



Journal of Universal Computer Science, vol. 30, no. 2 (2024), 262-286
submitted: 13/4/2023, accepted: 9/10/2023, appeared: 28/2/2024 CC BY-ND 4.0

Visualizing Portable Executable Headers for Ransomware Detection: A Deep Learning-Based Approach


Tien Quang Dam

(The University of Danang – University of Science and Technology, Danang, Vietnam
 <https://orcid.org/0000-0003-2670-7314>, tiendam.vn@gmail.com)


Nghia Thinh Nguyen

(The University of Danang – University of Science and Technology, Danang, Vietnam
 <https://orcid.org/0009-0007-6509-5759>, nghiathinh2000@gmail.com)


Trung Viet Le

(The University of Danang – University of Science and Technology, Danang, Vietnam
 <https://orcid.org/0009-0006-1587-292X>, trunglv2000vn@gmail.com)


Tran Duc Le*

(The University of Danang – University of Science and Technology, Danang, Vietnam
Université du Québec à Trois-Rivières, Canada
 <https://orcid.org/0000-0003-3735-0314>, letranduc@dut.udn.vn,
tran.duc.le@uqtr.ca, corresponding author)

Sylvestre Uwizeyemungu

(Université du Québec à Trois-Rivières, Canada
 <https://orcid.org/0000-0002-1532-8848>, sylvestre.uwizeyemungu@uqtr.ca)

Thang Le-Dinh

(Université du Québec à Trois-Rivières, Canada
 <https://orcid.org/0000-0002-5324-2746>, thang.ledinh@uqtr.ca)

Abstract: In recent years, the rapid evolution of ransomware has led to the development of numerous techniques designed to evade traditional malware detection methods. To address this issue, a novel approach is proposed in this study, leveraging machine learning to encode critical information from Portable Executable (PE) headers into visual representations of ransomware samples. The proposed method selects highly impactful features for data sample classification and encodes them as images based on predefined color rules. A deep learning model named peIRCECon (PE Header-Image-based Ransomware Classification Ensemble with Concatenating) is also developed by integrating prominent architectures, such as VGG16 and ResNet50, and incorporating the concatenating method to enhance ransomware detection and classification performance. Experimental results using self-collected datasets demonstrate the efficacy of this approach, achieving high accuracy of 99.85% in distinguishing between ransomware and benign samples. This promising approach holds the potential to significantly improve the effectiveness of ransomware detection and classification, thereby contributing to more robust cybersecurity defense systems.

Keywords: Ransomware; Deep Learning; Machine Learning; Ensemble Model; Image-based diagnose; PE Header

Categories: D.4.6, I.2.10, I.4.10, I.6.4, L.4.0
DOI: 10.3897/jucs.104901

1 Introduction

In today's digital landscape, malicious software, also known as malware, has emerged as a significant threat and a potential instrument for orchestrating network attacks. Among the different forms of malware, ransomware has drawn much focus due to its ability to inflict severe damage on individuals, companies, and business ecosystems at large [Oz et al., 22]. Notably, modern malware is becoming increasingly sophisticated, employing many techniques to conceal its presence and persist on computer systems. Consequently, traditional malware analysis methods struggle to keep pace with the onslaught of novel attacks and emerging variants [Murali et al., 20].

The effectiveness of malware analysis often depends on the expertise and knowledge of the analyst, which can limit the accuracy of results, particularly when examining previously unencountered malicious samples [Gibert et al., 20]. It emphasizes the urgent need for novel approaches and techniques to strengthen the malware analysis process, guaranteeing a more robust defense against evolving threats in the cybersecurity landscape.

As machine learning algorithms and artificial intelligence have gained popularity across various domains in recent years, their application in network security and malware analysis has become a logical progression [Tayyab et al., 22]. In particular, deep learning and neural networks offer promising solutions for addressing complex cybersecurity challenges [Dixit & Silakari, 21]. The ability of these neural network models to accurately categorize and distinguish ransomware from benign files in Windows operating systems, especially in the context of Portable Executable (PE) files, depends critically on data preprocessing [Berman et al., 19].

Visualization techniques in the context of malware classification and detection have been shown to significantly enhance processing time and speed, according to [Vu et al., 20]. Compared to traditional machine learning methods, which are often challenged in processing images, visualization techniques offer an alternative means of transforming raw data into feature vectors without requiring extensive expertise in design and engineering. Therefore, incorporating visualization techniques is paramount in enhancing the efficacy of malware classification and detection [Akshay Kumar & Rangasamy, 23].

This study aims at proposing an approach called *peIRCECon* (**PE** Header-**I**mage-based **R**ansomware **C**lassification **E**nsemble with **C**oncatenating) to enhance the classification and detection of ransomware by transforming data into images that encapsulate the semantic features of PE headers in order to contribute to the ongoing efforts in combating ransomware. Thus, an ensemble deep learning model is employed to extract additional features from the image data, thereby improving the accuracy of results compared to existing algorithms.

In this study, we aim to enhance the classification and detection of ransomware by transforming data into images that encapsulate the semantic features of PE headers. We then employ an ensemble deep learning model to extract additional features from the image data, thereby improving the accuracy of results compared to existing algorithms.

The rationale behind our image-based analysis method is multifaceted. First, the image representation of binary malware samples facilitates pattern recognition that may otherwise remain undetected in their raw binary form. The features extracted in this manner can be processed more efficiently by image-centric models. Second, this approach offers an added layer of security by mitigating the risk associated with the direct execution of malware, a critical factor in cybersecurity. Third, it allows the application of transfer learning techniques using pre-existing, pre-trained artificial intelligence models, which can greatly enhance the performance of our detection system. Lastly, image data offer opportunities for data augmentation, thereby enhancing the robustness of the model by providing a more diverse set of training examples.

The primary contributions of this study are:

- Employ machine learning models to determine the most impactful features from the PE Header of ransomware files, substantially influencing the detection and classification of ransomware.
- Encode the critical features from PE headers into visual representations of ransomware samples.
- Design the *peIRCECon* model that integrates prominent Convolutional Neural Networks (CNN) architectures, including ResNet-50 and VGG16, in combination with concatenation methods to deliver precise and comprehensive assessments for ransomware malware classification.

The remainder of this paper is organized as follows. Section 2 presents related work, and Section 3 details the *peIRCECon* approach. Section 4 discusses the experimental results of the approach, and Section 5 ends the paper with conclusions and future research.

2 Related works

2.1 Malware detection and classification

Various approaches have been proposed and explored in the realm of malware detection and classification. Firstly, [Nataraj et al., 11] proposed an approach for visualizing binary code as images by mapping each cluster of eight binary bits to a single-pixel representation in the image. This straightforward approach demonstrated high effectiveness and reliability. However, when faced with obfuscated or masked malware types, this approach struggles to achieve the desired level of accuracy.

Secondly, [Kim, 18] demonstrated that employing machine learning algorithms on PE header information could effectively enhance the performance of classification tasks. The study yielded positive results across all selected models, underscoring the potential of machine-learning techniques for improving malware classification based on PE header information. This finding encourages further exploration of machine learning methods in combination with PE header features to advance the state-of-the-art in malware detection and classification.

Thirdly, Convolutional Neural Networks (CNNs) have emerged as a more efficient alternative for classifying malware with high accuracy [Nataraj et al., 11], [Bensaoud et al., 20]. The image-based approach, employed in numerous shallow learning methods [Casolare et al., 22], facilitates the extraction of relevant features directly from the data,

eliminating the need for manual feature engineering. It streamlines the classification process and enhances overall performance, making CNNs and image-based techniques particularly promising for advancing the malware analysis field [Zhao et al., 23].

Fourthly, [Gibert et al., 20] conducted a comprehensive survey highlighting the significance and impact of static analysis methods, focusing on utilizing PE headers for malware classification and identification. Through this survey, the authors provided a strong foundation for applying machine learning techniques to analyze and extract information from PE headers, ultimately contributing to developing malware classification and identification strategies.

Indeed, the traditional methods discussed in the studies mentioned above can be time-consuming and less effective, given the exponential growth in the number of malware samples. As [Vu et al., 20] observed, approaches employing malware visualization techniques have demonstrated greater processing time and speed efficiency. Traditional machine learning methods typically struggle to handle the information in image pixels and lack the capacity for incremental learning. Converting raw data into feature vectors demands extensive expertise in design and engineering.

In contrast, deep learning models can be trained directly on input image data, leveraging their generalization and learning capabilities to process malware images more effectively. Consequently, this study focuses on harnessing the strengths of deep learning in the context of malware image processing, aiming to develop more efficient and robust methods for detecting and classifying malicious software.

2.2 Image processing based on deep learning

Deep learning-based image processing has gained significant traction in various practical applications, including face recognition, object classification [Zhang & Zheng, 22]. As a result, numerous innovative models have emerged to tackle increasingly complex challenges, such as handwritten character recognition [Khandokar et al., 21] and information extraction from job applications or identification cards [Yang et al., 22]. Owing to the remarkable advantages of image processing, numerous alternative and optimal solutions have been developed for addressing traditional issues, including detecting and classifying malware. The advancements in deep learning models and their applicability to image processing can potentially revolutionize the field of cybersecurity by providing enhanced methods for malware detection and classification [Singh et al., 19].

In the past, machine learning methods were employed to classify malware after converting it into grayscale or RGB images containing specific information about the malware. Techniques such as K-means and SVM were utilized for classification, achieving 95% accuracy with a dataset containing 25,000 malware samples and 12,000 benign files [Kancherla & Mukkamala, 13]. However, given the increasing complexity and volume of malware, these machine learning methods may struggle to meet the required accuracy levels. Image-based deep learning provides a promising solution through its end-to-end approach, which has been widely applied in classification problems using CNN-based models such as ResNet [He et al., 16], VGG [Simonyan and Zisserman, 14], [Salota & Singh, 23] and EfficientNet [Tan and Le, 19], [Chaganti et al., 22]. These models have been trained on the ImageNet dataset, comprising over 10 million diverse images, and have demonstrated remarkable accuracy.

Recent studies employing image-based methodologies for malware classification have provided evidence that this approach shows considerable promise and potential for advancement in the field [Reilly et al., 23], [Patil et al., 23]. However, applying image-based deep learning to the challenge of detecting and classifying malware, particularly ransomware, presents two primary scenarios depending on the size of the labeled malware dataset:

- (1) Large-scale labeled malware dataset: if the collected malware data consists of a vast number of labeled samples (approximately several million images), it is possible to train deep learning models using the entire dataset and all parameters of the CNN models. This approach allows the models to learn and extract relevant features specific to ransomware directly from the extensive training data, potentially resulting in more accurate and robust classification performance.
- (2) Small-scale labeled malware dataset: if the collected malware data consists of a relatively small number of labeled samples (approximately a few thousand), employing transfer learning becomes more suitable. Transfer learning leverages pre-trained models already trained on large datasets, such as *ImageNet*¹ or *COCO*², and fine-tunes them for ransomware detection and classification. This technique enables the models to capitalize on the knowledge acquired from the more extensive datasets, thereby improving their performance and accuracy even when dealing with limited ransomware samples.

This study adopts the second scenario due to the inherent challenges in collecting a large-scale dataset of malware samples, particularly ransomware. Indeed, in scenarios where labeled malware datasets are limited in size, transfer learning becomes a more viable option. For instance, [Bensaoud et al., 20] demonstrated that when combining CNN and SVM for ResNet and VGG16 models without transfer learning, the classification accuracy was only 26.66% and 14.31%, respectively. These results highlight the limitations of utilizing deep learning models without transfer learning when dealing with small-scale labeled malware datasets.

Numerous studies have explored the application of the end-to-end transfer learning method for malware detection and classification. For instance, [Rezende et al., 17] achieved very good results, with classification accuracies of 90.77% for VGG16 with Softmax and 98.62% for ResNet with Softmax. These findings underscore the effectiveness of transfer learning in enhancing the performance of deep learning models for this specific task.

Furthermore, ensemble models that integrate multiple feature types have proven to be highly effective in addressing malware classification challenges. Studies such as [Vasan et al., 20] have demonstrated the power of ensemble models in this context. By combining the strengths of various feature types and learning models, ensemble

¹ <https://image-net.org/index.php>

² <https://cocodataset.org/>

methods can improve classification accuracy and deliver more robust and reliable results in the ever-evolving landscape of malware detection.

Besides, the emergence of advanced malware obfuscation and packing techniques has diminished the accuracy and reliability of existing detection and classification methods [Singh et al., 18]. To address these challenges, there is a need for more sophisticated deep learning models or improved feature extraction strategies that can effectively focus on the most relevant features.

The present study capitalizes on the advantages of ResNet and VGG16 deep learning architectures within the ensemble learning model [Vasan et al., 20] alongside conventional machine learning techniques to address the ransomware classification problem. Furthermore, the study explores the potential of the Vision Transformer (ViT) model, which incorporates a multi-head attention mechanism to focus better on the most informative features within the input data. By adopting this attention-based approach, the Vision Transformer model [Dosovitskiy et al., 20] can more effectively identify and prioritize critical features, potentially improving performance in detecting and classifying sophisticated and obfuscated malware. This approach aspires to harness the strengths of each paradigm, providing a more resilient and precise methodology for detecting and classifying ransomware with enhanced performance.

Through the proposed method, the study aims at enhancing the state of the art in ransomware detection and classification, ultimately providing more effective tools to combat the growing threat of advanced malware and ransomware.

3 peIRCECon approach

This section presents the peIRCECon approach, including different steps such as selecting the best features from the PE header, generating grayscale representatives for PE files, and using deep learning in classification.

3.1 Preparing the datasets

Collecting sample data for the proposed model is an essential step in establishing the dependability and precision of the experiments. The sources for gathering ransomware samples are pre-classified sources: *VirusShare*, *Bazaar*, *Hybrid Analysis*, and *Amy-run*. This study utilizes three distinct datasets for the experiments.

- The first dataset (DT1) includes 3590 ransomware and 3753 benign files
- The second dataset (DT2) includes 3590 ransomware and 4724 samples of other malware types
- The third dataset (DT3) includes 1219 ransomware samples divided into nine families: Locky, Ryuk, Babuk, Cerber, Conti, WannaCry, Nitro, GandCrab, and Stop.

The initial two datasets underwent classification using the *Microsoft Software Removal Tool*, subsequently categorizing ransomware into two distinct groups: ransomware and other malware types. In addition, executable files were crawled on Windows and added to the dataset as benign data. Regarding the third dataset, only

samples that undoubtedly belonged to a specific ransomware family were retained to ensure the accuracy of ransomware families.

For the experiments, we employed a 5-fold cross-validation approach because the available data is limited, and this ensures the distribution characteristics of the data for both the training and validation sets in each fold. Increasing the number of folds would result in smaller validation sets, thereby impacting the quality of the evaluation.

3.2 Selecting the best features from the PE header using machine learning

The Portable Executable (PE) format³ is a file format for executables, object code, DLLs, and other binary files in Windows operating systems. It is designed to be portable across all supported Windows architectures, including 32-bit and 64-bit versions. The PE format is an extension of the Common Object File Format (COFF), which was originally developed for Unix systems and later adapted for Windows.

The PE format serves as a standard for Windows executables and provides a structured way for the operating system to understand and load the contents of a binary file. The format consists of several headers and sections that contain information about the file, such as its dependencies, entry point, code, and data sections.

Key components of the PE format include:

- **DOS Header:** The first part of the PE file is the DOS header, which contains information about the file's compatibility with the MS-DOS operating system. It also includes a stub program that displays a message indicating that the program cannot run in DOS mode when executed in a DOS environment.
- **PE Header:** It follows the DOS header and contains essential information about the file, such as its architecture, size, and the location of its various sections. The PE header is divided into the file header and the optional header.
- **File Header:** The file header provides general information about the file, such as its target machine, the number of sections, and the file's timestamp.
- **Optional Header:** It contains additional information about the file, such as its entry point, base address, section alignment, and other characteristics.
- **Section Headers:** Following the PE header, there are a series of section headers that describe the individual sections of the file, such as their size, location, and access permissions. Typical sections include `.text` (code), `.data` (initialized data), and `.rsrc` (resource data).
- **Sections:** The actual sections of the file contain the code, data, and resources for the executable. The section headers describe these sections and are loaded into memory when the file is executed.

The PE format plays a crucial role in the Windows operating system, providing a standard way for the system to load and execute binary files. Additionally, the PE format is often used in malware analysis, as its headers and sections can reveal valuable information about the behavior and characteristics of potentially malicious executables.

The PE headers contain numerous attributes, many of which are categorical in nature. For example, the *Machine* attribute typically has two values, 332 or 34404, representing different target machine architectures, and the *SizeOfOptionalHeader* attribute often assumes two common values, 224 and 240. Categorical machine learning

³ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

models are well-suited for handling this data type, as they can efficiently capture the relationships between the categorical features and the target variable.

In this study, we have chosen to employ the *ExtraTrees* [Geurts et al., 06], *XGBoost* [Chen & Guestrin, 16], and *CatBoost* [Prokhorenkova et al., 18] models to evaluate the importance of features extracted from the PE headers. These models were selected due to their strong performance in machine learning competitions⁴ hosted on Kaggle, a popular platform for data science and machine learning challenges.

This study employs a feature extraction process that involves training the selected machine learning models multiple times on shuffled versions of the dataset (refer to Figure 1). The reason for this choice is to increase the stability and generalizability of the features identified as important for malware and ransomware classification.

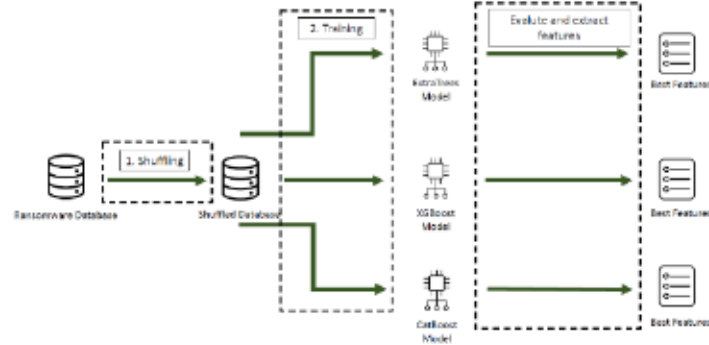


Figure 1: The training steps

First, a general accuracy threshold is established for all the models selected for training. For instance, in the case of ransomware and other types of malware classification, the approach may set an overall accuracy target of 99.6%. With this target in mind, each model is trained to achieve this general accuracy threshold at least three times.

Shuffling the training and testing datasets after each training iteration ensures that the models are exposed to different training and testing data combinations. This prevents the models from overfitting to a particular data set and helps improve their generalization performance on new and unseen data. Additionally, shuffling the data can help to reduce bias in the training process by ensuring that the models are not learning patterns specific to the order or structure of the data.

The selected machine learning models (*ExtraTrees*, *XGBoost*, and *CatBoost*) will be better equipped to extract meaningful and stable features from the PE headers through this iterative and dynamic training process.

To extract the most important features, the *peIRCECon* approach evaluates the impact of each feature on the trained models. The approach aims to quickly identify the most influential features by averaging the impact values of each feature across all models. The average impact level of feature values (ILOFV) is computed using the following formula:

$$\text{Average of ILOFV} = \frac{\sum_{i=1}^n \text{ILOFV}_i}{n} \quad (1)$$

⁴ <https://www.kaggle.com/competitions>

Where:

- *ILOFV* (Impact Level of Feature Value): The feature's impact value in a model (default = 0 if not in the list)
- *n*: The total number of trained models that achieved the general accuracy (In this case, we chose nine models, as we selected three trained models that achieved the general accuracy for each of the chosen ML models: ExtraTrees, XGBoost, and CatBoost)

After calculating the average ILOFV for each feature, the values are compiled in Table 1. Despite numerous features (56 features), only the 12 most significant ones will be presented here, for example. Based on this table, the most influential features that consistently contribute to the models' performance in detecting and classifying ransomware are selected. By focusing on these key features, the peIRCECon can become a more accurate and efficient model for tackling the challenges of ransomware classification and detection.

No	Features	Impact level (%)
1	'Checksum'	11.56
2	'SizeOfUninitializedData'	10.63
3	'SizeOfStackCommit'	7.84
4	'DllCharacteristics'	7.55
5	'MinorLinkerVersion'	7.31
6	'SectionAlignment'	5.45
7	'SectionMaxRawsize'	4.83
8	'ImportsNbDLL'	4.76
9	'ImageBase'	4.61
10	'SectionsMinVirtualsize'	4.09
11	'SizeOfHeaders'	3.93
12	'MinorOperatingSystemVersion'	3.01

Table 1: The impact level of features in PE headers

3.3 Generating grayscale representatives for PE files

In this subsection, the paper delves into the process of generating grayscale representations for Portable Executable (PE) files as a critical step in preparing the data for image-based malware classification techniques. Leveraging the image-based approach allows for capturing relevant features and structural information from the PE files, which can be instrumental in enhancing the performance of deep learning models for malware detection and classification. Building upon the work of [Xiao et al., 21], the study explores the advantages of integrating informative features like PE sections border information into malware image representations. Their novel image-based technique, which embeds PE sections border data, aimed to capture additional structural and semantic details for improved malware classification and detection. The experiments showed that incorporating the border of the section enhanced classification accuracy, even with a relatively simple deep-learning model.

PE headers possess valuable information that can be employed for static analysis. The peIRCECon approach also proposes incorporating additional features from PE

headers into the sample representations to enhance the capabilities of deep learning models, thereby improving classification and detection accuracy. To achieve this, the proposed approach selected the most relevant features from PE headers, encoded these features as colors, and reconstructed the sample representations to include this additional information.

3.3.1 From binary to grayscale

[Nataraj et al., 11] converted binaries into pixels, as illustrated in Figure 2. The width of the image is specified, as shown in Table 2. Samples belonging to the same variant exhibit similar pixel distributions. However, due to obfuscation techniques, the representation of binary byte distribution may no longer accurately depict the malware variant, causing the results of this naive method to be quickly bypassed. As a result, many approaches have adopted this method as an intermediate step for generating representative images of data samples.



Figure 2: Visualizing binary sample using a grayscale image

File Size Range	Image Width
<10 kB	32
10 kB – 30 kB	64
30 kB – 60 kB	128
60 kB – 100 kB	256
100 kB – 200 kB	384
200 kB – 500 kB	512
500 kB – 1000 kB	768
>1000 kB	1028

Table 2: Image width corresponding with binary size

3.3.2 Adding section lines to the image

In this section, we build upon the idea presented by [Xiao et al., 21] to utilize section information from the PE header to create section borders in grayscale images. The PE header in an executable file contains semantic information that assists the operating system in executing the file properly. Executable file sections are described in the PE header using three main attributes: the pointer to the section's start, size, and name. By adding section border lines, we can clearly delineate the sections within the image. The process for incorporating section borders into the image is illustrated in Figure 3. For

each section defined in the PE header, border lines with specific colors based on the section's name are added. The width of these lines is calculated based on the file size.

The formula to determine the line thickness is as follows:

$$thickness = \frac{file_size}{1024 \times 50} * interval + padding \quad (2)$$

Where:

- *interval* is the normalization coefficient that helps adjust the line thickness proportionally to the file size.
- *padding* represents the minimal possible value for thickness, ensuring that the line is always visible, even for small file sizes.

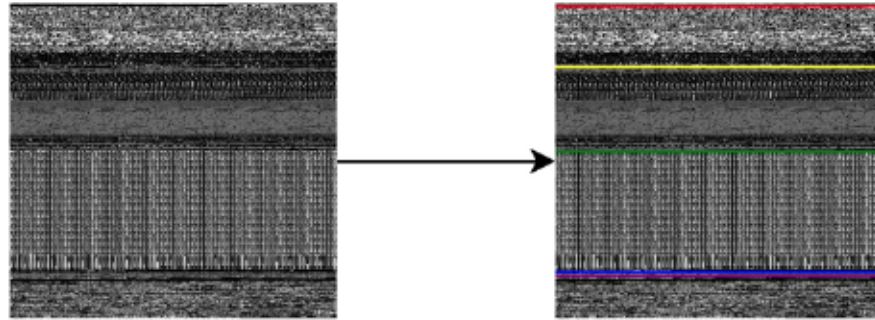


Figure 3: Adding section lines for the ransomware image

3.3.3 Encoding PE header features into image

In this subsection, the paper discusses the process of encoding PE header features into the ransomware image after incorporating section lines. This step aims to further enhance the image representation by incorporating additional valuable information from the PE header. In this research, the following types of information are extracted to enhance the representation of the ransomware image further:

- Descriptive information such as *Machine*, *Characteristics*, and *SizeOfImage*.
- Statistical information related to sections, such as *Entropy*, *RawSize*, and *VirtualSize*.

Out of the 56 features extracted from the executable files, the proposed approach focuses on encoding only the most essential features into the image representation, as determined in subsection 3.1. This selective approach ensures that the image contains the most relevant and informative features for classification and detection tasks while avoiding potential issues arising from including limited or redundant information. By incorporating these key features into the image, the peIRCECon approach aims to enhance the deep learning models' ability to classify and detect ransomware accurately.

In order to encode the values of the PE header's features into the image representation, the min-max normalization is first performed to scale all values to a range of 0-255. Subsequently, these normalized values are mapped to the Turbo Color

Scheme⁵. This method tends to work well, particularly when dealing with samples subjected to obfuscation techniques. As illustrated in Figure 4, adding feature lines to the image representations makes it easier to recognize that the samples may belong to the same malware family, despite the presence of obfuscation. By integrating these informative features, the approach aims to improve the deep learning models' performance in malware classification and detection tasks.

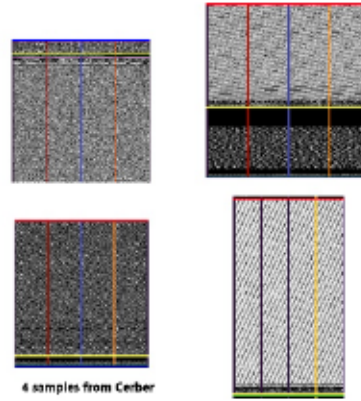


Figure 4: Samples of the Cerber ransomware variant

The values of the embedded features are standardized using min-max normalization according to the following equation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} * 255 \quad (3)$$

In this equation, x represents the original feature value, and x' represents the transformed value within the range of 0 - 255. Once the values are normalized, the corresponding color is selected from the Turbo color map to encode the features into the image as vertical color lines.

Figure 5 illustrates the proposed pipeline for our approach. In the proposed pipeline, we initially gather a ransomware dataset in binary format, which is then converted into grayscale images. Simultaneously, we extract PE features from these binary files, which undergo a machine learning selection process to identify the most informative features. These selected features are color-encoded and integrated into the grayscale images to enrich the data representation. Finally, we apply our proposed model.

⁵ <https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html>

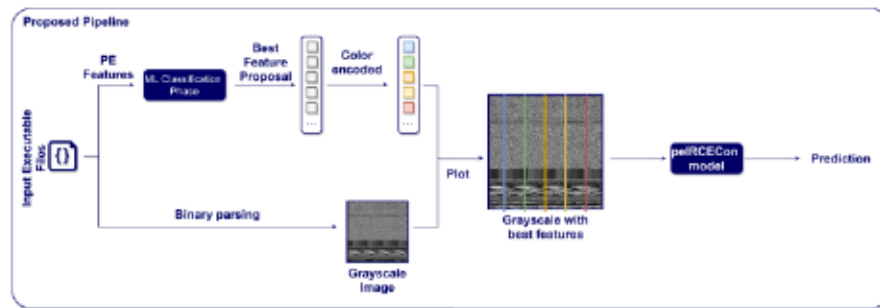


Figure 5: Proposed pipeline

3.4 Using deep learning in classification

The paper delves into applying deep learning to address classification in this subsection. Various models and techniques are discussed that have demonstrated remarkable success in diverse classification tasks. These models and techniques form the foundation of the proposed approach, which aims to enhance the performance and efficiency of classification processes. The study will identify the most suitable approach to be integrated into our methodology's proposed approach by exploring each model's strengths and weaknesses to ensure an effective and reliable solution for classification issues.

3.4.1 Deep learning models

This study develops a combined model using the concept of the model called IMCEC (Image-Based Malware Classification using Ensemble of CNN Architectures) [Vasan et al., 20] that leverages both the VGG16 and ResNet-50 architectures, which were pre-trained on the ImageNet dataset for object detection tasks. It is important to note that there are some changes in the peIRCECon approach in comparison with IMCEC, including: (i) adding the critical PE Header features and section lines to ransomware images; (ii) using a concatenating method instead of adding method in the last layer to preserve the characteristics of prior layers while reducing the total number of parameters. In the proposed approach, the convolutional layers of both VGG16 and ResNet-50 models are utilized to extract bottleneck features from ransomware images. These features were then used as input for classifying ransomware families, ransomware and benign files, as well as ransomware and other types of malware. Furthermore, a Vision Transformer and standalone models, such as ResNet-50 and VGG16, are also trained to compare their performance with our peIRCECon model based on transfer learning. The study aims to identify the most effective approach for accurate and robust malware classification and detection by evaluating these different models.

The VGG16 model [Simonyan and Zisserman, 14] consists of 16 trainable layers, which include five blocks of convolutional layers and three fully connected layers. The convolutional layers employ filters with a kernel size of 3, padding of 1, and a stride of 1, ensuring that the spatial dimensions of each activation map remain consistent with those of the previous layer. Moreover, the input size is reduced, and the dimensions of layers are increased throughout the series of blocks, accelerating the model training

process. A max-pooling layer with a 2x2 kernel filter, no padding, and a stride of 2 is used, guaranteeing that the spatial dimensions of the activation map are halved compared to the previous layer. However, VGG16 has over 138 million parameters, while ResNet-50 provides improvements through the use of residual blocks.

ResNet-50 [Krizhevsky et al., 17] is a deep convolutional network that employs shortcut connections to skip convolutional layer blocks, addressing the vanishing gradient problem. The fundamental building blocks, known as bottleneck blocks, follow two design rules: (i) for similar output feature map sizes, the same number of filters are applied to layers, and (ii) if feature map sizes are halved, the number of filters is doubled. Down-sampling is performed directly by convolutional layers with a stride of 2, and batch normalization is applied immediately after each convolution, preceding the ReLU (Rectified Linear Unit) activation function. An identity shortcut is used when the output and input dimensions are similar. If dimensions need to be increased, a projection shortcut is applied for dimension matching via 1x1 convolutions. Although ResNet-50 has more layers than VGG16, with over 30 layers, each layer block incorporates batch normalization to normalize parameters within a specific range.

For models like ResNet and VGG16, using convolutional layers helps improve learning performance. However, in some cases, a model does not need to be overly complex or concentrate on all the positions within an image. Instead, it can concentrate on the most distinctive features of the object to be classified in the image. The Vision Transformer [Dosovitskiy et al., 20] achieves this using the Multihead Attention mechanism [Subakan et al., 21]. Inspired by LSTM (Long Short-Term Memory) and RNN (Recurrent Neural Network) models, the Vision Transformer divides the image into patches and feeds them sequentially into the model as input. Each patch passes through a Flatten layer, transforming the image into an input vector. Since the Vision Transformer is sensitive to positions that LSTM and RNN do not take into account, position information is added to each input vector corresponding to each patch. Using Multihead Attention, the model concentrates on individual pixels in the image and assigns corresponding weights, eliminating unnecessary features if the pixel's weight is very low or nearly zero.

This attention mechanism identifies the most important features of objects in the image. However, at initialization, the position embeddings do not contain any information about the 2D positions of the patches, requiring all spatial interactions between them to be learned from scratch. The Vision Transformer benefits from parallel processing, which makes it faster than LSTM and RNN models that rely on sequential processing.

Ensemble learning [Vasan et al., 20] is a method that combines multiple models, supplementing each other to improve learning performance. This approach is suitable when each model has its own strengths and weaknesses. We employ ensemble learning instead of adjusting parameters for each model to achieve better learning, which requires significant research time and a large amount of training data for the most accurate evaluation. This study constructs the peIRCECon using the ensemble learning approach by combining VGG16 with ResNet-50 and concatenating method (refer to Figure 6). The efficacy of ResNet is primarily attributed to its rapid training time, rendering it particularly suitable for image data of relatively small dimensions.

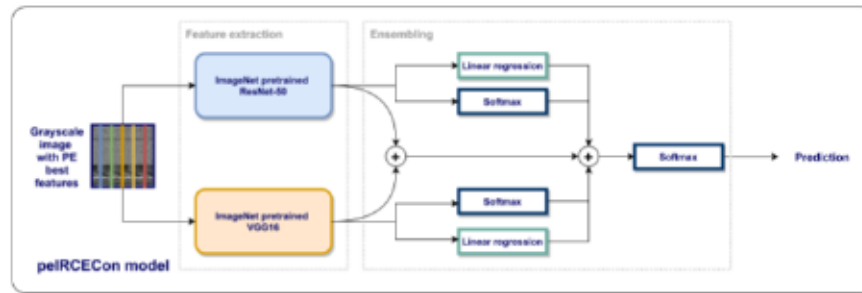


Figure 6: peIRCECon model

In contrast, VGG16 demonstrates a strong aptitude for boundary segmentation tasks and is well-equipped to handle data with comparatively larger feature sizes. Integrating these two models can significantly simplify the classification problem, resulting in enhanced accuracy. Nevertheless, the amalgamation of two models with extensive parameters may give rise to overfitting, a phenomenon wherein the model becomes excessively tailored to the training data and struggles to generalize to unseen data. Consequently, devising an appropriate training strategy is paramount when constructing the model.

3.4.2 Using concatenation on the last layer

One of our main contributions is opting for the *Concatenate* method over the *Add* method, commonly used in the ResNet model as in IMCEC. The *Add* method combines the input and output of a layer to create shortcut connections and tackle the vanishing gradient problem. In contrast, the proposed model replaces it with the *Concatenate* method, inspired by the DenseNet model [Zhang et al., 21]. It utilizes dense connections followed by normalization using a Softmax layer to standardize the final output.

The *Concatenate* method aggregates outputs from all previous layers instead of introducing more complex terms. This approach preserves the characteristics of prior layers while reducing the total number of parameters, thereby making the model more computationally efficient. The *Add* method can generally be perceived as sharing information that might lead to distortion. In contrast, the *Concatenate* method represents a more literal information sharing, directly combining outputs from previous layers.

3.4.3 Transfer learning

Predictions may perform well on the test set in numerous machine learning problems but yield disastrous results in real-world applications [Long et al., 15]. Several factors could contribute to this discrepancy:

- Insufficient dataset size (1)
- Imbalanced dataset (2)
- Overly complex model for the given data size (3)
- Challenges in data optimization (4)

Transfer learning effectively addresses the first and third problems, which may be the challenges in this study, by utilizing pre-trained models with weights learned from massive datasets. These models can be divided into two sections. The base model,

composed of Conv2D layers, serves as the first section and is responsible for extracting features from the input data. The second section, the Fully Connected (FC) layer, calculates the probability distribution for classification, with the number of outputs equal to the number of classes. Our approach performs feature extraction on malicious data, necessitating modifications to the FC layer structure in original models like ResNet and VGG16. To obtain optimal results, deeper layers are unfrozen, as malicious data is not part of ImageNet's classes. The FC layer employs Softmax, Linear Regression, ReLU, and Normalization functions, avoiding convolutional layers for simplicity. The models' parameters are detailed in Table 3:

Model	Parameters
VGG16	138,616,113
ResNet-50	24,034,889
ViT	87,084,785
peIRCECon	147,466,166

Table 3: Architecture statistics

3.4.4 Normalizing dataset based on ImageNet format

To utilize a pre-trained model based on ImageNet, we preprocess the dataset images to conform to the ImageNet format, which yields the best results. The pixel values in the malware binary images range from 0 to 255, which can affect the model's output. Consequently, we scale the images to fall within the range $[0, 1]$ and resize them to the default dimensions of 224×224 before training the model. Furthermore, we normalize the pixels using the means and standard deviations (*std*) of ImageNet, ensuring optimal performance during the training process: $mean = [0.485, 0.456, 0.406]$, $std = [0.229, 0.224, 0.225]$.

$$output[n] = \frac{input[n] - mean[n]}{std[n]} \quad (4)$$

With n : channel number of the input image, $n=3$ if RGB image.

3.4.5 Class weights for the imbalance dataset

To address the issue mentioned above of data imbalance (problem 2 in section 3.4.3), where the minority group is too small compared to the majority group, we apply balanced class weights to the model after each output prediction step. This adjustment mitigates the bias toward the majority group and helps the model classify instances from all classes accurately. The balanced class weights are calculated as follows:

$$w_j = \frac{n_{samples}}{n_{classes} * n_{samples_j}} \quad (5)$$

Where:

w_j : weight of class j^{th}

$n_{samples}$: the total amount of data to train

$n_{classes}$: the number of classes to classify

n_{samples_j} : the number of samples of class j^{th} .

4 Experiments and evaluations

This study adopts various evaluation metrics to assess the performance of the approach used to convert malicious binaries into images and assess the proposed model's performance. These metrics include the accuracy, recall, precision, false positive rate, and receiver operating characteristic (ROC) curve. These metrics are widely accepted within the research community for providing a comprehensive and detailed analysis of model outcomes [Tharwat, 21]. They are calculated based on the following parameters:

- True Positive (TP): The label is Positive, and the prediction is also Positive.
- True Negative (TN): The label is Negative, and the prediction is also Negative.
- False Positive (FP): The label is Negative, but the prediction is Positive.
- False Negative (FN): The label is Positive, but the prediction is Negative.

The proposed approach will be evaluated by accuracy, precision, and recall.

Accuracy is the percentage of correct predictions on all predictions:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

Precision is the percentage of correctly predicting a label on all predictions that predicted that label:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (7)$$

Recall is the percentage of correctly predicting a label on all data that is actually that label:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (8)$$

F1 score represents the harmonic mean of precision and recall, thereby serving as a surrogate for simultaneously evaluating both precision and recall. Consequently, it enhances comprehension in scenarios where the data exhibits an imbalance.

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (9)$$

In the subsequent experiments, all metrics are computed as the macro-average across all classes. To mitigate the issue of overfitting and arrive at more informed judgments regarding the classifier's efficacy, we employ 5-fold cross-validation. Given the limited number of samples available for each experiment, we opt for a value of $k = 5$, which entails an 80/20 split between training and testing data within each fold.

4.1 Benign and ransomware classification

In the first scenario, the paper concentrates on classifying ransomware and benign files (Dataset DT1). Three methods are compared: not using section information (basic), using only section information, and combining section information with PE header features. The experiment will be conducted using four models: ResNet-50, VGG16,

Vision Transformer (ViT), and the proposed peIRCECon model to test their ability to distinguish between benign and ransomware files.

As our solution is fundamentally based on image classification, we have expanded our experimental comparison methods by including a state-of-the-art image classification model for comparison. Specifically, we selected the Vision Transformer (ViT) model as a benchmark for comparison, as it currently holds leading positions in image classification algorithms as per the rankings on *PaperWithCode*⁶. Furthermore, among various configurations of ViT models, the ViT-B/16 model stands out due to its relatively smaller size and high performance [Dosovitskiy et al., 20]. This makes it an ideal representative of state-of-the-art image classification models for comparison purposes.

Results (see Figure 7) confirm that integrating section information notably boosts models' classification performance, including PE header features that sharpen the differentiation between ransomware and benign files. In detail, using only basic features, peIRCECon, ResNet50, ViT-B/16, and VGG16 achieved respective accuracies of 0.957, 0.938, 0.931, and 0.910, with peIRCECon leading.

Using features only from the section table in the PE header, all models improved in accuracy, with peIRCECon again achieving the top result of 0.988. It indicates that the section table contains highly discriminative information for identifying ransomware. The models ResNet50 and VGG16 had lower accuracy of 0.974 and 0.951, respectively, suggesting they could not fully capitalize on the section table features. The ViT-B/16 transformer model was closer to peIRCECon at 0.954 accuracy.

When using both sections and the best features, peIRCECon obtained the highest accuracy of 0.993, outperforming all other models by a significant margin. This demonstrates peIRCECon's strength in integrating spatial details from the section table with other informative PE header features. While the other models also improved in accuracy, none surpassed 0.983, achieved by ResNet50.

PeIRCECon notably outperformed other models for both benign and ransomware classes, achieving the highest F1-score of 0.956, bolstered by strong precision (0.972) and recall (0.940) rates. ResNet50 trailed closely with a 0.940 F1-score for the benign class. Other models showed F1-scores ranging between 0.908 and 0.940 across classes.

Overall, peIRCECon demonstrated balanced precision and recall, leading to the best macro-average F1-score of 0.956. It effectively detected both benign and ransomware classes. The other convolutional and vision transformer models had lower but comparable macro-F1 scores from 0.908 to 0.929.

⁶ <https://paperswithcode.com/>

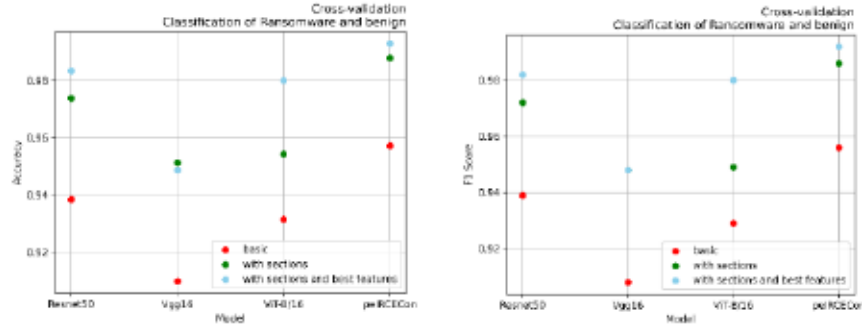


Figure 7: Results for different datasets when distinguishing between ransomware and benign files

Through the initial experimentation, the significance of incorporating PE header features and sections into ransomware images has been identified. The next challenge is determining the optimal number of PE header features to apply to achieve the best results.

In order to evaluate the model on the dataset after training, the Grad-CAM method [Selvaraju et al., 17] is employed to analyze the model's focus on specific locations within ransomware images. To perform the Grad-CAM computation, the Fully Connected Layer is replaced with Global Average Pooling for the Convolutional Network layer, which yields the weight of each layer. Utilizing these factors, Grad-CAM calculates the derivative of y^c (probability of class c) with respect to the degree of the feature map at position A of a $Conv2D$ layer. Finally, it performs backpropagation and averages the results to obtain the weights a_k^c , which represents the heat distribution of objects within an image.

$$a_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \quad (9)$$

Z represents the size of the feature map. The results obtained from the GradCam calculations are then passed through the ReLU (Rectified Linear Unit) activation function to generate a GradCam heat map. The ReLU function helps maintain non-linear properties while eliminating negative values, resulting in a more interpretable and visually coherent representation of the features' importance in the image. This heat map highlights the areas in the image where the model focuses its attention, allowing for a better understanding and analysis of the model's decision-making process in classifying ransomware and benign files.

$$L_{Grad-Cam}^c = ReLU(\sum_k a_k^c A^k) \quad (10)$$

In Figure 8, the darker regions (in red) indicate the areas where the model concentrates on the object for classification. The VGG16 model focuses on almost the entire image, resulting in significant differences between the images containing PE headers and sections without this information. For the ResNet-50 model, the addition of PE header features considerably impacts the classification results, as the model can identify areas containing essential information. In contrast, the results of the ViT model

remain relatively similar because the model still focuses on specific regions, and the addition of PE header features does not substantially affect the outcome.

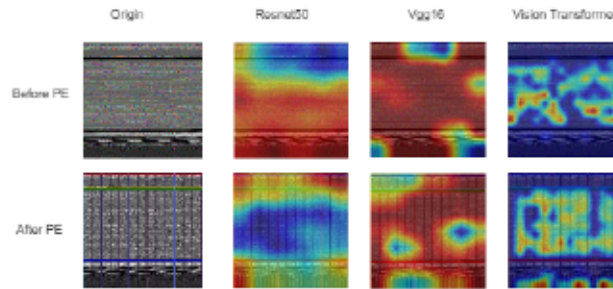


Figure 8: GradCam method evaluation results

4.2 Classification of ransomware families

In the second scenario, the focus shifts to classifying distinct ransomware families (Dataset DT3). A dataset comprising ransomware family samples from meticulously categorized families is prepared to accomplish this task. The impact of the features on classification performance is then analyzed by incorporating 5 or 10 sample features. It should be noted that these numbers are presented as examples only, and in practice, the study may utilize different numbers of features.

Figure 9 reveals that the number of features can be considered a hyperparameter, which needs to be adjusted during the application of the method. The number of PE header features added significantly impacts the final outcome, and this relationship is not linear.

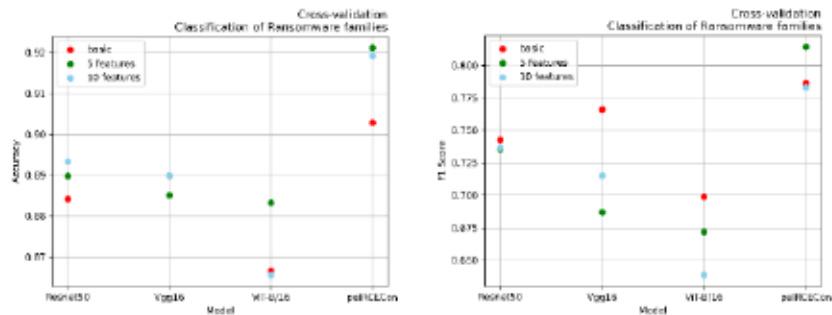


Figure 9: Result when applying the model on the dataset to classify the ransomware families

When classifying ransomware families using basic PE header features, peIRCECon achieved the highest accuracy of 0.903 versus 0.867-0.890 for Resnet50, VGG16, and ViT-B/16. The peIRCECon demonstrated the highest accuracy when utilizing the five best header features, successfully capturing nuanced differences between ransomware families. It outperforms the 0.883-0.890 scores of the other kinds of models. The better performance is because our model is integrated with two kinds of structured deep learning models, including VGG16 and ResNet50. These models extract different

feature information that can be combined to create a better model. Finally, utilizing the ten best features, peIRCECon reached an accuracy of 0.919, considerably higher than ResNet50 at 0.893 accuracy, VGG16 at 0.890 accuracy, and ViT-B/16 lagging behind at 0.866 accuracy. The generic ViT-B/16 transformer struggled to incorporate the color-coded header lines within its image patch approach, hampering accuracy gains from supplementary features.

Using basic PE header features, peIRCECon attained the highest macro F1-score of 0.786 with balanced precision (0.81) and recall (0.78), outperforming other models at 0.699-0.766. With the five best features, it achieved an even higher top macro F1-score of 0.814, significantly exceeding ResNet50 at 0.736, VGG16 at 0.687, and ViT-B/16 at 0.672, with superior precision (0.85) and recall (0.81) across many families. Finally, utilizing the ten best features, peIRCECon reached a high macro F1-score of 0.783 compared to just 0.715-0.737 for other models, again demonstrating strong precision (0.8) and recall (0.79) balance. Across basic, five best, and ten best feature experiments, peIRCECon designed for PE files consistently attained the highest F1-scores, underscoring the value of specialized modeling for ransomware family classification.

4.3 Classifying ransomware in the presence of other malware types

In the third scenario, the objective is to differentiate ransomware from other types of malware (Dataset DT2). This task is crucial in understanding the specific characteristics of ransomware and effectively separating it from a broader range of malicious software. The experimental results (refer to Figure 10) indicate that the peIRCECon model achieved a high accuracy of 0.962 using only basic PE header features, surpassing the performance of other models ranging from 0.918 to 0.950. By incorporating the five most informative features, peIRCECon achieved an even highest accuracy of 0.979, surpassing ResNet50 (0.956), VGG16 (0.904), and ViT-B/16 (0.941) in terms of accuracy, thus leveraging the most informative signals. Furthermore, utilizing the top 10 features, peIRCECon achieved a high accuracy of 0.975, outperforming other models, which achieved an accuracy ranging from 0.927 to 0.959. It highlights the ability of peIRCECon to extract differences between ransomware and other malware types effectively. Remarkably, when incorporating 10 PE header features, the VGG16 model demonstrates improved performance. However, its performance declines when only five features are utilized.

The peIRCECon achieved a superior 0.961 F1-score with basic features by balancing precision (0.964) and recall (0.966) between classes. PeIRCECon further improved F1-score to 0.978 using five selective features, with class-balanced precision (0.980) and recall (0.978). Finally, with the ten best features, it attained a high value of F1-score (0.974), precision (0.973), and recall (0.973), mastering the nuanced separation of ransomware from other malware. By optimizing all three metrics through its tailored design, peIRCECon substantiated the value of specialized modeling over general architectures for accurate ransomware detection from executables.

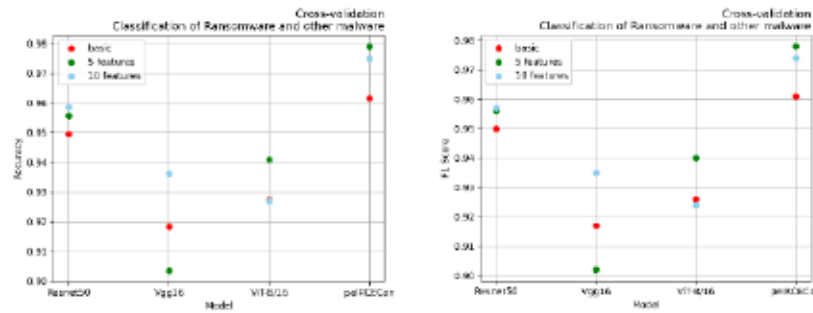


Figure 10: Result when applying the model on the dataset to classify the ransomware and other malware types

5 Conclusions

We have developed a ransomware classification and detection approach - peIRCECon (PE Header-Image-based Ransomware Classification Ensemble with Concatenating). This approach integrates valuable information from PE Headers into image representations of ransomware and leverages advanced machine learning models to effectively select the most suitable features for encoding into ransomware images. An ensemble deep learning model, combining VGG16 and ResNet50, is constructed using the concatenating method to aggregate outputs from all preceding layers. It results in a model that preserves prior layers' characteristics while reducing the total number of parameters, thereby enhancing computational efficiency.

Our findings indicate that incorporating optimal features from the PE header can significantly improve the accuracy of deep learning models in classifying and detecting ransomware. Among the strengths of our work, the proposed peIRCECon approach outperformed generic models across different sets of features and datasets, demonstrating its effectiveness in distinguishing ransomware from benign files, other types of malware, and even different ransomware families.

However, the work presented also has its limitations. With the continuous development and evolution of ransomware, obfuscation, information-hiding techniques, and packing methods are becoming more advanced. In such cases, our proposed preprocessing method may not fully represent the features of evolving ransomware variants. Hence, while our approach has shown promising results, it is not immune to the rapid advancements in ransomware techniques, which may affect its future performance.

Future research should focus on improving preprocessing and feature extraction techniques to better account for these evolving threats. Optimization of the number of selected best features is also an important task. Exploring other deep learning architectures and ensemble methods could also yield further improvements in ransomware detection and classification. Our work has made significant contributions to the field of malware classification, providing insights into developing robust, efficient, and accurate ransomware detection methods. We hope other researchers and practitioners can leverage our findings to advance the development of robust solutions for this critical cybersecurity issue.

References

- [Akshay Kumar & Rangasamy, 23] Akshay Kumar, E., & Rangasamy, J.: Light-Weight Deep Learning Models for Visual Malware Classification, In *Advances in Signal Processing, Embedded Systems and IoT: Proceedings of Seventh ICMEET-2022*, 485-495, 2023, https://doi.org/10.1007/978-981-19-8865-3_44.
- [Bensaoud et al., 20] Bensaoud, A., Abudawood, N., & Kalita, J.: Classifying malware images with convolutional neural network models, *International Journal of Network Security*, vol. 22, no. 6, 1022-1031, 2020, <https://doi.org/10.48550/arXiv.2010.16108>.
- [Berman et al., 19] Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L.: A survey of deep learning methods for cyber security, information, vol. 10, no. 4, 122, April 2019, <https://doi.org/10.3390/INFO10040122>.
- [Casolare et al., 22] Casolare, R., Ciaramella, G., Iadarola, G., Martinelli, F., Mercaldo, F., Santone, A., & Tommasone, M.: On the Resilience of Shallow Machine Learning Classification in Image-based Malware Detection, *Procedia Computer Science*, vol. 207, 145-157, 2022, <https://doi.org/10.1016/j.procs.2022.09.047>.
- [Chaganti et al., 22] Chaganti, R., Ravi, V., & Pham, T. D.: Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification, *Journal of Information Security and Applications*, vol. 69, 103306, 2022, <https://doi.org/10.1016/j.jisa.2022.103306>.
- [Chen & Guestrin, 16] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system, In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 13, 2016, San Francisco, California, USA, 785-794, <https://doi.org/10.1145/2939672.2939785>.
- [Dixit & Silakari, 21] Dixit, P., & Silakari, S.: Deep learning algorithms for cybersecurity applications: A technological and status review, *Computer Science Review*, no. 39, 100317, February 2021, <https://doi.org/10.1016/j.cosrev.2020.100317>.
- [Dosovitskiy et al., 20] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ..., & Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale, *arXiv preprint arXiv:2010.11929*, October 2020.
- [Geurts et al., 06] Geurts, P., Ernst, D., & Wehenkel, L.: Extremely randomized trees, *Machine learning*, vol. 63, 3-42, 2006, <https://doi.org/10.1007/s10994-006-6226-1>.
- [Gibert et al., 20] Gibert, D., Mateu, C., & Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, *Journal of Network and Computer Applications*, August 2020, vol. 153, 102526, <https://doi.org/10.1016/j.jnca.2019.102526>.
- [He et al., 16] He, K., Zhang, X., Ren, S., & Sun, J.: Deep residual learning for image recognition, In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778, 2016, <https://doi.org/10.1109/CVPR.2016.90>.
- [Kancherla & Mukkamala, 13] Kancherla, K., & Mukkamala, S.: Image visualization based malware detection, In *2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, 40-44, April 2013, <https://doi.org/10.1109/CICYBS.2013.6597204>.
- [Khandokar et al., 21] Khandokar, I., Hasan, M., Ernawan, F., Islam, S., & Kabir, M. N.: Handwritten character recognition using convolutional neural network, In *Journal of Physics: Conference Series*, vol. 1918, no. 4, 042152, IOP Publishing, June 2021, <https://doi.org/10.1088/1742-6596/1918/4/042152>.

- [Kim, 18] Kim, S.: PE header analysis for malware detection, Master's Projects, San Jose State University, 624, 2018.
- [Krizhevsky et al., 17] Krizhevsky, A., Sutskever, I., & Hinton, G. E.: Imagenet classification with deep convolutional neural networks, *Communications of the ACM*, vol. 60, no. 6, 84-90, 2017, <https://doi.org/10.1145/3065386>.
- [Long et al., 15] Long, J., Shelhamer, E., & Darrell, T.: Fully convolutional networks for semantic segmentation, In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431-3440, 2015, <https://doi.org/10.1109/tpami.2016.2572683>.
- [Murali et al., 20] Murali, R., Ravi, A., & Agarwal, H.: A malware variant resistant to traditional analysis techniques, In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 1-7, February 2020, <https://doi.org/10.1109/ic-ETITE47903.2020.264>.
- [Nataraj et al., 11] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S.: Malware images: visualization and automatic classification, In *Proceedings of the 8th international symposium on visualization for cyber security*, 1-7, July 2011, <https://doi.org/10.1145/2016904.2016908>.
- [Oz et al., 22] Oz, H., Aris, A., Levi, A., & Uluagac, A. S.: A survey on ransomware: Evolution, taxonomy, and defense solutions, *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, 1-37, September 2022, <https://doi.org/10.1145/3514229>.
- [Patil et al., 23] Patil, V., Shetty, S., Tawte, A., & Wathare, S.: Deep Learning and Binary Representational Image Approach for Malware Detection, In *2023 International Conference on Power, Instrumentation, Control and Computing (PICC)*, 1-7, April 2023, <https://doi.org/10.1109/PICC57976.2023.10142644>.
- [Prokhorenkova et al., 18] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A.: CatBoost: unbiased boosting with categorical features, *Advances in neural information processing systems*, vol. 31, 2018.
- [Reilly et al., 23] Reilly, C., O Shaughnessy, S., & Thorpe, C.: Robustness of Image-Based Malware Classification Models trained with Generative Adversarial Networks, In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference*, 92-99, June 2023, <https://doi.org/10.1145/3590777.3590792>.
- [Rezende et al., 17] Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., & De Geus, P.: Malicious software classification using transfer learning of resnet-50 deep neural network, In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1011-1014, December 2017, <https://doi.org/10.1109/ICMLA.2017.00-19>.
- [Salota & Singh, 23] Salota, R., & Singh, I.: Efficient Image based Malware Classification Using a Modified VGG based deep Learning Model, *Harbin Gongcheng Daxue Xuebao/Journal of Harbin Engineering University*, vol. 44, no. 5, 419-431, 2023.
- [Selvaraju et al., 17] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization, In *Proceedings of the IEEE international conference on computer vision*, 618-626, October 2017, <https://doi.org/10.1109/ICCV.2017.74>.
- [Simonyan and Zisserman, 14] Simonyan, K., & Zisserman, A.: Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*, September 2014, <https://doi.org/10.48550/arXiv.1409.1556>.

[Singh et al., 19] Singh, A., Handa, A., Kumar, N., & Shukla, S. K.: Malware classification using image representation, In *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Proceedings 3*, 75-92, Springer International Publishing, June 2019, https://doi.org/10.1007/978-3-030-20951-3_6.

[Singh et al., 18] Singh, J., & Singh, J.: Challenge of malware analysis: malware obfuscation techniques, *International Journal of Information Security Science*, vol. 7, no. 3, 100-110, September 2018.

[Subakan et al., 21] Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., & Zhong, J.: Attention is all you need in speech separation, In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 21-25, June 2021, <https://doi.org/10.48550/arXiv.2010.13154>.

[Tan and Le, 19] Tan, M., & Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks, In *International conference on machine learning*, 6105-6114, May 2019, <https://doi.org/10.48550/arXiv.1905.11946>.

[Tayyab et al., 22] Tayyab, U. E. H., Khan, F. B., Durad, M. H., Khan, A., & Lee, Y. S.: A survey of the recent trends in deep learning based malware detection, *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, 800-829, September 2022, <https://doi.org/10.3390/jcp2040041>.

[Tharwat, 21] Tharwat, A.: Classification assessment methods, *Applied computing and informatics*, vol. 17, no. 1, 168-192, January 2021, <https://doi.org/10.1016/j.aci.2018.08.003>.

[Vasan et al., 20] Vasan, D., Alazab, M., Wassan, S., Safaei, B., & Zheng, Q.: Image-Based malware classification using ensemble of CNN architectures (IMCEC), *Computers & Security*, vol. 92, 101748, May 2020, <https://doi.org/10.1016/j.cose.2020.101748>.

[Vu et al., 20] Vu, D. L., Nguyen, T. K., Nguyen, T. V., Nguyen, T. N., Massacci, F., & Phung, P. H.: HIT4Mal: Hybrid image transformation for malware classification, In *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 11, e3789, November 2020, <https://doi.org/10.1002/ett.3789>.

[Xiao et al., 21] Xiao, M., Guo, C., Shen, G., Cui, Y., & Jiang, C.: Image-based malware classification using section distribution information, *Computers & Security*, vol. 110, 102420, November 2021, <https://doi.org/10.1016/j.cose.2021.102420>.

[Yang et al., 22] Yang, Y., Wu, Z., Yang, Y., Lian, S., Guo, F., & Wang, Z.: A Survey of Information Extraction Based on Deep Learning, *Applied Sciences*, vol. 12, no. 19, 9691, September 2022, <https://doi.org/10.3390/app12199691>.

[Zhang et al., 21] Zhang, C., Benz, P., Argaw, D. M., Lee, S., Kim, J., Rameau, F., Bazin, J. C., & Kweon, I. S.: Resnet or densenet? introducing dense shortcuts to resnet, In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 3550-3559, January 2021, <https://doi.org/10.1109/WACV48630.2021.00359>.

[Zhang & Zheng, 22] Zhang, Y., & Zheng, X.: Development of Image Processing Based on Deep Learning Algorithm, In *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 1226-1228, April 2022, <https://doi.org/10.1109/IPEC54454.2022.9777479>.

[Zhao et al., 23] Zhao, Z., Zhao, D., Yang, S., & Xu, L.: Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model, *Security and Communication Networks*, 2023, <https://doi.org/10.1155/2023/6390023>.