malware detector," in *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2022, pp. 194–199.

[30] M. Saqib, B. C. M. Fung, P. Charland, and A. Walenstein, "GAGE: genetic algorithm-based graph explainer for malware analysis," in *Proceedings of the 40th IEEE International Conference on Data Engineering (ICDE)*. Utrecht, Netherlands: IEEE Computer Society, May 2024, pp. 2258–2270.

[31] J. D. Herath, P. P. Wakodikar, P. Yang, and G. Yan, "CFGExplainer: explaining graph neural network-based malware classification from control flow graphs," in *Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2022, pp. 172–184.

[32] H.-C. Kuo and Y.-J. Lin, "The optimal estimation of fuzziness parameter in fuzzy c-means algorithm," in *Proceedings of the International Joint Conference on Rough Sets (IJCRS)*. Springer, 2017, pp. 566–575.

[33] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE, 2000, pp. 38–49.