



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

MASTER THESIS IN COMPUTER ENGINEERING

Demystifying Malware Image Visualization: Enhancing Reproducibility and Explainability

MASTER CANDIDATE

Matteo Brosolo

Student ID 2029470

SUPERVISOR

Prof. Mauro Conti

University of Padova

CO-SUPERVISOR

Dr. Vinod P

University of Padova

ACADEMIC YEAR
2022/2023

When everything went to hell and the CPU began spewing out random bits, the result, on a CLI machine, was lines and lines of perfectly formed but random characters on the screen known to cognoscenti as "going Cyrillic." But to the MacOS, the screen was not a teletype, but a place to put graphics; the image on the screen was a bitmap, a literal rendering of the contents of a particular portion of the computer's memory. When the computer crashed and wrote gibberish into the bitmap, the result was something that looked vaguely like static on a broken television set - a "snow crash."

— Neal Stephenson, *Snow Crash*

Abstract

The persistent increase in malicious files encountered in the digital realm has posed an ongoing obstacle for security researchers from the advent of personal computers to the present day. Malware infection is a threat, particularly in platforms such as Android and Windows. These two OSs represent 64,25% of the whole worldwide consumer market share. Windows, specifically, is the most targeted platform. Researchers in cyber security must identify and categorize malware strains as soon as they discover them to defend public institutions, business organizations, and ordinary people. In the last few years, an effective way to classify malicious files in their respective families has been using neural networks, particularly CNNs. Utilizing neural networks in sensitive domains such as cyber security presents significant challenges in other fields like medicine and autonomous vehicles. Two of the major ones, rarely explored in the literature, are the problem of reproducibility and explainability of the results. In this thesis, we aim to replicate state-of-the-art CNN models from existing literature while employing class activation maps, a powerful and essential tool in explainability, to provide insightful explanations for their results.

In this study, six contemporary Convolutional Neural Networks (CNNs) were employed to assess the reproducibility and replicability within the field. The research focused on identifying challenges encountered during this process. Subsequently, HiResCAM was utilized to evaluate the CNNs performance and explore the underlying reasons for varied interpretations of inputs by different models. The findings from this analysis were combined with relevant metrics, including SSIM, to explore the potential of employing explanatory techniques to enhance emerging classifiers like the visual transformer.

To my brother

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Context	2
1.2 Contribution	5
1.3 Outline	6
2 Background	7
2.1 Malware Analysis	7
2.1.1 Static Analysis	7
2.1.2 Dynamic Analysis	8
2.1.3 Malware Visualization	9
2.1.4 Machine Learning	11
2.1.5 Deep Learning	12
2.2 Reproducibility	14
2.3 Explainability	15
3 Related Work	17
3.1 Machine Learning	17
3.2 Deep Learning	19
4 Methodology	29
4.1 Datasets	30
4.1.1 MalImg	30
4.1.2 Big2015	31
4.1.3 VX-Zoo	32
4.2 Replicated CNN Models	34

CONTENTS

4.2.1	Selection Methodology	34
4.2.2	Classification Models	36
4.3	Models Interpretation	39
4.3.1	CAM Techniques	40
4.3.2	SSIM	43
4.4	Classifier Improvement	44
4.4.1	Masks	45
4.4.2	ViT Classifier	46
5	Experiments	49
5.1	Reproduction	50
5.1.1	Environment	50
5.1.2	Evaluation Metrics	50
5.1.3	Hyperparameters Fine Tuning (E1)	52
5.1.4	Result Replication (E2)	54
5.1.5	Train Test Validation Problem (E3)	55
5.2	Explainability	60
5.2.1	Extraction of CAMs (E4)	60
5.2.2	SSIM (E5)	65
5.3	Attention Ensemble Architecture	67
5.3.1	Baseline ViT implementation (E6)	68
5.3.2	ViT implementation with ensemble masking (E7)	71
6	Conclusions and Future Works	79
6.1	Summary of Results	79
6.2	Future Work	80
	References	83
	Acknowledgments	93
A	HiResCAM Cumulative Heatmaps	95

List of Figures

1.1	Windows malware evolution	2
2.1	Example of grayscale images taken from two families found in the MallImg dataset. Note the similarity in shape and content of the image inside a specific class.	10
2.2	Example of CNN architecture	13
2.3	Visual Transformer Architecture [20]	14
2.4	Example of HiResCAM heatmaps overlayed to their respective samples	16
4.1	Diagram of the Replicability pipeline adopted in this thesis	35
4.2	Diagram of the explainability pipeline adopted in this thesis	39
4.3	Example of GradCAM heatmap extracted from malware families	42
4.4	Example of HiResCAM heatmap extracted from malware families using EfficientNetB0	43
4.5	Diagram of the classification enhancement through HiresCAM masks implemented	45
4.6	Example of mask of sample malware family	46
5.1	DenseNet121 [68] Training evolution on Big2015	56
5.2	EfficientNetB0 [55] Training evolution on Big2015	56
5.3	VGG16 [71] Training evolution on MallImg	56
5.4	Gibert [26] Training evolution on MallImg	57
5.5	Confusion matrix obtained from the training of DenseNet121 on Big2015 dataset	60
5.6	Confusion matrix obtained from the training of EfficientNetB0 on MallImg dataset	61

LIST OF FIGURES

5.7	Confusion matrix obtained from the training of DenseNet121 on VX-Zoo dataset	62
5.8	ROC curve obtained from the training of DenseNet121 on Big2015 dataset	63
5.9	ROC curve obtained from the training of EfficientNetB0 on Mal-Img dataset	63
5.10	ROC curve obtained from the training of DenseNet121 on VX-Zoo dataset	64
5.11	Confusion matrix obtained from the training of ViT on VX-Zoo dataset without mask augmentation.	68
5.12	Confusion matrix obtained from the training of ViT on Mallng dataset without mask augmentation	69
5.13	Confusion matrix obtained from the training of ViT on Big2015 dataset without mask augmentation	70
5.14	Confusion matrix obtained from the training of ViT on Big2015 dataset with mask augmentation generated with EfficientNetB0 and DenseNet121	72
5.15	Confusion matrix obtained from the training of ViT on Mallng dataset with mask augmentation generated with EfficientNetB0 and DenseNet121	73
5.16	Confusion matrix obtained from the training of ViT on VX-Zoo dataset with mask augmentation generated with EfficientNetB0 and DenseNet121	74
5.17	Confusion matrix obtained from the training of ViT on Big2015 dataset with mask augmentation generated with EfficientNetB0 and ResNet50	75
5.18	Confusion matrix obtained from the training of ViT on Mallng dataset with mask augmentation generated with EfficientNetB0 and ResNet50	76
5.19	Confusion matrix obtained from the training of ViT on VX-Zoo dataset with mask augmentation generated with EfficientNetB0 and ResNet50	77

List of Tables

3.1	Overview of Surveyed Research Papers in Malware Image-Based Detection: A Comparative Analysis of Methods and Approaches	24
4.1	Mallng Dataset	31
4.2	Big2015 Dataset	32
4.3	VX-Zoo Dataset	34
5.1	Hyperparameters of the analyzed CNN regarding the architecture of the neural network. Values in bold have been tuned or inferred	53
5.2	Hyperparameters of the analyzed CNN regarding the training of the neural network. Values in bold have been tuned or inferred .	54
5.3	Performance metrics of replicated CNN models trained on Mal-Img. P represents the value found in the paper while R represents the replicated value.	55
5.4	Performance metrics of replicated CNN models trained on Big2015. P represents the value found in the paper while R represents the replicated value.	55
5.5	Performance metrics of CNN models with split 70/30 on Big2015	58
5.6	Performance metrics of CNN models with split 70/30 on Mallng	58
5.7	Performance metrics of CNN models with split 70/30 on VX-Zoo	59
5.8	Prediction timings of neural networks on the 3 datasets. Values in seconds	59
5.9	Cumulative GradCAM SSIM values extracted from the VX-Zoo dataset	66
5.10	Cumulative GradCAM SSIM values extracted from the Mallng dataset	66

LIST OF TABLES

5.11	Cumulative GradCAM SSIM values extracted from the Big2015 dataset	66
5.12	Cumulative HiResCAM SSIM values extracted from the VX-Zoo dataset	66
5.13	Cumulative HiResCAM SSIM values extracted from the MallImg dataset	67
5.14	Cumulative HiResCAM SSIM values extracted from the Big2015 dataset	67
5.15	Final Results	78
A.1	HiResCAM Cumulative Heatmaps generated for Big2015	96
A.2	HiResCAM Cumulative Heatmaps generated for MallImg	98
A.3	HiResCAM Cumulative Heatmaps generated for VX-Zoo	99

1

Introduction

The proliferation of malware has continued uninterrupted since the dawn of the personal computer and has only increased with the advent of the internet. The variety of malware has also increased, and nowadays, malicious files are categorized into classes based on targets, techniques used, or platforms [13]. Navigating the web without the due precautions can be dangerous, and if individuals find annoying the consequences of an infection, businesses and governments cannot permit even a small breach in security. For this reason, in recent years, there has been a substantial surge in investments directed toward cybersecurity, with a particular focus on developing robust anti-malware solutions. This increased investment in cybersecurity can be attributed more specifically to several factors. Firstly, the rising number of high-profile cyber attacks targeting critical infrastructure, financial institutions, and large corporations has highlighted the devastating impact of malware and the urgent need for effective countermeasures. The significant financial losses, reputational damage, and potential compromise of sensitive information associated with these attacks have driven decision-makers to allocate substantial resources toward bolstering their cybersecurity posture. Furthermore, the growing interconnectivity of digital ecosystems, the proliferation of the Internet of Things (IoT), and the emergence of transformative technologies such as artificial intelligence and blockchain have heightened the intricacy and magnitude of cyber threats, demanding proactive and all-encompassing security measures. Moreover, regulatory frameworks and compliance requirements have been tightened in various industries, compelling organizations to invest in robust anti-malware solutions to meet stringent se-

1.1. CONTEXT

curity standards. The continuous growth of the global cybersecurity market, which encompasses a wide array of services and solutions, including malware analysis and detection, highlights the recognition of the gravity of the malware threat and the commitment towards mitigating its risks. In this context, the need for detection, identification, and analysis of malware samples is ever more needed. A fundamental task in malware analysis is the categorization of mal-

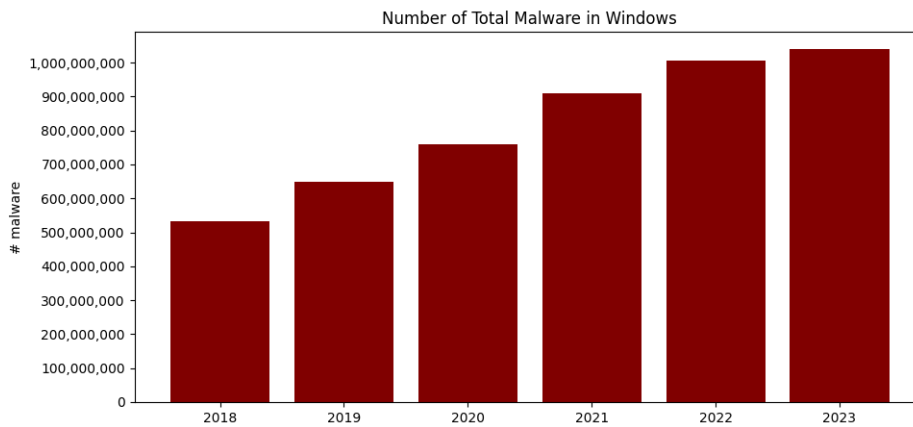


Figure 1.1: Windows malware evolution

ware in families. This assignment is important because it helps categorize and organize different types of malware based on their characteristics and behavior, helping security researchers and analysts better understand how the malware operates and develop effective countermeasures to prevent or mitigate its impact. Malware classification can also help identify trends and patterns in the types of malware being used by attackers. This information is used to develop proactive measures to prevent similar attacks in the future. Malware classification differs from malware detection, identifying malware's presence on a system. While malware classification focuses on categorizing and organizing different types of malware, malware detection focuses on identifying specific instances of malware to prevent or mitigate their impact.

1.1 CONTEXT

In recent years, in addition to traditional signature-based approaches, more contemporary methods rooted in machine learning and neural networks have been incorporated into the defender's toolkit, resulting in remarkable successes

in countering malicious software. While traditional signature-based approaches were effective at classifying known malware, they struggled to keep up with modern threats' rapid evolution and polymorphic nature. However, machine learning algorithms, particularly neural networks, have revolutionized the field by enabling the proactive detection and mitigation of previously unseen and zero-day malware.

By training on extensive sets of labeled data, neural networks have demonstrated exceptional abilities in learning intricate patterns and identifying subtle indicators of malware presence. These models can even extract high-level features from raw data, such as file behavior, network traffic, and system logs, to distinguish between malicious strains. Through continuous learning and adaptation, neural networks can dynamically update their knowledge base, effectively detecting emerging threats. Furthermore, integrating deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has further enhanced the accuracy and efficiency of anti-malware systems [24] [43].

CNNs excel in capturing spatial and structural information from malware samples [49], while RNNs are adept at modeling dependencies in sequential code or network behavior[32]. Combining these techniques has enabled more robust and comprehensive malware analysis, including advanced features like the fusion of static and dynamic analysis, behavioral profiling, and anomaly detection.

A widely used technique that harnesses the power of CNNs is visualization. Researchers can generate an image with the data extracted from a malicious file and use it as a sample for the CNN [49][19][71][18][40]. This technique takes advantage of the convolutional neural network models' intrinsic qualities such as robust feature extraction, spatial invariance, adaptation, and efficient processing. One of the significant drawbacks of neural networks is the inherent challenge of explaining the classification results. While neural networks excel in complex pattern recognition tasks, their inner workings and decision-making processes can be challenging to interpret and understand. Unlike traditional machine learning algorithms, such as decision trees or logistic regression, which provide explicit rules or feature importance rankings, neural networks lack explicit transparency [31]. This lack of openness arises from the highly interconnected layers and numerous parameters within the network. As a result, determining which specific features contribute to a particular classification decision can be

1.1. CONTEXT

elusive.

The absence of interpretability presents challenges, particularly in critical domains like healthcare, legal systems, or cybersecurity, where the need for explainability and transparency is paramount. The inherent difficulty in elucidating the results of neural network classifications necessitates the development of techniques and methodologies for interpretability. These methods should enable researchers and practitioners to gain insights into the decision-making processes of these intricate models, all while striking a balance between accuracy and transparency. The demand for explainable outcomes is steadily growing, especially among legislators who seek to ensure fairness and accountability in the decisions made by artificial intelligence systems. A prime example of this is the European Union's General Data Protection Regulation (GDPR) [76], which is European legislation aimed at safeguarding individuals' privacy rights and governing the processing of their data. Among the key provisions outlined in the document is the "Right to explain," which grants data subjects the right to avoid decisions that are solely based on automated processing. Similar legislation is being deliberated in the United States [35] and China [8].

The reproducibility of the results obtained by the best model is paramount, especially in a crucial field like cybersecurity, in which millions are spent to get the best results possible. In the context of malware analysis, where the complexity of threats and the sophistication of cyber-attacks are constantly evolving, reproducibility acts as a litmus test for the reliability of detection methods, classification algorithms, and mitigation strategies. Reproducible research allows experts to scrutinize methodologies, identify potential biases, and validate the effectiveness of proposed solutions. Furthermore, it fosters a culture of collaboration and knowledge-sharing within the research community, enabling the collective development of robust and reliable tools to combat ever-changing malware threats. Enabling the reproducibility of papers in malware analysis not only bolsters confidence in research outcomes but also paves the way for continuous innovation and advancements in cybersecurity practices, ultimately safeguarding digital infrastructures worldwide.

With this study, we aim to tackle these two main challenges to model robustness: reproducibility and explainability. We tested reproducibility by selecting six of the best-performing CNNs found in the literature and implementing the architectures using the information found online and in the papers presenting them. We analyzed the issues that emerged in the implementation and studied

the results, comparing them to the ones reported. We employed a technique called Class Activation Maps (CAM) to address the challenge of explainability. In particular, we are the first to implement HiResCAM for accurately explaining the state-of-the-art CNN used in developing malware scanners. HiResCAM provides insights into the decision-making process of CNNs by highlighting the regions of an input image that contribute the most to a specific classification outcome. By visualizing the activation patterns learned by the network, CAMs offer interpretability and shed light on the essential features or areas that influenced the classification decision. This technique has proven particularly valuable in applications such as image recognition and object detection, allowing researchers and practitioners to understand the underlying reasoning behind a CNN's predictions.

In short, the objective of this thesis is to answer the following questions:

- **RQ1:** Can researchers reasonably reproduce or replicate the models presented in a paper?
- **RQ2:** What are the current issues affecting the field of malware image visualization?
- **RQ3:** How can heatmaps help shed light on the functioning of CNNs in visualizing malware, offering insights into their mechanisms?
- **RQ4:** Can these insights be extrapolated to enhance classifiers performance?

1.2 CONTRIBUTION

The main contributions of this paper are the following:

1. We carry out a reproducibility study on five high-quality papers in the domain of malware image-based classification, highlighting the challenges that researchers can have in the implementation of these models;
2. We study the comparability of results between papers produced by different research groups;
3. We introduce an additional dataset with different characteristics from the other two used in the study to further validate our results;
4. We implement the discussed models and show that visualization methods can have optimal results in the field of malware classification;

1.3. OUTLINE

5. We produce heatmaps using two different CAM methodologies to shed light on the process by which the individual neural networks are classifying the samples;
6. We analyze the cumulative heatmaps to comprehend the way different neural networks arrive at their final prediction and the errors they might make;
7. We introduced a new and upcoming technique in malware visualization called visual transformer and show how the study of CNNs heatmaps can help improve classifiers results.

1.3 OUTLINE

The rest of the paper is organized in the following way.

1. **Chapter 1** we provided an introduction to the context in which this study operates and the problems that we will address;
2. **Chapter 2** introduces the fundamental concepts and techniques used in malware analysis and the ideas to investigate the reproducibility and replicability of models found in the literature. Additionally, here will be presented the notion of explainability and the reasons and theory behind its use;
3. **Chapter 3** summarizes a survey done on the malware image classification domain, highlighting the particular gaps and blind spots of the field;
4. **Chapter 4** advances the methodology used throughout the thesis and justifies it in light of the questions that the survey brought up;
5. **Chapter 5** presents a detailed description of the experiments and comments on the results.
6. **Chapter 6** concludes with the final takeaways and calls attention to future questions.



Background

2.1 MALWARE ANALYSIS

Two main approaches divide malware analysis: static and dynamic. In static analysis, malware is studied without executing it, involving examining code using disassemblers and decompilers. Researchers must comprehend the behavior by analyzing indicators like opcodes, API calls, functions, and the overall structure of the executable. Suspicious code patterns can be identified and used as signatures to detect other malware of a similar strain. This technique, known as signature-based malware classification, constitutes the traditional method employed by antivirus software to identify malicious code. In contrast, dynamic analysis, an alternative to static analysis, entails observing malware's behavior during its execution. Unlike static analysis, dynamic analysis necessitates running the malware in a controlled environment, such as a virtual machine, to monitor its actions and detect malicious activities. Dynamic analysis circumvents common challenges associated with static analysis, such as obfuscated code, anti-debugging mechanisms, or anti-virtualization measures.

2.1.1 STATIC ANALYSIS

Static analysis is a method used in malware analysis to examine the code or binary of the malicious program without executing it. It involves analyzing the structure and content of malware samples to identify potentially malicious activities and understand how the malware operates without seeing it in action.

2.1. MALWARE ANALYSIS

Some recognized tools used in static analysis are disassemblers, decompilers, and debuggers. These tools help to reveal the instructions used by the malware to perform its malicious activities, such as stealing data, modifying system settings, or propagating to other systems. Additionally, static analysis can help identify indicators of compromise that enable the detection of malware presence on a system [50]. These can include file names, registry keys, network traffic patterns, and other characteristics unique to the malware. During static analysis, a researcher can extract signatures or marks of behavior from the malware's binary or code. One can use these signatures to detect malware on a system by comparing them against a database of known malware signatures [82].

2.1.2 DYNAMIC ANALYSIS

Dynamic analysis is a method used in malware analysis to examine the behavior of malware during its execution. Unlike static analysis, dynamic analysis involves running the malware in a controlled environment to observe its behavior and identify malicious activities. A sandbox or virtual environment executes the malware during dynamic analysis, simulating a real-world operating system. This allows the analyst to observe the malware's behavior in a controlled environment and monitor its interactions with the operating system and network. Sandbox services, such as VirusTotal [74], Hybrid Analysis[30], Any.Run, [3] Cuckoo Sandbox[15], and Joe Sandbox[33] offer a convenient way to perform dynamic malware analysis and provide detailed reports on the malware's behavior. The analyst can use various tools and techniques to monitor the sample's behavior, including system monitors, network monitors, and debuggers. These tools can help identify the malware's activities, such as creating files, modifying system settings, or communicating with a command-and-control (C2) server. Dynamic analysis can also help determine the malware's evasion techniques, such as anti-debugging or anti-virtualization techniques [12]. By observing the malware's behavior in a controlled environment, the analyst can better understand the type and threat level of the malicious code. In the case of dynamic analysis, the visualization approach. One can exploit the ideas in static analysis by organizing the extracted information as an image when working with novel data.

2.1.3 MALWARE VISUALIZATION

Malware visualization is a technique used in malware analysis to visually represent the behavior and characteristics of malware [43]. Visualization-based malware detection is driven by the need to analyze and understand malware behavior, detect anomalies, and mitigate threats. Key motivations include the ability to analyze complex data and identify patterns of malicious activity. Visualization enables comprehensive views of systems and networks, facilitating the identification of relationships and potential attack vectors. It involves creating graphical representations of various aspects of the malware, such as its network activity, file activity, system calls, or the simple binary file. Analysts can use malware visualization to represent the code structure of the malware, aiding in the identification of the techniques used by the malware to evade detection and the vulnerabilities it exploits. In addition to assisting in detecting and analyzing malware, malware visualization can also help with the malware classification task. Visualizing malware as images enables the differentiation of its components. Malware developers often create new variants by making small changes to the code, allowing for the identification of related malware strains. Visualization techniques are effective in capturing minor malware alterations. Analysts can identify similarities and differences between different malware strains by representing the behavior and characteristics of malware in a visual format and using machine learning methods. This can help with the classification of new and unknown malware, as well as the identification of malware families and variants. Unlike existing classification methods that necessitate disassembly or execution, this approach eliminates the need for both yet exhibits remarkable enhancements in performance. Moreover, it proves to be robust against prevalent obfuscation tactics like section encryption [71][80]. The pipeline used in malware classification through malware visualization comprises the following steps: information gathering, image generation, feature extraction and classification.

Researchers can employ different methods to transform malware into an image representation, with the first consideration being the information available to them. This information can range from the raw bytecode of the malware to a memory dump of the executed code from a sandbox. Generally, the techniques described in this paragraph for generating malware images can be applied to any type of information with some minor adaptations. Nevertheless, the type, quan-

2.1. MALWARE ANALYSIS

tity, and quality of information available during image creation can markedly alter the results that a hypothetical classifier will yield at the end of the classification pipeline.

Selecting an optimal technique for information representation is of utmost importance, as it determines how the data will be organized visually. The methods applied at this level strive to present the information as images while preserving crucial characteristics, such as data interrelationships and connections. Researchers must balance achieving a robust image layout that can be effectively utilized by feature extractors and classifiers while also considering the potential loss of information that accompanies the transformation of data representation. The standard methodology applied to transform a malware into a grayscale image is the so-called *line-by-line technique*. A malware executable can be depicted as a binary sequence of zeros and ones. This sequence can be transformed into a matrix and interpreted visually as an image. Researchers have noticed striking visual resemblances in the texture of images representing malware from the same family. This phenomenon could be attributed to the widespread practice of repurposing code to generate fresh malware iterations. The next step in developing a reliable malware classifier using malware visu-

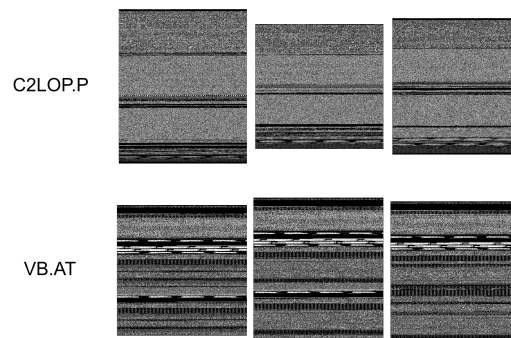


Figure 2.1: Example of grayscale images taken from two families found in the Mallimg dataset. Note the similarity in shape and content of the image inside a specific class.

alization techniques is to extract relevant features from the generated image of the executable. This is done by applying visual or textural feature extraction methods to the images, which capture and transform features into a concise representation for identifying patterns in a specific region of interest. These techniques are used to distinguish between malware and benign files, as well as between different malware families.

The last step involves classifying the samples using the extracted features. At this stage, any machine learning or deep learning classifiers commonly used in other domains can be applied, as the information has been transformed and synthesized into a format suitable for input into a classifier.

2.1.4 MACHINE LEARNING

Machine learning (ML) is increasingly used in malware analysis to automatically classify and identify malicious software based on its behavior and characteristics [69]. Machine learning, in general, forms the foundation for a wide range of applications, spanning from image recognition and natural language processing to medical diagnostics and autonomous vehicles. Rooted in advanced statistical methods and computational algorithms, machine learning empowers systems to learn from data, identify patterns, and make intelligent decisions, all without explicit programming. The ability to process immense datasets and unearth intricate patterns, which might be beyond human perception, is a testament to its power. In machine learning, two fundamental paradigms govern the learning process: supervised and unsupervised learning. The most commonly used algorithms in malware image-based analysis are supervised.

In supervised learning, the algorithm is trained on a labeled dataset, meaning it learns from input-output pairs. Given a set of input variables X and corresponding output labels y , the algorithm learns a mapping function $f(X) \rightarrow y$. Common supervised learning algorithms used in modern malware image classification include Support Vector Machine (SVM), Random Forests (RF) and K-Nearest Neighbour (KNN) [22][65][78].

SVM is a robust supervised learning algorithm for classification and regression tasks. SVM finds the optimal hyperplane that best separates classes in the feature space, maximizing the margin between classes; Random Forests are an ensemble learning method that constructs multiple decision trees during training and outputs the individual trees' mode (classification) or mean prediction (regression). Another notable algorithm is KNN, a non-parametric, instance-based learning algorithm for classification and regression tasks. Given a new data point, KNN classifies it by identifying the 'k' nearest neighbors from the training data and determining the majority class or averaging the values of these neighbors.

2.1. MALWARE ANALYSIS

Machine learning algorithms can also analyze malware features such as file size, file type, file activity, system calls, network traffic, and other characteristics to achieve optimal results. By analyzing these features, machine learning algorithms can identify patterns and traits indicative of malware. For example, during dynamic analysis, machine learning algorithms can analyze the behavior of malware as it interacts with the system and network. Machine learning algorithms can identify patterns that distinguish malicious behavior from legitimate activity by monitoring the malware's actions. Similarly, during static analysis, machine learning algorithms can analyze the code and structure of malware to identify patterns and characteristics associated with malicious software.

2.1.5 DEEP LEARNING

In addition to traditional machine learning approaches, researchers applied various types of neural networks to the malware classification problem with considerable effort directed toward malware visualization [10][26]. In this particular niche, the two major architecture explored have been CNNs and LSTMs. Their application in malware analysis has demonstrated superior results compared to traditional machine learning models. In the last decade, the application of deep learning has skyrocketed, driven by exceptional results in multiple fields such as NLP and image classification. Neural networks, particularly CNNs, have been successfully applied to malware visualization, guaranteeing often better results than classical machine learning models. Deep neural networks hold immense promise for the future despite their significant requirements for increased computational resources and larger datasets.

The advantage of using a CNN is present when treating spatially-structured data. The origin of Convolutional Neural Networks has been attributed to LeCun et al. [16]. CNNs utilize local connectivity and weight sharing to capture spatial correlations in images efficiently. This is done by leveraging the concept of convolution, which involves sliding a set of filters across the input and applying element-wise multiplication and summation operations. They employ pooling layers to downsample data while preserving important features. CNNs learn hierarchical representations, extracting complex features from images. Additionally, they automatically learn discriminative filters, making them effective for tasks like object recognition and image classification. Keeping in mind the malware classification through visualization pipeline, it is essential to note how

the application of a CNN essentially permits an end-to-end classification of the samples, integrating feature extraction and classification in a single architecture. The Long Short Term memory neural network is another commonly found architecture in the literature. Even though the use of this type of model has obtained its fame in the field of NLP, its application on malware analysis has produced notable results

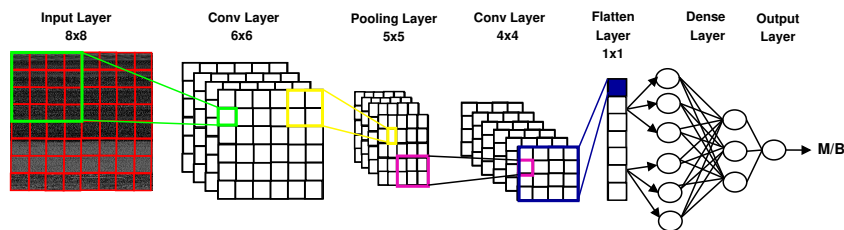


Figure 2.2: Example of CNN architecture

An additional deep learning architecture that is catching on more recently is the use of transformers in the particular form of Visual Transformers (ViT) [20]. ViT could have disruptive consequences for the field because of its particular inner working that differs substantially from the CNN. A Transformer is primarily employed in Natural Language Processing (NLP) and relies heavily on attention mechanisms, comprising two main components: an encoder and a decoder. The encoder processes the input sequence, capturing its salient features through self-attention mechanisms, ultimately generating a set of context-rich representations. The decoder, on the other hand, takes these representations and incrementally produces the output sequence, ensuring accuracy through self-attention. The synergy between the encoder and decoder empowers the model to proficiently comprehend and generate sequential data. In computer vision, transformers can weigh the importance of different image regions by using self-attention, enabling them to capture complex spatial relationships and long-range dependencies. Self-attention is a mechanism that permits a neural network to weigh the significance of different parts within an input sequence during its representation. Unlike traditional attention mechanisms, where the attention weights are calculated based on the relationship between two separate sequences, self-attention allows a sequence to focus on different parts of itself. By considering all elements in the input sequence during processing, self-attention enable the model to access global contextual information. This is in contrast to traditional sequential models like RNNs, where information flow is limited by the fixed order of processing.

2.2. REPRODUCIBILITY

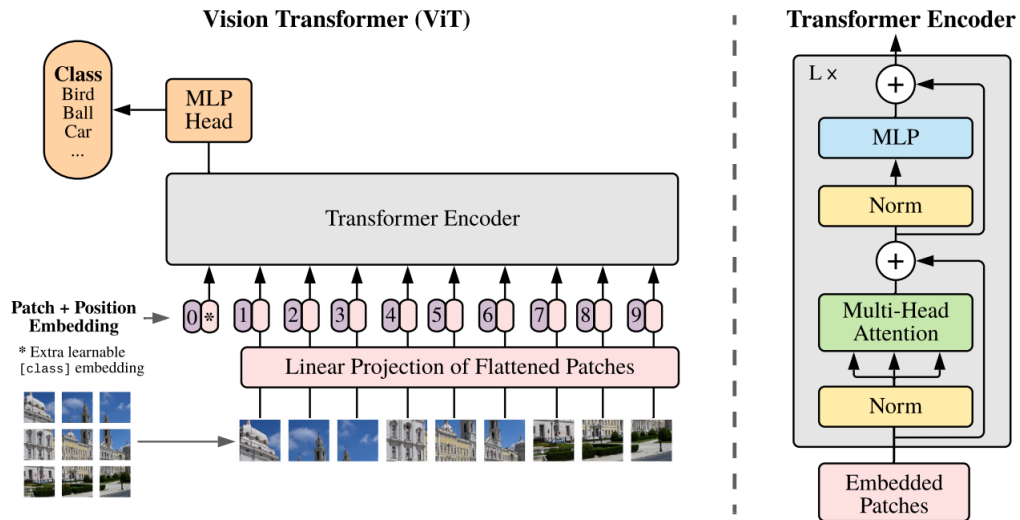


Figure 2.3: Visual Transformer Architecture [20]

In ViT, both its encoder and decoder utilize stacked self-attention and point-wise fully connected layers. Unlike CNN, where properties such as locality, two-dimensional neighborhood structure, and translation equivariance are inherently embedded within all layers, ViT primarily incorporates these traits through Multi-Layer Perceptron (MLP) layers. These MLP layers are local and exhibit translational equivariance. In contrast, ViT’s self-attention layers are global, establishing comprehensive dependencies between input and output by considering the entire input sequence [57].

2.2 REPRODUCIBILITY

Few authors have asked themselves the question of reproducibility. The preliminary study that is considered at the beginning of the majority of papers presenting new models doesn’t usually include a discussion about the implementation of other models but often discusses only the results taken at face value. The large majority of researchers’ lack of sharing of the codebases further exacerbates this problem. The reproducibility problem has been studied in depth in other fields. Borrowing from the advice given by researchers from different domains, we can see how following some guidelines for implementing models could help other authors verify and reproduce the research of other groups. It is essential to understand that reproducibility has varied over time

and can mean different things to different people. Daoudi et al. [17] have helpfully defined three concepts to represent different ways researchers can implement a model with various degrees of faithfulness. These three concepts are *repeatability*, *reproducibility* proper and *replicability*. The difference between those three lies in which team carries out the new experiment and which experimental setup is used. If the same team performs the original experiment on the same experimental setup, then researchers talk about repeatability. If another group reproduces the experiment on the same setup, it is called reproducibility. If, in the end, another team experiments on a different experimental setup, we are talking about replicability. Reproducibility appears to be the key concern if the scientific community wants to validate the results obtained by a research group.

The reproducibility of the results obtained by the best model is of paramount importance, especially in the context of malware analysis, where the complexity of threats and the sophistication of cyber-attacks are constantly evolving, reproducibility acts as a criterion for evaluation for the reliability of detection methods, classification algorithms, and mitigation strategies. Reproducible research allows experts to scrutinize methodologies, identify potential biases, and validate the effectiveness of proposed solutions. Furthermore, it fosters a culture of collaboration and knowledge-sharing within the research community, enabling the collective development of robust and reliable tools to combat ever-changing malware threats. Facilitating the reproducibility of research papers in malware analysis enhances trust in research findings. It lays the foundation for ongoing innovation and progress in cybersecurity practices, ultimately ensuring the security of global digital infrastructures.

2.3 EXPLAINABILITY

The need for transparency and accountability in automated decision-making systems has made explainability a crucial element in machine learning. Understanding and interpreting the rationale behind a machine learning model's predictions is particularly significant in healthcare, finance, and autonomous systems. Explainability holds paramount importance in cybersecurity, where ambiguity is unacceptable. Understanding how security systems make decisions is crucial to prevent serious issues. Explainability makes systems more transparent by providing a deep understanding of how they work and highlighting potential weaknesses or malicious activity. Although some machine learning

2.3. EXPLAINABILITY

algorithms inherently offer explanations for their outputs, widely used neural networks lack inherent explainability. This lack of transparency has raised concerns and led to a demand for interpretability techniques to enhance trust, mitigate bias, and ensure the ethical deployment of machine learning models.

The purpose of any explanation generally falls into one of two categories: (i) providing insights into model training and generalization. These explanations offer practitioners valuable information for decision-making in model training and validation, such as the quantity of labeled data, hyperparameter values, and model selection. The other category, (ii) explanations elucidating model predictions, helps practitioners understand why the model made a specific prediction, typically related to the input data. The explanations used in our research fall into this category, enabling communication of model predictions, even to non-experts. Analyzing individual predictions can reveal patterns in the overall behavior of the model. In our case, dealing with image data, explanations take the form of saliency maps or heatmaps. Saliency maps highlight crucial regions in an image that influenced the network’s prediction. It’s important to note that while heatmaps are a prevalent method for presenting model explanations, their interpretation is subjective and dependent on the practitioner.

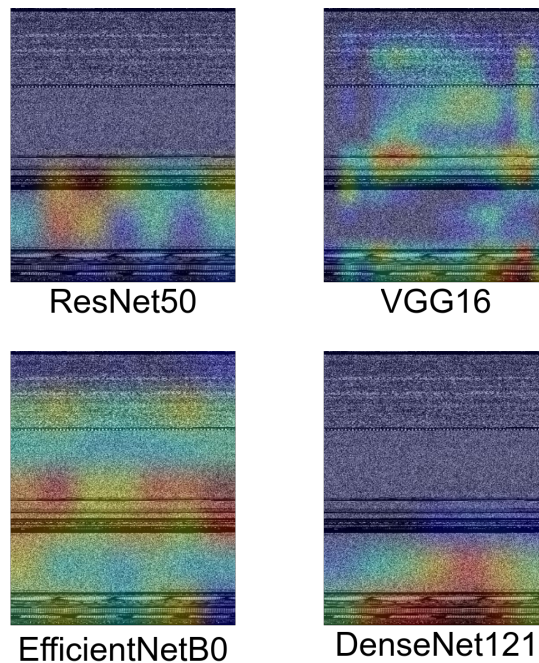


Figure 2.4: Example of HiResCAM heatmaps overlaid to their respective samples

3

Related Work

This chapter serves as a comprehensive exploration into the cutting-edge methodologies within the fields of machine learning and deep learning, specifically tailored to the domain of malware image classification. As we delve into the subsequent sections, our focus will shift distinctly into two pivotal areas: the sophisticated algorithms of deep learning and the more traditional yet robust machine learning techniques. By understanding the nuances and capabilities of these two approaches and how researchers have taken advantage of them, readers will gain invaluable insights into the diverse strategies employed to identify, analyze, and classify malware from visual data, forming a critical foundation for the discussions that follow. By conducting a thorough analysis of these techniques, this chapter endeavors to give readers a nuanced comprehension of the state-of-the-art advancements in malware image classification while spotlighting the inherent blind spots and knowledge gaps within the field. The subsequent chapters will meticulously delve into these issues, dissecting and exploring them in detail together with the solution developed as the thesis' work.

3.1 MACHINE LEARNING

The introduction of machine learning techniques has been proven fruitful in various domains and cybersecurity use is increasing [69].

In 2011 Nataraj et al. [49] introduced malware analysis and the use of visu-

3.1. MACHINE LEARNING

alization techniques, accompanying it with appreciable results in classification. This paper used an algorithm to transform the binary file of malicious samples in gray-scale images. GIST texture features have been extracted from the image dataset to implement an SVM classifier. This approach aids in categorizing novel and unfamiliar malware, as well as recognizing distinct lineages and versions within malware families. The model's overall accuracy has been tested on the MalImg dataset presented in the same study. Over 9339 samples divided into 25 families, the classifier has obtained 97.18% accuracy.

In the case of visual malware analysis, a modern approach can be found in [47] where authors use classifiers such as random forests, SVM, and decision trees to obtain results over 90% accuracy depending on the feature extraction method. The innovative idea presented in this paper is the use of an ensemble of global (GIST) and local (D-SIFT) feature extraction techniques. They reduce pattern dimensions through cluster assignment and regularization to address computational issues. Results show the model achieves 98% accuracy on the Malimg database but has a slightly slower runtime.

Fu et al.[23] presented a technique for visualizing malware both globally and locally to achieve precise classification. Their approach involved representing malware as RGB-colored images and extracting global features through the Gray-level Co-occurrence Matrix for texture and color moments for color attributes. Additionally, they incorporated locally extracted special byte sequences from code and data sections, transforming all these features into feature vectors using Simhash. By integrating global and local features, they enabled efficient classification using algorithms such as random forest, K-nearest neighbor, and support vector machine. This approach resulted in the highest accuracy of 97.47% and the highest F-measure of 96.85% when tested on a dataset comprising 7087 samples from 15 different malware families.

In [81] the authors presented two approaches to identify malware code. The first solution involved solving malware classification through an integrated learning method, demonstrating high accuracy in classification, particularly with multi-model ensemble techniques. The second solution addressed the visualization of feature matrices using the t-SNE algorithm to determine the number of malware families, followed by clustering using the k-means algorithm. The paper also introduced a novel concept called the visual character matrix, which successfully determined the number of malware families. The authors suggested future applications for these methods in detecting malicious

code in IoT devices and wireless sensor networks.

Roseline et al. [60] proposed an anti-malware system that employs a layered ensemble of random forests, mimicking deep learning techniques but with improved performance. This approach doesn't require hyperparameter tuning or backpropagation and operates with reduced model complexity. The proposed system achieved detection rates of 98.65%, 97.2%, for Malimg and BIG 2015, outperforming other state-of-the-art.

Machine learning techniques have demonstrated a reasonable ability to detect and categorize malware, with their notable strengths in efficiency and their models' inherent interpretability. Nonetheless, their utility has been overshadowed by deep learning methods' increasingly prevalent and superior performance. Furthermore, the requirement for meticulous feature extraction, often necessitating expert assistance, represents another vulnerability of machine learning approaches.

3.2 DEEP LEARNING

In addition to traditional machine learning approaches, more recently, neural networks, particularly CNNs have been explored. Their application in malware analysis has demonstrated superior results compared to conventional machine learning models.

In [26], the authors improve the visualization technique by using an originally designed CNN. The feature extraction has been done through the neural network without the need for other algorithms. The overall model has improved accuracy by 1.30% over the earliest Nataraj et al. model, proving the usefulness of CNNs.

In [34], authors explore the effectiveness of Convolutional Neural Networks (CNNs) in this context. The proposed approach converts malware binaries into grayscale images and trains a CNN for classification. Experiments conducted on standard malware datasets (Malimg and Microsoft malware) show that the method outperforms existing techniques, achieving accuracy rates of 98.52% and 99.97% on the respective datasets. In this case, as with the previously discussed Gibert paper [26], the CNN proposed has been developed from scratch by the authors.

Ni et al. [51] proposes a malware classification algorithm, Malware Classification using SimHash and CNN (MCSC), to tackle the task of malware classification using visualization. MCSC converts disassembled malware codes

3.2. DEEP LEARNING

into grayscale images using SimHash and then utilizes CNNs to identify their families. The algorithm incorporates multi-hash, major block selection, and bilinear interpolation to enhance performance. Experimental results demonstrate MCSC's effectiveness in classifying malware families, even with unevenly distributed samples. The algorithm achieves a classification accuracy of up to 99.26% and an average accuracy of 98.86% on a dataset of 10,805 samples, outperforming other compared algorithms.

In [66], the authors discuss the vulnerability of Internet of Things (IoT) devices to cyberattacks, particularly Distributed Denial of Service (DDoS) attacks, due to their lack of basic security measures. They propose a lightweight method to detect DDoS malware in IoT environments. The approach similar to the previously discussed techniques converts malware binaries into grayscale images and uses a convolutional neural network for classification. Experimental results demonstrate that the system achieves 94.0% accuracy in distinguishing goodware and DDoS malware and 81.8% accuracy in classifying goodware and two prominent malware families.

The model presented in [73] combines similarity mining and deep learning architectures, using various distance measures to calculate similarities between malware variants. These measures generate similarity matrices, and the distances between these images help identify malware families. Sequential Minimal Optimization (SMO), Normalized Polynomial kernel plus Deep learning architectures in a bi-directional CNN achieves nearly 99% accuracy. The proposed method is computationally efficient compared to traditional machine learning techniques and can be continuously trained in real-time to handle new malware threats.

In 2020 Vasan et al. [72] experimented with a slightly simplified version of the VGG16 CNN. The authors utilized RGB images instead of gray-scale ones by applying a simple colormap on the originals. They then employed transfer learning and fine-tuning to obtain a competitive model. With transfer learning, researchers recycle the CNN's weights trained in another similar task, in this case, the ImageNet challenge [61], and use them for the model they are developing. Fine-tuning is a further improvement over transfer learning because it enables the authors to re-train the neural network with the new weights. Usually, to save on computational costs, only some layers are trained. In the case of this paper, only the dense and the first convolutional layers are trained. The final accuracy obtained for the MallImg dataset is 98.82%.

In the same year, authors in [71], used an ensemble technique to harness the power of different feature extractors in the form of different CNNs, and other classifiers. The fine-tuned models employed in the study have been VGG16 and ResNet50. The final accuracy result on the MalImg dataset has been 99.50%, marking a big improvement over previous models and, most importantly, without the need for complicated image generation procedures. In general, researchers moved in two directions to improve the results of their model. The first group moved towards improving classification by using better-performing classifiers, mainly CNNs. The other group focused on better methodologies to extract features and create images with more information. Despite the resource-intensive nature and data requirements of deep neural networks, they hold great promise for future advancements in the field.

In [48] Naeem et al. address the malware detection problem in the Industrial Internet of Things (IIoT). Traditional malware detection techniques are unsuitable for IIoT devices due to their computing complexity and limited resources. The authors propose an architecture integrating malware visualization into a Deep Convolutional Neural Network (DCNN) model. The technique involves converting APK and PE files into color images, which the DCNN model then processes to extract dynamic image features of malware. The proposed method achieves high detection accuracy, with 97.81% accuracy on the IIoT dataset and 98.47% accuracy on the Windows dataset.

In [80], the authors propose a visualization method using Colored Label boxes (CoLab) to mark sections of Portable Executable (PE) files, emphasizing section distribution information in converted malware images. A classification method named Malware classification using CoLab image, VGG16, and Support Vector Machine (MalCVS) is developed. Experimental results using malware datasets from Virusshare and the Microsoft Malware Classification Challenge demonstrate MalCVS's effectiveness. The approach achieves average accuracies of 96.59% and 98.94% on the two datasets, respectively, indicating high accuracy in classifying malware into families.

In [68], the authors propose a method utilizing Convolutional Neural Networks (CNNs), transforming bytes files into grayscale and RGB images for classification. They introduce a new technique called B2IMG for file transformation and a CycleGAN-based data augmentation method to handle imbalanced data sizes among malware families. Testing on BIG2015 and DumpWare10 datasets showed significant improvements in classification accuracy, reaching 99.86% for

3.2. DEEP LEARNING

BIG2015 and 99.60% for DumpWare10, demonstrating the effectiveness of the proposed method.

In [64], the authors introduce a novel approach to malware detection by combining static and dynamic analysis techniques. Traditional methods have limitations, with static analysis being fast but unable to detect obfuscated malware variants and dynamic analysis being effective but slow and resource-intensive. The proposed method visualizes PE files as colored images, extracts deep features using a fine-tuned deep learning model, and detects malware using support vector machines (SVM). Integrating deep learning and machine learning eliminates the need for extensive feature engineering and domain knowledge. Experimental validation involving 12 machine learning and 15 deep learning models, conducted on various benchmark datasets, shows state-of-the-art performance. The proposed framework achieved an accuracy of 99.06% on the Malimg dataset and demonstrated statistical significance through rigorous testing methods.

In [19], Deng et al. propose a malware classification method, Malware Classification based on Three-channel Visualization and Deep learning (MCTVD), which employs small, uniform-sized malware images and a shallow convolutional neural network. Experimental results demonstrate MCTVD's effectiveness, achieving an accuracy of 99.44% on Microsoft's public malware dataset under 10-fold cross-validation.

The current landscape of malware image-based detection is undeniably dominated by deep learning methods, particularly Convolutional Neural Networks (CNNs). Researchers have extensively proven that these complex architectures, borrowed from diverse fields, or simpler ad-hoc models, consistently yield state-of-the-art performance levels. Continuous refinement of CNNs is ongoing, with ensemble techniques emerging as the preferred approach for achieving optimal results. Simultaneously, the exploration of Recurrent Neural Networks (RNNs) for malware visual classification is underway, with nascent yet promising efforts in visual transformers. The remarkable results attained by neural networks have established them as the preferred option for both researchers and businesses, replacing conventional machine learning models. This transition has unquestionably improved the effectiveness of malware categorization. However, it has also concealed the inherent transparency found in machine learning models, leading to the emergence of a "black-box" approach in the field.

Despite the remarkable successes, there remains a significant challenge re-

garding explainability. Few researchers have delved into the intricacies of why specific choices are made, such as selecting the best-performing neural networks for ensembling. This lack of clarity often leads to situations where combining the top-performing neural networks does not consistently generate the best final results. The field, therefore, grapples with the trade-off between superior performance and the understanding of the decision-making processes underlying these advanced models.

Table 3.1 contains the information about the classification of the model and the ideas explored in the classification phase. Only the main contribution of each paper is considered. The columns are: Paper - paper reference; D/C - if the paper talks about detection of malware [D] or classification in families [C]; DT/RF - if the paper uses a decision tree or random forest technique for classification; KNN - if the paper uses a KNN classifier; SVM - if the paper uses an SVM classifier; CNN - if the article uses a CNN as a classifier, it will be specified which CNN architecture is used if it is a known architecture; RNN - if the paper uses a RNN as a classifier, it will be specified which RNN architecture is used if available; Other ML - all the machine learning techniques other the KNN, DT/RF and SVM categories; Other DL - all the deep learning techniques that do not fit in the CNN and RNN categories (ex. vision transformers); Split - the training test split used to train the model, the notation is train/validation/test, [f] indicates the use of k-fold cross validation instead of leave one out; E - if the problems of explainability and interpretability is dealt with.

Table 3.1: Overview of Surveyed Research Papers in Malware Image-Based Detection: A Comparative Analysis of Methods and Approaches

Paper	D/C	DT/RF	KNN	SVM	CNN	RNN	Other ML	Other DL	Split	E
[26]	C				✓				10f	
[31]	C				✓				80/-/20, 80/20/-	Grad CAM
[65]	C	RF	✓	✓	✓					
[68]	C				DenseNet 121				80/-/20 10f	
[54]	D,C				VGG3, ResNet50				70/-/30	
[6]	D,C				✓				5f	
[4]	C				CNN, CNN- LSTM CNN- BiLSTM				80/-/20	
[42]	C				Attention + CNN				80/-/20	

					+ SPP					
[1]	C				EfficientNet B1				75/-/25	
[67]	C				ResNet50				75/-/25	
[2]	D,C				DenseNet 201				70/-/30	
[80]	C			✓	VGG16, Linear- SVM				10f	
[79]	C					BiLSTM			73/20/20, 62/20/20	✓
[32]	D				SEResNet 50	BiLSTM			80/10/10, 70/15/15, 60/20/20	✓
[7]	C				VGG16				50/-/50	
[28]	D				AlexNet				50/-/50	
[25]	D						XGBoost, GBT		10f	
[53]	C	RF	✓	✓					5f	
[83]	C				3conv+3FC				10f	
[40]	C	RF	✓	✓	VGG16, VGG19,				90/-/10	

					Inception V3, ResNet50				
[45]	C	DT	✓			BiLSTM		✓	80/-/20
[14]	C				Shallow-CNN				70/-/30
[84]	C				Inception V3 ResNet50 VGG16 Xception	✓			80/-/20 t-SNE
[58]	D				DenseNet EfficientNet Inception MobileNet ResNet VGG				75/-/25 t-SNE
[11]	C				✓				65/20/20 Ablation study
[10]	C				EfficientNet B1				70/-/30 ✓

[55]	C				EfficientNet B7				70/-/20	
[56]	C				ShuffleNet				70/-/30	
[19]	C				MCTVD				10f	
[44]	C				✓				80/-/20	
[62]	C	✓	✓	✓			✓		80/-/20	✓
[39]	C				✓				10f	
[41]	D,C	✓					✓			✓
[36]	D,C						✓		5f	✓
[64]	D,C			✓					60/20/20, 10f	
[46]	D	RF	✓	✓	Xception, Inception + ResNetV2 EfficientNet V2S		NB LR ,SGD		10f	✓
[9]	D			✓	1conv+ 2Dense	LSTM		4 DNN models	73/-/27	✓
[27]	C				✓				80/-/20	

4

Methodology

In this chapter, we delve into the methodology employed to meticulously analyze and address the research questions posed in the introduction 1. Our exploration commences with a comprehensive overview of the three datasets, detailing the methods employed for data acquisition and emphasizing the nuances that distinguish them. Proceeding from dataset elucidation, we meticulously explicate the rationale underpinning our selection of six Convolutional Neural Networks (CNNs) derived from five researched papers of high repute. Each CNN undergoes a thorough dissection, delving deep into its architectural intricacies and nuanced presentations. Following this dissection of the chosen models, we describe the strategic choices we made in replicating these models, illuminating the challenges faced and the decisions taken to ensure faithful replication. In our particular case, we tried to reproduce the model selected but the task was impossible for the lack of information present in the papers. For this reason, what we achieved in this thesis is more properly defined as replicability. In the rest of the document, the two words are used interchangeably keeping in mind that formally the work aims to replicate the models and not reproduce them

Transitioning from replicability considerations, we introduce the array of explainability tools utilized, predominantly focusing on CAMs, specifically HiResCAM [21] and GradCAM. Our exploration navigates through their unique implementations and distinctive features, shedding light on how these tools were instrumental in our analysis. Furthermore, we introduce the Structural Similarity Index (SSIM) metric and articulate its significance within the context of

4.1. DATASETS

our research. Furthermore, we present the cumulative heatmap for a particular malware class and cumulative-SSIM, essential metrics used to assess the CNNs' ability to distinguish between different malware families.

Towards the chapter's end, we clarify our rationale for adopting a Vision Transformer (ViT) as a benchmark classifier. Additionally, we provide an in-depth exploration of the ensemble architecture meticulously crafted to attain the final results, offering a holistic view of our research methodology and its intricacies.

4.1 DATASETS

The most commonly used datasets in the literature are Big2015 and MalImg. We decided to employ both of them to give meaningful results that prove the robustness of our research. We added to these two a new and original dataset, called VX-Zoo, that we created during the study of this project. The three datasets are presented in detail in the following parts.

4.1.1 MALIMG

MalImg stands out as the pioneering malware dataset exclusively designed for image visualization. Despite its age, it remains a cornerstone in the realm of Portable Executable malware image-based analysis, and it continues to be widely utilized. Introduced by Nataraj et al. in their seminal paper [49] in 2011, this dataset sparked significant interest in malware visual classification. Comprising 9,339 malware grayscale images categorized into 25 distinct classes, this dataset is publicly accessible.

Sourced from the Anubis analysis system [5], MalImg transforms each malware sample into a grayscale image by mapping individual bytes to pixels. Notably, the dataset contains a mix of packed and unpacked malware, with specific families such as *Yuner.A*, *VB.AT*, *Malex.gen!J*, *Autorun.K*, and *Rbot!gen* being packed using UPX [70]. Moreover, the dataset presents a notable imbalance, ranging from 2,949 samples belonging to the *Allapple.A* family to merely 80 samples in the *Skintrim.N* family. The specific composition of the dataset can be found in 4.1.

Family	Total	Type
Adialer.C	125	Dialer
Agent.FYI	116	Backdoor
Allaple.A	2949	Worm
Allaple.L	1591	Worm
Alueron.gen!J	198	Trojan
Autorun.K	106	Worm
C2LOP.gen!g	200	Trojan
C2LOP.P	146	Trojan
Dialplatform.B	177	Dialer
Dontovo.A	162	Trojan DL
Fakearean	381	Rogue
Instantaccess	431	Dialer
Lolyda.AA1	213	Stealer
Lolyda.AA2	184	Stealer
Lolyda.AA3	123	Stealer
Lolyda.AT	159	Stealer
Malex.gen!J	136	Trojan
Obfuscator.AD	142	Trojan DL
Robot!gen	158	Backdoor
Skintrim.N	80	Trojan
Swizzor.gen!E	128	Trojan DL
Swizzor.gen!I	132	Trojan DL
VB.AT	408	Worm
Wintrim.BX	97	Trojan DL
Yuner.A	800	Worm

Table 4.1: MalImg Dataset

4.1.2 Big2015

The Microsoft Malware Classification Challenge Dataset, often called Big2015 or MMCC [59], is the second most cited repository in the field. This dataset encompasses a robust collection of 10,860 malware executables, categorized into nine distinct families. Originating from the Microsoft Malware Classification Challenge in 2015, participants were tasked with training their classifiers, of varied types, on these 10,860 malware samples. The challenge required participants to predict on a separate set, the results of which were evaluated remotely.

Each executable within this dataset possesses a unique identification, comprising a 20-character hash value and an integer signifying the malware family name. To ensure the safety of researchers, the PE (Portable Executable) headers

4.1. DATASETS

of all binaries were removed, rendering them inert and harmless. Furthermore, the dataset includes metadata that captures diverse information extracted from the binaries, including function calls, strings, and more. Assembly files associated with each binary are also provided.

It is imperative to note that the Big2015 malware dataset wasn't originally tailored for malware visualization tasks. Consequently, we had to extract grayscale malware images from this dataset, employing a technique akin to that utilized by authors in [49] for MalImg. The Big2015 files provide binaries in hexadecimal form, enabling a swift translation into grayscale images through a straightforward script.

Similar to MalImg, Big2015 suffers from a pronounced imbalance in class distribution and encompasses obfuscated malware, notably presented within the *Obfuscator.ACY* class. Further details on the dataset composition are presented in 4.2.

Family	Total	Type
Gatak	1013	Trojan
Khelios_ver3	2942	Virus
Khelos_v1	398	Virus
Lollipop	2478	Adware
Obfuscator.ACY	1228	Obfuscated Malware
Ramnit	1541	Trojan
Simda	42	Stealer
Tracur	751	Trojan
Vundo	475	Trojan

Table 4.2: Big2015 Dataset

4.1.3 VX-Zoo

The final dataset employed in this study is an original creation meticulously explicitly crafted for this document. While Big2015 and MalImg possess unique strengths and weaknesses, we recognized the necessity of introducing a new and more contemporary malware set to establish additional benchmarks. The fundamental principles guiding the development of this malware dataset included:

1. **Incorporating New Malware:** Acknowledging the evolving malware landscape, we aimed to create a dataset that reflects the present-day scenario.

Both MalImg and Big2015 have started to show signs of aging, and testing on them may not provide a realistic perspective on new malware variants;

2. **Year-wise Identification:** Recognizing the significance of analyzing malware from the same class but across different years, we made an effort to include malware samples from diverse years whenever possible. This approach enables researchers to assess classifier performance when confronted with malware that has undergone mutations over time;
3. **Maintaining Imbalanced Characteristics:** Emulating the natural variation observed in the wild, where malware classes are not evenly distributed, we preserved the imbalanced nature of sample distribution. This decision aligns with the reality of cybersecurity threats, where certain malware classes are widespread, while others are more niche;
4. **Ensuring Variety in Malware Types:** Following the precedent set by previously cited datasets, we aimed for diversity in the types of malware classes represented. The goal was to create a classifier capable of identifying malware in general, rather than being specialized for a specific type. While ensuring variety, we maintained the ability of the classifier to distinguish between classes without overly emphasizing the differentiation of malware types;
5. **No Distinction Between Obfuscated and Non-obfuscated Malware:** Unlike some datasets that categorize all obfuscated malware into a single category, we opted not to distinguish between obfuscated and non-obfuscated malware. Our objective was for the classifiers to correctly identify malware classes, regardless of whether the malware was obfuscated or not;
6. **Increased Sample Size and Intermediate Class Count:** In our endeavor to enhance the datasets robustness, we aimed for a larger sample size than both Big2015 and MalImg. Additionally, we settled on a number of classes that fell between the extremes, striking a balance that allowed for a comprehensive evaluation of classifier performance.

By adhering to these core principles, we aimed to create a dataset that not only addresses the limitations of existing datasets but also serves as a valuable resource for researchers in the field of malware classification and analysis.

To compile our samples, we accessed the malware binaries from VirusShare [75], stored within Zip files available through torrents. Spanning a period of eight years from 2015 to 2022. For each year, we randomly chose two Zip files uploaded during that specific year, utilizing them as the foundational data source for that particular time frame.

The labeling system utilized was based on the Kaspersky[37] nomenclature, obtained from VX-Underground[77]. Leveraging these labels, we could discern and filter out non-PE (Portable Executable) malware and benign files present

4.2. REPLICATED CNN MODELS

within the zipped archives. This curation step was pivotal in ensuring the dataset’s quality and relevance.

Subsequently, we honed in on malware families that exhibited consistency across the eight years under consideration. Specifically, we focused on families that appeared at least three times, except for three classes (*Inject.ahqtx*, *Elkern.b* and *Vtfloder.ekl*). This criterion ensured the dataset’s integrity and relevance to contemporary malware classifications.

The resulting dataset showcased the diverse characteristics of malware families over time. Some classes exhibited sporadic spikes in specific years, while others displayed consistent patterns or concentrated occurrences in consecutive years. This dynamic evolution of malware families was cataloged and is detailed in Table 4.3 for a comprehensive reference.

Family	Total	'15	'16	'17	'18	'19	'20	'21	'22	Type
AntiFW.b	5777	2784	1419	1574	0	0	0	0	0	Trojan
Expiro.w	364	325	0	0	0	0	0	1	38	Virus
Lamer.cb	754	16	0	0	17	27	636	44	14	Virus
Parite.b	380	361	7	1	1	7	1	2	0	Virus
Sality.sil	855	809	7	1	10	15	11	2	0	Virus
Virut.ce	1612	849	11	5	7	15	647	39	39	Virus
WB.NA	908	885	0	0	2	1	1	12	7	Worm
Wabot.a	442	60	0	0	0	0	363	4	15	Bckdr
Vtfloder.ekl	465	465	0	0	0	0	0	0	0	Trojan
Alman.b	226	218	3	0	0	2	2	1	0	Virus
Elkern.b	219	219	0	0	0	0	0	0	0	Virus
Nimul.f	3044	37	0	0	4	2	7	2984	10	Virus
Debris.b	208	208	0	0	0	0	0	0	0	Worm
Inject.ahqtx	275	0	0	275	0	0	0	0	0	Trojan
VB.cuvt	130	0	0	0	0	0	120	4	6	Trojan

Table 4.3: VX-Zoo Dataset

4.2 REPLICATED CNN MODELS

4.2.1 SELECTION METHODOLOGY

To ensure the reliability of our research regarding the replicability of models in malware image classification, we meticulously chose representative architectures for implementation and testing. We opted for diversity in approaches over

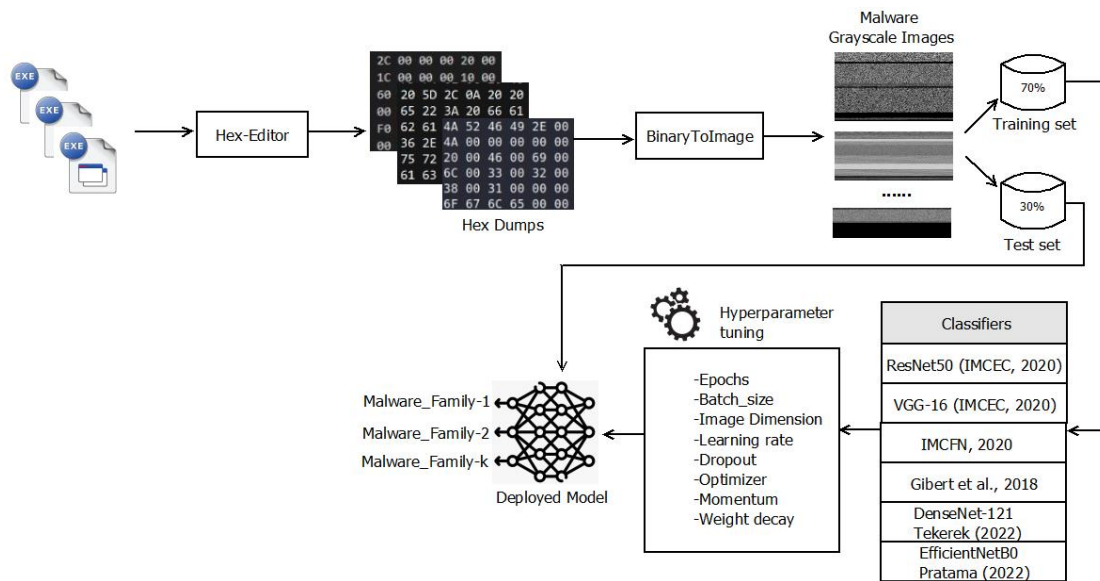


Figure 4.1: Diagram of the Replicability pipeline adopted in this thesis

rigid adherence to performance metrics, emphasizing a comprehensive representation of the field. The following principles guided our selection criteria:

- **High-Quality Papers:** The chosen papers were required to meet stringent quality standards, recognized in the field for their citations and prestigious venue. Furthermore, these papers had to lack glaring errors or deficiencies easily discernible by a malware analysis researcher;
- **State-of-the-Art Results:** The selected papers had to yield results comparable to the best outcomes prevalent at the time of publication, ensuring alignment with the contemporary state of the art;
- **Known Datasets:** The datasets used in these papers had to align with the established benchmarks for malware image visualization, namely Big2015 or Mallmg, ensuring a standardized comparison ground;
- **Variety of Architecture:** Diverse architectural approaches were a paramount consideration. We focused on incorporating different types of architectures for each model, encompassing a wide spectrum of CNNs commonly employed in state-of-the-art malware visualization. This included well-established models such as *ResNet50*, *VGG15*, *DenseNet121*, and *EfficientNetB0*, representing the core of our replicability research. Additionally, a slightly modified neural network akin to a well-known model (in our case, *IMCFN*, a modified version of *VGG16*) and a completely original CNN in the form of the *Gibert* CNN were incorporated;
- **Fine-Tuning and Transfer Learning:** The selected CNNs were deliberately chosen to include instances where fine-tuning and transfer learning were applied, enhancing the depth and versatility of our analysis;

4.2. REPLICATED CNN MODELS

- **Different Years:** To capture the evolutionary trajectory of neural network models, our selection spanned across various years. Consequently, we curated one CNN from 2018, three from 2020, and two from 2022, offering a nuanced insight into the progression of techniques over time;

4.2.2 CLASSIFICATION MODELS

In this section, we will present the selected six models and the papers in which they are described.

GIBERT 2018

We chose the model presented in [26] as a representative example of a well-established original CNN used in malware visual analysis. The authors meticulously elaborate on the architectural decisions behind the CNN, allowing us to effortlessly replicate the structure. The paper lacks specific hyperparameter values, despite the authors mentioning their tuning efforts. The authors test the model with two datasets, the already named Big2015 and MalImg, and input image sizes. We opted to focus our analysis on 256×256 images instead of 128×128 . This choice not only aligns more closely with other studies, which all used 224×224 images, but also results in superior classifier performance. The paper offers crucial information for reproducibilitythe model architecture. While the *absence of a comprehensive hyperparameter report hinders full replication*, the provided structure enables researchers to delve into the CNN's nuances. The authors calculate essential metrics like accuracy, precision, recall, and F1-score, along with supplying detailed confusion matrices. However, an opportunity is *missed concerning the training's evolution statistics*, which could enhance the paper's comprehensiveness.

IMCEC 2020

We selected the study by Vasan et al. [71] due to its publication in a highly regarded journal and its reputation as an exemplary ensemble architecture in the field. The authors meticulously outline their approach, detailing the implementation of transfer learning and fine-tuning, specifically focusing on SVMs and the fusion methodology employed for the results. The paper delves into the Principal Component Analysis (PCA) to reduce features extracted from

headless VGG16 and ResNet50 models. Crucially, the authors provide all hyperparameters, enhancing the reproducibility of their work. Despite initiating with well-known and easily implementable CNNs like *VGG16* and *ResNet50*, Vasan et al.'s model incorporates various intermediate steps in constructing the final model. While these complexities could pose challenges for replication, the authors meticulously document each step, eliminating potential pitfalls for researchers. In our replication efforts, we focused on implementing these intermediate neural networks, omitting alterations to the CNN heads or the addition of SVMs. This choice was motivated by our intention to assess the fundamental architectures themselves rather than the ensemble technique as a standalone entity. In this paper, it's clear that "VGG16" and "ResNet50," while perhaps not the most attention-grabbing aspects, unquestionably form the core of our model. Their faithful reproduction forms the foundation upon which the rest of the ensemble method stands. Consequently, these foundational models, with their profound influence, represent the most critical aspect of the architecture. Moreover, Vasan and colleagues enrich the reader's by providing a comprehensive array of metrics. In addition to those offered by Gibert 2018 and discussed previously, they include ROC curves and training statistics, fostering a deeper understanding of the model's performance dynamics.

IMCFN 2020

The model discussed in [72] emanates from the same research group responsible for IMCEC 2020, sharing commonalities that reflects in the paper and the provided information, notably the final result metrics and the training metrics. However, *certain hyperparameters*, notably dropout rates and momentum values, are featured in IMCEC 2020 but *remain conspicuously absent here*. It can be inferred that these parameters might echo their counterparts in IMCEC 2020 due to the resemblance between the CNN architectures. The original structure, denoted as IMCFN 2020, represents an adaptation of *VGG16* detailed by the authors, making its replication feasible. Our research strategy, pivoted towards grayscale images, deliberately ignoring the colormap augmentation. This deliberate exclusion aimed to ensure result comparability with other models under consideration, thereby refining our analysis and providing a more streamlined basis for evaluation. This model was also chosen because the fine-tuning is done by keeping frozen all the layers except the last three.

4.2. REPLICATED CNN MODELS

TEKEREK 2022

Among the papers we scrutinized, [68] stands out for its meticulous detailing of the utilized model and of the training and testing steps. The authors delve deeply into the structure of the *DenseNet121* CNN employed for classification, providing an exceptionally comprehensive account. This paper emerged as the most detailed among the ones analyzed, encapsulating all essential details necessary for a comprehensive replicability study. A notable facet of this paper is the authors' transparency in sharing the framework employed for neural network implementation. In alignment with our approach for the other models under consideration, we made a conscious choice not to implement the augmentation technique outlined in the paper, specifically Cycle GAN. The authors also present results for the model without augmentations, affording us the opportunity to continue our study along this trajectory. This decision, founded on methodological consistency, further solidifies the robustness of our research path and ensures a meaningful comparison of results across the analyzed models.

PRATAMA 2022

Replicating the model introduced by Pratama and team in [55] was a fairly straightforward process. Their study encompassed the entire EfficientNet CNN family. For our analysis, we narrowed our focus to *EfficientNetB0*. Even though it isn't the top-performing CNN, it shares a common image input size with other models we examined, namely 224×224 . The authors thoroughly explain the CNN architecture and techniques, making replication quite accessible. However, some hyperparameters like dropout, momentum, and weight decay weren't provided. It's possible the authors chose default values from their framework, hence the absence of specific numbers. The approach outlined in the paper is robust, despite it being a conference paper. It illustrates a trend observed in recent years, where the field of malware image visualization has gained prominence as a research area. Authors are increasingly meticulous in sharing implementation specifics and essential information necessary for accurate result replication. This trend highlights a growing commitment within the research community to enhance the transparency and replicability of their findings.

The differences and similarities between the models will be clarified and dissected in the experiments Chapter 6.

4.3 MODELS INTERPRETATION

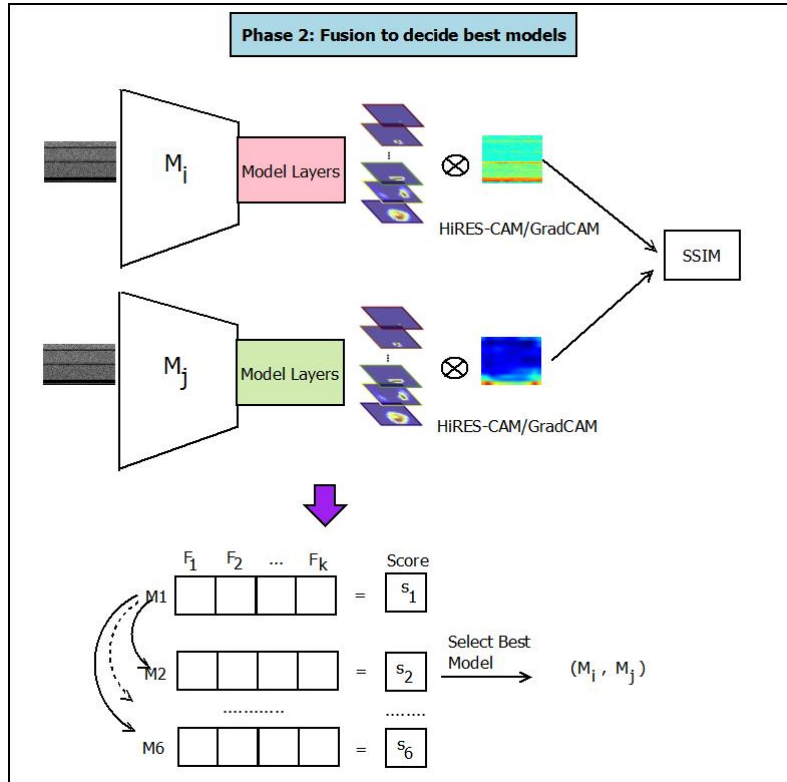


Figure 4.2: Diagram of the explainability pipeline adopted in this thesis

We aim to contribute significantly to the field by offering comprehensive insights into the operational mechanisms of Convolutional Neural Networks (CNNs) and highlighting distinctions between various models. This endeavor is crucial for several reasons, all of which hold immense significance:

- **Distinguishing Models based on Quality:** CNNs, while highly effective, often exhibit marginal differences in performance. It becomes essential to discern between these models based on their quality to make informed decisions about their application and efficacy;
- **Addressing the Black Box Nature:** CNNs, by their nature, function as black boxes, making them susceptible to manipulation if researchers lack clarity regarding their inner workings. Understanding the intricacies of these models is pivotal to ensuring their robustness and integrity in real-world applications;
- **Leveraging CNNs to their Full Potential:** Awareness of the limitations and weaknesses inherent in CNNs empowers researchers to extract maximum utility from these powerful tools. By recognizing and addressing these

4.3. MODELS INTERPRETATION

limitations, researchers can fine-tune CNNs, enhancing their accuracy and reliability across various tasks;

- **Addressing Specific Issues:** Even a highly effective CNN may yield satisfactory results overall but struggle with specific subsets of data. Understanding the intricacies of how the CNN functions within these subsets enables researchers to pinpoint underlying reasons. Armed with this knowledge, tailored solutions can be developed, bridging the performance gap for these specific samples and aligning it with the overall CNN performance.

In the context of explaining neural network models, especially those designed for image analysis, employing heatmaps, specifically Class Activation Maps (CAMs), emerges as an invaluable technique. CAMs, a well-recognized method in computer vision, facilitate the discernment of crucial regions within an image instrumental in predicting a specific class. Typically utilized for object localization, CAMs pinpoint the location of an identified object within an image. Our objective diverges slightly, as *we aim to comprehend the areas considered pivotal by CNNs during the prediction process* rather than identifying specific objects within bitmap images.

The rationale behind this approach is identifying recurring image patterns prevalent across all malware samples within a particular class. This understanding, in turn, aids in discriminating one class from another. The utilization of CAMs becomes pertinent in this context as they offer a nuanced insight into the neural network's decision-making process, shedding light on the specific image regions influencing the classification outcome.

4.3.1 CAM TECHNIQUES

The process of extracting a CAM heatmap from a CNN involves several steps. In instances involving simple CAMs, the neural network incorporates a final convolutional layer succeeded by a dense layer with softmax activation. In this thesis, we used two variants of CAM, GradCAM and HiResCAM with a particular focus on HiResCAM. Now, we will explain why we chose these two methods and why they are needed instead of classical CAM.

To visualize heatmaps by themselves, we also used the Kronecker product [29] to upscale the resulting image instead of the classical resizing method. The Kronecker product is a mathematical operation that combines two matrices to create a larger matrix. Given two matrices A and B , the Kronecker product, denoted by $A \otimes B$, results in a block matrix where each element of matrix

A is multiplied by the entire matrix B . The resulting matrix has dimensions $m \times n$ (where m and n are the dimensions of matrices A and B , respectively) and its elements are formed by multiplying each element of A with the entire matrix B .

GRADCAM

Gradient-weighted Class Activation Mapping (GradCAM) is an extension of the basic Class Activation Mapping (CAM) technique. While CAM provides valuable insights into the regions of an image that contribute to a particular class prediction, it has limitations. GradCAM addresses these limitations and provides more accurate and detailed visualizations.

Unlike CAM, GradCAM incorporates gradient information to weight the importance of the feature maps. It computes the gradients of the predicted class score with respect to the activations in the final convolutional layer. These gradients represent how sensitive the predicted class score is to changes in the feature map activations. It then calculates the importance of each feature map by taking the global average pooling (GAP) of the gradients as shown in Equation 4.1 where D_1 and D_2 represent the height and width of the feature map. This provides a weight for each feature map \mathbf{A}^f , refer Equation 4.2, indicating its contribution to the final prediction. The importance-weighted feature maps are then combined to create the GradCAM visualization. GradCAM highlights the important regions and provides a more localized and detailed visualization compared to CAM. By incorporating gradient information, it can focus on more precise regions within the activated feature maps, leading to sharper and more accurate visualizations of object boundaries and parts. In Figure 4.3, we display cumulative GradCAM images of example malware families.

$$\alpha_m^f = \frac{1}{D_1 D_2} \sum_{d_1=1}^{D_1} \sum_{d_2=1}^{D_2} \frac{\partial s_m}{\partial A_{d_1 d_2}^f} \quad (4.1)$$

$$\tilde{\mathcal{A}}_m^{GradCAM} = \sum_{f=1}^F \alpha_m^f \mathbf{A}^f \quad (4.2)$$

To summarize, the important improvements of GradCAM over CAM are:

- **Incorporation of Gradient Information:** As the main addition to CAM, GradCAM incorporates gradients, which CAM does not. This gradient-

4.3. MODELS INTERPRETATION

based approach allows GradCAM to capture fine-grained details and more precisely localizes essential regions.

- **Sharper Object Boundaries:** GradCAM can capture object boundaries more accurately because of the gradient weighting. It can provide detailed visualizations of object edges and parts, making it particularly useful for tasks like segmentation.
- **Reduction of Overemphasis on Discriminative Parts:** CAM tends to highlight large discriminative parts, which might include irrelevant background regions. GradCAM, by utilizing gradients, reduces the tendency to overemphasize these areas, leading to more focused and relevant visualizations.

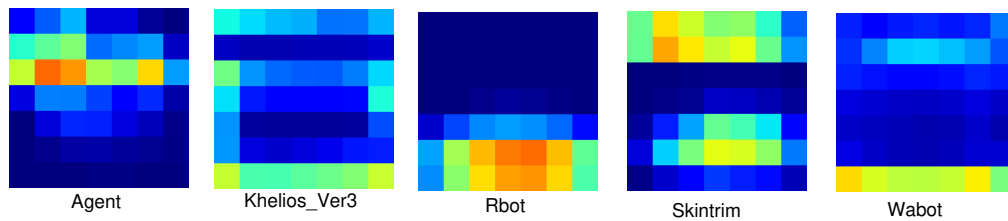


Figure 4.3: Example of GradCAM heatmap extracted from malware families

HiResCAM

In the realm of computer vision, Draelos et. al [21] investigated the differences of GradCAM and HiResCAM and proved that HiResCAM is a more general version of GradCAM, which is itself a more general version of CAM. HiResCAM, like GradCAM provides detailed attention information over the spatial dimensions of the input data, allowing for precise localization of important regions used by the model for predictions. The idea stems from GradCAM's global average pooling step, which sometimes blurs the attention maps, limiting their ability to show specific locations within the image that contribute to predictions. The fundamental limitation HiResCAM addresses lies in GradCAM's handling of feature map importance weights and their interaction with the feature map components. In Grad-CAM, each component of the final explanation must match the relative magnitudes and patterns of the feature map. This constraint leads to blurred visualizations as individual changes in the feature map (like rescaling or sign changes) are averaged out. *HiResCAM preserves rescaling and sign changes in the individual elements of the feature map*, ensuring that the high-resolution attention maps accurately reflect the model's computations.

By maintaining these fine-grained details, HiResCAM provides a visualization that is not only high-resolution but also faithfully represents the model’s decision-making process at the level of individual features. HiResCAM’s ability to preserve fine-grained information is crucial for tasks where a detailed understanding of the model’s reasoning is necessary. Practically, the formulas used by HiResCAM to calculate the attention map is Equation 4.3:

$$\tilde{\mathcal{A}}_m^{HiResCAM} = \sum_{f=1}^F \frac{\partial s_m}{\partial \mathbf{A}^f} \odot \mathbf{A}^f \quad (4.3)$$

Note how the formula is the same as the GradCAM, \mathbf{A}^f is again the feature map, but without the GAP multiplication step. The multiplication between the feature map and the gradients is done element-wise as indicated with \odot . Some examples of extracted heatmaps can be seen in Figures 4.4.

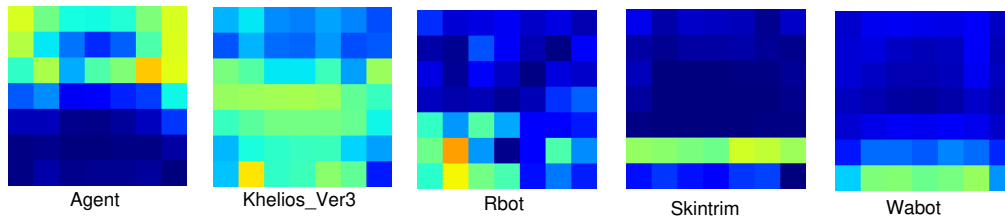


Figure 4.4: Example of HiResCAM heatmap extracted from malware families using EfficientNetB0

4.3.2 SSIM

In our approach to comparing heatmaps and leveraging their information, we employ the Structural Similarity Index (SSIM) [63], a perception-based model designed for comparing distinct images. Originally used to assess image degradations, distortions, and variations like contrast masking within different versions of the same image, SSIM evaluates perceived image quality by considering closely situated pixels and the varying visibility of areas (lightness) across different input image versions. SSIM values range from -1 to +1, reaching 1 only when two images are identical. Our method can be implemented with any deep learning model equipped with convolutional layers and capable of utilizing GradCAM or HiResCAM. This method was developed out of the imperative to identify similarities and differences within heatmaps. A mathematical model is

4.4. CLASSIFIER IMPROVEMENT

essential in objectively quantifying these distinctions, providing a structured basis for judges to assess similarity or dissimilarity. Particularly crucial in contexts involving images, such as malware images, which are inherently incomprehensible to human interpretation, this approach allows for precise and reliable evaluations, addressing the unique challenges posed by non-interpretable visual data.

We chose to employ SSIM on cumulative heatmaps derived for each class within our datasets. These cumulative heatmaps, formed by summing and averaging the heatmaps of individual samples within a class, offer valuable insights. They enable researchers to discern prevalent focus areas when classifying one family over another. The introduction of cumulative-SSIM allows researchers to assess both quantitatively and qualitatively the similarity between two cumulative heatmaps generated by distinct CNNs for the same family. This approach allows researchers to measure how these two neural networks diverge in assessing various elements within the same image during classification.

Analyzing cumulative SSIM along with the corresponding heatmaps empowers us to delve into the diverse strategies employed by the CNNs. This comprehensive approach allows us to draw insights and infer the reasons behind specific outcomes observed during the training and testing phases. By comparing these cumulative SSIM values and examining the associated heatmaps, we gain valuable perspectives into the underlying mechanisms guiding the CNNs' decision-making processes.

4.4 CLASSIFIER IMPROVEMENT

In our academic inquiry, we sought to extend the utility of heatmaps beyond their previously explained role as tools for elucidating Convolutional Neural Networks' (CNN) performance. Our objective was twofold: firstly, to demonstrate that these heatmaps possess the potential to enhance classifiers' interpretability, and secondly, to explore their applicability in improving the overall performance of diverse architectures. Specifically, our study aimed to illustrate CNN's capacity to augment not only its final prediction but also the predictive capabilities of alternative architectures. We accomplished this by integrating the CNN-produced heatmap into various techniques, such as ensemble methods, thereby empowering classifiers to make more informed decisions. Through this innovative approach, we aimed to showcase the ability of CNN guidance to

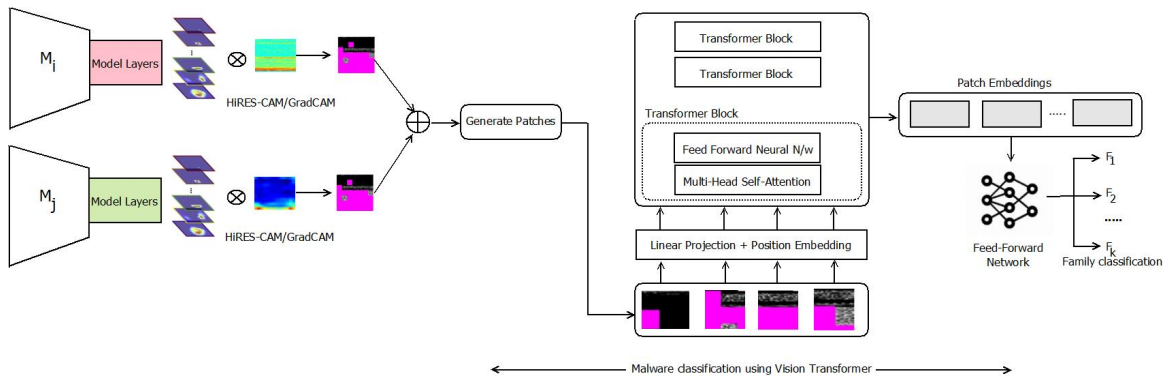


Figure 4.5: Diagram of the classification enhancement through HiresCAM masks implemented

substantially enhance the decision-making processes of classifiers, surpassing the outcomes attainable without such guidance.

4.4.1 MASKS

In our study, we introduce a novel approach involving the concept of a mask applied to malware image samples. The purpose of this mask is to selectively cover specific regions of a malware image, rendering them inaccessible to the classifier during the analysis process. This strategic masking prevents the classifier from being misled by anomalous patterns occurring in positions where they typically do not appear.

Our method for creating these masks involves a fusion of two heatmaps generated independently by two distinct Convolutional Neural Networks (CNNs). These individual heatmaps are integrated into a unified heatmap, through a logical OR , which serves as the basis for determining the mask's coverage areas. To establish the boundaries of the mask, we set a threshold value of 0.3 on the average heatmap. A heatmap can be conceptualized as a matrix of identical dimensions to the sample it is derived from. In this matrix, each pixel corresponds to a value ranging between 0 and 1. A pixel with a value of 1 signifies its consistent utilization by the CNN in making predictions, while a value of 0 implies non-usage. We chose a threshold of 0.3 to exclude only the segments of the image rarely utilized. Through empirical observation, we determined that this threshold, on average, provided us with the desired level of precision.

Pixels within the heatmap exhibiting values above this threshold indicate positions where the CNNs have identified relevant features essential for classi-

4.4. CLASSIFIER IMPROVEMENT

fication. In such instances, these pixels remain exposed, allowing the classifier to consider them during analysis. Conversely, pixels with values below the 0.3 threshold are considered non-essential for classification. Consequently, these pixels are masked, ensuring that the classifier does not factor in information from these regions, thus enhancing the classifier’s robustness against deceptive patterns. Through this innovative masking technique, we mitigate the risk of classifier manipulation, thereby advancing the accuracy and reliability of malware detection systems. Example of used masks can be seen in Figure 4.6.

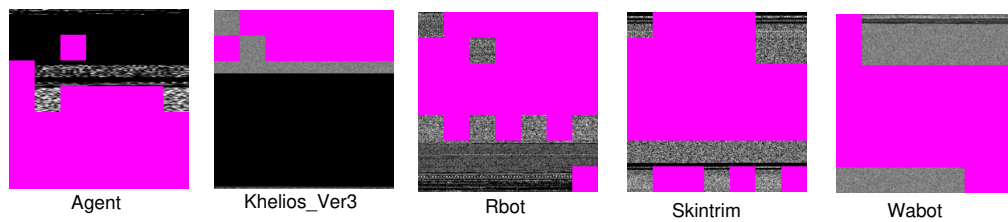


Figure 4.6: Example of mask of sample malware family

4.4.2 ViT CLASSIFIER

In our research, we utilized the newly created masked dataset to enhance the classification efficacy of a Visual Transformer model. This implementation followed the framework provided by [38], resulting in a minimalist architecture that incorporates a scaled-down version of the Visual Transformer initially introduced in [20].

It is imperative to clarify that our primary objective was not to achieve state-of-the-art classification performance, but rather to enhance the outcomes of the baseline classifier. An important question that might arise is why we opted for a classifier that, in certain datasets, exhibits clearly inferior performance compared to the CNNs under scrutiny. The rationale behind this choice stems from our intention to explore a less-considered yet intriguing model within the realm of malware image classification.

The decision to employ both transformers and CNNs is rooted in the fundamental disparities in how these models analyze images. By juxtaposing their methodologies, we paved the way for future researchers to integrate these distinct approaches, potentially filling the blind spots inherent in each method. Our study represents a foundational step in this direction, offering a preliminary approach. We anticipate that future research endeavors will refine the

methodologies and fundamental architectures underlying these experiments.

It is pertinent to mention that contemporary literature already showcases researchers achieving cutting-edge results with more intricate and comprehensive versions of the Visual Transformer architecture, surpassing the simplicity of the model utilized in this study [57]. Despite its relative simplicity, our work serves as a testament to the potential of these models when applied innovatively, pointing toward promising directions for future advancements in the field.

5

Experiments

In this chapter, we dive into our experiments aimed at answering the research questions introduced earlier. We start by explaining our experimental setup, including the tools and methods we used to replicate the selected models. Dealing with missing or unclear information from the original papers, we address these challenges effectively.

Next, we delve into the metrics employed for comparing our findings with those presented in the original papers. To ensure an equitable comparison, we make adjustments to the models to eliminate any biased advantages, ensuring a fair competition.

Moving on, we explore the HiResCAM and GradCAM heatmaps, aiming to understand how human researchers can interpret these visualizations. Our analysis focuses on identifying meaningful patterns within the heatmaps, offering insights into the models' inner workings.

To provide an objective perspective, we apply the Structural Similarity Index (SSIM) to quantify the differences between the CNNs when analyzing different malware families. This quantitative analysis helps us grasp the distinctions between the models and their approaches to malware image classification.

Finally, we introduce an innovative technique involving visual transformers. By incorporating masks generated from heatmaps, we evaluate how this approach can enhance the performance of this architectural paradigm. Our experiments led to the development of a prototype, demonstrating the potential of this new method to significantly improve results in malware image classification. List of experiments summarized:

5.1. REPRODUCTION

- **E1:** Tuning of the hyperparameters for the selected CNNs;
- **E2:** Replication of the model’s training;
- **E3:** Training on updated models;
- **E4:** Extraction of HiResCAM and GradCAM with corresponding analysis;
- **E5:** SSIM study on HiResCAM;
- **E6:** Training of ViT without enhancements;
- **E7:** Training of ViT with mask technique.

5.1 REPRODUCTION

5.1.1 ENVIRONMENT

All models were implemented in Keras utilizing a Tensorflow backend, in particular version 2.9.1. We had to use this Tensorflow version because of a known problem with the compatibility of the later version and the Efficient-NetB0 model. The training and evaluation processes were conducted on the Google Colaboratory Pro+ plan, operating within a Python 3.7 environment. The runtime used in Colab was always with High-RAM enabled and T4 GPU Hardware accelerator. Data preprocessing, image creation, heatmap extractions, and all the scripting to facilitate the training of multiple neural networks with multiple datasets were carried out on an ASUS TUF Dash F15, running Windows 11. The laptop operates 16 GB of RAM and 12th Gen Intel Core i7.

5.1.2 EVALUATION METRICS

In this thesis, we used accuracy, F1-score, recall, precision, false positive, and receiver operating characteristics curve (ROC) for performance evaluation of the implemented models. These evaluation metrics have been widely utilized in the research community to offer comprehensive assessments of various methods. In addition, we considered the prediction time to evaluate the time performance of the models. Another fundamental tool used to investigate the performance of the architectures is the confusion matrix. It summarizes the true positive, true negative, false positive, and false negative predictions, providing a clear

overview of the model's accuracy and error rates. Finally, for best performing models we also provide the graph to show the evolution of validation loss and accuracy.

- **True Positive:** The model correctly identifies that a sample belongs to the correct class among multiple classes;
- **True Negative:** The model correctly identifies that a sample does not belong to the wrong class among multiple classes;
- **False Positive:** The model wrongly identifies that a sample belongs to the wrong class among multiple classes;
- **False Negative:** The model wrongly identifies that a sample does not belong to the correct class among multiple classes.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \quad (5.1)$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (5.2)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (5.3)$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.4)$$

It has to be noted that in the context of an unbalanced dataset, where the number of instances in different classes significantly varies, accuracy can be misleading as an evaluation metric. As shown by the formula, accuracy simply measures the overall correctness of the model by considering both true positives and true negatives concerning all predictions. However, in unbalanced datasets, where one class considerably outnumbers the others, accuracy can be high even if the model predominantly predicts the majority class, ignoring the minority class completely. This situation can lead to a false sense of model performance. On the contrary, the F1-score, which combines both precision and recall, provides a more balanced measure, making it sensitive to the performance of minority classes. It provides a comprehensive evaluation, especially when identifying the minority class instances is crucial, like in a sensitive task such as malware classification.

5.1.3 HYPERPARAMETERS FINE TUNING (E1)

To replicate the results outlined in the state-of-art papers, the initial crucial step involves gathering substantial information about the environment used. Regrettably, among the five papers scrutinized, only two provided clear and detailed specifications regarding the software and hardware employed. This glaring lack of comprehensive data and significant disparities in the software and hardware configurations across the papers renders genuine reproduction unattainable. The absence of essential information in some papers further exacerbates this challenge.

In light of these limitations, the study presented in this thesis leans more toward replicability than strict reproducibility. Despite these hurdles, diligent efforts were made to establish a standardized environment for each implementation. Specifically, this was achieved by utilizing the Colab cloud environment offered by Google. This choice was made to ensure consistency and comparability in the experimental setups, even in the face of the disparities and gaps in the information provided by the analyzed papers.

Once we had compiled the necessary information about the environment and settled on our hardware and software configurations, the next step involved delving into the intricacies of the architectures we intended to implement. Our approach at this stage was to emulate the methodology of a resourceful researcher. We conducted a meticulous analysis of the papers, exhaustively scoured the internet for additional insights, and even reached out to the authors in an attempt to obtain access to their source code. Unfortunately, our attempts to engage with the authors yielded no response; none of the authors involved in this study provided any comments or assistance. Despite our proactive efforts, the outcome of our internet research proved to be equally disappointing. We uncovered only sporadic code snippets that lacked the depth of detail necessary for practical implementation. These sparse pieces of code, although available, lacked the detailed and practical information we needed to help significantly with our own implementations.

We pinpointed the essential hyperparameters required for the implementations. The detailed values are presented in Table 5.1 and Table 5.2. For unspecified hyperparameters, we adhered to the default settings provided by the Keras model we utilized whenever possible. We deviated from this approach only for hyperparameters specified in at least one paper. For these, we

developed a hyperparameter fine-tuning pipeline using Keras-tuner, especially `RandomSearchCV()` [52].

In conclusion, the hyperparameter values found in the papers or tuned are the following:

- **Learning Rate (LR)** is a hyperparameter determining the size of steps taken during optimization; it profoundly influences training by balancing the trade-off between rapid convergence and overshooting the optimal solution, with a too-high learning rate leading to divergence and too low leading to slow or stuck training progress;
- **Dropout** is a regularization technique where randomly selected neurons are ignored during training, reducing overfitting; it influences training by enhancing model generalization and preventing reliance on specific features, ultimately leading to more robust and accurate predictions;
- **Optimizers (Optim)** are algorithms that adjust the model’s weights to minimize the loss function; they influence training by determining the efficiency and speed of convergence, with different optimizers offering various strategies to navigate the complex, high-dimensional space of neural network parameters, affecting the model’s accuracy and training time;
- **Momentum (M)** is a parameter in optimization algorithms like SGD, enhancing convergence by adding a fraction of the previous update to the current update, which helps overcome local minima and speeds up learning by allowing the model to carry the momentum of earlier steps, leading to more stable and efficient training;
- **Weight Decay (WD)** is a regularization technique that penalizes large weights by adding a proportional term to the loss function, preventing overfitting by encouraging the model to favor smaller weights, ultimately leading to a more generalized and accurate neural network;

CNN	Dim	Dropout
VGG16	224×224	0.2
ResNet50	224×224	0.2
IMCFN	224×224	0.2
Gibert	256×256	0.3
EfficientNetB0	224×224	0.0
DenseNet121	224×224	0.3, 0.2

Table 5.1: Hyperparameters of the analyzed CNN regarding the architecture of the neural network. Values in bold have been tuned or inferred

From Tables 5.1 and 5.2, it is evident that among the six models examined, only one model required no hyperparameter tuning [68]. In the case of two other papers [71], although the training-test-validation split was not explicitly

5.1. REPRODUCTION

CNN	Batch	LR	Optim	M	WD	Split
VGG16	32	5x10e-6	SGD	0.9	5x10e-6	63/07/30
ResNet50	32	5x10e-6	SGD	0.9	5x10e-6	63/07/30
IMCFN	32	5x10e-6	SGD	0.9	5x10e-6	63/07/30
Gibert	32	5x10e-6	Adam	0.9	2x10-e3	10f/100
EfficientNetB0	64	1x10e-4	Adam	0.9	2x10e-6	68/12/20
DenseNet121	32	1x10e-3	SGD	0.9	1x10e-6	10f/20

Table 5.2: Hyperparameters of the analyzed CNN regarding the training of the neural network. Values in bold have been tuned or inferred

stated, all other essential hyperparameters for complete model implementation were provided. Another citation lacked only one hyperparameter [55], leading to speculation that the default value of the specific implementation library might have been utilized, although this remains unconfirmed without explicit author confirmation. Another model [72], attributed to the Vasan research group, was missing dropout and momentum definitions; however, fine-tuning efforts revealed consistent values with those found in other CNNs from the same research group, serving as additional validation. Interestingly, one paper did not provide any of the analyzed hyperparameters [26].

The analysis of hyperparameter tuning results indicates a significant gap in the information provided by authors, particularly in the papers under review. The findings suggest that crucial details essential for replicating a model are often missing. Additionally, when available, the provided information is frequently ambiguous, especially concerning aspects like the train-validation-test split. Consequently, researchers aiming to replicate a state-of-the-art malware image classification model face challenges as they need to adapt, infer, or fine-tune a considerable portion of the required hyperparameters on their own.

5.1.4 RESULT REPLICATION (E2)

In Table 5.3 and Table 5.4 are presented the results for the replication study we did in during this thesis. The models provided by [55] and [68] were tested on Big2015 while the models from [71] and [72] were tested on Maling. The architecture provided by [26] was tested on both. We can also see how not

CNN	Accuracy		Precision		Recall		F1 Score	
	P	R	P	R	P	R	P	R
VGG16	0.984	0.982		0.957		0.952		0.954
ResNet50	0.982	0.986		0.963		0.962		0.962
IMCFN	0.978	0.982	0.982	0.954	0.981	0.952	0.982	0.953
Gibert	0.948	0.998		0.965		0.962	0.958	0.962

Table 5.3: Performance metrics of replicated CNN models trained on MalImg. P represents the value found in the paper while R represents the replicated value.

CNN	Accuracy		Precision		Recall		F1 Score	
	P	R	P	R	P	R	P	R
EfficientNetB0	0.992	0.982	0.965	0.960	0.965	0.954	0.965	0.955
DenseNet121	0.997	0.985	0.950	0.951	0.960	0.950	0.960	0.950

Table 5.4: Performance metrics of replicated CNN models trained on Big2015. P represents the value found in the paper while R represents the replicated value.

all the analyzed metrics are presented in some papers. Our findings revealed disparities between the results obtained through implementation and those reported in the papers, with deviations ranging from -0.029 to $+0.004$ when considering the F1 score. It is evident that the replicated models consistently underperformed in comparison to the claims made in the original papers. This disparity underscores the challenge of replicating a model solely based on shared hyperparameters, even in cases where these parameters are widely available. The difficulty in accurate replication arises due to the absence of both code and detailed architectural information. This situation emphasizes the necessity for comprehensive code sharing as the sole viable solution to enable successful model reproduction within the field.

We share the training graphs of some of our replicated models showing the evolution of training accuracy and validation accuracy. 5.1 5.2 5.3 5.4.

5.1.5 TRAIN TEST VALIDATION PROBLEM (E3)

A significant challenge that surfaced during training preparation was the inconsistency in dataset splitting and testing methodologies. This issue was not unique to our study; Table 3.1 in Chapter 3 illustrates that many other analyzed papers faced similar or worse problems.

For instance, in IMCEC and IMCFN, the authors partitioned the dataset into train/test and train/validation sets, respectively. This introduced ambiguity

5.1. REPRODUCTION

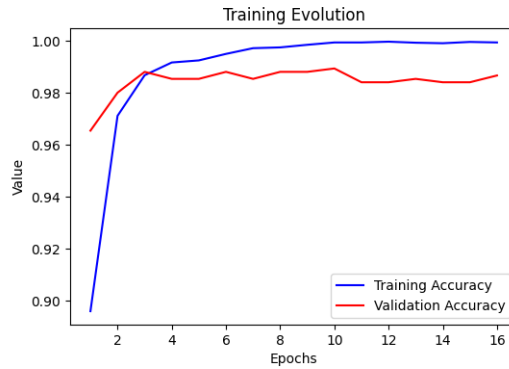


Figure 5.1: DenseNet121 [68] Training evolution on Big2015

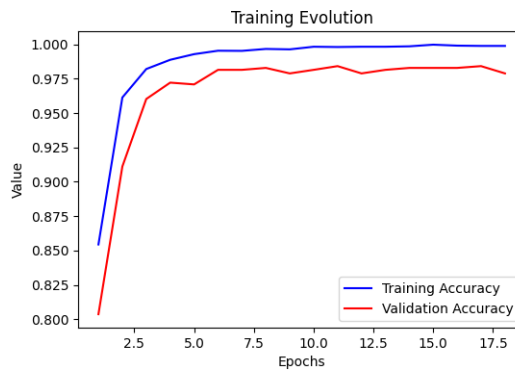


Figure 5.2: EfficientNetB0 [55] Training evolution on Big2015

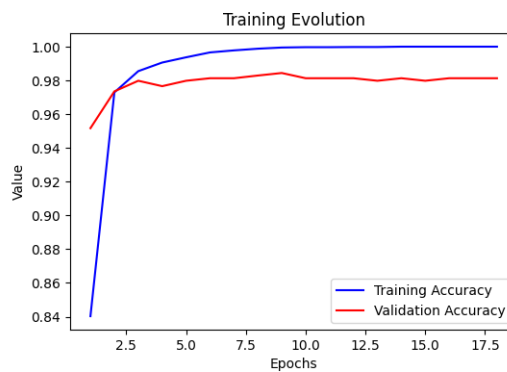


Figure 5.3: VGG16 [71] Training evolution on Mallng

regarding the authors' definition of "validation." If validation referred to the final "test" split, a common but unfortunately vague convention in literature, it raised questions about the appropriate size of the validation set. On the other hand, if the authors intended a distinct validation split, the question raised is about how to assess the final trained model. This confusion was particularly

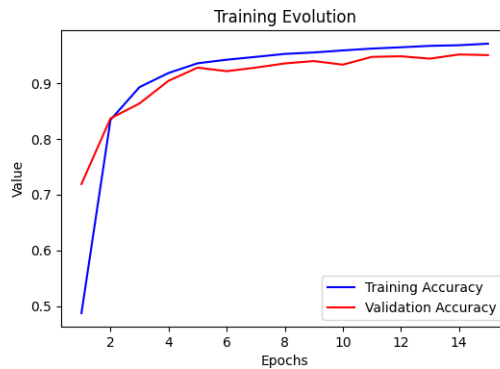


Figure 5.4: Gibert [26] Training evolution on Mallimg

perplexing given that the authors of both papers were the same. To simulate a realistic approach, we opted for a 70/30 split for training and testing, reserving 10% of the training set for validation.

The attention to detail in [55] and [68] papers greatly facilitated the training and testing phases, specifying in detail the train/validation/test splits and the training methodologies.

In the examined older paper, [26] employed an intriguing testing approach, utilizing both 10- and 5-fold cross-validation for their model. However, what set them apart was their final evaluation method. Instead of employing a leave-one-out test set or showcasing results from one of the folds, they chose to evaluate the model on the entire dataset. While this approach allowed a comprehensive assessment of all samples, it introduced a fairness concern. Testing the model on a dataset largely overlapping with the training set raised questions about the usefulness of their findings.

This discovery raised doubts about other papers that lacked explicit specifications of the complete three-way split (train, test, validation) and only defined two of these categories. While it's not inherently wrong for an author to assess their model on the total dataset it was trained on, clarity in the methodology is crucial. Authors should transparently communicate their testing approaches to maintain the integrity of their research findings.

After successfully replicating the results of the analyzed architectures, we optimized the models to ensure fair and consistent comparisons. Specifically, we adjusted the Gibert architecture [26] by adding two convolutional layers, aligning the final layer's dimensions with the rest of the considered models. While this modification had a minor impact on the model's performance, it

5.1. REPRODUCTION

CNN	Big2015			
	Accuracy	Precision	Recall	F1 Score
VGG16	0.965	0.940	0.922	0.930
ResNet50	0.979	0.959	0.944	0.951
IMCFN	0.973	0.954	0.946	0.950
Gibert	0.965	0.919	0.924	0.921
EfficientNetB0	0.983	0.965	0.957	0.961
DenseNet121	0.985	0.972	0.958	0.965

Table 5.5: Performance metrics of CNN models with split 70/30 on Big2015

CNN	Mallmg			
	Accuracy	Precision	Recall	F1 Score
VGG16	0.982	0.957	0.952	0.954
ResNet50	0.986	0.963	0.962	0.962
IMCFN	0.982	0.954	0.952	0.953
Gibert	0.975	0.922	0.925	0.923
EfficientNetB0	0.990	0.974	0.973	0.973
DenseNet121	0.984	0.963	0.956	0.956

Table 5.6: Performance metrics of CNN models with split 70/30 on Mallmg

enabled meaningful comparisons when extracting heatmaps, as elaborated in the subsequent sections.

Crucially, we maintained consistency by training all models using identical train/validation/test splits across all datasets. In particular, we used the split used originally for IMCEC and IMCFN. This approach ensured comparable results and eliminated external influences, guaranteeing a rigorous evaluation of the models.

The conclusive findings regarding the replicability and reliability of the analyzed models found in Tables 5.5, 5.6, and 5.7 show how the metrics measured change, even significantly when hyperparameters, in this case training-validation-test split, are changed.

Our investigation revealed a consistent pattern in the superior performance of specific models across various datasets. Notably, DenseNet121 and EfficientNetB0 consistently outperformed other models in all datasets analyzed,

CNN	VX-Zoo			
	Accuracy	Precision	Recall	F1 Score
VGG16	0.945	0.911	0.909	0.910
ResNet50	0.945	0.923	0.908	0.913
IMCFN	0.940	0.903	0.894	0.898
Gibert	0.950	0.920	0.915	0.916
EfficientNetB0	0.969	0.953	0.959	0.955
DenseNet121	0.966	0.955	0.956	0.955

Table 5.7: Performance metrics of CNN models with split 70/30 on VX-Zoo

CNN	Datasets		
	Big2015	MalImg	VX-Zoo
VGG16	0.302	0.585	0.437
ResNet50	0.464	0.253	0.399
IMCFN	0.422	0.418	0.401
Gibert	0.395	0.669	0.689
DenseNet121	0.459	0.344	0.231
EfficientNetB0	0.265	0.269	0.226

Table 5.8: Prediction timings of neural networks on the 3 datasets. Values in seconds

reaffirming their status within the hierarchy of high-performing models. Focusing on these two models, it became evident that even minor alterations in fundamental hyperparameters, such as the training-validation-test split, could yield noteworthy impacts. For instance, within our implementation, a single change in this parameter resulted in a notable increase of +0.011 and +0.01 in the F1 score. These changes are substantial, considering that all classifiers under consideration exhibited precision values within the range of 0.1 to 0.15.

We also tested all the models on all the available datasets in order to continue with our research providing in particular result for our new dataset VX-Zoo 5.7

We provide the confusion matrices and ROC curves of the best models for each dataset, in particular DenseNet121 for Big2015 Figures 5.6 5.8 and VX-Zoo Figures 5.7 5.10, and EfficientNetB0 for MalImg Figures 5.5 5.9. The results shown by this metrics are in line with what is found in the literature and provide brand new results for the VX-Zoo dataset. Taking into consideration Vx-Zoo we can see how even the best model struggles to distinguish between classes 9, 10 and 11 a situation analogous to Big2015, where two classes, in particular Swizzor.gen!E and Swizzor.gen!I cause the most trouble to researchers and their

5.2. EXPLAINABILITY

models.

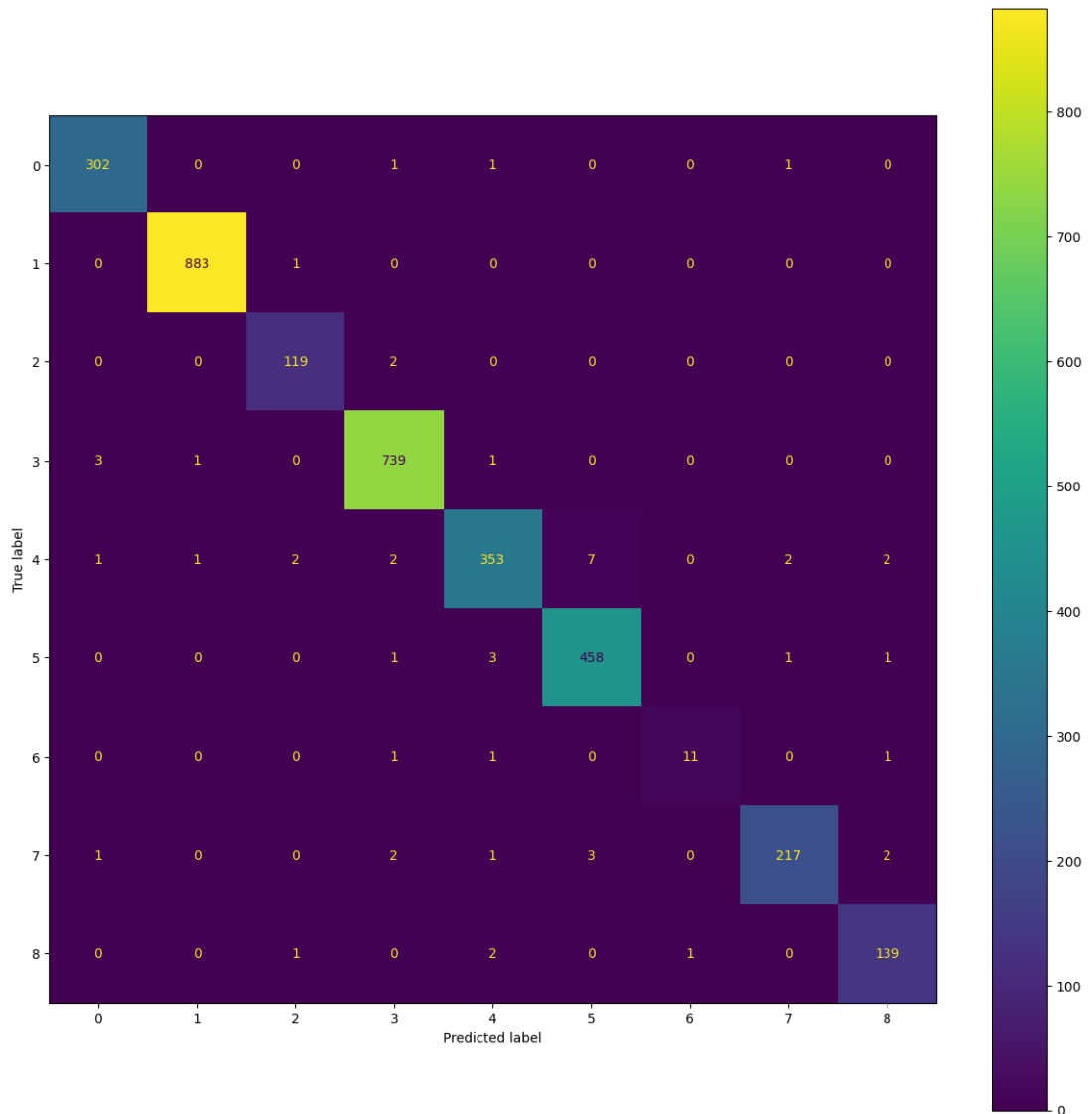


Figure 5.5: Confusion matrix obtained from the training of DenseNet121 on Big2015 dataset

5.2 EXPLAINABILITY

5.2.1 EXTRACTION OF CAMs (E4)

The objective of this section is to differentiate how various CNNs made use of image patterns. To accomplish this, we utilized the GradCAM and

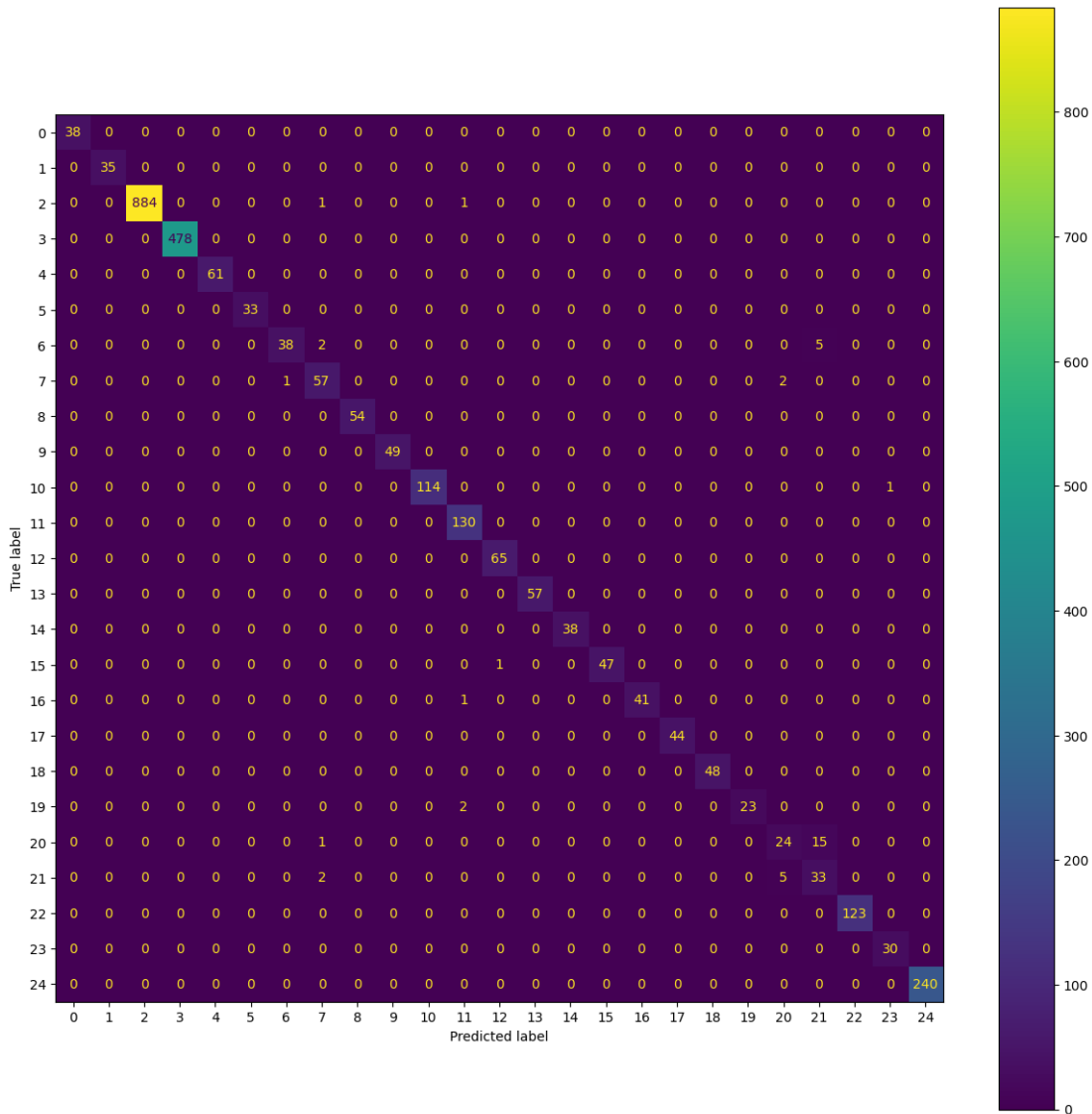


Figure 5.6: Confusion matrix obtained from the training of EfficientNetB0 on MallImg dataset

HiresCAM techniques mentioned earlier. We generated cumulative heatmaps for each family across the datasets, enabling us to comprehensively analyze heatmaps produced by any of the neural networks under examination for a selected malware family.

Generating cumulative heatmaps for each class was crucial for understanding whether diverse neural networks consistently utilized specific regions to classify particular samples. By employing these techniques and meticulously studying the heatmaps, we gained valuable insights into the underlying patterns and decision-making processes of the CNNs under scrutiny.

5.2. EXPLAINABILITY

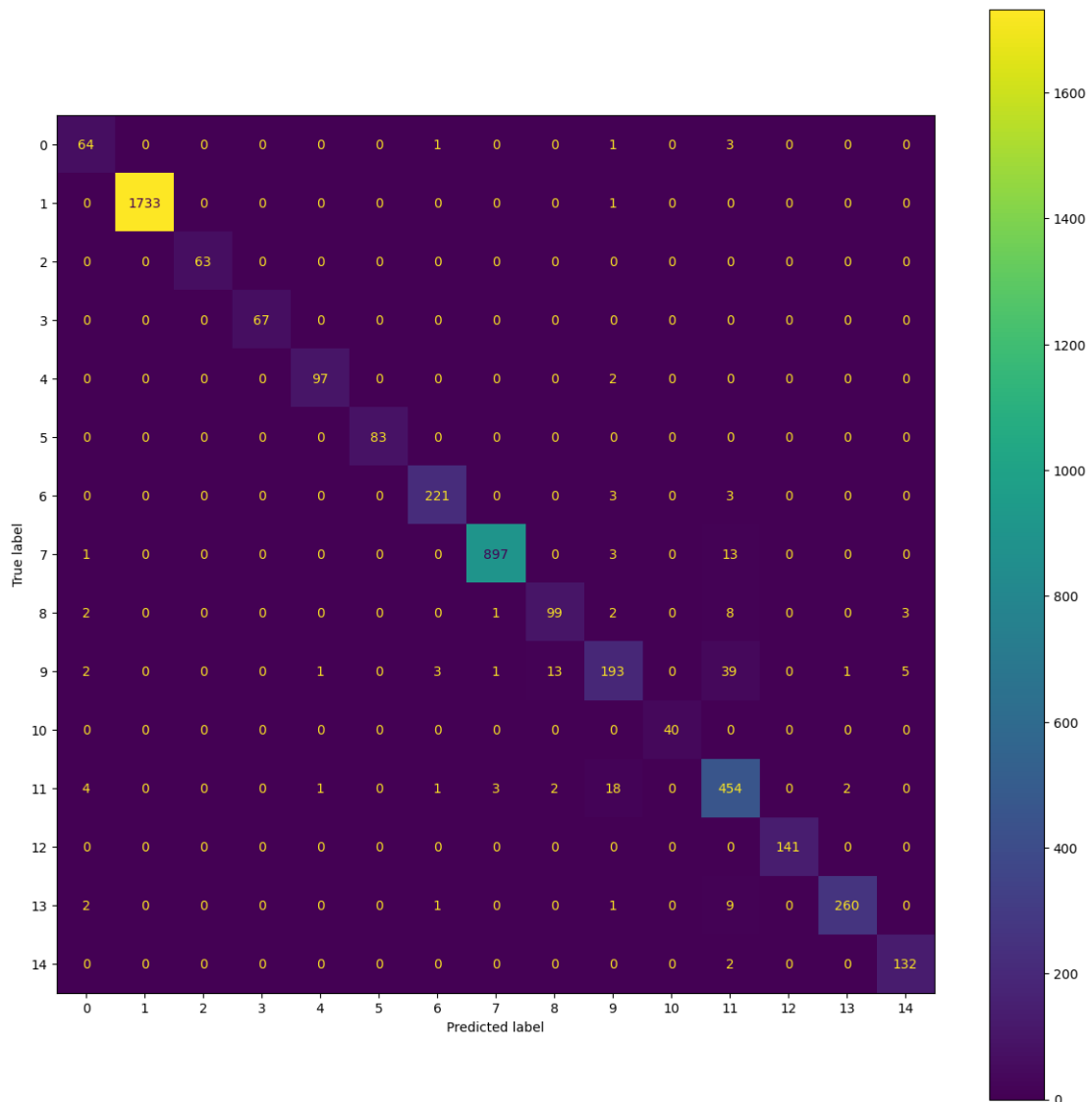


Figure 5.7: Confusion matrix obtained from the training of DenseNet121 on VX-Zoo dataset

The decision to employ both GradCAM and HiResCAM stems from GradCAM’s widespread recognition as the most prominent Class Activation Mapping (CAM) technique with extensive research backing its application. GradCAM’s popularity is due to its proven usefulness across various fields, producing visually smoother images that are generally easier for humans to comprehend. However, in the domain of malware analysis, where images are inherently cryptic even to experts, the clarity offered by smoother heatmaps is not as impactful. Malware images are almost incomprehensible for humans, making the interpretability of smooth heatmaps less relevant, especially when compared to

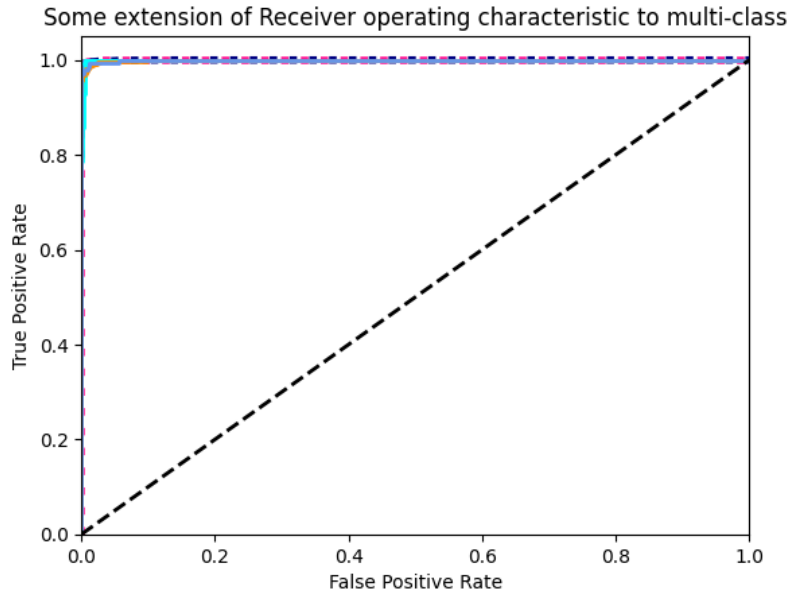


Figure 5.8: ROC curve obtained from the training of DenseNet121 on Big2015 dataset

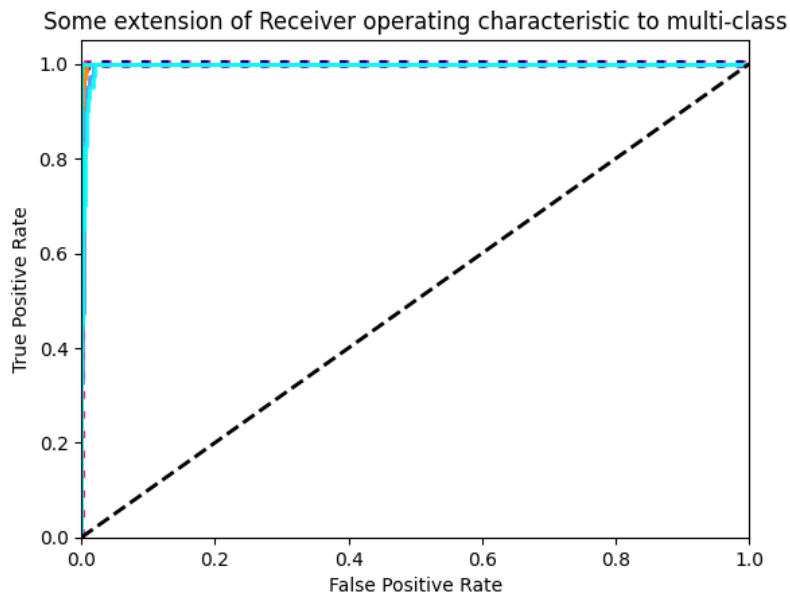


Figure 5.9: ROC curve obtained from the training of EfficientNetB0 on MallImg dataset

situations involving everyday object recognition.

In response, we introduced HiResCAM as a complement to GradCAM. HiResCAM shares similarities with GradCAM but eliminates the averaging step, providing explanation heatmaps that, although less smooth, faithfully represent

5.2. EXPLAINABILITY

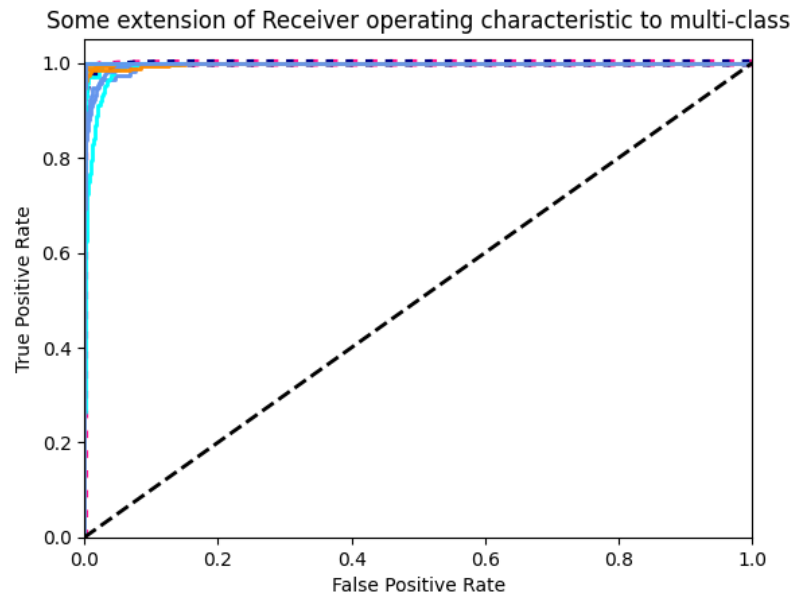


Figure 5.10: ROC curve obtained from the training of DenseNet121 on VX-Zoo dataset

the specific areas effectively utilized by the CNNs under scrutiny. Our primary aim in this project was to employ HiResCAM for faithful explanations of the neural network models we scrutinized. Nonetheless, we intentionally paired the HiResCAM analysis with GradCAM to showcase the substantial variability in outcomes that different CAM techniques can produce.

This strategic pairing of techniques serves as a word of caution to researchers, urging them to meticulously select the appropriate explanation technique, especially in fields like malware visual analysis, where even experts find it challenging to definitively evaluate an explanation heatmap. By showcasing the disparities between GradCAM and HiResCAM outputs, we emphasize the necessity of careful consideration when choosing CAM techniques, ensuring accurate interpretations in complex and cryptic domains like malware analysis.

The heatmaps produced are 7×7 matrices that represent the areas of the feature map generated considering the last convolutional layer of the CNN. The cumulative HiResCAM heatmaps of some of the best performing CNNs are presented in A.1, A.2 and A.3. The analysis of heatmaps generated through HiResCAM, presented in A, reveals discernible patterns, particularly within specific malware families. For instance, in the *AntiFW.b* family, DenseNet121 and ResNet50 predominantly utilize the upper portion of the malware image to predict this specific class, indicating a consistent strategy. In contrast, Ef-

efficientNetB0 appears to employ a different approach for accurate classification within the same family. Shifting focus to the previously mentioned *Swizzor.gen!E* and *Swizzor.gen!I* families, the cumulative heatmaps exhibit striking similarity. This observation corroborates our findings in the confusion matrix analysis conducted with DenseNet121, reinforcing the reliability of our results.

An intriguing observation arises when examining the operation of the same CNN across different classes within the same dataset. For instance, in the case of the *Rbot!gen.C* and *Yuner.A* classes from the MalImg dataset, ResNet50 consistently directs its attention to the lower section of the image. This behavior highlights the classifier ability to concentrate its focus on a specific subset of an image, disregarding surrounding elements. This focused attention allows the network to accurately classify the sample by concentrating on the distinctive patterns unique to the specific family, thus showcasing the network’s capacity to discern crucial features within localized sections of the input data.

5.2.2 SSIM (E5)

We computed each family’s cumulative Structural Similarity Index (cumulative-SSIM) value in every dataset. Tables 5.12, 5.13 and 5.14 provides a summary of these results for HiResCAM heatmaps, while tables and 5.9, 5.10, and 5.11 provides a summary of these results for GradCAM. The certain elements in the tables are depicted in boldface to represent two CNN with low cumulative-SSIM, while elements in italics represent higher cumulative-SSIM. We can see the difference in cumulative-SSIM produced by using the two different methods on the same dataset and CNNs.

Cumulative-SSIM values are determined through a systematic procedure. Initially, for each class within a dataset, a cumulative heatmap is generated. Subsequently, the researcher computes Structural Similarity Index (SSIM) values between the heatmaps generated by different CNNs belonging to the same class. This process yields a specific value, denoted as single-class-SSIM, signifying how distinctively the analyzed CNN pairs interpret the same class. To obtain the cumulative-SSIM value between two CNNs, all the individual single-class-SSIM values are summed. Each single-class-SSIM value corresponds to a unique pair of CNNs and represents the disparity in their interpretations across all classes present in the dataset.

The cumulative SSIM values offer valuable insights into the comparability

5.2. EXPLAINABILITY

EfficientNetB0	1,5463				
Gibert	2,169	1,617			
IMCFN	1,852	1,585	2,215		
ResNet50	3,357	1,405	2,394	1,828	
VGG16	1,975	1,81	2,206	3,446	1,762
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.9: Cumulative GradCAM SSIM values extracted from the VX-Zoo dataset

EfficientNetB0	0.19				
Gibert	0.6	0.612			
IMCFN	1.353	0.86	0.808		
ResNet50	2.293	0.63	0.93	1.925	
VGG16	1.277	1.159	1.02	2.614	1.705
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.10: Cumulative GradCAM SSIM values extracted from the Mallimg dataset

EfficientNetB0	2.447				
Gibert	1.225	1.865			
IMCFN	1.786	1.568	1.584		
ResNet50	2.137	2.528	1.419	1.46	
VGG16	1.895	2.037	2.269	3.304	1.753
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.11: Cumulative GradCAM SSIM values extracted from the Big2015 dataset

EfficientNetB0	1.792				
Gibert	2.829	2.31			
IMCFN	2.152	1.921	3.04		
ResNet50	2.925	1.344	2.688	1.975	
VGG16	2.121	2.219	3.262	2.962	1.891
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.12: Cumulative HiResCAM SSIM values extracted from the VX-Zoo dataset

EfficientNetB0	2.23				
Gibert	2.302	2.378			
IMCFN	1.997	2.08	2.651		
ResNet50	2.434	2.864	2.122	1.775	
VGG16	1.791	1.717	2.282	2.398	1.635
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.13: Cumulative HiResCAM SSIM values extracted from the Mallimg dataset

EfficientNetB0	1.982				
Gibert	2.701	2.396			
IMCFN	2.382	2.318	3.273		
ResNet50	2.193	1.813	2.404	1.909	
VGG16	2.32	2.02	2.906	2.848	1.939
	DenseNet121	EfficientNetB0	Gibert	IMCFN	ResNet50

Table 5.14: Cumulative HiResCAM SSIM values extracted from the Big2015 dataset

of CNNs’ heatmaps. A higher value indicates a greater similarity between the heatmaps generated by two CNNs, signifying a shared interpretation of malware families. Conversely, a lower value signifies divergent interpretations between the heatmaps, indicating dissimilar comprehension of the malware families. This crucial distinction will be explored further in the subsequent chapter through the implementation of the previously discussed masking technique for Vision Transformers (ViTs).

5.3 ATTENTION ENSEMBLE ARCHITECTURE

The heatmaps obtained from HiResCAM and GradCAM have the potential to facilitate an exploration of a neural network’s internal processes. This exploration, in turn, can offer valuable insights for crafting more advanced networks that excel in both explainability and performance. Gaining a grasp of the intricate internal mechanisms of a complex neural network can provide context to the CAM results, highlighting areas in which architectural enhancements are required.

We introduced another architecture, namely the ViT, that recently appeared in the literature related to malware analysis [57] to probe the potential of an innovative technique that, starting from CAM heatmaps might be a future help-

5.3. ATTENTION ENSEMBLE ARCHITECTURE

ful tool for the malware analyst. The result of the ViT testing together with the prediction using the masks presented in this chapter, can be found in Table 5.15

5.3.1 BASELINE ViT IMPLEMENTATION (E6)

As a first step to gauge the potential of the masking methodology exploiting HiResCAM heatmaps, we implemented a barebone version of the visual transformer. The results of this classifier on our three datasets will work as the baseline on which we will benchmark our ensemble methodology.

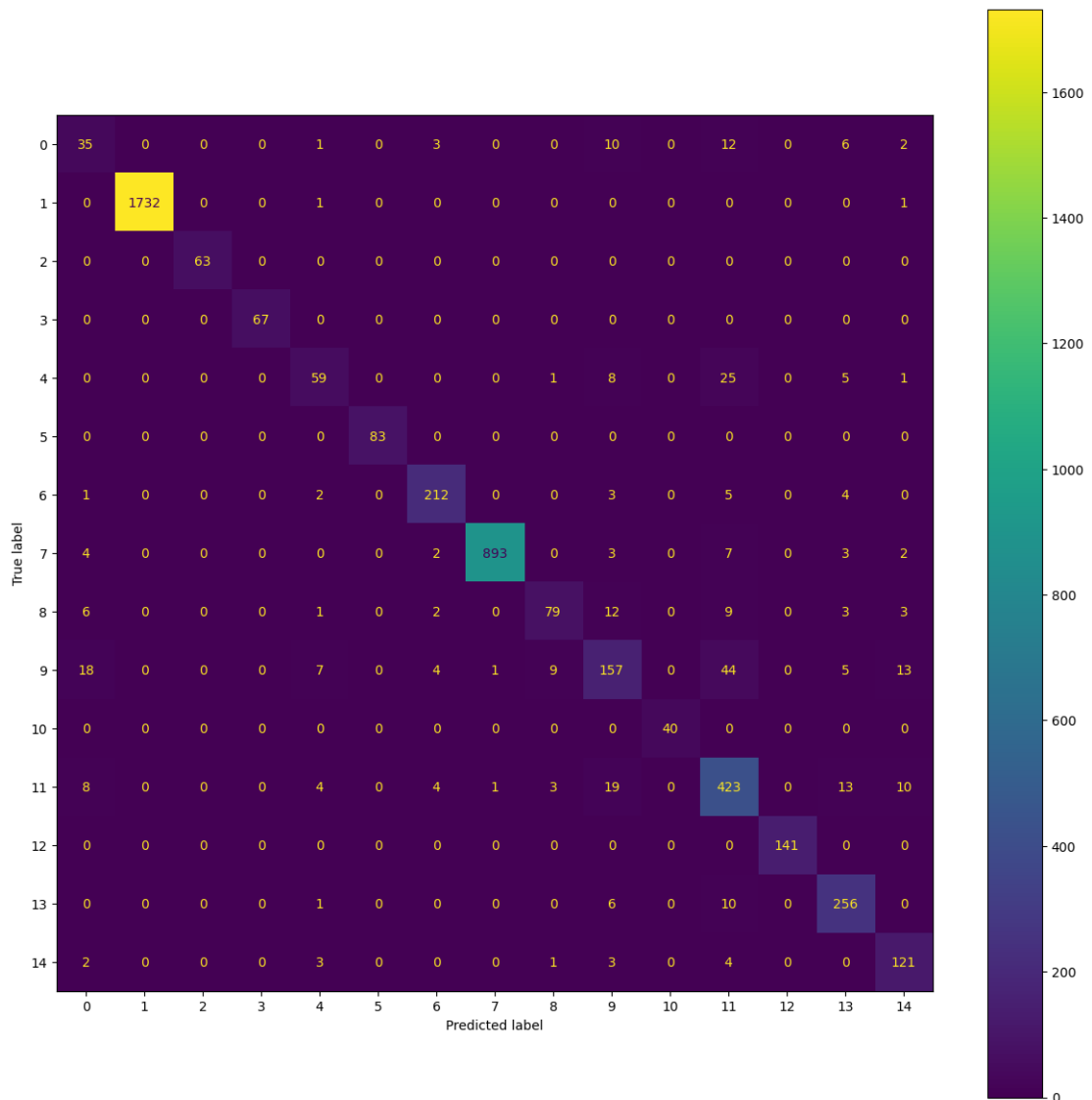


Figure 5.11: Confusion matrix obtained from the training of ViT on VX-Zoo dataset without mask augmentation.

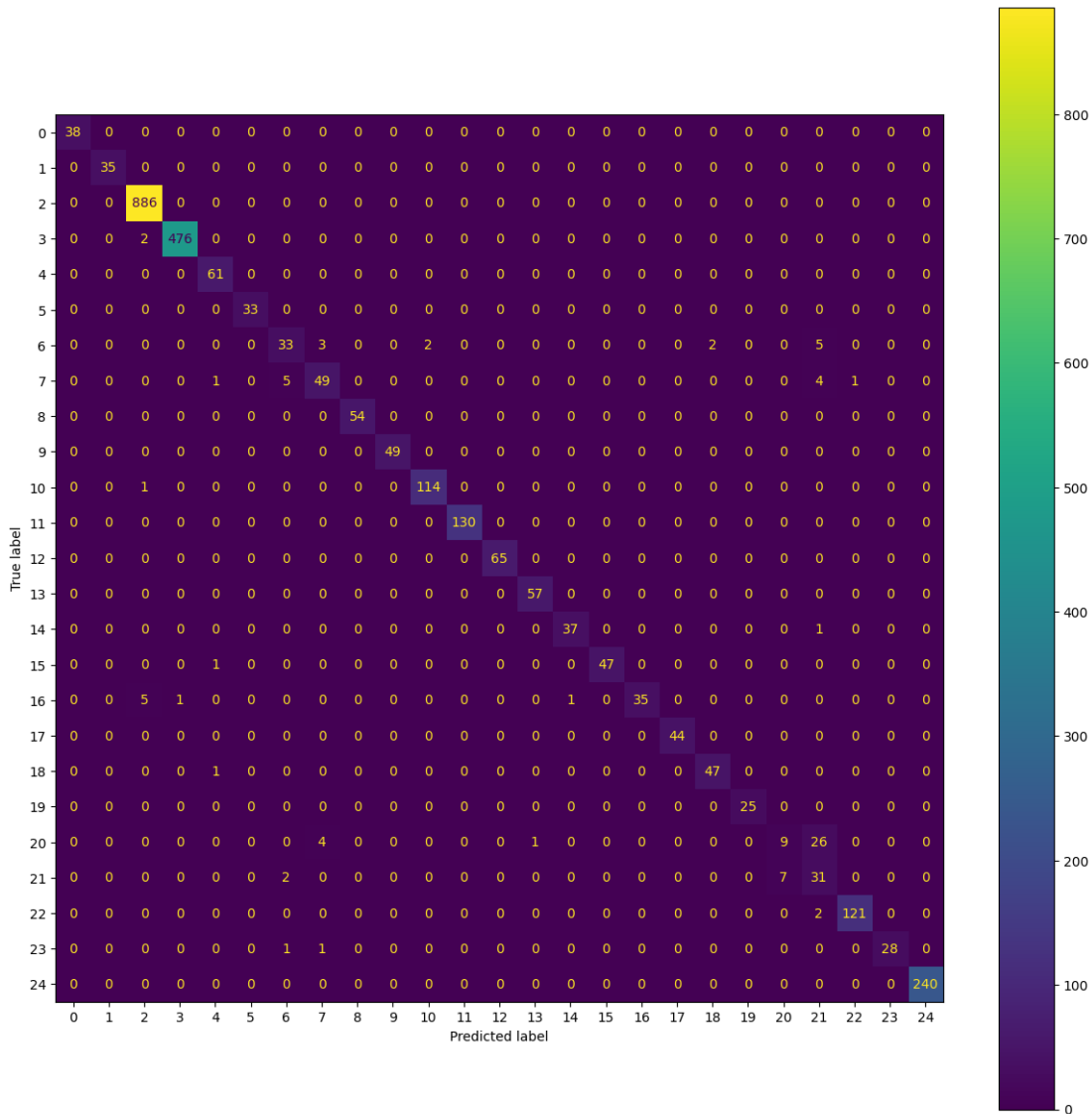


Figure 5.12: Confusion matrix obtained from the training of ViT on Mallng dataset without mask augmentation

The results for this step are presented in Table 5.15 together with the masking ensemble technique and in confusion matrices Figures 5.11, 5.12, and 5.13 for each analyzed dataset. The attained results, while not reaching the state-of-the-art levels achieved by the CNNs analyzed in the preceding chapters, remain acceptable. Across all datasets, the accuracy consistently surpasses 0.9, and the F1 score exceeds 0.8. This performance, although not exceptional, provides a solid baseline against which to measure the effectiveness of the masking technique when applied to Vision Transformers (ViTs). Notably, the model performs well on the Mallng dataset, achieving an F1 score of 0.929, comparable to the

5.3. ATTENTION ENSEMBLE ARCHITECTURE

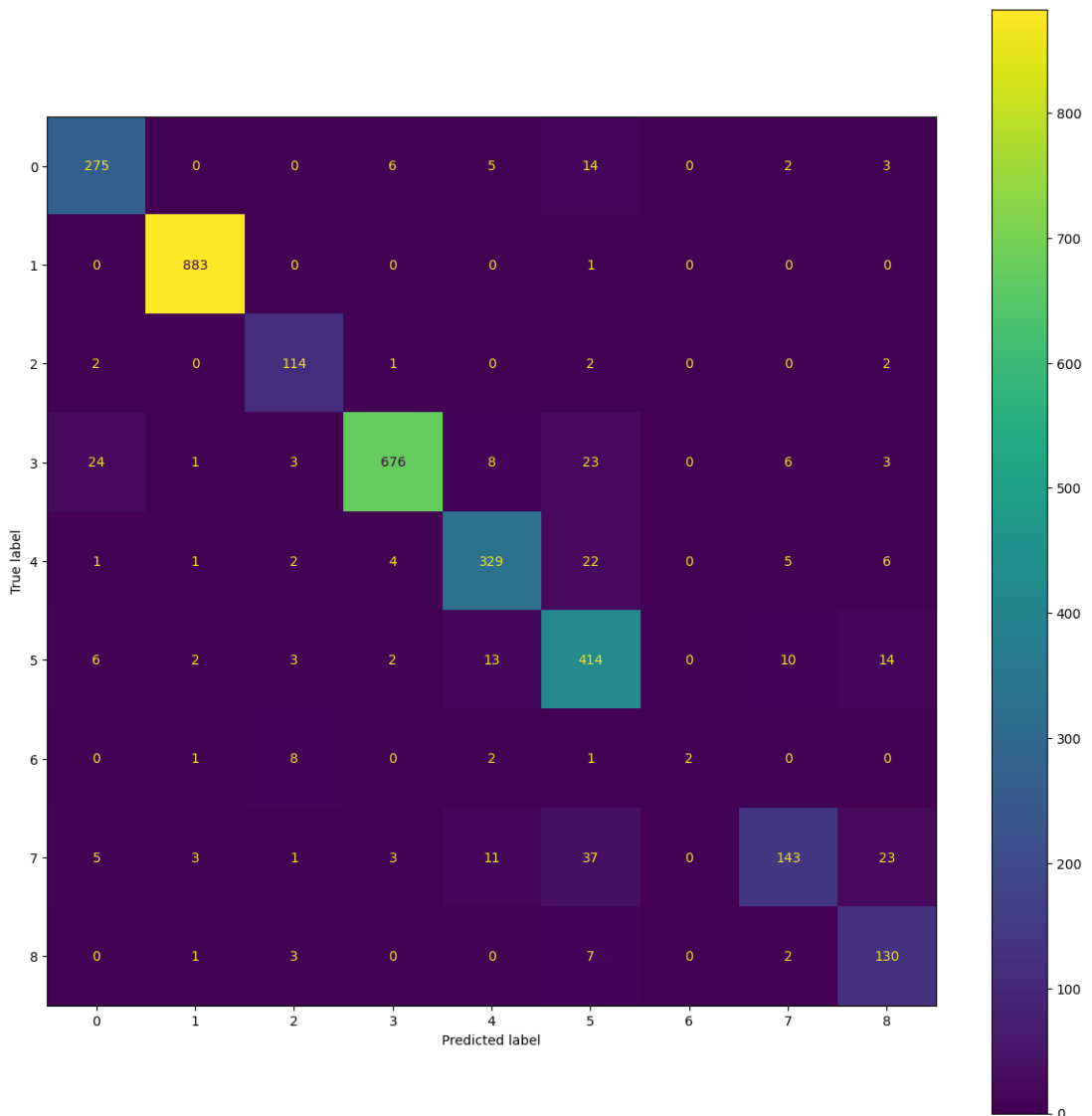


Figure 5.13: Confusion matrix obtained from the training of ViT on Big2015 dataset without mask augmentation

performance of the CNNs. In contrast, the disparities are more pronounced in the other datasets, with a notable difference of 0.803 for Big2015 and a substantial margin of 0.871 for VX-Zoo when considering the F1 score. Concerning the confusion matrices, it is evident across all datasets that the inferior outcomes stem from the model’s confusion within specific classes, rather than a broader inaccuracy within the entire family of classes under consideration, exacerbating a problem we found also with better performing classifiers.

5.3.2 ViT IMPLEMENTATION WITH ENSEMBLE MASKING (E7)

We employed the same methodology used for heatmap extraction to generate masks applied to samples in each dataset. This approach aimed to demonstrate the collaborative potential of CNNs and ViT at a profound level, transcending mere result ensembles or feature extraction. Our goal was to demonstrate that by leveraging the strengths of CNNs in conjunction with ViT, we could improve the performance of the latter model.

In this approach, CNNs generated heatmaps and subsequent masks using their unique strengths. These masks highlighted specific areas of significance. Meanwhile, the ViT model, trained over the masked images, categorized the newly filtered samples, leveraging its own strengths in the concentrated regions pinpointed by CNNs. The intention was to encourage the ViT model to disregard parts deemed irrelevant by CNNs and enhance its performance in areas where it excelled, thereby optimizing the overall analysis process.

In the particular implementation we decided to use we combined heatmaps generated by couples of CNNs, in particular *EfficientNetB0+DenseNet121*, and *EfficientNetB0+ResNet50*. The first pair was decided because across the three datasets those two CNNs were the best performing on average. The second couple was selected because the HiResCAM cumulative-SSIM between them was the lowest in two of the three datasets. We decided to use pair couple because a lower cumulative-SSIM means a greater difference between the heatmaps. Consequently, the areas of the images the CNNs use to classify the malware. Interestingly, these were also the pair of CNNs with the highest cumulative-SSIM in the third dataset. Consequently, we were even more motivated to test these two to determine if this particular characteristic would be reflected in the results. We choose to use heatmaps that look quite different instead of those that look very similar because we don't want to narrow down what the ViT can see too much. Instead, we want to provide guidance on what areas to pay attention to while keeping most of the image intact.

The results in Table 5.15 demonstrate a noticeable enhancement achieved through our masking approach. Specifically, all combinations of applied masks outperform the baseline. In the case of Big2015, the ViT classifier exhibited improvement with the mask derived from *EfficientNetB0*, resulting in a 0.09 increase

5.3. ATTENTION ENSEMBLE ARCHITECTURE

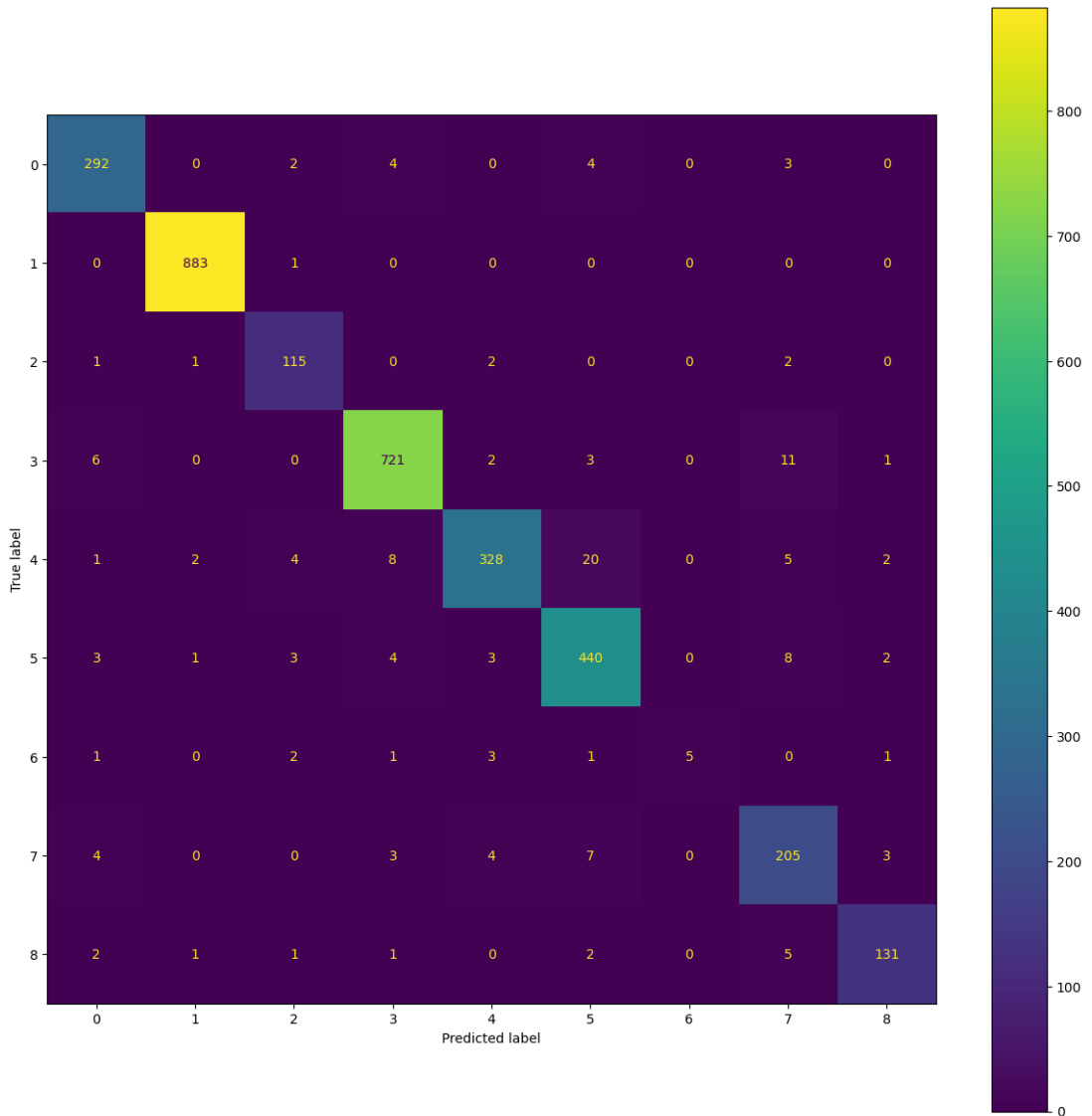


Figure 5.14: Confusion matrix obtained from the training of ViT on Big2015 dataset with mask augmentation generated with EfficientNetB0 and DenseNet121

in the F1-score. This improvement is evident in the confusion matrix when compared to the baseline. While the enhancements in other datasets are less significant, they remain consistent, with a maximum improvement of 0.024 in MalImg and 0.021 in VX-Zoo.

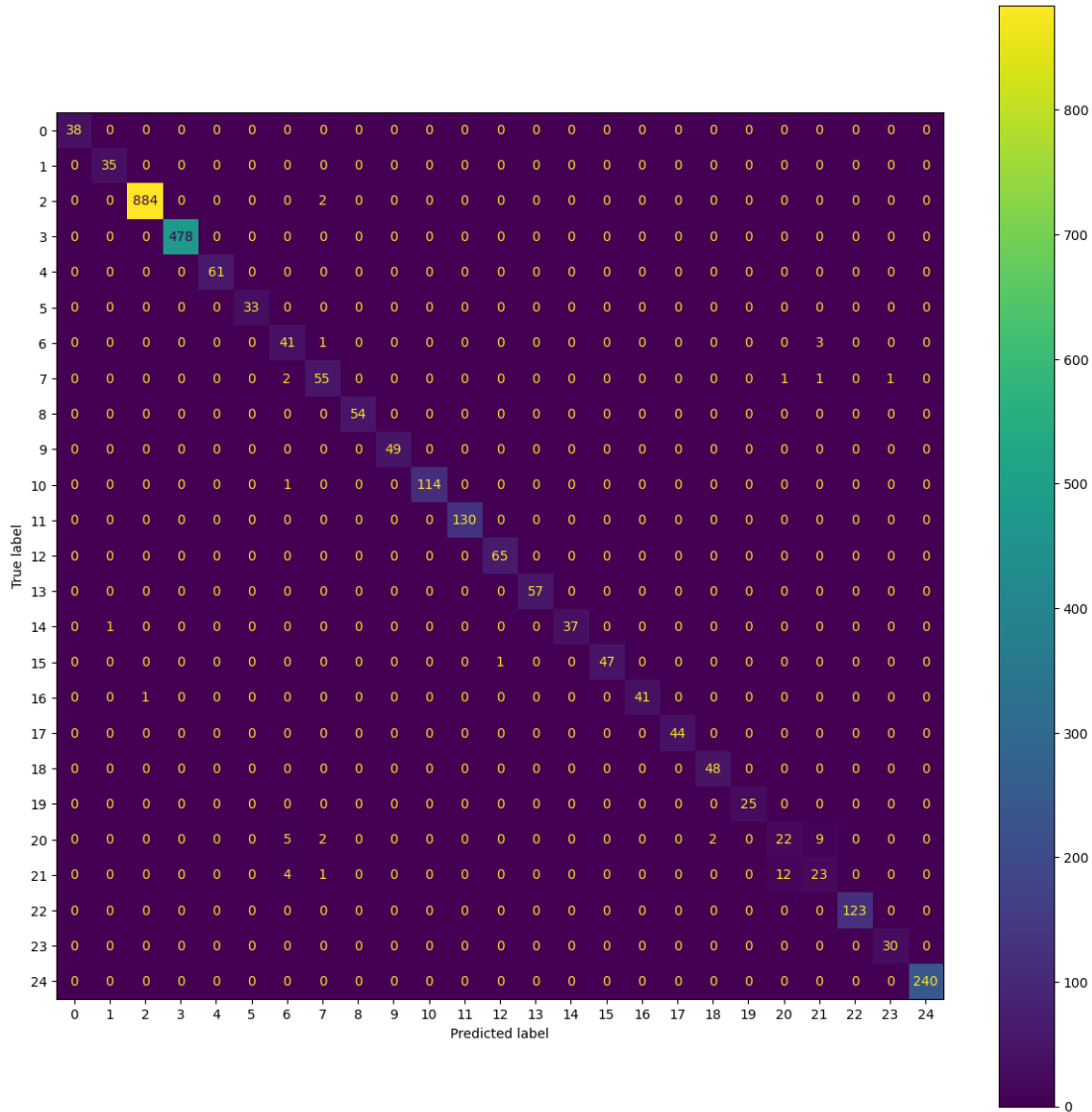


Figure 5.15: Confusion matrix obtained from the training of ViT on Mal-Img dataset with mask augmentation generated with EfficientNetB0 and DenseNet121

5.3. ATTENTION ENSEMBLE ARCHITECTURE

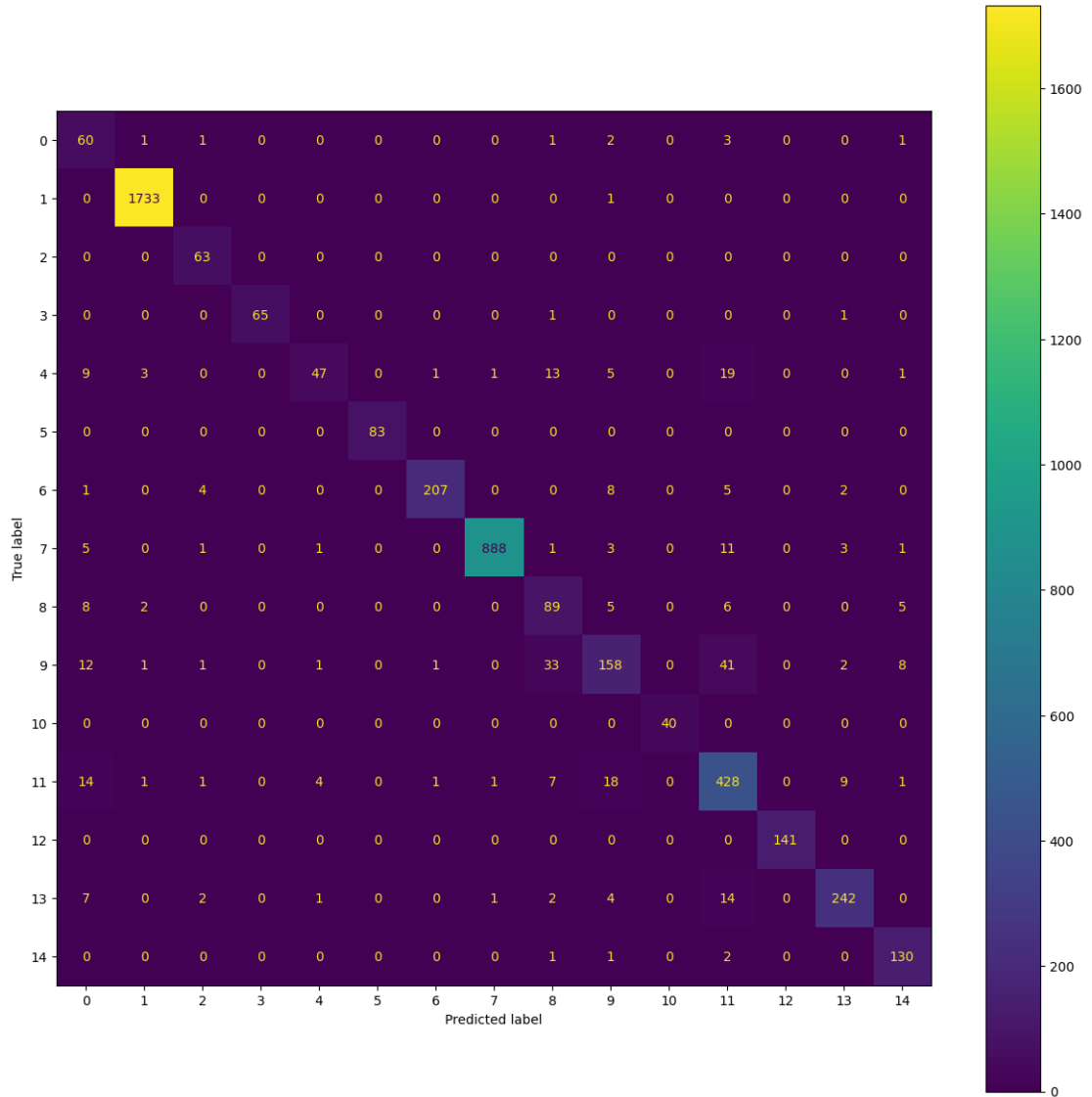


Figure 5.16: Confusion matrix obtained from the training of ViT on VX-Zoo dataset with mask augmentation generated with EfficientNetB0 and DenseNet121

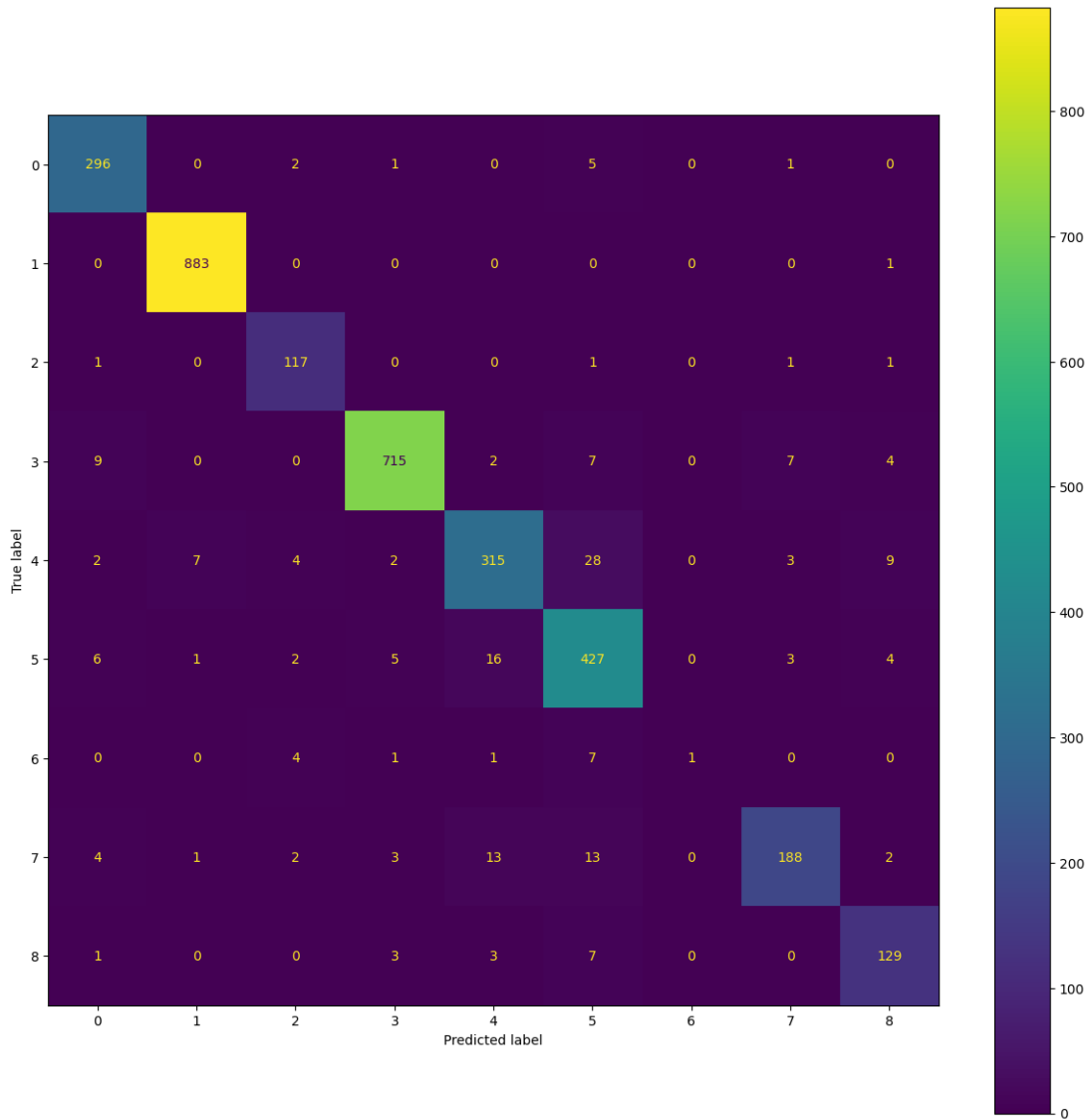


Figure 5.17: Confusion matrix obtained from the training of ViT on Big2015 dataset with mask augmentation generated with EfficientNetB0 and ResNet50

5.3. ATTENTION ENSEMBLE ARCHITECTURE

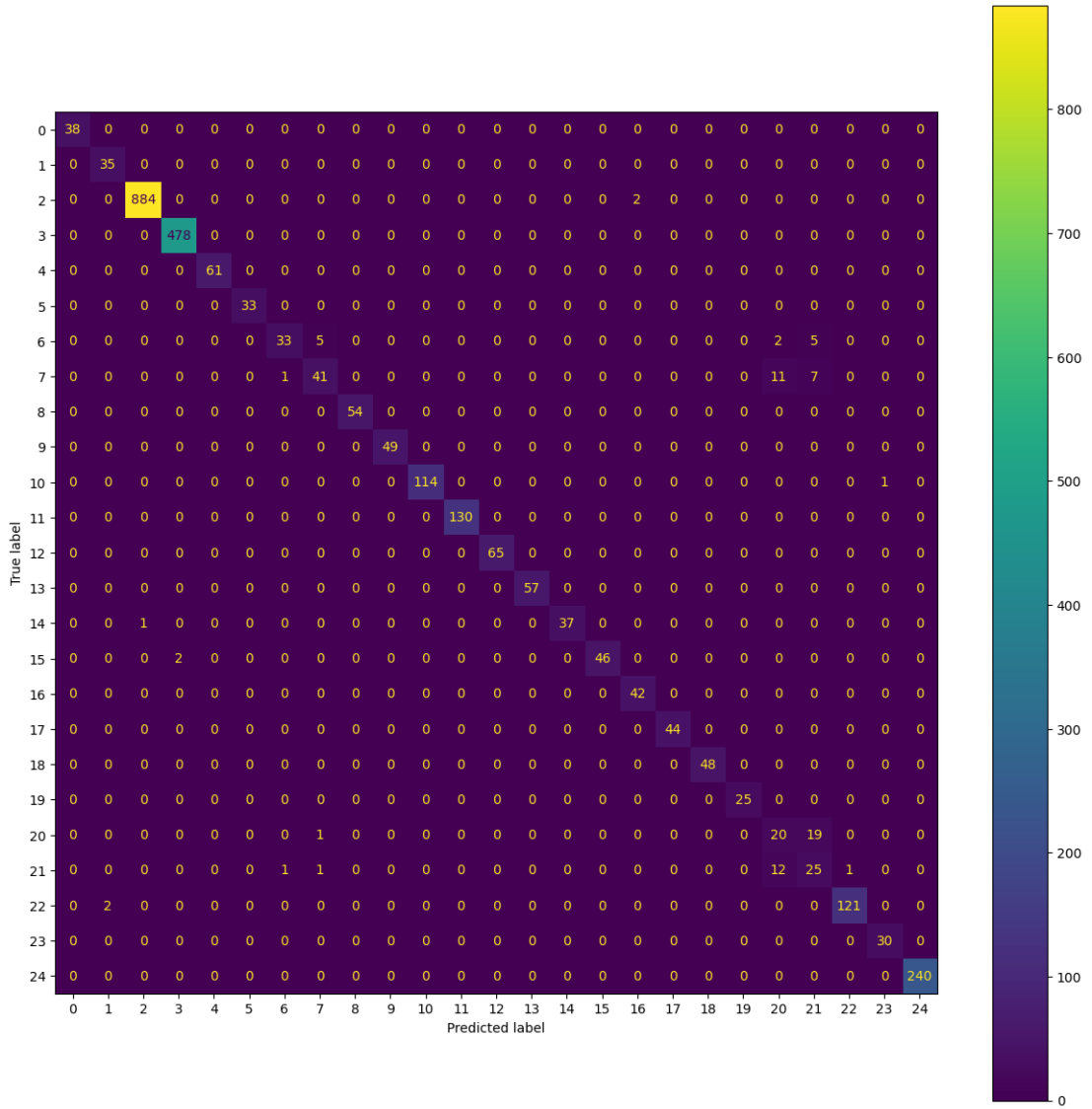


Figure 5.18: Confusion matrix obtained from the training of ViT on MallImg dataset with mask augmentation generated with EfficientNetB0 and ResNet50

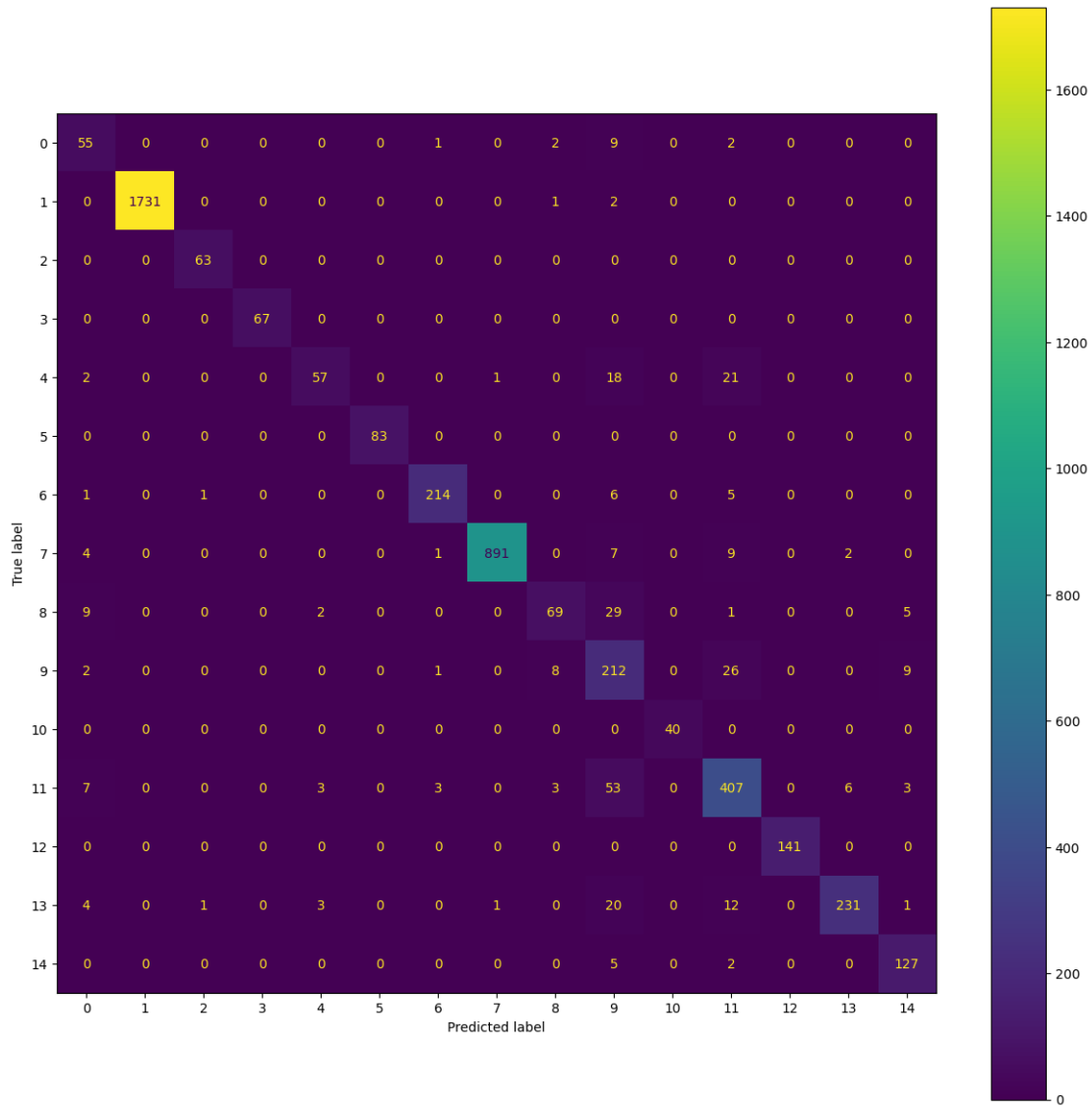


Figure 5.19: Confusion matrix obtained from the training of ViT on VX-Zoo dataset with mask augmentation generated with EfficientNetB0 and ResNet50

5.3. ATTENTION ENSEMBLE ARCHITECTURE

Dataset	Metrics			
	Accuracy	Precision	Recall	F1 Score
Big2015	<u>0.907</u>	0.884	0.802	<u>0.803</u>
Big2015 (ED)	0.954	0.942	0.877	0.892
Big2015 (RE)	0.939	0.927	0.830	0.833
MalImg	<u>0.972</u>	0.940	0.929	<u>0.929</u>
MalImg (ED)	0.982	0.953	0.955	0.953
MalImg (RE)	0.974	0.942	0.938	0.938
VX-Zoo	<u>0.927</u>	0.877	0.868	<u>0.871</u>
VX-Zoo (ED)	0.930	0.882	0.888	0.876
VX-Zoo (RE)	0.933	0.905	0.890	0.892

Table 5.15: Performance metrics of ViT with and without masking augmentation. In the Augmented Dataset column, ED means that the dataset has been augmented with the masks generated through EfficientNetB0 and DenseNet121 and RE means that the dataset has been augmented with the masks generated through EfficientNetB0 and ResNet50.



Conclusions and Future Works

6.1 SUMMARY OF RESULTS

This thesis delves into two crucial aspects: reproducibility and explainability, which form the foundation for the future of deep learning in malware image classification. Despite the development of highly accurate classifiers over the years, it is imperative for researchers to shift their focus towards consolidating existing state-of-the-art models documented in the literature. This consolidation effort necessitates conscientious researchers who, in the process of developing new architectures and presenting results, consider the broader scientific community. They should provide sufficient information to enable other knowledgeable parties to replicate their findings. The challenges encountered in ensuring reproducibility and replicability are multifaceted. Variability in datasets used for testing and training poses a significant obstacle. While it may not always be feasible to test models on different datasets, authors must strive to experiment with as many diverse datasets as possible. Another area requiring clarity is the division of data for training, validation, and testing purposes. Since this choice depends on specific variables, there is no universally “correct” way to split the data. However, authors can enhance comparability by testing their preferred method alongside commonly used approaches like the 70/30 split or 10-fold cross-validation. Additionally, disclosing hyperparameters is fundamental for a comprehensive study of any architecture. This disclosure ensures transparency and allows for a more thorough evaluation of the research work.

6.2. FUTURE WORK

This thesis delves into the examination of explainability in black box models, specifically focusing on Convolutional Neural Networks (CNNs). Employing an explainability tool holds immense potential for both researchers and users, providing invaluable insights into interpreting results and guiding subsequent actions and decisions. Researchers can utilize the outcomes of explainability studies to make informed modifications to their models, tailoring them to specific contexts. For users, understanding the rationale behind a model's choices and comprehending the advantages and disadvantages of established models are paramount. Our study highlighted the impact of the choice of an explainability tool on output data, underscoring the need for continued meticulous experimentation in this area.

The cumulative heatmaps generated through High-Resolution Class Activation Maps (HiResCAM) offer an unprecedented glimpse into the inner workings of the analyzed neural network. By comparing these heatmaps across different CNNs, researchers can deduce reasons behind specific occurrences and understand the varying performance of different architectures in specific scenarios.

Furthermore, our investigation delved into how model explanation can enhance the performance of different models. Surprisingly, it was evident that a model does not necessarily need to propose predictions directly to aid another model; leveraging its explanation alone can lead to performance enhancements. Although our masking technique is still in its prototype stage, it has demonstrated significant efficacy on certain datasets. This technique possesses the potential to further enhance malware classifiers, establishing explainability as a pivotal factor in their design.

6.2 FUTURE WORK

Further experimentation, particularly on a broader array of datasets and with a focus on Convolutional Neural Networks (CNNs), is imperative. While specific parameters can be chosen for CNN selection in replication efforts, the essential need for extensive testing remains even without such specified parameters. An intriguing avenue for progress lies in establishing guidelines for researchers to ensure that their described models are replicable with a high degree of confidence. This is especially crucial for widely-used datasets, where a standard approach for training set division should be formulated and adopted by practitioners. Although an uncomplicated solution would be the direct sharing of

documented code, this method is not always accessible to researchers, thereby necessitating the formulation of standardized rules.

On the front of explainability, conducting further testing using High-Resolution Class Activation Maps (HiResCAM) and Gradient-weighted Class Activation Mapping (GradCAM) can illuminate the relationship between how distinct neural networks discern malware families. Our hypothesis posits that the dissimilarity in learning approaches between these networks, quantified in our study using the cumulative Structural Similarity Index (cumulative-SSIM), can serve as a valuable parameter for determining the most suitable neural network combination.

In summary, a more competitive implementation of Vision Transformer (ViT) warrants exploration. Investigating whether masking datasets during training, employing our proposed method, not only leads to qualitative improvements but also quantifiable enhancements in results is an area ripe for further study.

References

- [1] Vasundhara Acharya, Vinayakumar Ravi, and Nazeeruddin Mohammad. “EfficientNet-based Convolutional Neural Networks for Malware Classification”. In: *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2021, pp. 1–6. DOI: [10.1109/ICCCNT51525.2021.9579750](https://doi.org/10.1109/ICCCNT51525.2021.9579750).
- [2] V Anandhi, P Vinod, and Varun G Menon. “Malware visualization and detection using DenseNets”. In: *Personal and Ubiquitous Computing (2021)*, pp. 1–17.
- [3] ANY.RUN. <https://any.run/>. Accessed: 2023-5-15. 2023.
- [4] Pooja Bagane et al. “Classification of Malware using Deep Learning Techniques”. In: *2021 9th International Conference on Cyber and IT Service Management (CITSM)*. 2021, pp. 1–7. DOI: [10.1109/CITSM52892.2021.9588795](https://doi.org/10.1109/CITSM52892.2021.9588795).
- [5] Ulrich Bayer et al. “Dynamic Analysis of Malicious Code”. In: *Journal in Computer Virology* 2.1 (Aug. 2006), pp. 67–77. ISSN: 1772-9904. DOI: [10.1007/s11416-006-0012-2](https://doi.org/10.1007/s11416-006-0012-2). URL: <https://doi.org/10.1007/s11416-006-0012-2>.
- [6] Ahmed Bensaoud and Jugal Kalita. “Deep multi-task learning for malware image classification”. In: *Journal of Information Security and Applications* 64 (2022), p. 103057. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2021.103057>. URL: <https://www.sciencedirect.com/science/article/pii/S2214212621002428>.
- [7] Prima Bouchaib and Mohammed Bouhorma. “Transfer Learning and SMOTE Algorithm for Image-based Malware Classification”. In: *Proceedings of the 4th International Conference on Networking, Information Systems & Security*. NISS2021. KENITRA, AA, Morocco: Association for Computing

REFERENCES

- Machinery, 2021, pp. 1–6. ISBN: 9781450388719. DOI: 10.1145/3454127.3457631. URL: <https://doi.org/10.1145/3454127.3457631>.
- [8] Igor Calzada. “Citizens; Data Privacy in China: The State of the Art of the Personal Information Protection Law (PIPL)”. In: *Smart Cities* 5.3 (2022), pp. 1129–1150. ISSN: 2624-6511. DOI: 10.3390/smartcities5030057. URL: <https://www.mdpi.com/2624-6511/5/3/57>.
- [9] Rajasekhar Chaganti, Vinayakumar Ravi, and Tuan D. Pham. “A multi-view feature fusion approach for effective malware classification using Deep Learning”. In: *Journal of Information Security and Applications* 72 (2023), p. 103402. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2022.103402>. URL: <https://www.sciencedirect.com/science/article/pii/S2214212622002460>.
- [10] Rajasekhar Chaganti, Vinayakumar Ravi, and Tuan D. Pham. “Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification”. In: *Journal of Information Security and Applications* 69 (2022), p. 103306. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2022.103306>. URL: <https://www.sciencedirect.com/science/article/pii/S2214212622001570>.
- [11] Yuhan Chai et al. “From Data and Model Levels: Improve the Performance of Few-Shot Malware Classification”. In: *IEEE Transactions on Network and Service Management* 19.4 (2022), pp. 4248–4261. DOI: 10.1109/TNSM.2022.3200866.
- [12] Xu Chen et al. “Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware”. In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. 2008, pp. 177–186. DOI: 10.1109/DSN.2008.4630086.
- [13] Zhongqiang Chen et al. “Malware characteristics and threats on the internet ecosystem”. In: *Journal of Systems and Software* 85.7 (2012). Software Ecosystems, pp. 1650–1672. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2012.02.015>. URL: <https://www.sciencedirect.com/science/article/pii/S0164121212000441>.
- [14] Mauro Conti, Shubham Khandhar, and P Vinod. “A few-shot malware classification approach for unknown family recognition using malware feature visualization”. In: *Computers & Security* 122 (2022), p. 102887.

- [15] *Cuckoo Sandbox*. <https://cuckoosandbox.org/>. Accessed: 2023-5-15. 2023.
- [16] Y. Le Cun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 396–404. ISBN: 1558601007.
- [17] Nadia Daoudi et al. "Lessons learnt on reproducibility in machine learning based android malware detection". In: *Empirical Software Engineering* 26.4 (2021), p. 74.
- [18] Abdulbasit Darem et al. "Visualization and deep-learning-based malware variant detection using OpCode-level features". In: *Future Generation Computer Systems* 125 (2021), pp. 314–323. ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2021.06.032>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X21002272>.
- [19] Huaxin Deng et al. "MCTVD: A malware classification method based on three-channel visualization and deep learning". In: *Computers & Security* 126 (2023), p. 103084. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2022.103084>. URL: <https://www.sciencedirect.com/science/article/pii/S016740482200476X>.
- [20] A Dosovitskiy et al. "Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).
- [21] Rachel Lea Draelos and Lawrence Carin. *Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks*. 2021. arXiv: 2011.08891 [eess.IV].
- [22] Olorunjube James Falana et al. "Mal-Detect: An intelligent visualization approach for malware detection". In: *Journal of King Saud University - Computer and Information Sciences* 34.5 (2022), pp. 1968–1983. ISSN: 1319-1578. DOI: <https://doi.org/10.1016/j.jksuci.2022.02.026>. URL: <https://www.sciencedirect.com/science/article/pii/S1319157822000702>.
- [23] Jianwen Fu et al. "Malware Visualization for Fine-Grained Classification". In: *IEEE Access* 6 (2018), pp. 14510–14523. DOI: 10.1109/ACCESS.2018.2805301.

REFERENCES

- [24] Daniel Gibert, Carles Mateu, and Jordi Planes. “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”. In: *Journal of Network and Computer Applications* 153 (2020), p. 102526. ISSN: 1084-8045. DOI: <https://doi.org/10.1016/j.jnca.2019.102526>. URL: <https://www.sciencedirect.com/science/article/pii/S1084804519303868>.
- [25] Daniel Gibert et al. “Fusing feature engineering and deep learning: A case study for malware classification”. In: *Expert Systems with Applications* 207 (2022), p. 117957.
- [26] Daniel Gibert et al. “Using convolutional neural networks for classification of malware represented as images”. In: *Journal of Computer Virology and Hacking Techniques* 15 (2019), pp. 15–28.
- [27] Jingcai Guo et al. *MDENet: Multi-modal Dual-embedding Networks for Malware Open-set Recognition*. 2023. arXiv: 2305.01245 [cs.CR].
- [28] Hashem Hashemi, Mohammad Ebrahim Samie, and Ali Hamzeh. “IFMD: image fusion for malware detection”. In: *Journal of Computer Virology and Hacking Techniques* 19.2 (2023), pp. 271–286. DOI: [10.1007/s11416-022-00445-y](https://doi.org/10.1007/s11416-022-00445-y).
- [29] Harold V Henderson, Friedrich Pukelsheim, and Shayle R Searle. “On the history of the Kronecker product”. In: *Linear and Multilinear Algebra* 14.2 (1983), pp. 113–120.
- [30] *Hybrid Sandbox*. <https://www.hybrid-analysis.com/>. Accessed: 2023-5-15. 2023.
- [31] Giacomo Iadarola et al. “Towards an interpretable deep learning model for mobile malware detection and family identification”. In: *Computers & Security* 105 (2021), p. 102198. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102198>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821000225>.
- [32] Yifei Jian et al. “A novel framework for image-based malware detection with a deep neural network”. In: *Computers & Security* 109 (2021), p. 102400. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102400>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821002248>.
- [33] *Joe Sandbox*. <https://www.joesandbox.com/>. Accessed: 2023-5-15. 2023.

- [34] Mahmoud Kalash et al. "Malware classification with deep convolutional neural networks". In: *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE. 2018, pp. 1–5.
- [35] Margot E. Kaminski. "The Right to Explanation, Explained". In: *Berkeley Technology Law Journal* 34.1 (2019), pp. 189–218. ISSN: 10863818, 23804742. URL: <https://www.jstor.org/stable/26755229> (visited on 10/07/2023).
- [36] ElMouatez Billah Karbab, Mourad Debbabi, and Abdelouahid Derhab. "SwiftR: Cross-platform ransomware fingerprinting using hierarchical neural networks on hybrid features". In: *Expert Systems with Applications* 225 (2023), p. 120017. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2023.120017>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417423005195>.
- [37] *Kaspersky*. <https://www.kaspersky.it/>. Accessed: 2023-5-15. 2023.
- [38] *Keras ViT Implementation*. https://keras.io/examples/vision/image_classification_with_vision_transformer/. Accessed: 2023-5-15. 2023.
- [39] Jeongwoo Kim, Joon-Young Paik, and Eun-Sun Cho. "Attention-Based Cross-Modal CNN Using Non-Disassembled Files for Malware Classification". In: *IEEE Access* 11 (2023), pp. 22889–22903. DOI: [10.1109/ACCESS.2023.3253770](https://doi.org/10.1109/ACCESS.2023.3253770).
- [40] Sanjeev Kumar and B. Janet. "DTMIC: Deep transfer learning for malware image classification". In: *Journal of Information Security and Applications* 64 (2022), p. 103063. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2021.103063>. URL: <https://www.sciencedirect.com/science/article/pii/S2214212621002465>.
- [41] Hyunjong Lee et al. "Robust IoT Malware Detection and Classification Using Opcode Category Features on Machine Learning". In: *IEEE Access* 11 (2023), pp. 18855–18867. DOI: [10.1109/ACCESS.2023.3247344](https://doi.org/10.1109/ACCESS.2023.3247344).
- [42] Qi Li et al. "CNN-Based Malware Variants Detection Method for Internet of Things". In: *IEEE Internet of Things Journal* 8.23 (2021), pp. 16946–16962. DOI: [10.1109/JIOT.2021.3075694](https://doi.org/10.1109/JIOT.2021.3075694).

REFERENCES

- [43] Gopinath M. and Sibi Chakkaravarthy Sethuraman. "A comprehensive survey on deep learning based malware detection techniques". In: *Computer Science Review* 47 (2023), p. 100529. ISSN: 1574-0137. DOI: <https://doi.org/10.1016/j.cosrev.2022.100529>. URL: <https://www.sciencedirect.com/science/article/pii/S1574013722000636>.
- [44] Zhidong Ma et al. "Visualizable Malware Detection based on Multi-dimension Dynamic Behaviors". In: *2022 International Conference on Networking and Network Applications (NaNA)*. 2022, pp. 247–252. DOI: [10.1109/NaNA56854.2022.00049](https://doi.org/10.1109/NaNA56854.2022.00049).
- [45] Abhishek Mallik, Anavi Khetarpal, and Sanjay Kumar. "ConRec: malware classification using convolutional recurrence". In: *Journal of Computer Virology and Hacking Techniques* 18 (2022), pp. 297–313. ISSN: 2214-2126. DOI: [10.1007/s11416-022-00416-3](https://doi.org/10.1007/s11416-022-00416-3). URL: <https://doi.org/10.1007/s11416-022-00416-3>.
- [46] Caio C. Moreira, Davi C. Moreira, and Claudomiro de S. de Sales Jr. "Improving ransomware detection based on portable executable header using xception convolutional neural network". In: *Computers & Security* 130 (2023), p. 103265. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2023.103265>. URL: <https://www.sciencedirect.com/science/article/pii/S016740482300175X>.
- [47] Hamad Naeem et al. "Identification of malicious code variants based on image visualization". In: *Computers & Electrical Engineering* 76 (2019), pp. 225–237. ISSN: 0045-7906. DOI: <https://doi.org/10.1016/j.compeleceng.2019.03.015>. URL: <https://www.sciencedirect.com/science/article/pii/S004579061733776X>.
- [48] Hamad Naeem et al. "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model". In: *Ad Hoc Networks* 105 (2020), p. 102154. ISSN: 1570-8705. DOI: <https://doi.org/10.1016/j.adhoc.2020.102154>. URL: <https://www.sciencedirect.com/science/article/pii/S1570870519306948>.
- [49] Lakshmanan Nataraj et al. "Malware images: visualization and automatic classification". In: *Proceedings of the 8th international symposium on visualization for cyber security*. 2011, pp. 1–7.

- [50] Quoc-Dung Ngo et al. "A survey of IoT malware and detection methods based on static features". In: *ICT Express* 6.4 (2020), pp. 280–286. ISSN: 2405-9595. DOI: <https://doi.org/10.1016/j.icte.2020.04.005>. URL: <https://www.sciencedirect.com/science/article/pii/S2405959520300503>.
- [51] Sang Ni, Quan Qian, and Rui Zhang. "Malware identification using visualization images and deep learning". In: *Computers & Security* 77 (2018), pp. 871–885.
- [52] Tom O'Malley et al. *KerasTuner*. <https://github.com/keras-team/keras-tuner>. 2019.
- [53] Stephen OShaughnessy and Stephen Sheridan. "Image-based malware classification hybrid framework based on space-filling curves". In: *Computers & Security* 116 (2022), p. 102660. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2022.102660>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404822000591>.
- [54] Anson Pinhero et al. "Malware detection employed by visualization and deep neural network". In: *Computers & Security* 105 (2021), p. 102247. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102247>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821000717>.
- [55] Handhika Yanuar Pratama and Jackson Sidabutar. "Malware Classification and Visualization Using EfficientNet and B2IMG Algorithm". In: *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. 2022, pp. 75–80. DOI: [10.1109/ICACSIS56558.2022.9923524](https://doi.org/10.1109/ICACSIS56558.2022.9923524).
- [56] Lingfeng Qiu et al. "Malware Classification based on a Light-weight Architecture of CNN: MalShuffleNet". In: *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*. 2022, pp. 1047–1050. DOI: [10.1109/CVIDLICCEA56201.2022.9824719](https://doi.org/10.1109/CVIDLICCEA56201.2022.9824719).
- [57] Akshara Ravi, Vivek Chaturvedi, and Muhammad Shafique. "ViT4Mal: Lightweight Vision Transformer for Malware Detection on Edge Devices". In: *ACM Transactions on Embedded Computing Systems* 22.5s (2023), pp. 1–26.

REFERENCES

- [58] Vinayakumar Ravi and Rajasekhar Chaganti. "EfficientNet deep learning meta-classifier approach for image-based android malware detection". In: *Multimedia Tools and Applications* 82 (2023), pp. 24891–24917. ISSN: 0167-4048. DOI: <https://doi.org/10.1007/s11042-022-14236-6>.
- [59] Royi Ronen et al. "Microsoft malware classification challenge". In: *arXiv preprint arXiv:1802.10135* (2018).
- [60] S. Abijah Roseline et al. "Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm". In: *IEEE Access* 8 (2020), pp. 206303–206324. DOI: 10.1109/ACCESS.2020.3036491.
- [61] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115 (2015), pp. 211–252.
- [62] Furqan Rustam et al. "Malware detection using image representation of malware data and transfer learning". In: *Journal of Parallel and Distributed Computing* 172 (2023), pp. 32–50. ISSN: 0743-7315. DOI: <https://doi.org/10.1016/j.jpdc.2022.10.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0743731522002118>.
- [63] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. "Image quality assessment through FSIM, SSIM, MSE and PSNRa comparative study". In: *Journal of Computer and Communications* 7.3 (2019), pp. 8–18.
- [64] Kamran Shaukat, Suhuai Luo, and Vijay Varadharajan. "A novel deep learning-based approach for malware detection". In: *Engineering Applications of Artificial Intelligence* 122 (2023), p. 106030. ISSN: 0952-1976. DOI: <https://doi.org/10.1016/j.engappai.2023.106030>. URL: <https://www.sciencedirect.com/science/article/pii/S0952197623002142>.
- [65] Jaiteg Singh et al. "Classification and Analysis of Android Malware Images Using Feature Fusion Technique". In: *IEEE Access* 9 (2021), pp. 90102–90117. DOI: 10.1109/ACCESS.2021.3090998.
- [66] Jiawei Su et al. "Lightweight Classification of IoT Malware Based on Image Recognition". In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 02. 2018, pp. 664–669. DOI: 10.1109/COMPSAC.2018.10315.

- [67] Sudhakar and Sushil Kumar. “MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things”. In: *Future Generation Computer Systems* 125 (2021), pp. 334–351. ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2021.06.029>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X21002247>.
- [68] Adem Tekerek and Muhammed Mutlu Yapici. “A novel malware classification and augmentation model based on convolutional neural network”. In: *Computers & Security* 112 (2022), p. 102515. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102515>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821003394>.
- [69] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. “Survey of machine learning techniques for malware analysis”. In: *Computers & Security* 81 (2019), pp. 123–147. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2018.11.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404818303808>.
- [70] UPX. <https://github.com/upx/upx>. Accessed: 2023-5-15. 2023.
- [71] Danish Vasan et al. “Image-Based malware classification using ensemble of CNN architectures (IMCEC)”. In: *Computers & Security* 92 (2020), p. 101748. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.101748>. URL: <https://www.sciencedirect.com/science/article/pii/S016740482030033X>.
- [72] Danish Vasan et al. “IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture”. In: *Computer Networks* 171 (2020), p. 107138.
- [73] Sitalakshmi Venkatraman, Mamoun Alazab, and R. Vinayakumar. “A hybrid deep learning image-based analysis for effective malware detection”. In: *Journal of Information Security and Applications* 47 (2019), pp. 377–389. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2019.06.006>. URL: <https://www.sciencedirect.com/science/article/pii/S2214212618304563>.
- [74] *Virus Total*. <https://www.virustotal.com>. Accessed: 2023-5-15. 2023.
- [75] *VirusShare*. <https://virusshare.com>. Accessed: 2023-5-15. 2023.

REFERENCES

- [76] Paul Voigt and Axel Von dem Bussche. "The eu general data protection regulation (gdpr)". In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.
- [77] *VX-Underground*. <https://www.vx-underground.org/>. Accessed: 2023-5-15. 2023.
- [78] Bidong Wang et al. "Image-Based Ransomware Classification with Classifier Combination". In: *Proceedings of the 3rd International Conference on Advanced Information Science and System*. AISS '21. Sanya, China: Association for Computing Machinery, 2022. ISBN: 9781450385862. DOI: 10.1145/3503047.3503083. URL: <https://doi.org/10.1145/3503047.3503083>.
- [79] Peng Wang, Zhijie Tang, and Junfeng Wang. "A novel few-shot malware classification approach for unknown family recognition with multi-prototype modeling". In: *Computers & Security* 106 (2021), p. 102273. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102273>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821000973>.
- [80] Mao Xiao et al. "Image-based malware classification using section distribution information". In: *Computers & Security* 110 (2021), p. 102420. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102420>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821002443>.
- [81] Hangfeng Yang et al. "A Novel Solutions for Malicious Code Detection and Family Clustering Based on Machine Learning". In: *IEEE Access* 7 (2019), pp. 148853–148860. DOI: 10.1109/ACCESS.2019.2946482.
- [82] *YARA Ruels*. <https://virustotal.github.io/yara/>. Accessed: 2023-5-15. 2023.
- [83] Fangtian Zhong et al. "Malware-on-the-brain: Illuminating malware byte codes with images for malware classification". In: *IEEE Transactions on Computers* 72.2 (2022), pp. 438–451.
- [84] Jinting Zhu et al. "A few-shot meta-learning based siamese neural network using entropy features for ransomware classification". In: *Computers & Security* 117 (2022), p. 102691. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2022.102691>. URL: <https://www.sciencedirect.com/science/article/pii/S016740482200089X>.

Acknowledgments

I would like to express my deepest gratitude to my advisors, Mauro Conti and Vinod P., for their unwavering support, guidance, and invaluable insights throughout this research journey. I am also immensely thankful to the entire SPRITZ Research Group for having welcomed me and for providing a continuous source of inspiration.

My heartfelt thanks go to my family, for their encouragement and understanding during the challenging phases of this thesis. Their belief in me kept me motivated.

I am deeply grateful to my friends, especially the VS, for the incredible years we've shared and the countless adventures we've embarked on together.



HiResCAM Cumulative Heatmaps

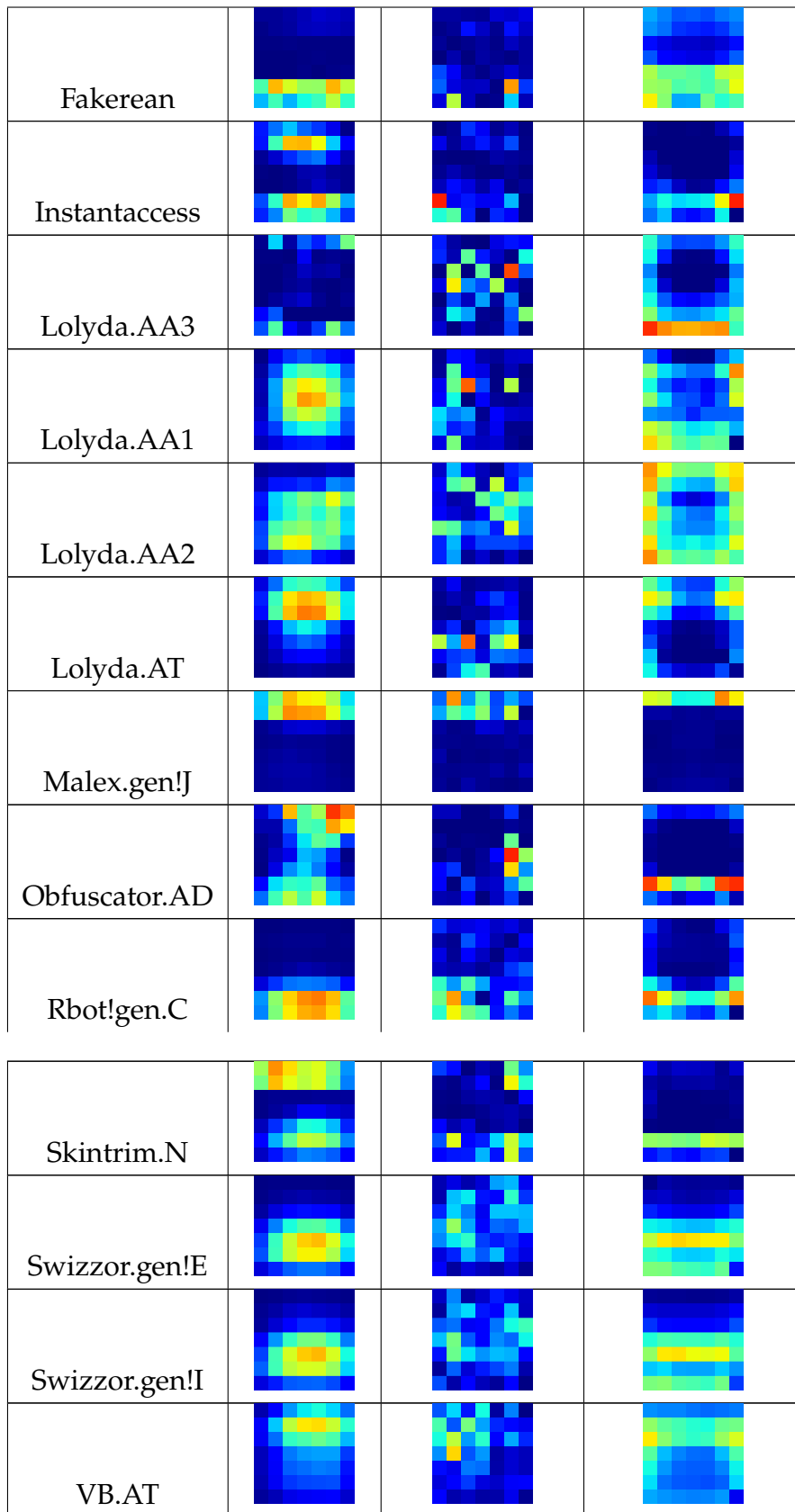
Family	ResNet50	DenseNet121	EfficientNetB0
Gatak			
Khelios_ver3			
Khelos_v1			
Lollipop			
Obfuscator.ACY			
Ramnit			
Simda			
Tracur			

Vundo			
-------	--	--	--

Table A.1: HiResCAM Cumulative Heatmaps generated for Big2015

Family	ResNet50	DenseNet121	EfficientNetB0
Adialer.C			
Agent.FYI			
Allaple.A			
Allaple.L			
Alueron.gen!J			
Autorun.K			
C2LOP.gen!g			
C2LOP.P			
Dialplatform.B			
Dontovo.A			

APPENDIX A. HIRES-CAM CUMULATIVE HEATMAPS



Wintrim.BX			
Yuner.A			

Table A.2: HiResCAM Cumulative Heatmaps generated for Mallmg

Family	ResNet50	DenseNet121	EfficientNetB0
Alman.b			
Debris.b			
Elkern.b			
Expiro.w			
inject.ahqtx			
Lamer.cb			
Nimnul.f			
Parite.b			
Salaty.sil			

APPENDIX A. HIRESCAM CUMULATIVE HEATMAPS

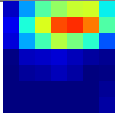
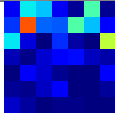
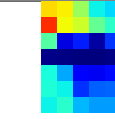
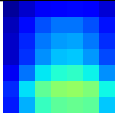
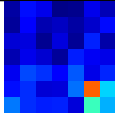
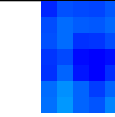
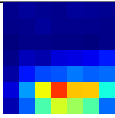
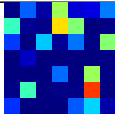
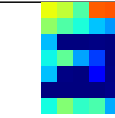
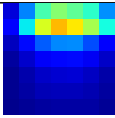
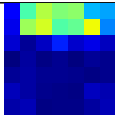
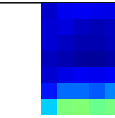
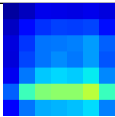
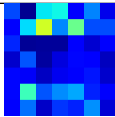
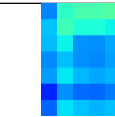
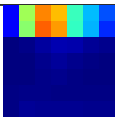
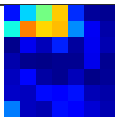
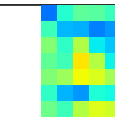
VB.cuvt			
Virut.ce			
Vtflooder ekl			
Wabot.a			
WBNA.ipa			
AntiFW.b			

Table A.3: HiResCAM Cumulative Heatmaps generated for VX-Zoo