Z-MDZS: Zero-day Malware Detection using Zero-Shot Machine Learning Schemes

Attaullah Buriro*, Flaminia Luccio*, Gabriele Costa‡ and Riccardo Focardi*
* Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University, Venezia, Italy
Email: {attaullah.buriro, luccio, focardi}@unive.it
† Systems Security Modelling and Analysis (SySMA), IMT School for Advanced Studies, Lucca, Italy
Email: gabriele.costa@imtlucca.it

***Abstract*—Zero-day malware is a serious cybersecurity concern since it can evade detection techniques using trained and expert systems. In this paper, we propose Z-MDZS - a scheme to effectively identify zero-day malware using a zero-shot[1] machine learning approach. Our objective is to detect previously unseen malware based on its properties and relationships to known malware variants, by applying zero-shot learning methods. We evaluate the effectiveness of Z-MDZS, using different machine learning methods, including Random Forest, Deep Neural Networks, and Convolutional Neural Networks. Our results demonstrate that even with smaller feature sets, the zero-shot ML strategy yields solid results, particularly when Random Forest is used as the classifier. Furthermore, we discovered that balancing class samples using Generative Adversarial Network greatly increases classifier accuracy, highlighting its significance.**

***Index Terms*—Malware, Zero-day, Generative Adversarial Network, Deep Learning**

I. Introduction

The rapid advancement of technology has exposed an increasing number of interconnected computer systems to various cybersecurity threats. Among the most serious threats, there is malware, or malicious software, which performs harmful actions when executed on targeted computer systems. Malware is constantly evolving, developing new and often surprising behaviors, posing a significant challenge for the digital industry [1].

Detecting malware is a complex and challenging task due to its continuous evolution and adaptation to new technologies and environments [2]. Traditional detection methods, such as signature-based [3] or behavior-based techniques [4]–[6], often fail to detect new and evolving threats [7]. These methods are particularly ineffective against unknown (zero-day) malware variants, which can bypass traditional detection mechanisms and cause severe damage or loss before they are discovered and countered [8]. As existing malware detection methods face several limitations and difficulties, there exists a dire need for novel and innovative approaches [9].

Machine Learning (ML) algorithms have shown to be highly effective in detecting malware due to their ability to analyze vast amounts of data and identify patterns that may not be detectable through traditional methods [10]. However, commonly used evaluation methods often lead to overfitting and are not realistic in the context of zero-day and zero-shot learning scenarios [11], [12]. It is crucial to design evaluation methods that mimic real-world conditions and leverage zero-shot learning principles to ensure the robustness and reliability of ML-based malware detection systems.

In this study, we present Z-MDZS, a machine learning-based scheme for zero-day malware detection using zero-shot learning principles. Our evaluation involved three advanced classifiers: Random Forest (RF), Deep Neural Network (DNN), and Convolutional Neural Network (CNN), using the Blue Hexagon Open Dataset for Malware AnalysiS (BODMAS) [13], comprising 134,000 samples of benign and malicious Portable Executable (PE) files. PE files are the standard format for executable files on Windows operating systems and contain various information and metadata about the executable, such as headers, sections, imports, exports, resources, and certificates [14]. The BODMAS dataset includes malware samples from 581 families, with the number of samples per family ranging from one to 4,729. We selected 57 families with more than 100 samples each for our experiments and employed Generative Adversarial Networks (GANs) to synthesize additional data, resulting in a balanced dataset with 4,729 samples per family. We achieved accuracy rates of 78.73%, 81.27%, and 86.88% on original samples and 97.63%, 90.3%, and 92.81% on augmented samples using as few as 25 features selected using Sequential Forward Feature Selection (SFS), Correlation Attribute Evaluator (CAE) and Information Gain Attribute Evaluator (IGAE), respectively, and adapting RF as the classifier.

In summary, the main contributions of this work are the following:

- The introduction of Z-MDZS, a machine learning-based scheme for zero-day malware detection using zero-shot learning principles. Unlike traditional methods, Z-MDZS utilizes completely unseen samples for testing, reflecting real-world scenarios.
- The evaluation of Z-MDZS using three advanced classifiers: RF, DNN, and CNN. The evaluation is conducted on the BODMAS dataset, which includes 134,000 samples

[1]Zero-shot learning (ZSL) in machine learning refers to the ability of a model to correctly predict malware families, it has never encountered during its training phase.

of benign and malicious PE files.
- The evaluation of Z-MDZS on subsets of features selected using three conceptually different feature selection schemes.
- The utilization of GANs for data synthesis. By generating high-quality synthetic data, GANs enhance the training process, leading to improved accuracy of the classifiers.

II. Related Work

In this section, we review the most relevant papers in the field related to our proposal.

A. ML-based Malware Detection Schemes

This study by Mills et al. [15] introduces NODENS, a lightweight malware detection system that can detect previously unseen malware. Additionally, it can be deployed on affordable hardware such as a Raspberry Pi. Using RF as the classifier, the study claims to provide interpretability without sacrificing accuracy or detection runtime, and reports an average detection speed of 3-8 seconds. Authors conduct the evaluation using a time-based split to reflect real-world scenarios, rather than cross-validation or random splits.

Vadrevu et al. [16] introduce AMICO, a system for measuring and detecting malware downloads in live web traffic. AMICO learns to distinguish between malware and benign file downloads based on the download behavior of network users. The download behavior was profiled based on network-related features such as Server IPs, URLs, and domain-related features. AMICO was deployed at the edge of an academic network for almost nine months and continuously recorded hundreds of new malware downloads per week. The evaluation was conducted using a labeled dataset of past benign and malware file downloads, and the system achieved up to 90% True Positives Rate (TPR) at a False Positives Rate (FPR) of just 0.1%.

In our previous study [17], we proposed MalwD&C, a machine learning-based approach for malware detection and categorization. MalwD&C scheme focuses on the secure installation of PE files. The proposed method uses multiple ML classifiers to analyze PE files and classify them as benign or malware. The evaluation was conducted on the same BODMAS dataset in two settings: two-class classification (malware detection) and multi-class classification (malware categorization). The RF classifier outperformed all other chosen classifiers, and achieved 99.56% and 97.69% accuracies in the two-class and multi-class settings, respectively.

Deep Learning has also been exploited in Malware analysis domain [18] [19]. The study [18] proposes an Android malware detection system using deep learning techniques to address the growing threat of Android malware. The system leverages deep neural networks to analyze and classify applications as benign or malicious. The study [19] introduces two self-attention transformer-based classifiers, SeqConvAttn and ImgConvAttn, designed to replace traditional CNN-based classifiers for real-time malware classification. The evaluation was conducted on the Microsoft Malware Classification Challenge (BIG 2015) and subsets from the BODMAS dataset. The study demonstrates that self-attention mechanisms can enhance classification accuracy while maintaining low inference latency, making them suitable for real-time applications.

In summary, the existing literature on malware detection highlights a variety of approaches, including the use of traditional ML classifiers, deep learning techniques, and innovative methods like downloader graph analytics and memory analysis. However, many of these studies rely on conventional random train/test splits or cross-validation methods, which may not accurately reflect real-world scenarios where malware evolves over time. Additionally, data augmentation and the issue of data skewness are often not explicitly addressed, potentially limiting the generalizability of the models.

B. Data Synthesis using GAN

The use of GANs for data augmentation in the malware detection domain has been comparatively underexplored. The following studies highlight the potential of GANs and other generative models in creating robust, efficient, and accurate malware detection systems [20]–[22]. The study by Burks et al. [20] explores the use of GANs and Variational Autoencoders (VAEs) for data augmentation, revealing that GAN-generated samples improve the accuracy of a Residual Network (ResNet-18) classifier by 6%, while VAE-generated samples enhance accuracy by 2%.

In another study, Stalin et al. [21] employed Wasserstein GANs (WGANs) to generate synthetic data, which was then used to train a CNN. This approach not only reduced storage demands by creating image representations of features but also improved detection performance by 1.5% to 7%, achieving an F1 score of 0.9751. Furthermore, Gupta et al. [22] combined Long Short-Term Memory (LSTM) networks with GANs, leveraging the VirusShare dataset to enhance detection accuracy to 98%. This method also explored ensemble learning to reduce biases and increase model complexity.

Our proposed approach, Z-MDZS, represents a significant advancement in the field of malware detection by employing a zero-day and zero-shot learning framework, ensuring that the models are adept at detecting previously unseen malware. By utilizing GANs for data synthesis, we effectively address the challenge of data scarcity and variability by balancing the dataset. Furthermore, our evaluation using advanced classifiers, such as DNN and CNN, on a larger dataset of 134k samples from the BODMAS dataset (without transforming these samples to images) demonstrates considerable performance improvements, highlighting the effectiveness of our method in detecting zero-day malware.

III. Our Proposal of Z-MDZS

The diagram in Figure 1 depicts our Z-MDZS approach for zero-day malware detection. The process starts with the acquisition of PE files from the BODMAS dataset, encompassing both benign and malicious files, with the latter identified as known malware. Our approach involves feature extraction

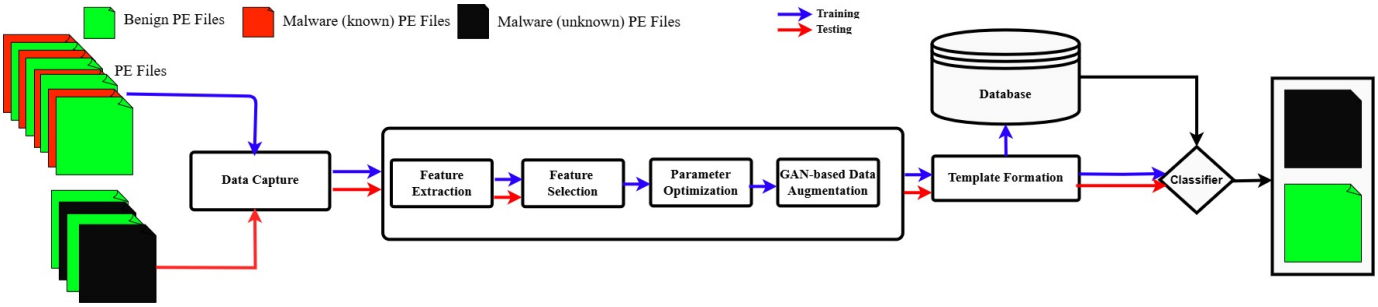


Fig. 1: Building blocks of our approach.

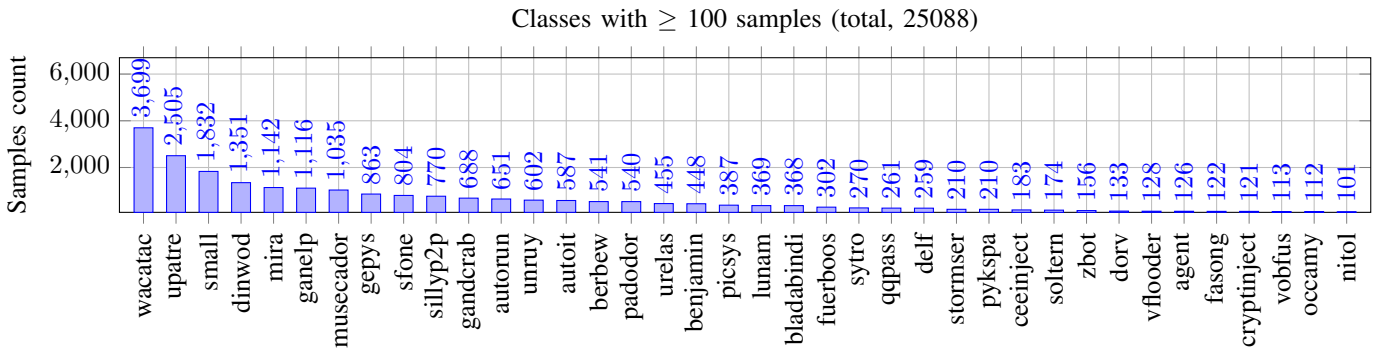


Fig. 2: Distribution of samples of malware families in first 8 months

utilizing the LIEF[2] library, followed by feature selection to reduce the initial feature vector from 2381 features to a more manageable 25-feature vector. Further, our approach involves classifier parameter optimization, as both the DNN and CNN architectures necessitate parameter tuning. Subsequently, data synthesis is performed to enhance samples from minority classes, ensuring that the decisions of the classifier are not biased towards majority classes. The optimized classifiers are then tested on the same extracted and selected features from unseen and benign samples to understand the benign or malicious nature of a testing file. It is worth noting that we conduct classifiers evaluation both before and after data augmentation to comprehend the effectiveness of our developed GAN.

IV. Experimental Evaluation

In this section, we present our experimental results.

A. Dataset

We used the BODMAS [13] to evaluate our approach. The dataset comprises of 57,293 malware samples from 581 families and 77,142 benign files, collected between August 2019 and September 2020. This dataset includes binaries (disarmed malware only), feature vectors, and metadata. Each observation is represented as a 2381-feature vector, accompanied by its label (benign or malicious) and the malware family in metadata file.

We combined 25088 malware samples (created by combining all malware samples as depicted in Figure 2) collected in first 8 months, and 25k benign samples to create a custom dataset and use it for feature selection and parameter optimization only. Note that the we used the full length feature vector (2381-features long) for feature selection and parameter optimization.

B. Features Extraction

We exploit the same set of features that were originally extracted by the authors of the BODMAS [13] and EMBER [23] datasets using LIEF [3] library, each 2381 features long, to construct feature vectors for training the selected ML models.

C. Features Selection

Feature subset selection involves identifying the most relevant features for training ML models. This technique serves at least two main purposes: it reduces the training time of the selected classifiers compared to using the full set of features, and it lowers computational costs without sacrificing accuracy.

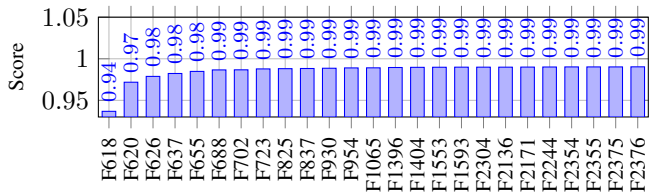
In this study, we exploit the following three well-known feature selection schemes:

- **Sequential forward feature selection**[4] **(SFS)**: is a technique used to iteratively add features to an empty set, and selecting the feature that maximizes model performance at each step. This process continues until adding more features does not improve the model's performance. We manage to extract the same set of top 25 features, as shown in Figure 3a to an earlier study [17].

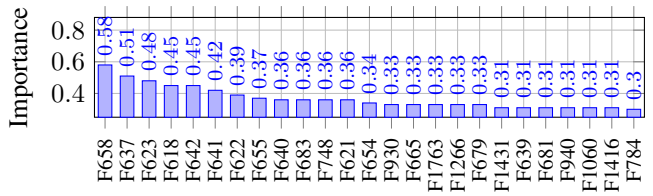
[2]https://github.com/lief-project/LIEF/releases

[3]https://github.com/lief-project/LIEF

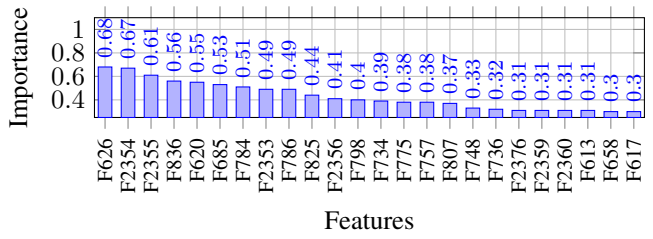
[4]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html



(a) Top 25 SFS features [17]



(b) Top 25 CAE features



(c) Top 25 IGAE features

Fig. 3: Comparison of Top 25 features selected by different methods

- **Correlation Attribute Evaluator**[5] **(CAE)**: assesses the value of an attribute by calculating the Pearson's correlation coefficient between that attribute and the class label. Attributes with higher correlation coefficients are considered more relevant for the classification task. In Figure 3b, we show the importance of our top 25 CAE selected features.
- **Information Gain Attribute Evaluator**[6] **(IGAE)**: Information Gain is a feature selection method that measures the reduction in entropy or uncertainty of the target variable when a particular feature is known. It quantifies how much information a feature provides about the class, with higher values indicating more informative features. Figure 3c lists our top 25 selected IGAE features.

D. Classifier Selection and Parameter Optimization

Classifiers are essential machine learning models or algorithms that are designed to learn from data and assign labels to new samples. In our approach, we chose RF, DNN, and CNN as our classifiers due to their strong performance in previous studies [15], [17], [24]. To ensure the efficient use of these models as our classifiers, we conducted parameter optimization. Optimizing a classifier's parameters involves identifying the optimal values that enhance the classifier's performance. Various methods exist for parameter optimization, including grid search, random search, cross-validation, Bayesian optimization, evolutionary algorithms, and meta-learning.

Our parameter optimization process covered a range of parameters, as detailed in Table I. The subset of parameters with the best validation accuracy were selected for the final model creation. It is important to note that we used the same customized dataset for both feature selection and parameter optimization.

E. Data Augmentation using GAN

Introduced by Goodfellow et al. [25], GANs are an unsupervised method for generating synthetic data. This technique has been widely explored across various fields, however, the generation of one-dimensional (1D) tabular data, particularly in malware detection [20] [21] [22], has received limited attention.

Our implementation of a GAN used the Python programming language and the Keras library[7]. Technically speaking, in this work, the developed generator and discriminator networks are compact neural networks. The generator comprises seven hidden layers with 512, 512, 192, 96, 448, 352 and 224 units, respectively, while the discriminator consists of same number of hidden layers with 224, 352, 448, 96, 192, 512 and 512 units. We experimented with various configurations of hidden layers and units in the generator to determine this optimal setup, which demonstrated excellent results in generating synthetic samples.

The generator's first layer accepts a fixed-size (200-dimensional) noise vector, and its final output is a 25-dimensional vector that closely resembles the 25-dimensional vector of "real" feature selected samples. These 25-dimensional vectors from both "real" and "non-real" samples are then fed into the discriminator, whose output is binary, indicating whether the sample is "real" or "non-real".

We considered only the 57 classes with more than 100 samples. Since the *Sfone* class contains a maximum of 4729 samples, we generated additional samples for all other classes to balance the dataset. For instance, if class X had 100 samples, we used GAN to generate 4629 additional samples. This process was repeated for different feature sets, resulting in three versions for each class.

F. Classification Protocol

The BODMAS dataset comprises malware samples from 581 families, with sample sizes per family ranging from 1 to 4,729. For our study, we selected 57 families containing more than 100 samples each. This selection allowed us to create a training set consisting of 38,571 benign samples (labeled as 0) and known malware samples (all malware families except the excluded one, labeled as 1). To construct the test set, we

[5]https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CorrelationAttributeEval.html

[6]https://machinelearningmastery.com/information-gain-and-mutual-information/

[7]https://keras.io/

TABLE I: Parameter optimization of all chosen classifiers.

Classifiers	*Parameters and their range*		*Best*	Best validation Accuracy (%)
RF	*n_estimators*	$\{50i \mid 1 \leq 1 \leq 20\}$	450	86.79
	bootstrap	$\{True, False\}$	$True$	
	max_depth	$\{5, 10, 15, \infty\}$	∞	
DNN	*num_layers*	$\{2, \ldots, 10\}$	7	89.56
	num_units	$\{32i \mid 1 \leq i \leq 16\}$	$512, 512, 192, 96, 448, 352, 224$	
	learning_rate	$\{10^{-i} \mid 2 \leq i \leq 4\}$	10^{-4}	
CNN	*filters*	$\{32, 64, 128\}$	*128*	89.16
	kernel_size	$\{3, 5\}$	3	
	pool_size	$\{2, 3\}$	*2*	
	dense_units	$\{32i \mid 1 \leq i \leq 4\}$	*128*	
	dropout_rate	$\{0.1i \mid 1 \leq i \leq 5\}$	0.2	

combined the remaining 38,571 benign samples (labeled as 0) with all samples from the excluded family and the combined samples of all families with fewer than 100 samples (labeled as 1). Our experiments were conducted in 57 iterations, with each iteration excluding one malware family from the training set and combining it with the samples of all families having fewer than 100 samples (unknown malware) to form the test set. This approach ensures that the model is tested on its ability to detect previously unseen malware classes, adhering to the principles of zero-shot learning. By iterating through each malware family, we thoroughly evaluated the model's performance across a diverse set of malware families.

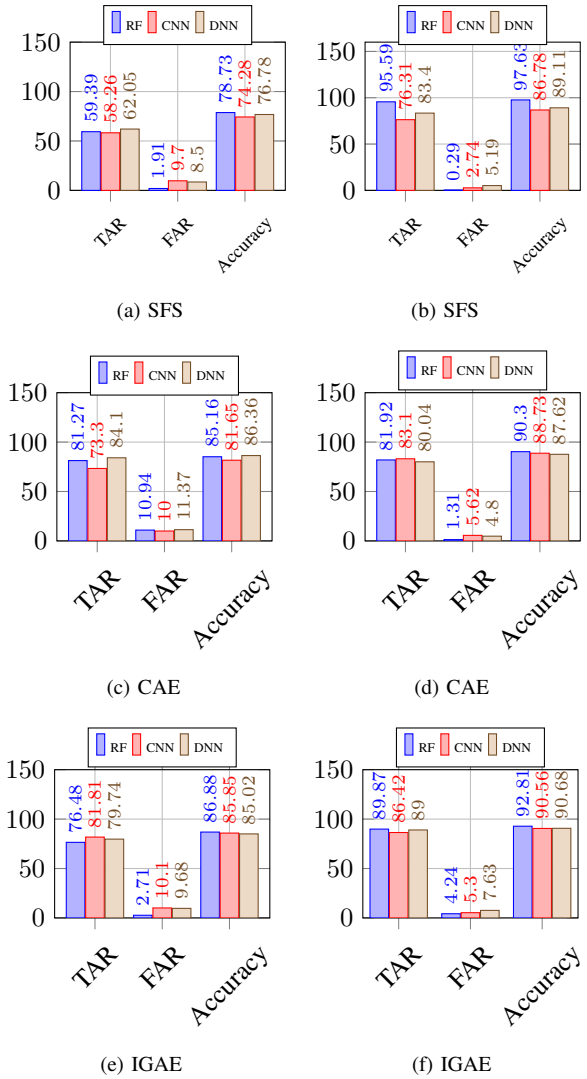
V. Results & Discussion

We report the results of malware detection using several metrics: True Accept Rate (TAR), False Reject Rate (FRR), False Accept Rate (FAR), True Reject Rate (TRR), Accuracy, and F1 Score. TAR represents the fraction of malware samples correctly identified as malware, while FRR denotes the fraction of malware samples incorrectly classified as benign. Similarly, FAR indicates the fraction of benign samples incorrectly classified as malware, and TRR represents the fraction of benign samples correctly identified as benign. Accuracy is the ratio of correct classifications to the total classification attempts. Since FRR and TRR can be computed as $(1-TAR)$ and $(1-FAR)$, respectively, we only present TAR and FAR to avoid redundancy.

We summarise our classification results in Figure 4. Figures 4a, 4c, 4e depict the obtained results without any augmentation Figures 4b, 4d, 4f with data augmentation.

In Figures 4a, 4c , and 4e, we show TAR, FAR, and accuracy. The results indicate that while the classifiers achieve moderate TAR and accuracy, there are notable areas for improvement. Specifically, the FAR remains relatively high, and the accuracy suggest that the classifiers struggle with TAR and FAR balance.

Figures 4b, 4d, and 4f illustrate the performance metrics after implementing data augmentation. A significant improvement is observed across all metrics for each classifier. The TAR increases, indicating a higher rate of correctly accepted instances. The FAR decreases, demonstrating enhanced precision and fewer false positives. Additionally, the accuracy



(a) SFS (b) SFS

(c) CAE (d) CAE

(e) IGAE (f) IGAE

Fig. 4: Comparison of classifiers' performance without (see Figures 4a, 4c, 4e) and with data augmentation (see Figures 4b, 4d, 4f). The legend indicates the models used in each experiment.

shows significant improvements, reflecting better overall performance and robustness of the classifiers. This consistent upward trend in performance metrics post-augmentation highlights the effectiveness of data GAN-based augmentation technique.

Our obtained results clearly demonstrate the significance of data augmentation. The improvements in TAR, FAR, and accuracy across all classifiers validate the use of data augmentation as a critical step in the ML pipeline. Based on the comparative analysis of the classifiers, the RF model consistently achieved higher performance across all evaluated metrics, including TAR, FAR, and accuracy. Given its reliability, we have decided to utilize this classifier in our final implementation. We leave this final implementation and evaluation of its effectiveness in real-world settings, as future work.

VI. Conclusions

In this study, we presented Z-MDZS, a ML-based scheme for zero-day malware detection using zero-shot learning principles. Our approach was evaluated using three advanced classifiers — RF, DNN, and CNN — on the BODMAS dataset. Additionally, we utilized GANs for data synthesis, which proved effective in generating high-quality data and enhancing the accuracy of our classifiers. This approach could potentially addresses the challenges of data scarcity and variability in malware detection. RF outperformed its opponents and remain consistent across all features spaces.

The Z-MDZS scheme, with its robust performance and ability to generalize to unseen threats, represents a significant advancement in the field of cybersecurity. As a future work we plan to explore additional data augmentation techniques to further enhance classifier performance and robustness. Further, investigating hybrid models that combine the strengths of different classifiers, such as RF and CNN, could also be beneficial. Furthermore, developing real-time implementation strategies and assessing the scalability and efficiency of the models for larger datasets will be crucial. Finally, applying the classifiers to different domains and datasets to validate their generalizability and adaptability is another important area for future research.

Acknowledgment

This work is partially supported by projects “SEcurity and RIghts In the CyberSpace - SERICS” (PE00000014 - CUP H73C2200089001), “Interconnected Nord-Est Innovation Ecoscheme - iNEST” (ECS00000043 - CUP H43C22000540006), and PRIN/PNRR “Automatic Modelling and ∀erification of Dedicated sEcUrity deviceS - AM∀DEUS” (P2022EPPHM - CUP H53D23008130001), all under the National Recovery and Resilience Plan (NRRP) funded by the European Union - NextGenerationEU.

References

[1] A. Djenna, A. Bouridane, S. Rubab, and I. M. Marou, “Artificial intelligence-based malware detection, analysis, and mitigation,” *Symmetry*, vol. 15, no. 3, p. 677, 2023.

[2] A. Wolsey, “The state-of-the-art in ai-based malware detection techniques: A review,” *arXiv preprint arXiv:2210.11239*, 2022.

[3] A. K. Chakravarty and et al., “A study of signature-based and behaviour-based malware detection approaches,” *Int. J. Adv. Res. Ideas Innov. Technol.*, vol. 5, no. 3, pp. 1509–1511, 2019.

[4] G. Jacob and et al., “Behavioral detection of malware: from a survey towards an established taxonomy,” *Journal in computer Virology*, vol. 4, pp. 251–266, 2008.

[5] H. S. Galal and et al., “Behavior-based features model for malware detection,” *Journal of Computer Virology and Hacking Techniques*, vol. 12, pp. 59–67, 2016.

[6] H.-Y. Kwon and et al., “Advanced intrusion detection combining signature-based and behavior-based detection methods,” *Electronics*, vol. 11, no. 6, p. 867, 2022.

[7] A. Souri and R. Hosseini, “A state-of-the-art survey of malware detection approaches using data mining techniques,” *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 1–22, 2018.

[8] A. Sabbah and et al., “Android malware detection: a literature review,” in *International Conference on Ubiquitous Security*, pp. 263–278, Springer, 2022.

[9] N. Z. Gorment, A. Selamat, and O. Krejcar, “A recent research on malware detection using machine learning algorithm: Current challenges and future works,” in *Advances in Visual Informatics: 7th International Visual Informatics Conference, IVIC 2021, Kajang, Malaysia, November 23–25, 2021, Proceedings 7*, pp. 469–481, Springer, 2021.

[10] M. S. Akhtar and T. Feng, “Malware analysis and detection using machine learning algorithms,” *Symmetry*, vol. 14, no. 11, p. 2304, 2022.

[11] H. Rathore and et al., “Malware detection using machine learning and deep learning,” in *Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6*, pp. 402–411, Springer, 2018.

[12] H. Li and et al., “Keeping deep learning models in check: A history-based approach to mitigate overfitting,” *IEEE Access*, 2024.

[13] L. Yang and et al., “Bodmas: An open dataset for learning based temporal analysis of PE malware,” in *2021 IEEE Security and Privacy Workshops (SPW)*, pp. 78–84, IEEE, 2021.

[14] C. Connors and D. Sarkar, “Machine learning for detecting malware in PE files,” *arXiv preprint arXiv:2212.13988*, 2022.

[15] A. Mills and et al., “Efficient and interpretable real-time malware detection using random-forest,” in *2019 International conference on cyber situational awareness, data analytics and assessment (Cyber SA)*, pp. 1–8, IEEE, 2019.

[16] P. Vadrevu and et al., “Measuring and detecting malware downloads in live network traffic,” in *Computer Security–ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings 18*, pp. 556–573, Springer, 2013.

[17] A. Buriro and et al., “Malwd&c: a quick and accurate machine learning-based approach for malware detection and categorization,” *Applied Sciences*, vol. 13, no. 4, p. 2508, 2023.

[18] S. Hou and et al., “Deep neural networks for automatic android malware detection,” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 803–810, 2017.

[19] Q. Lu and et al., “Self-attentive models for real-time malware classification,” *IEEE Access*, vol. 10, pp. 95970–95985, 2022.

[20] R. Burks and et al., “Data augmentation with generative models for improved malware detection: A comparative study,” in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0660–0665, IEEE, 2019.

[21] K. Stalin and M. B. Mekoya, “Improving android malware detection through data augmentation using wasserstein generative adversarial networks,” *arXiv preprint arXiv:2403.00890*, 2024.

[22] I. Gupta and et al., “Leveraging lstm and gan for modern malware detection,” *arXiv preprint arXiv:2405.04373*, 2024.

[23] H. S. Anderson and P. Roth, “Ember: an open dataset for training static pe malware machine learning models,” *arXiv preprint arXiv:1804.04637*, 2018.

[24] M. H. L. Louk and B. A. Tama, “Tree-based classifier ensembles for pe malware analysis: a performance revisit,” *Algorithms*, vol. 15, no. 9, p. 332, 2022.

[25] I. Goodfellow and et al., “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.