Explainable Deep Learning Models for Dynamic and Online Malware Classification

Quincy Card, Daniel Simpson, Kshitiz Aryal, Maanak Gupta
Department of Computer Science
Tennessee Tech University
Cookeville, TN USA
qacard42@tntech.edu, dnsimpson42@tntech.edu, karyal42@tntech.edu, mgupta@tntech.edu

Sheikh Rabiul Islam
Department of Computer Science
Rutgers University
Camden, NJ USA
sheikh.islam@rutgers.edu

2024 IEEE International Conference on Smart Computing (SMARTCOMP) | 979-8-3503-4994-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SMARTCOMP61445.2024.00045

***Abstract*—In recent years, there has been a significant surge in malware attacks, necessitating more advanced preventive measures and remedial strategies. While several successful AI-based malware classification approaches exist—categorized into static, dynamic, or online analysis—most successful AI models lack easily interpretable decisions and explanations for their processes. Our paper aims to delve into explainable malware classification across various execution environments (such as dynamic and online), thoroughly analyzing their respective strengths, weaknesses, and commonalities. To evaluate our approach, we train Feed Forward Neural Networks (FFNN) and Convolutional Neural Networks (CNN) to classify malware based on features obtained from dynamic and online analysis environments. The feature attribution for malware classification is performed by explainability tools, SHAP, LIME and Permutation Importance. We perform a detailed evaluation of the calculated global and local explanations from the experiments, discuss limitations and, ultimately, offer recommendations for achieving a balanced approach.**

***Index Terms*—Explainable AI, Explainable malware analysis, Interpretability, Explainability, SHAP, LIME, Permutation Importance, Dynamic malware classification, Online malware classification, Feature attribution**

I. Introduction

Malware poses a significant cybersecurity threat, requiring effective classification for remediation [1], [2]. Machine learning-based approaches [3] aim to categorize malware types for tailored response plans. Malware analysis is typically categorized into static, dynamic, and online approaches [4]–[8]. Static analysis examines resting malicious files, while dynamic analysis executes malware in a simulated environment. Online analysis, on the other hand, monitors systems in real-time, allowing access to internet resources. Both physical test beds [9] and cloud environments [6] are used for this purpose, although costs can be prohibitive.

Despite the importance of accurate malware classification, understanding the classification processes is equally crucial. Deep learning models, often used for their accuracy, can be challenging to interpret due to their complexity [10]. Explainable AI/ML methods aim to address this challenge by elucidating decisions and identifying significant features contributing to model outcomes. This enhances user trust and aids security analysts in countering malware threats [11].

In this work, we employ explainable techniques to attribute features in deep learning models used for dynamic and online malware classification. We first train Feed Forward Neural Network (FFNN) and Convolutional Neural Network (CNN) models on dynamic and online analysis features to create malware detectors. In the subsequent phase, we utilize Shapley Additive exPlanations (SHAP) [12], Local Interpretable Model-Agnostic Explanations (LIME) [13], and Permutation Importance to interpret black-box model decisions. SHAP assigns feature contributions using cooperative game theory, LIME constructs local interpretable models around specific predictions, and Permutation Importance [14] assesses feature impact through value shuffling for global explanations. We leverage *DeepSHAP* [12] for both global and local interpretations, complemented by Permutation Importance for global insights and LIME for local insights. Our contributions include:

- We evaluate the effectiveness of deep learning models for classifying malware categories from both a dynamic and an online data set.
- We extend this analysis by explaining model predictions on a global level with SHAP and Permutation Importance, and on a local level with LIME and SHAP.

The paper is organized as follows. Section II reviews related works in (a) dynamic analysis, (b) online analysis, and (c) explainability techniques for interpreting model predictions. Section III outlines the methodology and introduces the dynamic and online datasets. Results and model explanations for each dataset are presented in Section IV. The paper concludes with a summary and discussion of future work in Section V.

II. Related Works

This section provides a review of dynamic and online analyses, concluding with a discussion on explainable AI. Table I compares relevant works across attributes like domain, analysis level, models, features, explainability techniques, and malware platform. It is important to clarify terms often conflated in the literature. Detection involves determining malware's presence or absence, similar to binary classification. This work distinguishes detection from classification, where the goal is to differentiate malware samples. Classification groups malware by family or category, with family denoting

2693-8340/24/$31.00 ©2024 IEEE
DOI 10.1109/SMARTCOMP61445.2024.00045

TABLE I: Related works compared. A $\surd$ indicates that a specific paper has this feature or model, and a blank cell shows that this attribute or model does not exist.

	Analysis			Domain		Features					Models Used							XAI Technique							Platform		
Paper	Detection	Category Classification	Family Classification	Dynamic Analysis	Online Analysis	Performance Metrics	API Calls	System Calls	Grayscale Images	Others	Logistic Regression	Decision Tree	Random Forest	LightGBM	XGBoost	Support Vector Machine	Deep Learning	SHAP	LIME	LRP	Grad-CAM	Heatmap	Integrated Gradients	Permutation Importance	Linux	Android	Windows
Schofield et al. (2021) [15]		$\surd$		$\surd$			$\surd$					$\surd$	$\surd$				$\surd$										$\surd$
Keyes et al. (2021) [5]		$\surd$		$\surd$			$\surd$			$\surd$		$\surd$	$\surd$													$\surd$	
Pirch et al. (2021) [16]			$\surd$	$\surd$						$\surd$							$\surd$			$\surd$							$\surd$
Alenezi et al. (2021) [17]		$\surd$		$\surd$				$\surd$					$\surd$		$\surd$		$\surd$	$\surd$								$\surd$	
Kimmel et al. (2021) [18]	$\surd$				$\surd$	$\surd$							$\surd$			$\surd$	$\surd$								$\surd$		
Prasse et al. (2021) [19]		$\surd$	$\surd$		$\surd$					$\surd$			$\surd$				$\surd$						$\surd$				$\surd$
Karn et al. (2021) [20]	$\surd$				$\surd$			$\surd$				$\surd$			$\surd$		$\surd$	$\surd$	$\surd$						$\surd$		
Ullah et al. (2022) [21]		$\surd$		$\surd$					$\surd$	$\surd$	$\surd$	$\surd$	$\surd$			$\surd$		$\surd$									$\surd$	
Naeem et al. (2022) [22]			$\surd$	$\surd$					$\surd$		$\surd$	$\surd$	$\surd$			$\surd$	$\surd$				$\surd$	$\surd$					$\surd$	
Brown et al. (2022) [23]		$\surd$			$\surd$			$\surd$						$\surd$											$\surd$		
Manthena et al. (2023) [6]	$\surd$				$\surd$	$\surd$							$\surd$			$\surd$	$\surd$	$\surd$							$\surd$		
Our Approach		$\surd$		$\surd$	$\surd$		$\surd$			$\surd$							$\surd$	$\surd$	$\surd$					$\surd$		$\surd$	$\surd$

variants sharing traits and category grouping malware by objectives (e.g., ransomware encrypting systems or files).

A. Dynamic Analysis

Several studies focus on dynamic malware analysis in traditional host-based environments. [5] conducted Android malware classification using diverse input features, including memory, API calls, network data, battery status, log writing and process count. Transparent models like Naive Bayes and Decision Tree were employed, as a well as black-box SVM model. [15] classified Windows malware categories using API calls and various transparent and deep learning models, such as a CNN. Despite advancements over static analysis, these approaches also face challenges in analyzing modern advanced malware that may detect they are in a closed environment and evade analysis, prompting the need for online malware analysis methods.

B. Online Analysis

Several studies focus on online malware analysis, addressing the limitations of dynamic analysis by continuously monitoring systems and utilizing various runtime features for machine learning models. In [18], performance metrics were employed to detect malware with six ML classifiers, including a CNN. [23] categorized Linux malware in a cloud environment using the LightGBM model trained on system call n-grams. These studies share the characteristic of conducting online analysis in a cloud environment, chosen for its relative resource efficiency as compared to traditional host-based environments. Almost all of the previously mentioned works used some black-box model but lacked the necessary post-hoc explanations. Notably, our work stands out by combining dynamic and online analysis in a non-cloud, host-based environment, along with explainability methods for interpreting deep learning models in malware classification.

C. Explainable AI

Some machine learning models, termed "transparent models," operate without the need for external or post-hoc explanation methods. Typically, these models, such as tree-based models, offer easy visualization through their tree structure. Conversely, approaches like SVM or neural networks are labeled as black-box methods, where the internal mechanisms are not readily visible or understandable, necessitating additional post-hoc explanations. These explanations can be categorized based on their locality and model-specificity. Local explanations focus on understanding the model's predictions within specific examples, such as classifying individual pixels in an image. In contrast, global explanations provide insights into general patterns, feature importance, and model structure without examining specific inputs [24]. There may be scenarios where only a global or local explanation suffices, but generally, both levels of locality are essential for model interpretability. Model-specific techniques are tailored to a particular model type, such as saliency maps for CNNs, while model-agnostic techniques are applicable across different architectures.

Several studies address machine learning model interpretability in malware analysis. In [16], an interpretable CNN model predicted tags for dynamic malware family categorization, with Layer-wise Relevance Propagation (LRP) used for explanation. [22] developed a dynamic Android malware classification method, employing models such as Random Forest, Decision Tree, SVM, CNN, and Logistic Regression on grayscale images, explained through local, model-specific methods like Grad-CAM and heatmaps. [17] and [21] explored SHAP for explaining malware classification, focusing on dynamic Android malware classification and transformers-based transfer learning, respectively. [19] classified online Windows malware using models like Random Forest, LSTM, CNN, and Transformer, explained by Integrated Gradients for global and local insights. [6] focued on real-time cloud-based malware detection, explaining predictions with SHAP for various black-box models. Karn et al. [20] explain online malware detection,

TABLE II: All classes of Dynamic data set

Category	Number of Samples
Riskware	7261
Adware	5838
Trojan	4412
Ransomware	1861
Trojan_Spy	1801
Trojan_SMS	1028
Trojan_Dropper	837
PUA	665
Backdoor	591
Scareware	462
FileInfector	129
Trojan_Banker	118

particularly cryptominer detection using SHAP for XGBoost predictions and LIME for other models. While these studies emphasize online analysis and explainability, none specifically address explaining both dynamic and online malware category classification in a traditional host-based environment.

III. Methodology

In this section, we discuss the methodology we use for our dynamic analysis, our online analysis, how we evaluate our models, and our approach to explainability.

A. Dynamic Analysis

We utilized the AndMal2020 dataset from the Canadian Center for Cybersecurity [25], comprising 12 Android malware categories with 141 features across 6 types. Addressing highly imbalanced class distribution shown in Table II, we employed SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset. For analysis, we trained both FFNN and CNN models. The FFNN included 6 hidden layers with ReLU activation and Softmax output layer, trained on 80% of the data for 135 epochs with a batch size of 10. The CNN architecture featured convolution, max pooling layers, and two fully connected layers with dropout, trained on 80% of the data for 75 epochs with a batch size of 10.

B. Online Analysis

We used the RaDaR dataset from the Indian Institute of Technology Madras [9], capturing real-time behavior of Windows malware on a physical testbed. This dataset facilitates analysis of modern malware capable of detecting sandbox environments and remaining dormant. However, the extensive resources required for this analysis, along with the larger data volume generated, result in increased computation times for classification and explainability methods. The dataset comprises five malware categories, with 55 features focusing on malware behavior at the hardware level. Like dynamic analysis, the online analysis dataset is highly imbalanced, as seen in Table III, which is addressed using SMOTE to create synthetic samples. Initially considering a Long Short-Term Memory (LSTM) model for its advantages in handling time-series data, we found SHAP less compatible with LSTM's data shape. Hence, we opted for models consistent across both analysis levels. Future work should explore models adept at handling time-series data and suitable explanation methods. For the CNN, the most effective hyperparameters included a convolution layer followed by max pooling, another convolution layer, max pooling, and two fully connected layers. *ReLU* activation was used for all hidden layers, and *Softmax* for the output layer. The model was trained on 80% of the dataset for 75 epochs with a batch size of 50. Similarly, the FFNN comprised 5 hidden layers, with *ReLU* activation, 2 fully connected layers, 1 dropout layer, and *Softmax* activation for the output layer. Trained on 80% of the dataset for 100 epochs with a batch size of 50, both models were tested on the remaining 20% of the online dataset.

TABLE III: All classes of Online data set

Category	Number of Snapshots
Cryptominer	158158
Deceptor	99099
Ransomware	13013
PUA	3003
Backdoor	1001

TABLE IV: Performance Metrics for Dynamic Analysis and Riskware-Specific Metrics

	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
FFNN without SMOTE	80.76	81.23	80.76	81.00
FFNN with SMOTE	90.57	91.00	90.57	90.63
CNN without SMOTE	81.68	81.79	81.68	81.74
CNN with SMOTE	90.04	90.03	90.04	90.03
Riskware - FNN with SMOTE	78.53	100.00	78.53	87.97
Riskware - CNN with SMOTE	81.90	100.00	81.90	90.04

C. Explainability Approach

We chose SHAP as our primary explanation method due to its robust and well-documented Python library. SHAP quantifies feature contributions through Shapley values, providing a mathematical framework to explain model predictions [12]. SHAP offers model-specific explanation methods, such as *DeepExplainer* for deep learning models, and provides explanations at both global and local levels. After model training, we generate an explainer using SHAP's *DeepExplainer*, computing SHAP values for 1000 samples from the test datasets of both dynamic and online analyses. This undersampling is necessary due to the resource-intensive nature of computing SHAP values, especially for complex architectures like CNNs.

LIME and Permutation Importance serve as supplementary explanation methods. LIME offers local explanations, while Permutation Importance provides global explanations. We chose these methods to ensure robust interpretations of the models and to verify the efficiency of SHAP under random sampling. For Permutation Importance, we permuted each feature 30 times for both dynamic and online datasets. For local explanations, we randomly selected misclassified observations from the subsample and generated graphs using SHAP and LIME.

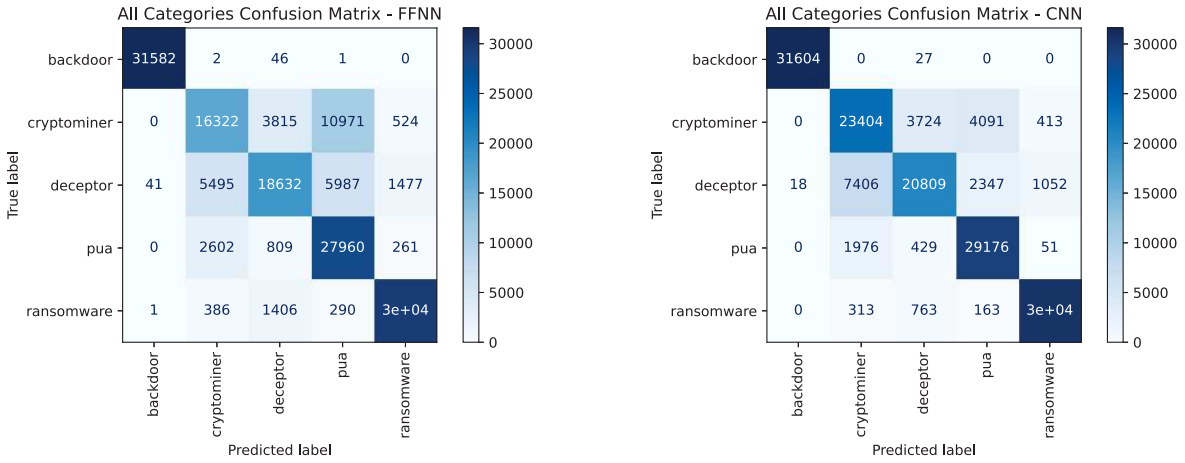


Fig. 1: Performance of models in Online Analysis

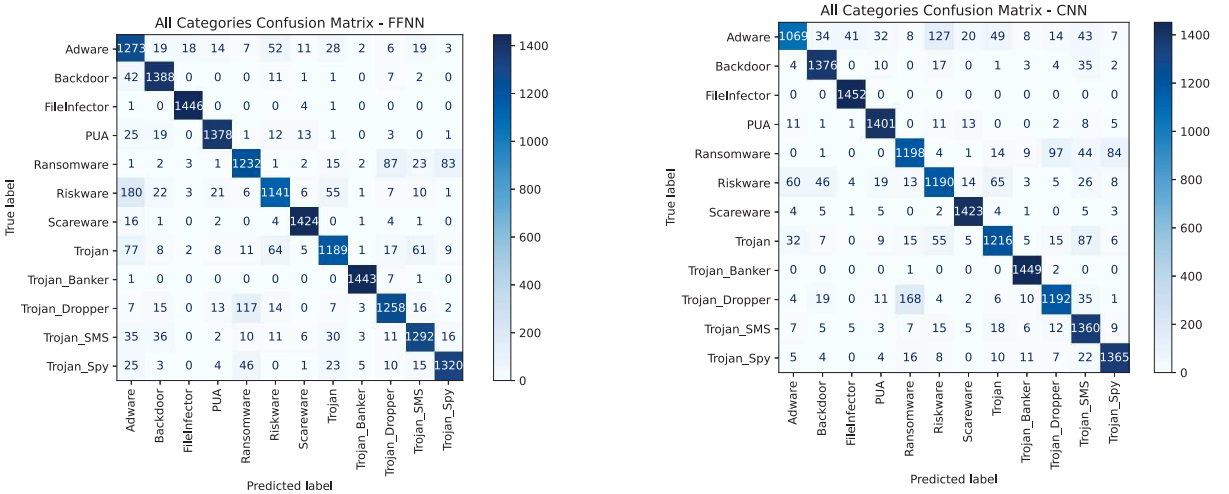


Fig. 2: Performance of models in Dynamic Analysis

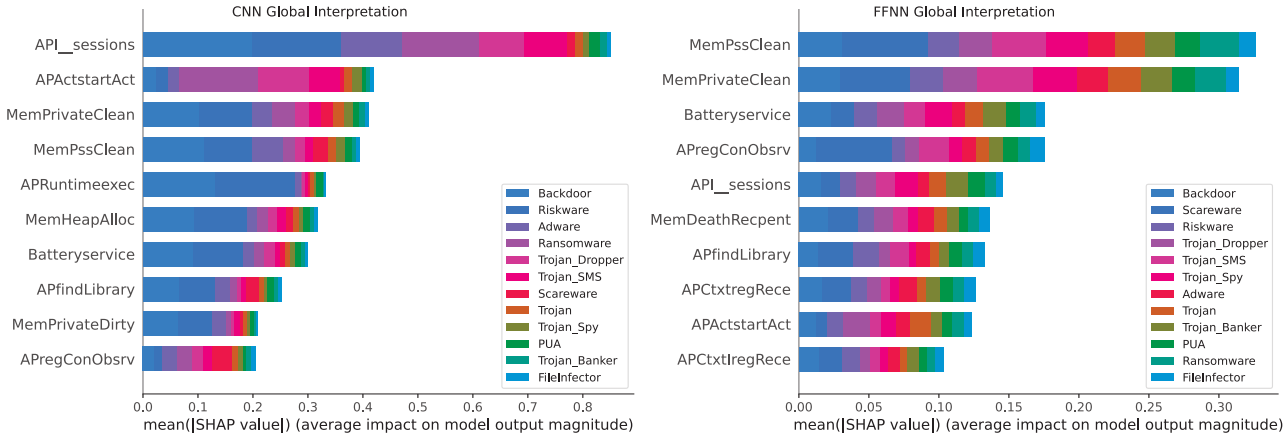


Fig. 3: A stacked bar graph depicting the top 10 online features identified by SHAP in model decision making

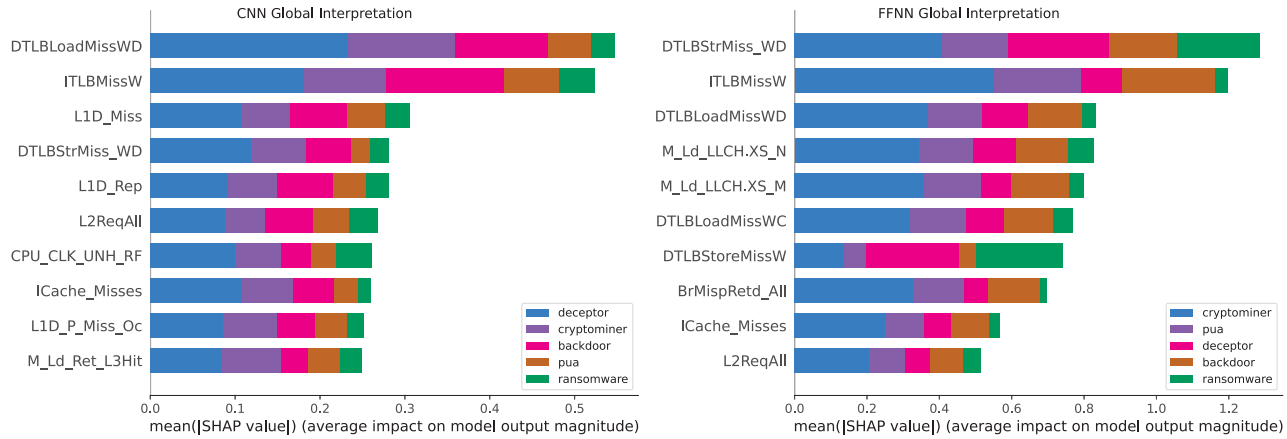


Fig. 4: A stacked bar graph depicting the top 10 online features identified by SHAP in model decision making

TABLE V: Performance Metrics for Online Analysis and Deceptor-Specific Metrics

	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
FFNN without SMOTE	77.45	77.15	77.45	77.30
FFNN with SMOTE	78.43	79.16	78.43	78.79
CNN without SMOTE	79.72	80.50	79.72	80.11
CNN with SMOTE	85.60	85.65	85.60	85.63
Deceptor - FNN with SMOTE	58.90	100.00	58.90	74.14
Deceptor - CNN with SMOTE	65.78	100.00	65.78	79.36

TABLE VI: Top 10 Dynamic Features Identified by Permutation Importance

FFNN Features (Importance)	**CNN Features (Importance)**
MemPssClean (0.236)	APCtxtregRece (0.178)
MemPrivateClean (0.191)	APregConObsrv (0.158)
APregConObsrv (0.140)	MemPssClean (0.155)
APCtxtregRece (0.115)	MemPrivateClean (0.132)
Batteryservice (0.110)	APfindLibrary (0.105)
MemDeathRecpent (0.108)	MemDeathRecpent (0.103)
APCtxtIregRece (0.103)	API__sessions (0.083)
APActstartAct (0.102)	APActstartAct (0.078)
API__sessions (0.086)	MemParcelCount (0.072)
APfindLibrary (0.079)	MemHeapAlloc (0.069)

TABLE VII: Top 10 Online Features Identified by Permutation Importance

FFNN Features (Importance)	**CNN Features (Importance)**
L1D_P_Miss_Oc (0.150)	ICache_Misses (0.188)
DTLBLoadMissWD (0.113)	DTLBLoadMissWD (0.181)
DTLBStoreMissW (0.112)	ITLBMissW (0.143)
DTLBLoadMiss_W (0.097)	L2ReqAll (0.138)
ITLBMissW (0.096)	M_Ld_LLCH.XS_N (0.132)
L2ReqPFms (0.090)	L1D_P_Miss_Oc (0.131)
Core_cyc (0.086)	L1D_Rep (0.130)
L1D_Rep (0.080)	Ref_cyc (0.103)
DTLBLoadMissWC (0.077)	L2ReqPFms (0.098)
BrMispRetd_All (0.069)	L1D_Rep (0.097)

IV. Results and Discussion

A. Evaluation of Performance Metrics

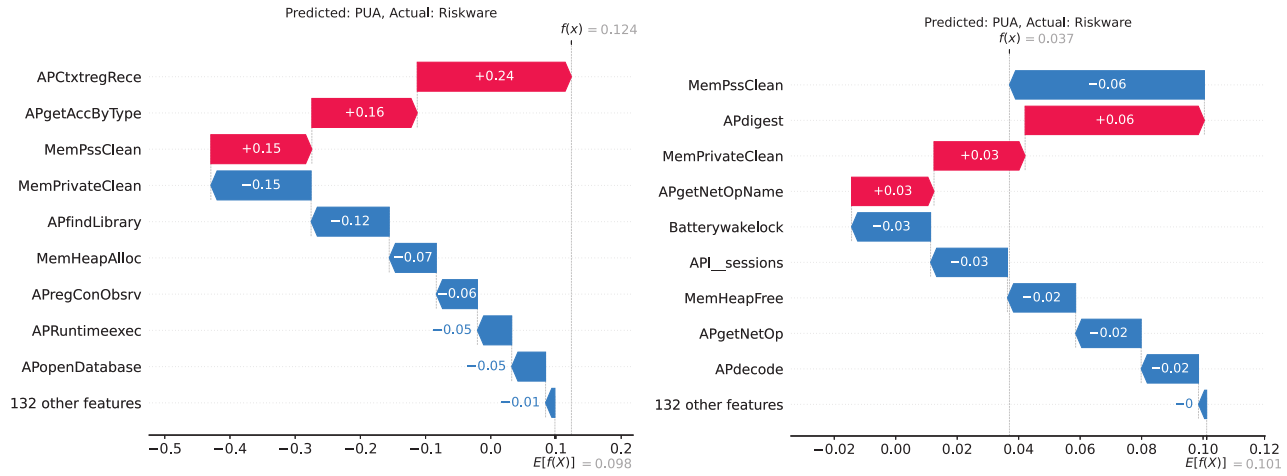
Table IV provides performance metrics for overall model performance in dynamic analysis, including a breakdown for one of the worst-performing classes. Utilizing the F1 score, a comprehensive evaluation beyond accuracy, the FFNN model achieved 90.63%, while the CNN model reached 90.03%. The comparison with and without SMOTE intervention indicates a notable performance increase with synthetic samples. Despite Riskware being the majority class in the unaltered dataset, its relative poor performance suggests potential overlap between SMOTE's synthetic samples for minority classes and the decision boundary of majority classes. Figure 2 suggests misclassification of majority class samples as minority classes, but the overall model improvement with SMOTE justifies this cost.

Table V presents performance metrics for the online analysis before and after SMOTE intervention. The FFNN achieved an F1 score of 78.79%, while the CNN achieved 85.63%. The CNN outperformed the FFNN, possibly due to its complexity, yet both models underperformed compared to those in the dynamic analysis. This could be attributed to the loss of time-series data and the introduction of synthetic samples via SMOTE, as depicted in Figure 1. We focused our explanation analysis on the Riskware class for the dynamic dataset and the Deceptor class for the online dataset due to their low F1 scores, indicating the need for further investigation. More analyses for other classes are available on our GitHub repository[1].

B. Global Explanation

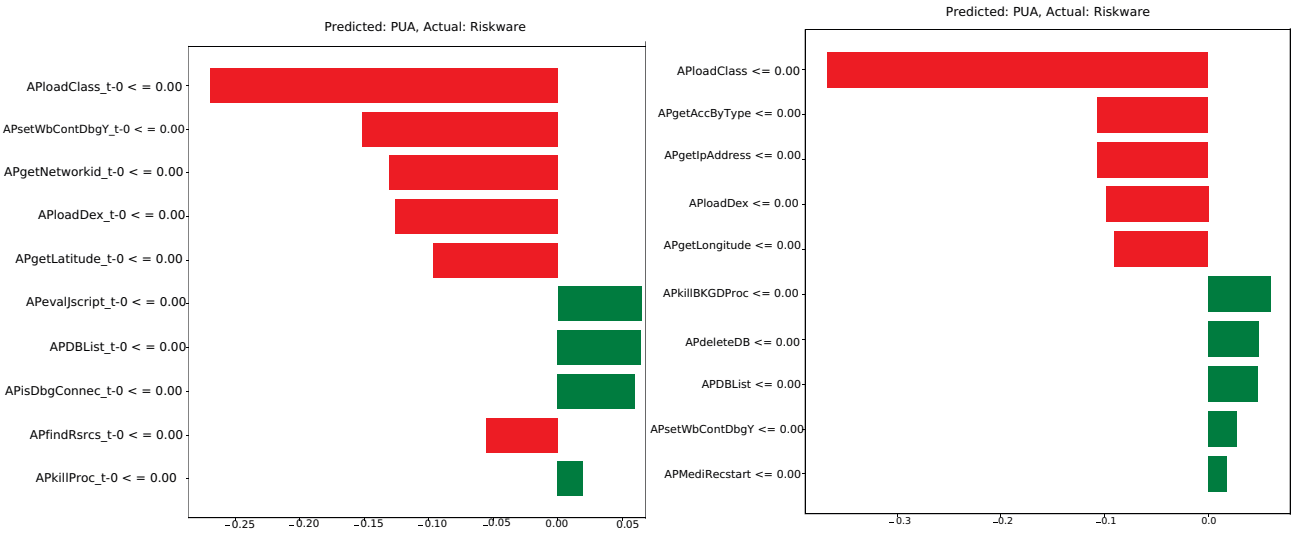
With reduced sample sizes, SHAP's *DeepExplainer* computed SHAP values in about 6 minutes for the CNN and 16 seconds for the FFNN on the dynamic dataset. The summary plots in Figure 3 illustrate feature importance, with features arranged by their effects' magnitudes across all classes. Notably, API calls and Memory features were top features for

[1]GitHub: https://github.com/SecurityCard/explainability-graphs.git



(a) CNN (b) FFNN

Fig. 5: Waterfall plots - Local interpretations of misclassified Riskware sample of dynamic data set

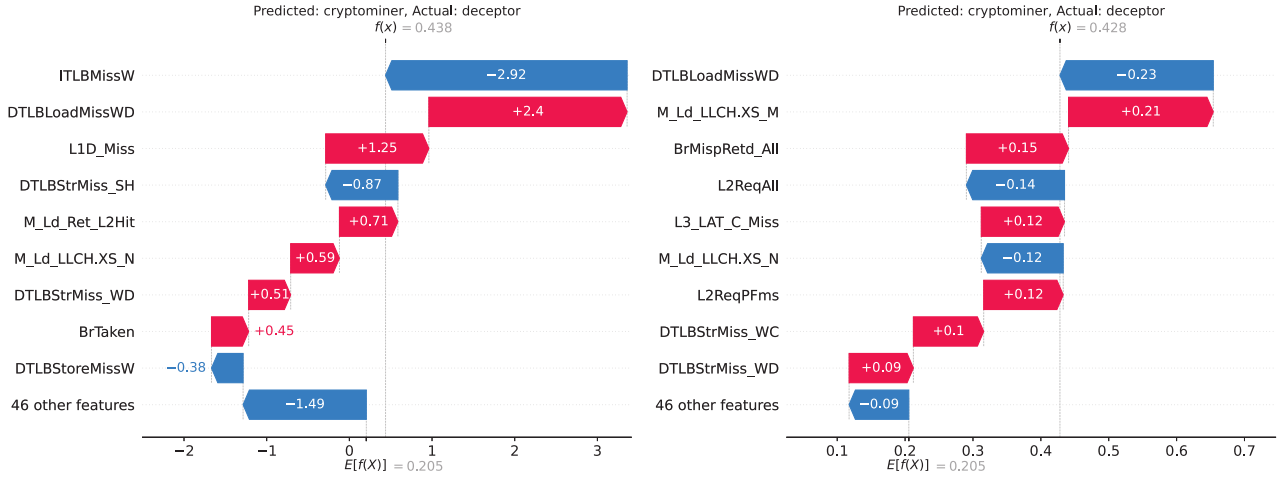


(a) CNN (b) FFNN

Fig. 6: LIME plots - Local interpretations of misclassified Riskware sample of dynamic data set

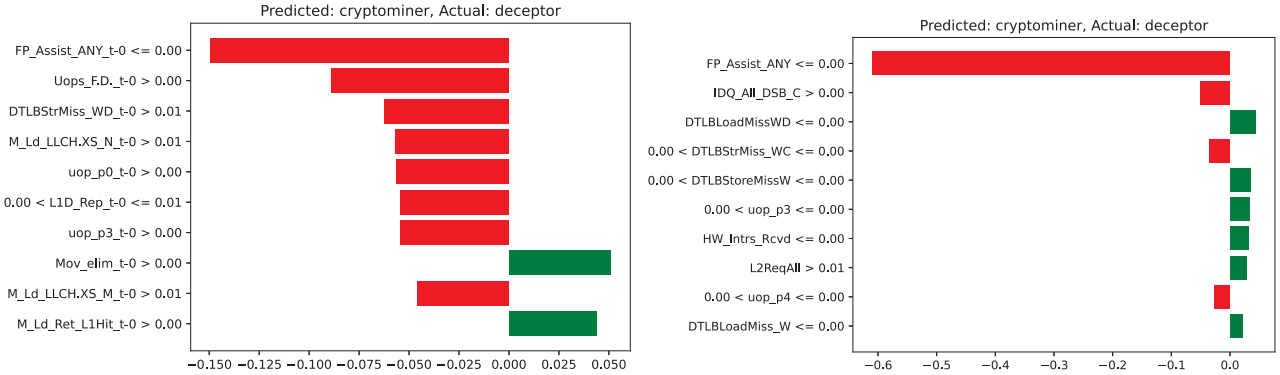
both models, with *API__sessions* particularly significant for the CNN model. Permutation Importance, validating SHAP's effectiveness, identified top features after permutation. The FFNN runtime was just under an hour, and for the CNN, two hours. Table VI lists top features and their mean importance when permuted, with 10 features shared with SHAP for the FFNN and 7 for the CNN, confirming a balanced trade-off between computation time and SHAP accuracy.

For the online dataset, *DeepExplainer* computed SHAP values in about 92 seconds for the CNN and 4 seconds for the FFNN. Permutation Importance took 40 minutes for the FFNN and an hour for the CNN. Figure 4 and Table VII reveal that among the globally identified top 10 features by Permutation Importance, 5 are within the top 10 identified by SHAP for the FFNN, and 6 for the CNN, indicating a balanced trade-off between computational accuracy and resource costs. Additionally, 4 features were shared among those identified by SHAP for both models, suggesting robustness in the online analysis models. However, this might not be as pronounced as in the dynamic analysis due to limitations with SHAP's handling of time-series-based neural networks. In real-world scenarios, our methodology of employing diverse explainability methods on an under-sampled dataset remains effective. The model-agnostic nature of our chosen explainability techniques allows for flexible and generalizable application across various model architectures tailored to specific use cases. Additionally, our



(a) CNN (b) FFNN

Fig. 7: Waterfall plots - Local interpretations of misclassified Deceptor sample of online data set



(a) CNN (b) FFNN

Fig. 8: LIME plots - Local interpretations of misclassified Deceptor sample of online data set

under-sampling approach mitigates the anticipated increase in dataset size often encountered in real-world implementations.

C. Local Explanation

For local explanations, SHAP generated waterfall plots, focusing on misclassified samples for the classes with the lowest F1 score. Figure 5 displays the waterfall plots for a dynamic sample misclassified by both models, showcasing SHAP values for each feature. Positive contributions (in pink) push the probability towards the predicted class, while negative contributions (in blue) push it away. The sum of SHAP values for each sample represents the difference between the base value $E[f(x)]$ at the bottom and the final output value $f(x)$ at the top. For Figure 5, the final output values indicate probabilities for the predicted class (PUA) and not the actual class (Riskware), being 0.124 and 0.037 for the CNN and FFNN, respectively. These scores while still positive, indicating the result of being classified as PUA, are not large quantities, meaning the classifier decision is not as certain. Once again the majority of the features are either of the Memory or the API types, with the respective model explanations even sharing two features, which are also globally important to model predictions. Figure 6 shows LIME graphs for the same misclassified sample, providing different feature importance compared to SHAP. Green bars (right-directed) indicate positive contributions towards the predicted class, while red bars (left-directed) suggest negative contributions for all classes not predicted. The LIME explanations indicate most of the decision to misclassify was due to negative feature contribution, supporting our conclusion from the waterfall plot analysis about the classifier decision not being certain. The top 10 features identified by LIME are all of the API type. This further indicates that it may be possible to increase model performance by removing the features that are less relevant to model decision making, which has significant implications for future research and for increasing the effectiveness of

real-world remediation strategies. Our methodology enhances efficiency in malware classification by pinpointing crucial analysis areas and fostering trust in black-box model decisions, which would otherwise be inscrutable. This enables focused analysis and improves overall classification efficacy.

For online analysis, Figure 7 displays waterfall plots for a misclassified Deceptor sample, showing similar characteristics to dynamic local explanations with classifier decision not being as certain and many features having equal and opposite contributions to each other. Figure 8 depicts LIME graphs for the same misclassified sample, with shared top two features providing strong negative impacts on classifying this sample as not the other classes. By combining the locality specific explainability methods of Permutation Importance and LIME to the both global and local method of SHAP, we have effectively reached an even deeper understanding and interpretation of these black-box models for both dynamic and online malware classification. This method of combining different explanaibility methods has significant implications for future works that even just choose to hone in on dynamic or online malware category classification.

V. Conclusion and Future Work

This paper applies FFNN and CNN models to dynamic and online malware datasets for classification, using SHAP, LIME, and Permutation Importance for explanations. This approach balances resource costs and analysis depth. Despite class imbalance in the dynamic dataset, SMOTE partially mitigates this issue, though with potential performance degradation. Another limitation arises from explainability methods unsuitable for time-series data. Future research should explore this limitation and evaluate model performance on diverse datasets.

Future work aims to investigate adversarial attacks' potential exploitation by malicious users to misclassify malware categories. By focusing on features of highest importance identified by explainability methods, our findings can contribute to creating more efficient datasets, facilitating real-world malware remediation and aiding cyber-analysts in coordinating responses to threats.

Acknowledgment

This work is supported by the NSF Scholarship for Service Program Award 2043324, 2025682 and 2230609.

References

[1] K. Aryal, M. Gupta, and M. Abdelsalam, "Analysis of label-flip poisoning attack on machine learning based malware detector," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 4236–4245.

[2] K. Aryal, M. Gupta, M. Abdelsalam, and M. Saleh, "Intra-section code cave injection for adversarial evasion attacks on windows pe malware file," *arXiv preprint arXiv:2403.06428*, 2024.

[3] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2016, pp. 577–582.

[4] A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. Gagnon, and F. Massicotte, "DIDroid: Android malware classification and characterization using deep image learning," in *10th International Conference on Communication and Network Security (ICCNS2020)*, Tokyo, Japan, November 2020, pp. 70–82.

[5] D. S. Keyes, B. Li, G. Kaur, A. H. Lashkari, F. Gagnon, and F. Massicotte, "Entroplyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics," in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 2021, pp. 1–12.

[6] H. Manthena, J. Kimmel, M. Abdelsalam, and M. Gupta, "Analyzing and explaining black-box models for online malware detection," *IEEE Access*, vol. 11, pp. 25 237–25 252, 2023.

[7] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Computers & Security*, 2024.

[8] K. Aryal, M. Gupta, and M. Abdelsalam, "A survey on adversarial attacks for malware analysis," *arXiv preprint arXiv:2111.08223*, 2022.

[9] S. Karapoola, N. Singh, C. Rebeiro, and K. V., "RaDaR: A real-world dataset for ai powered run-time detection of cyber-attacks," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 3222–3232.

[10] P. Hall and N. Gill, *An Introduction to Machine Learning Interpretability*. O'Reilly Media, Incorporated, 2019.

[11] A. Blanco-Justicia and J. Domingo-Ferrer, "Machine learning explainability through comprehensible decision trees," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2019, pp. 15–26.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] 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.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, p. 5–32, 2001.

[15] M. Schofield, G. Alicioglu, R. Binaco, P. Turner, C. Thatcher, A. Lam, and B. Sun, "Convolutional neural network for malware classification based on api call sequence," in *8th International Conference on Artificial Intelligence and Applications (AIAP)*, 2021, pp. 85–98.

[16] 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.

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

[18] J. C. Kimmell, M. Abdelsalam, and M. Gupta, "Analyzing machine learning approaches for online malware detection in cloud," in *2021 IEEE International Conference on Smart Computing*, 2021, pp. 189–196.

[19] P. Prasse, J. Brabec, J. Kohout, M. Kopp, L. Bajer, and T. Scheffer, "Learning explainable representations of malware behavior," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*, 2021, p. 53–68.

[20] R. R. Karn, P. Kudva, H. Huang, S. Suneja, and I. M. Elfadel, "Cryptomining detection in container clouds using system calls and explainable machine learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 674–691, 2021.

[21] F. Ullah, A. Alsirhani, M. M. Alshahrani, A. Alomari, H. Naeem, and S. A. Shah, "Explainable malware detection system using transformers-based transfer learning and multi-model visual representation," *Sensors*, vol. 22, no. 18, p. 6766, 2022.

[22] H. Naeem, B. M. Alshammari, and F. Ullah, "Explainable artificial intelligence-based iot device malware detection mechanism using image visualization and fine-tuned cnn-based transfer learning model," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[23] P. Brown, A. Brown, M. Gupta, and M. Abdelsalam, "Online malware classification with system-wide system calls in cloud iaas," in *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*, 2022, pp. 146–151.

[24] Y. Lin and X. Chang, "Towards interpreting ml-based automated malware detection models: A survey," *arXiv preprint arXiv:2101.06232*, 2021.

[25] Canadian Center for Cyber Security. (2020) CCCS-CIC-AndMal2020. [Online]. Available: https://www.unb.ca/cic/datasets/andmal2020.html