Malware Detection Through RGB Memory Dump Image Processing Using HOG Descriptor, UMAP, and Machine Learning Method

Adiva Veronia
Computer Engineering, Electrical Engineering Department
Universitas Indonesia
Depok, West Java 16424
adiva.veronia@ui.ac.id

Yan Maraden
Computer Engineering, Electrical Engineering Department
Universitas Indonesia
Depok, West Java 16424
maradens@ui.ac.id

***Abstract*— The increase in the number of internet users globally along with advances in technology provides great benefits in terms of access to information and communication. However, it also increases cybersecurity threats, demanding more sophisticated strategies in malware analysis. Static, dynamic, and memory forensic malware analysis is comprehensive in dealing with increasingly complex cyber-attacks. Memory forensics, among the other techniques, is a very crucial technique, especially in detecting hidden malware movements. This research presents a malware detection technique using the concept of forensic memory by using a memory dump dataset that has been converted to an RGB image, then utilizing computer vision techniques, namely the HOG descriptor, to extract its features. Supervised and unsupervised UMAP are used to reduce the dimensions of the data. Multinomial Logistic Regression, KNN, and SVM (RBF) are used as machine learning algorithms to classify the data. Among the three algorithms, KNN has the highest performance with an accuracy of 90.5% using unsupervised UMAP. This shows that the technique used has good effectiveness in detecting malware.**

Keywords—Malware Detection, Memory Dump, HOG Descriptor, UMAP, Machine Learning

I. INTRODUCTION

The acceleration in the development of technology and the internet can be observed every year. Data obtained from DataReportal shows that at the beginning of the fourth quarter of 2023, the number of internet users reached 5.16 billion, equivalent to 64.4% of the total world population [1]. Compared to 2022, when the number of internet users reached 4.95 billion, or 62.5% of the total world population [2], this demonstrates a significant growth in internet usage worldwide. Advanced internet and technology certainly have many benefits, such as making it easier for people to access information with just their fingertips and enabling long-distance communication through various available applications. However, the increasing number of internet users and the proliferation of technological advancements comes with cybersecurity risks.

According to 2023 cybersecurity statistics obtained from Packetlabs, a Canadian cybersecurity company, there were approximately 800,000 cyberattacks in 2023 [3]. This number indicates a significant upward trend. Additionally, there is an increase of 300,000 new malware every day. Furthermore, 4.1 million websites are known to contain malware [3]. This underscores the importance of enhancing cybersecurity efforts.

The application of malware analysis techniques, which include static, dynamic, and memory forensic approaches, constitutes a comprehensive strategy in addressing cyber threats [4]. Static analysis allows the identification of potential malware through code and file characteristics without executing the program, providing a crucial initial view for early detection. However, its drawback lies in its inability to reveal the actual behaviour of malware that only activates when executed and its vulnerability to obfuscated malware.

On the other hand, dynamic analysis can address the shortcomings of static analysis by executing malware in a controlled environment to observe its behaviour and impact directly. However, this technique typically requires more resources and carries a higher risk because it can infect or damage the system. Therefore, while dynamic analysis can provide deep insights into how malware operates, it still requires careful risk management to minimize potential negative impacts.

In the face of increasingly sophisticated and hidden malware, such as fileless malware that leaves no traces on disk, memory forensic approaches become crucial. This technique allows for the detection of malware solely through RAM checks, often where malware hides its activities [5, 6, 7, 8, 9]. However, manual memory forensic analysis can be time-consuming and prone to data interpretation errors.

Image-based malware detection approaches have become an innovation in cybersecurity. This technique involves converting memory dumps into visual representations, such as grayscale or RGB images, which can then be analysed for its features using image processing and machine learning techniques. This method leverages the advantage of visualizing hidden patterns in memory data that may be difficult to identify using traditional techniques. In contrast, non-image approaches rely on direct analysis of raw data through static and dynamic techniques to detect malware. This approach often requires deep analysis of binary code and malware runtime behaviour. Although effective, non-image approaches can be more complex and time-consuming. Research shows that transforming memory data into images can improve the accuracy and efficiency in detecting hidden malware more effectively [7, 8].

Another approach in non-image-based malware detection is using machine learning with raw memory dump data. This technique involves directly extracting features from memory dumps without converting them into visual forms. It relies on pattern analysis from binary data and requires intensive processes to identify relevant features that can differentiate between normal activity and malware activity [9]. While this approach can detect certain types of malware, it often faces challenges in recognizing complex and obfuscated malware. Image-based approaches, on the other hand, offer advantages in recognizing visual patterns that are difficult to identify by

non-image methods and can enhance overall detection performance.

In light of these issues, this research will use RGB images generated from memory dump file conversions, then use computer vision techniques to describe and analyse related patterns to produce vector-based information. Subsequently, machine learning algorithms can be used to classify various types of malwares from the obtained image information.

This approach is expected to help in developing intelligent and adaptive malware detection systems capable of recognizing patterns indicative of cyber threats without relying on conventional indicators such as indicators of compromise, user anomalies, or signature-based detection, which are often evaded by increasingly sophisticated malware. Additionally, this method is expected to address challenges arising from manual memory forensic analysis, such as long analysis times and the potential for data interpretation errors.

This paper is organized as follows: In Section 2, materials which underlies this research is introduced. Section 3 details the methods that will be used for this research include image information extraction, dimension reduction, to machine learning modeling. Further, Section 4 presents the experimental results and discussion of each stage of the method and how the results influence each other . Finally, Section 5 concludes the research and explain possible suggestions that can be used for further research.

II. MATERIALS

A. Malware

Malware, or malicious software, is a term that refers to programs or software designed to disrupt and/or damage the operation of a system, often causing data breaches within the system. Based on the need for a host file, malware can be categorized into two types: host dependent and fileless malware. Host dependent malware is a traditional type of malware that requires a host file for execution. The presence of a host file in the system makes it easier for cybersecurity software to identify and remove the malware. On the other hand, fileless malware does not require a host file for execution; instead, it operates within the computer's memory (RAM) and utilizes the system's built-in software to carry out attacks.

Based on its types, malware can be categorized as follows [10, 11]:

- Virus: Malware that can replicate itself and spread copies to other systems through user interaction.
- Worm: Malware that can replicate itself and spread copies to other systems without user interaction.
- Trojan: Malware that disguises itself as a safe application or code but is actually for malicious purposes.
- Spyware: Malware used to monitor and collect user activities, such as web activity monitoring and keylogging.
- Ransomware: Malware that encrypts user data and then demands a ransom to provide the decryption key.
- Adware: Malware that displays unwanted ads and causes users to download more malware.
- Botnet: Malware designed to take control of multiple computers and execute commands from a control server.

Malware can be spread through various methods, such as phishing emails, accessing malicious websites, exploiting vulnerabilities in user systems, and infected USB devices. For prevention, it is crucial for users to download antivirus or other anti-malware software as the first line of defense to detect and remove malware entering the system. Additionally, regularly updating the operating system and software is an effective way to avoid system breaches through system vulnerabilities.

B. HOG (Histogram of Oriented Gradients) Descriptor

The HOG (Histogram of Oriented Gradients) descriptor is a feature extraction technique commonly used in object detection within image processing and computer vision. This technique generates feature vectors by capturing the local appearance or visual cues of objects through the analysis of intensity gradient distribution and edge direction in the image. In other words, the HOG descriptor allows for pattern recognition in images based on their gradient characteristics, as shown in Figure 1. By examining these patterns, the HOG descriptor can identify important features within an object, such as edges, corners, or textures, which are useful for effectively detecting objects. This technique uses the magnitude and direction of gradients to compute its features.

Figure 1. Original Image (Left) and Image of Feature Extraction Process Results Using HOG Descriptor (Right)

Fundamentally, to generate feature vectors using HOG descriptor, three crucial steps are required. First, gradient computation, where the colour changes in each small part of the image are calculated. Second, creating an orientation histogram by grouping colour changes based on their direction, such as up, down, right, and left. Third, block normalization by calculating the average and standard deviation of each histogram block to ensure the feature vectors have a consistent scale. According to Dalal and Triggs [12], the original proponents of HOG descriptor, resizing the original image to 128x64 pixels yields optimal feature vectors for pedestrian detection.

To calculate the gradients of an image, consider a 3x3 block, where the magnitude and angle values of the image are combined using the following equations:

$$Gx = I(r, c + 1) - I(r, c - 1) \quad (1)$$

$$Gy = I(r - 1, c) - I(r + 1, c) \quad (2)$$

where G_x and G_y represent the gradient values along the x and y axes, I represent the intensity value at a pixel location, r represents the row, and c represents the column. From the

obtained G_x and G_y values, the magnitude and angle of each pixel are calculated using the following equations:

$$\text{Magnitude}(\mu) = \sqrt{G_x^2 + G_y^2} \quad (3)$$

$$\text{Angle}(\theta) = |\tan^{-1}(G_x/G_y)| \quad (4)$$

After obtaining the gradients for each pixel, the gradient matrix, which is a matrix of angles and magnitudes, is divided into 8x8 cells to form a block. For each block, a nine-point histogram is computed. This histogram has 9 bins representing angle ranges of 20 degrees each, as shown in Figure 2. The values in the magnitude matrix are assigned to the histogram according to the angle matrix values at the same row and column. The resulting calculation matrix will have a shape of 16x8x9.

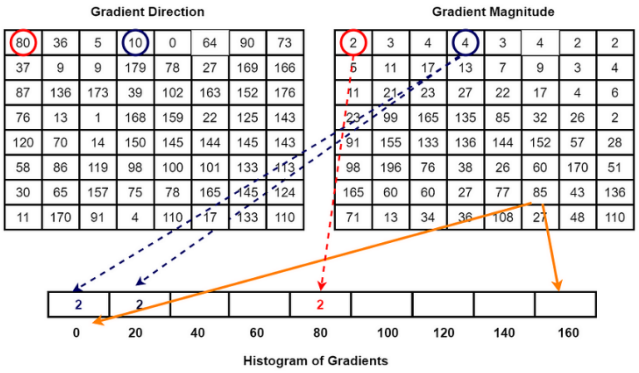

Figure 2. 9 Bin Histogram Calculation Method

Once the computation for all blocks is completed, 4 blocks from the 9-point histogram matrix are combined to form a 2x2 block. This is done in an overlapping manner with 8 pixels. For the 4 cells within a block, all 9-point histograms are combined for each constituent cell, forming a 36-feature vector as shown in Figure 2.

The final matrix is normalized, which serves to achieve a feature representation that is invariant to certain variations in the image while maintaining the strength of object detection and recognition. One of the drawbacks of the HOG descriptor is the requirement for uniform image sizes so that the feature vectors can be well-represented. Despite this, the HOG descriptor can be chosen as a method with the capability to visually detect images comprehensively, both from a local and global perspective.

C. UMAP (Uniform Manifold Approximation and Projection)

UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction technique used to map high-dimensional complex data into a lower-dimensional space. UMAP is based on manifold learning theory. A manifold is a mathematical representation of the complex data structure in a simpler space. According to McInnes, Healy, and Melville [13, 14], the UMAP algorithm is based on three assumptions about the data:

- The data is uniformly distributed on a Riemannian manifold.
- The Riemannian metric is locally constant (or can be approximated as such).
- The manifold is locally connected.

In the initial stage, UMAP builds a high-dimensional graph called a fuzzy simplicial complex [15], which is a weighted graph representation where the edge weights represent the likelihood of two points being connected. To determine this connectivity, UMAP expands the radius from each point and connects points when the radius overlap. Choosing the appropriate radius is crucial because a radius that is too small will result in small clusters, whereas a radius that is too large will connect data points together. This is addressed by UMAP selecting the radius locally based on the distance to the nth nearest neighbour of each point. From these results, UMAP forms a fuzzy graph by reducing the likelihood of connections as the radius increases. By ensuring that each point must connect to at least its nearest neighbours, UMAP balances preserving local structure with global structure.

UMAP has several parameters that affect the effectiveness of dimensionality reduction results, namely 'n_neighbors', 'min_dist', 'n_components', and 'metric' [16]. The 'n_neighbors' parameter controls the balance of local and global structures by determining the number of nearest neighbors during the dimensionality reduction process. The smaller the parameter value, the more UMAP focuses on local structure. The 'min_dist' parameter controls how closely data points can be packed together by calculating the minimum distance. The smaller the 'min_dist' value, the more clumped the data. The 'n_components' parameter controls the number of dimensions of the reduced data. The 'metric' parameter sets the type of metric used to calculate the distance between data points, such as Euclidean, Manhattan, Chebyshev, and Minkowski distances.

UMAP has two types of methods, supervised and unsupervised. In the supervised method, UMAP uses class label information in the data to produce projections that preserve relevant class label structures in the lower-dimensional space. This allows UMAP to account for class structure in low-dimensional analysis, which often provides better results [17] for classification or regression tasks. In the unsupervised method, UMAP operates only with the intrinsic geometry information of the data without considering class labels. The unsupervised mode of UMAP can be used for general data exploration and visualization, where the algorithm tries to represent the data efficiently in a lower-dimensional space without considering any class structure that may exist.

UMAP is an algorithm that can preserve both local and global structures of the data so that relationships between data in high dimensions can be maintained in lower-dimensional representations. Additionally, this algorithm has good scalability and fast processing times, especially for large amounts of data. However, producing good dimensionality reduction results requires selecting the appropriate parameters for different data. Furthermore, the non-deterministic nature of UMAP can cause variation in the results each time the algorithm is run, although this can be controlled by setting a random seed.

D. Logistic Regression

Logistic regression is a supervised machine learning algorithm that uses probabilities to determine its output. This algorithm is particularly useful when we have a binary output variable such as 0 or 1, true or false. Like linear regression, logistic regression also uses a line, but the line is S-shaped with values between 0 and 1, as shown in Figure 3. This S-shaped line is called the sigmoid function or logistic function.

To determine the classification of the prediction results, this algorithm uses a threshold value to convert the generated probability values into binary classes. For instance, if the threshold value is set at 0.5, probability values greater than 0.5 will be classified as the positive class (1), while probability values less than 0.5 will be classified as the negative class (0). The threshold value can be adjusted based on the needs and context of the application.

To determine the probability of an input value, logistic regression uses the sigmoid formula with the equation:

$$\sigma(z) = \frac{1}{1-e^{-z}} \quad (5)$$

where z is the input or logit with the equation:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n \quad (6)$$

Here, β_0 is the bias or intercept, β_1, β_2, …., β_n are the coefficients for each independent variable, and X, X_2, ..., X_n are the values of the independent variables. The probabilities can be calculated as follows:

$$P(y = 1) = \sigma(z) \quad (7)$$
$$P(y = 0) = 1 - \sigma(z) \quad (8)$$

Based on its categories, logistic regression can be distinguished into three types:

- Binomial or binary: classifies data with two classes, such as 0 or 1, true or false, and others.
- Multinomial: classifies data with more than two categories, for example, animal types such as cat, dog, or rabbit.
- Ordinal: classifies data with categories that are ordered or ranked, for example, customer satisfaction such as dissatisfied, satisfied, and very satisfied.

Logistic regression is one of the simpler machine learning algorithms compared to others, making it quite easy to implement. Additionally, this algorithm can process large volumes of data at high speeds with minimal computation. This makes it frequently used as a baseline model when comparing performance with more complex machine learning algorithms. Its ability to model linear relationships well and its easily understandable coefficient interpretation further contribute to its popularity. Logistic regression is a popular choice in various applications, especially when the main focus is on model interpretability and computational efficiency.

E. KNN (K- Nearest Neighbor)

KNN is one of the fundamental classification algorithms in supervised machine learning. KNN does not assume a specific data distribution and requires two main assumptions: that data points close to each other have similar or the same features, and a method of measuring distance [18]. This algorithm can handle both numerical and categorical data, although categorical data must first be converted to numerical data using methods such as label encoding or one-hot encoding.

In its application, KNN operates by identifying the k nearest neighbours from one point to another and making decisions based on the majority of these neighbours. A k value that is too small can make the model too sensitive to noise, leading to overfitting, while a large k value can result in underfitting, making the model unable to recognize data patterns. To find the optimal k value, techniques such as cross-validation, grid search, or the Elbow method can be used.

The main metric in KNN is the distance calculation using Euclidean, Manhattan, or Minkowski distance. This distance calculation determines how close or far one point is from another. Choosing the right distance metric is crucial as it can affect the performance of the KNN model. Euclidean distance is often used because it is simple and intuitive, while Manhattan and Minkowski distances can also be used depending on the characteristics of the data. Manhattan distance is suitable for data with many outliers, categories, and grid structure data. Minkowski distance is suitable for data requiring flexibility in distance metric selection.

KNN is a simple and intuitive algorithm, making it easy to understand and implement. However, this algorithm requires a lot of memory and can become slow when processing large datasets due to the distance calculations between data points, although large datasets can yield effective results. Additionally, KNN is sensitive to irrelevant features and requires data normalization to ensure that all data features contribute equally.

F. SVM (Support Vector Machine)

SVM is one of the fundamental supervised machine learning classification algorithms. SVM was developed in 1963 by Vladimir Vapnik and his team. This algorithm is known for its ability to handle high-dimensional data and perform well in various applications such as face recognition, text classification, and others. Essentially, SVM works by finding the optimal hyperplane to separate data from different classes with the largest margin.

A hyperplane in p dimensions will be a p-1 dimensional space situated in the p-dimensional space. The margin is the distance between the hyperplane and the nearest data points from each class. There are two types of margins: hard margin and soft margin. A hard margin is used when the data can be linearly separated without any misclassification. A soft margin is used when outliers or misclassifications are allowed. The margin function that is maximized aims to reduce the risk of classification errors. Support vectors, which are the data points closest to the decision boundary, determine the width of the margin. An illustration of the components of SVM can be seen in Figure 3.

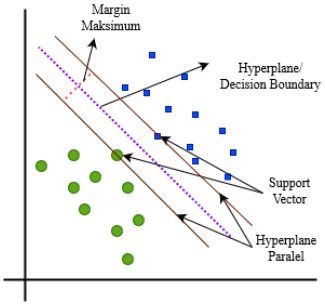

Figure 3. SVM Illustration

Based on the way data is separated, SVM can be divided into linear and kernel or non-linear SVM. Linear SVM uses a linear hyperplane to separate data into different classes because the data can be linearly separated in the feature space. Non-linear SVM is used when the data cannot be linearly separated using a kernel function. A kernel function maps data to a higher-dimensional space, especially for data that cannot be linearly separated, by returning the dot product of vectors

in the feature space. The kernel function is defined as, if there are data points x, z ∈ X and a mapping Φ: X → RN, then,

$$k(x,z)=\langle\Phi(x),\Phi(z)\rangle \quad (9)$$

Mapping data to a higher dimension can result in high computational resources being required, hence the need for a solution called the kernel trick. The kernel trick works by representing the data only through pairwise similarity comparisons between the original data x with the original coordinates in a lower-dimensional space, rather than directly applying the transformation Φ(x) and representing the data by those coordinates in the higher dimension. Based on the type of kernel, SVM can be divided into four types: linear, polynomial, radial basis function (RBF), and sigmoid kernels with function as follow:

- Linear:

$$k(x, z) = x.z \quad (10)$$

- Polynomial:

$$k(x, z) = (\gamma x \cdot z + r)^d \quad (11)$$

- RBF:

$$k(x, z) = \exp(-\gamma\|x - z\|^2) \quad (12)$$

- Sigmoid:

$$k(x,z) = \tanh(\gamma x \cdot z + r) \quad (13)$$

G. *Dumpware10 Dataset*

Dumpware10 is a dataset consisting of RGB images of memory dumps created by Bozkir et al [7]. The dataset contains 11 classes, with 10 malware classes (adware, worm, trojan, virus) and 1 benign class, in a ratio of 5:2:2:1. It consists of 3686 malware samples and 608 benign samples, divided into training and testing sets at an 80:20 ratio as in Table I. The data was collected by running PE files in the Windows Sandbox environment of Windows 10 (version 1903). Procdump was used with arguments -ma for full memory dumps and -w to wait for the process, with a 5000ms delay ensuring activity. Dump files (.dmp) were organized by class and varied from 10 MB to 100 MB. Memory sizes over 16GB were used for consistency.

TABLE I. DATASET CLASS AND CATEGORY

Class	Category	Jumlah Data		
		Training	Testing	Total
Adposhel	Adware	364	93	457
Allaple.A	Worm	349	88	437
Amonetize	Adware	349	87	436
AutoRun-PU	Worm	158	38	196
BrowseFox	Adware	152	38	190
Dinwod!rfn	Trojan	98	29	127
InstallCore.C	Adware	376	91	467
MultiPlug	Adware	390	98	488
VBA	Virus	399	100	499
Vilsel	Trojan	311	78	389
Benign	-	487	121	608
Total		**3433**	**861**	**4294**

The .dmp files were converted to RGB images using the "bin2png" Python script on Python 3, producing lossless PNG images. Four column sizes were chosen: 224, 300, 4096 pixels, and the square root of the original file size. Images were transformed into squares with 224x224 and 300x300 pixel sizes using Lanczos interpolation. RGB images were selected to retain more information per row, facilitating easy differentiation of similar samples. The initial sizes allow comparative studies on machine learning impacts, while resizing to 224x224 and 300x300 pixels helps with computer vision processing.

H. *Evaluation Metrics*

Evaluation metrics are tools or methods used to measure and assess the performance of a model or system. The performance of machine learning algorithms can be evaluated using quantitative metrics such as accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of the accuracy and effectiveness of the algorithm in classifying and detecting malware within the dataset used. The calculation process involves the number of data points in four categories: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The calculation formulas used are as follows [19]:

- Accuracy =

$$(TP + TN) / (TP + FP + FN + TN) \quad (14)$$

- Precision =

$$TP / (TP + FP) \quad (15)$$

- Recall =

$$TP / (TP + FN) \quad (16)$$

- F1-Score =

$$2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (17)$$

Accuracy is calculated as the number of correct classifications divided by the total number of cases. Precision calculates the number of true positive cases with the number of predicted positive cases. Recall calculates the number of true positive cases with the number of actual positive cases. F1-Score calculates the harmonic mean of precision and recall. The evaluation stage also involves the analysis of the confusion matrix to see the classification accuracy between classes generated by the model. The confusion matrix provides a more detailed picture of the number of correct predictions as well as incorrect predictions for each class in the model.

III. METHODS

A. *Research Design*

The research begins with dataset acquisition, followed by image feature extraction and labelling. The labelled data then undergoes dimensionality reduction. The reduced data is modelled using three machine learning algorithms: multinomial logistic regression, KNN, and SVM with an RBF kernel. Finally, the models are evaluated using metrics to determine their performance with flowchart as follows:

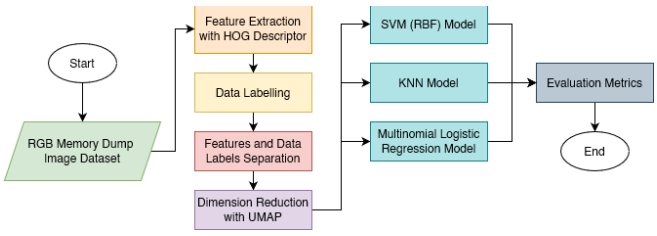

Figure 4. Research Flowchart

B. Dataset

The data size used for this research is 224x224 pixels from an initial size of 224 x height pixels and 300x300 pixels from an initial size of 330 x height pixels, as shown in Figure 5. These sizes were chosen to evaluate whether small and symmetrical images can accurately represent the complex memory dump data, with the goal of producing an effective malware detection system. The selection of these image sizes is based on the consideration that larger images would require more storage space for the dataset. Therefore, using smaller images is considered to save storage space. Additionally, smaller images can speed up the image processing. This research aims to find a balance between efficiency and accuracy in malware detection.

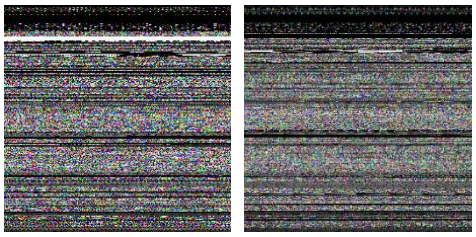

Figure 5. Example of RGB Memory Dump Image of 224x224 (Left) and 300x300 (Right)

C. Feature Extraction

The first step is feature-extraction using Histogram of Oriented Gradients (HOG) descriptor, chosen for its ability to capture relevant image information related to patterns and structures. The feature extraction process starts by resizing the original images to 256x256 pixels. This resizing ensures that HOG can measure local gradient distributions consistently across different image regions, maintaining the necessary detail for feature extraction while keeping the image size manageable to minimize resource usage.

The resized images are divided into small cells, and the gradient orientations for each cell are calculated to capture the direction and strength of colour changes. These calculations build a histogram of gradient orientations, representing the texture patterns in the image. The feature extraction process generates a descriptor vector, typically a long vector representing the gradient distribution within the image. Using HOG descriptor, this feature extraction process captures essential visual characteristics of the RGB memory dump samples, which are then used as input for malware detection analysis.

D. Data Labelling

The next step is labelling for the training and testing of machine learning algorithms and the dimensionality reduction process. This involves labelling the data based on the 11 classes available in the dataset. The data will be labelled into 5 categories: 'adware,' 'trojan,' 'worm,' 'virus,' and 'benign.' The labelling process is performed separately for training and testing data to facilitate both the dimensionality reduction using UMAP and the machine learning model execution. This ensures that each data subset (training and testing) has the appropriate labels, supporting an accurate and objective evaluation of the developed model's performance.

E. Dimension Reduction

Dimensionality reduction is a crucial step in processing complex data. Uniform Manifold Approximation and Projection (UMAP) is used to reduce the dimensionality of the data from feature extraction, employing two types: supervised and unsupervised. Supervised UMAP utilizes the malware type labels to guide the dimensionality reduction process, making it more responsive to data structures relevant to classification. In contrast, unsupervised UMAP works without using labels, allowing it to explore the natural structure of the data more generally.

UMAP is chosen for its ability to maintain relatively complex data structures in a lower-dimensional space, especially given the feature extraction results that have many features, exceeding 30,000, since the processed images are RGB images. By applying UMAP, it is expected to achieve a simpler yet still informative data representation, thereby improving the efficiency and performance of the machine learning algorithms used in malware detection analysis. Additionally, UMAP has demonstrated significant capability in handling large datasets.

F. Classification with Machine Learning

Several machine learning algorithms are used to classify the data samples. The algorithms employed include Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel, Multinomial Logistic Regression, and K-Nearest Neighbor (KNN). The selection of algorithms is based on the analysis needs and the characteristics of the data used. SVM with RBF kernel is chosen for its ability to handle data with nonlinear structures and high complexity. Multinomial Logistic Regression is used because it can provide good interpretation of the feature weights generated. KNN is selected for its simplicity yet effectiveness in classifying data based on proximity to the nearest neighbours.

G. Machine Learning Model Evaluation

In the final stage, the three machine learning models will be compared to determine which algorithm has the highest effectiveness and accuracy using five evaluation metrics: F1-Score, recall, precision, accuracy, and confusion matrix. These evaluation metrics provide detailed information on the capability of machine learning algorithms to recognize and distinguish between malware and benign classes with high accuracy.

IV. Results and Discussion

A. Feature Extraction

Feature extraction is performed using HOG descriptor to capture representative information from images. HOG descriptor is implemented using the hog library from ‘skimage.feature’. Images are iterated through 11 folders corresponding to their classes in both the training and testing datasets, split in an 80:20 ratio. The feature extraction function does three main functions: first, reading images from their paths; second, resizing them to 256x256 pixels; and third, extracting features using the hog() function with parameters documented in [20], including resized image size, 9 gradient orientation bins, 8x8 pixel cells for gradient computation, 2x2 cells per block for gradient normalization, visualization status, and channel axis determination (-1 for RGB images).

The extracted feature results can be visualized as depicted in Figure 4.2. Images originally sized 300x300 and 224x224 pixels, resized to 256x256 pixels, yield 34,596 feature vectors. Larger image sizes, like 512x512 pixels, generate 142,884 feature vectors. Feature vector count varies with image size and parameters like orientation, block, and cell counts, but not with image complexity, which influences feature vector representation values, indicating gradient or orientation changes.

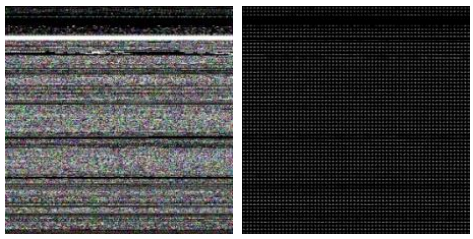

Figure 6. Example of Adposhel RGB Memory Dump Image (Left) and Feature Extraction Results with HOG Descriptor (Right)

Feature vectors differ between initial image sizes (300x300 vs. 224x224 pixels) before resizing to 256x256 pixels due to spatial information and resolution affecting detail level captured. Such differences arise from varying initial image information, impacting HOG features despite uniform final resizing.

The high-dimensional feature vectors extracted pose challenges for direct machine learning model processing, potentially triggering the "Curse of Dimensionality." This phenomenon includes issues like overfitting (overly complex models performing poorly on unseen data), performance degradation (increased features not always improving model accuracy), and increased computational resource demands (requiring more time and memory). Hence, dimensionality reduction techniques such as UMAP are crucial to mitigate these challenges.

B. Data Labelling

The results of the feature extraction process are labelled based on the type of malware across 11 data classes. Classes Adposhel, Amonetize, BrowseFox, InstallCore.C, and MultiPlug are labelled as 'adware'; classes Allaple.A and AutoRun-PU are labelled as 'worm'; classes Dinwod!rfn and Vilsel are labelled as 'trojan'; class VBA is labelled as 'virus'; and the class Benign is labelled as 'benign'. The labelling results in arrays containing feature vectors as data features and labels as targets, totalling 5 categories: adware, worm, trojan, virus, and benign, with ratios of 5:2:2:1:1 respectively.

The data labelling process is also separately applied to the training and testing data. To facilitate dimensionality reduction and machine learning training, feature vectors and labels are separated into different arrays. Labels are encoded into numeric values using the LabelEncoder() function with the following mappings: adware to 0, worm to 1, trojan to 2, virus to 3, and benign to 4. This encoding ensures effective data processing, especially for machine learning algorithms that require numeric labels. The outcome is separate arrays for training and testing data, each containing feature data and label data.

C. Dimension Reduction

Dimensionality reduction is performed using the Uniform Manifold Approximation and Projection (UMAP) method to decrease the number of features extracted. The primary goal is to enhance the effectiveness of malware detection using machine learning models. For processed training and testing data, UMAP is implemented using the UMAP library with both supervised and unsupervised processes. Training data is processed with the fit_transform() function, while testing data is processed with the transform() function. Parameters are based on research by Bozkir et al[7], as follows:

- n_components = 4: Reduces to 4 dimensions.
- n_neighbors = 55: Uses 55 nearest neighbors.
- min_dist = 1: Minimum distance between points.
- metric = 'manhattan': Distance metric using Manhattan distance.
- random_state = 42: Ensures consistent UMAP results.

In the first process, supervised UMAP is applied to both training and testing data, using labels as targets. UMAP successfully reduces the feature dimensions from 34,597 to 4. The resulting data in Figure 7 shows that data with the same labels tend to cluster together, indicating effective grouping based on labels.

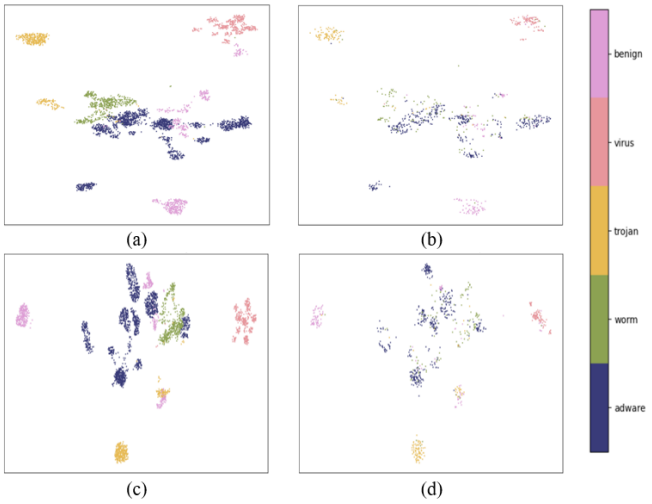

Figure 7. Visualization of Data Distribution Using UMAP Supervised (a) Training Data 224x224 (b) Testing Data 224x224 (c) Training Data 300x300 (d) Testing Data 300x300

In the second process, unsupervised UMAP is applied without using labels, relying solely on the training and testing data. The visualized data distribution in Figure 8 shows distinct clusters, but some labels are mixed. This occurs because UMAP maps data to a low-dimensional space based on the intrinsic geometric structure without label information, resulting in less clear clustering compared to supervised UMAP.

Comparing the results, supervised UMAP demonstrates the advantage of using labels in dimensionality reduction, producing more structured and classification-consistent data representations. In contrast, unsupervised UMAP, while revealing the data's intrinsic structure, results in less distinct groupings and less label alignment.

D. Machine Learning Model

The modelling was performed using three machine learning algorithms: Multinomial Logistic Regression, KNN,

and SVM (RBF). Before modelling, data standardization was carried out due to the dimensionality-reduced data having a value range between -10 and 20. Standardization is crucial for several reasons. Firstly, it aids in data interpretation by clarifying relationships between features. Secondly, it can enhance model performance, particularly for algorithms sensitive to feature scale such as KNN and SVM. Lastly, standardization prevents bias where features with larger value ranges could dominate others during the machine learning process.

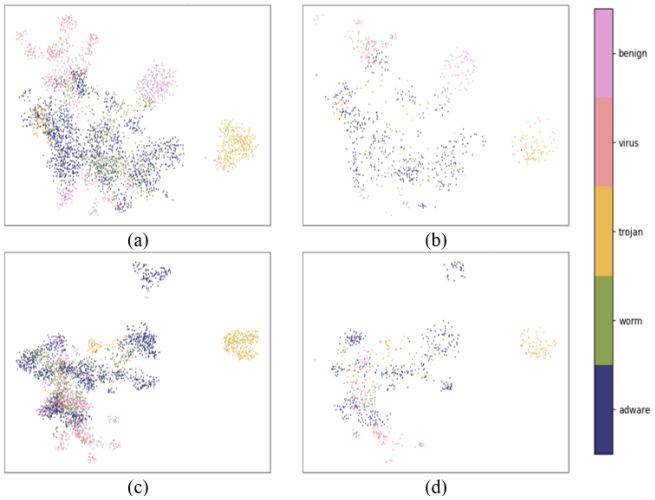

Figure 8. Visualization of Data Distribution Using UMAP Unsupervised (a) Training Data 224x224 (b) Testing Data 224x224 (c) Training Data 300x300 (d) Testing Data 300x300

Multinomial logistic regression was implemented using the LogisticRegression library from sklearn with the following parameters:

- 'multi_class='multinomial'' for handling multi-class classification.
- 'solver='lbfgs'' for estimating model coefficients using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm.
- 'max_iter=1000' to set a maximum of 1000 iterations for the solver.

The model successfully classified the data into 5 predefined labels. Evaluation metrics in Table II indicate that the model achieved the highest performance with 224x224 pixel images, dimensionally reduced using supervised UMAP, yielding an F1-Score of 0.865, recall of 0.877, precision of 0.877, and accuracy of 87.7%.

TABLE II. MULTINOMINAL LOGISTIC REGRESSION MODEL EVALUATION METRIC RESULT

Ukuran Gambar	Ukuran Awal Gambar (piksel)	F1-Score	Recall	Precision	Accuracy
UMAP Supervised	300x300	0.870	0.880	0.877	88%
	224x224	0.865	0.877	0.877	87.7%
UMAP Unsupervised	300x300	0.836	0.851	0.848	85.1%
	224x224	0.824	0.840	0.830	84%

The model's high accuracy can be attributed to the optimal parameter selection and the capability of multinomial logistic regression to handle multi-class classification. The 'lbfgs' solver efficiently estimates coefficients, particularly for large datasets with diverse features. The maximum iteration limit ensures the model has ample opportunity to converge to an optimal solution.

This algorithm is effective for classification problems where a linear relationship between features and the log-odds of the predicted response can be established. It is also robust against multicollinearity, where two or more features are correlated, and can provide well-calibrated probabilities for each class. The high evaluation metrics indicate the model's consistent and accurate performance in distinguishing the five predefined classes in the dataset.

KNN was implemented using the KNeighborClassifier library from sklearn. The model used the parameter n_neighbors = 3, specifying the number of nearest neighbors for predictions. It successfully classified the data into 5 predefined labels. Evaluation metrics in Table III show that the model achieved the highest performance with 300x300 pixel images, dimensionally reduced using unsupervised UMAP, yielding an F1-Score of 0.902, recall of 0.905, precision of 0.902, and accuracy of 90.5%.

TABLE III. KNN MODEL EVALUATION METRIC RESULT

Ukuran Gambar	Ukuran Awal Gambar (piksel)	F1-Score	Recall	Precision	Accuracy
UMAP Supervised	300x300	0.869	0.879	0.876	87.9%
	224x224	0.861	0.875	0.875	87.5%
UMAP Unsupervised	300x300	0.902	0.905	0.902	90.5%
	224x224	0.890	0.893	0.890	89.3%

The parameter n_neighbors = 3 was chosen for its balance between bias and variance, allowing the model to capture data patterns well without being too sensitive to noise. The success of the KNN model is not only due to this parameter but also the algorithm's simple yet effective nature in classifying data based on nearest neighbours. KNN performs well with non-linear datasets and adapts to the data distribution. The high evaluation metrics indicate KNN effectively handles feature differences between classes, producing accurate and consistent predictions.

SVM was implemented using the SVC library from sklearn. The model used the parameter 'kernel = 'rbf'' to run the RBF kernel with a radial basis function and 'gamma = 0.2' to control the reach of a single data point in the decision-making process. Evaluation metrics in Table IV show that SVM (RBF) achieved the highest accuracy with 300x300 pixel images, dimensionally reduced using supervised UMAP, with an F1-Score of 0.870, recall of 0.880, precision of 0.877, and accuracy of 88%.

TABLE IV. SVM (RBF) MODEL EVALUATION METRIC RESULT

Ukuran Gambar	Ukuran Awal Gambar (piksel)	F1-Score	Recall	Precision	Accuracy
UMAP Supervised	300x300	0.864	0.873	0.870	87.3%
	224x224	0.865	0.877	0.877	87.7%
UMAP Unsupervised	300x300	0.768	0.798	0.782	79.8%
	224x224	0.713	0.769	0.810	76.9%

The success of the SVM model with the RBF kernel is attributed to its ability to handle non-linear data from image feature extraction. The RBF kernel projects data into higher

dimensions, making it easier to separate with a hyperplane. The 'gamma' parameter is crucial in determining the influence of each data point, with higher values making the model more complex and sensitive to training data. The high evaluation metrics indicate that SVM (RBF) effectively captures important data characteristics, producing accurate predictions. SVM is also known for its robustness against overfitting, particularly with diverse feature sets and potential outliers in the data.

Overall, the three machine learning models achieved high accuracy, ranging from 70% to 90%. Additionally, the entire modelling process took less than 20 seconds. The best results for each model fell within the 80-90% range. This indicates that the models can classify data effectively, depending on image size and the type of UMAP used. The highest accuracy was achieved by KNN with 224x224 pixel images reduced using unsupervised UMAP, with an accuracy of 90.5% and an F1-Score of 0.902. The second-best performance was by SVM, followed by logistic regression. Figures 9 and 10 show the accuracy and F1-Score comparisons for all models.

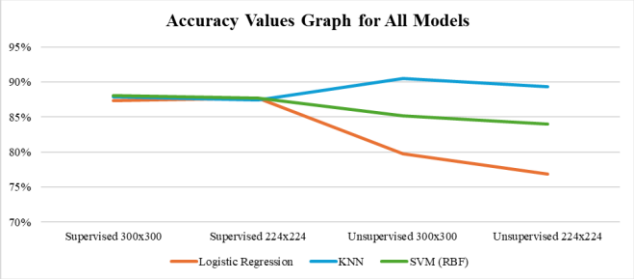

Figure 8. Graph of Accuracy Percentage Results of All Models

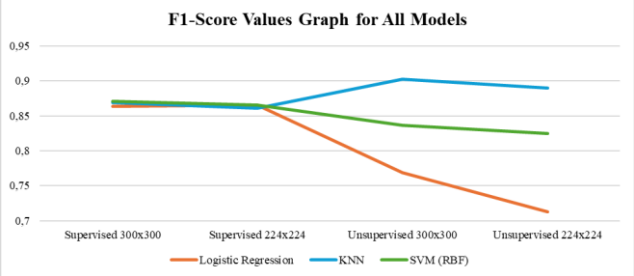

Figure 9. Graph of F1-Score Results of All Models

The results show that KNN performs well when data is dimensionally reduced using unsupervised UMAP. This effectiveness may be due to KNN's classification based on proximity in feature space and sensitivity to local data structure, which benefits from the detailed structure provided by unsupervised UMAP, especially with 300x300 pixel images. However, KNN still performs well with both image sizes using unsupervised UMAP.

Multinomial logistic regression and SVM with RBF kernel perform better when data is reduced using supervised UMAP. Multinomial logistic regression excels with 224x224 pixel images, while SVM performs best with 300x300 pixel images. Both algorithms rely on optimal class separation and margins, which benefit from the structure produced by supervised UMAP. While 224x224 images might lose some detail, this can simplify the data for logistic regression. Conversely, 300x300 images retain more detail, benefiting SVM with RBF kernel.

It's important to note that the metric evaluations might be influenced by data imbalance in the dataset. The dataset has an uneven distribution of labels, with 'adware' having the most data, followed by 'trojan' and 'worm', then 'virus' and 'benign', in a 5:2:2:1:1 ratio. This imbalance can bias classification. The confusion matrix in Figure 10 shows over 50 misclassifications into the 'adware' category, particularly for 'worm', likely due to similar feature characteristics.

Figure 10. Confusion Matrix Results for All Models (a) 224x224 Image with Supervised UMAP (b) 300x300 Image with Supervised UMAP (c) 224x224 Image with Unsupervised UMAP (d) 300x300 Image with Unsupervised UMAP

This imbalance can be addressed with under sampling, oversampling, or other techniques. However, the models still achieve high accuracy despite the imbalance, possibly due to UMAP's ability to preserve local and global data structures, aiding pattern recognition. Additionally, standardization before modelling helps by placing data on the same scale. Further research is needed to confirm the impact of these methods.

V. Conclusion

Based on the conducted research, the following conclusions were drawn: The performance of machine learning models is influenced by both the type of UMAP and the image size used. For instance, KNN achieved 90.5% accuracy and SVM (RBF) achieved 88% accuracy using 300x300 pixel images with different UMAP types. Multinomial Logistic Regression performed best with 87.7% accuracy using supervised UMAP and 224x224 pixel images, demonstrating effectiveness with smaller images and supervised UMAP. KNN excelled with 90.5% accuracy using unsupervised UMAP and 300x300 pixel images, particularly

effective with larger images and unsupervised UMAP. SVM (RBF) achieved 88% accuracy using supervised UMAP and 300x300 pixel images, proving effective with larger images and supervised UMAP. Among these models, KNN achieved the highest performance with an F1-Score of 0.902, recall of 0.905, precision of 0.902, and accuracy of 90.5%.

The research conducted has its shortcomings. Therefore, several recommendations could potentially improve future efforts in developing a malware detection system. Firstly, expanding the dataset with a greater variety of samples would be beneficial. Secondly, achieving better balance in dataset ratios could significantly enhance accuracy. Thirdly, exploring alternative algorithms such as SIFT, t-SNE, and deep learning for feature extraction, dimensionality reduction, and classification modelling may yield better results. Lastly, evaluating models with comprehensive metrics like ROC-AUC and precision-recall curves can provide a deeper understanding of overall model performance.

REFERENCES

[1] S. Kemp, "Digital 2023: Global Overview Report." Accessed: Jan. 27, 2024. [Online]. Available: https://datareportal.com/reports/digital-2023-global-overview-report

[2] S. Kemp, "Digital 2022: Global Overview Report." Accessed: Jan. 27, 2024. [Online]. Available: https://datareportal.com/reports/digital-2022-global-overview-report

[3] "239 Cybersecurity Statistics (2023)." Accessed: Jan. 27, 2024. [Online]. Available: https://www.packetlabs.net/posts/239-cybersecurity-statistics-2023/

[4] I. Kara, "Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges," *Expert Syst Appl*, vol. 214, Mar. 2023, doi: 10.1016/j.eswa.2022.119133.

[5] F. Bahtiar, N. Widiyasono, and A. P. Aldya, "Forensic Volatile Memory For Malware Detection Using Machine Learning Algorithm," *Jurnal Rekayasa Sistem & Industri (JRSI)*, vol. 6, no. 1, Jun. 2019, doi: 10.25124/jrsi.v5i02.311.

[6] S. Mele Pottaraikkal and A. Sujeer Sugatha, "Effectiveness of Multiple Memory-Images in detecting Fileless Malware," in *ISDFS 2023 - 11th International Symposium on Digital Forensics and Security*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ISDFS58141.2023.10131728.

[7] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, "Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision," *Comput Secur*, vol. 103, Apr. 2021, doi: 10.1016/j.cose.2020.102166.

[8] Y. Dai, H. Li, Y. Qian, and X. Lu, "A malware classification method based on memory dump grayscale image," *Digit Investig*, vol. 27, pp. 30–37, Dec. 2018, doi: 10.1016/j.diin.2018.09.006.

[9] T. Panker and N. Nissim, "Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments," *Knowl Based Syst*, vol. 226, Aug. 2021, doi: 10.1016/j.knosys.2021.107095.

[10] A. Wolf, "13 Types of Malware Attacks — and How You Can Defend Against Them." Accessed: Jan. 27, 2024. [Online]. Available: https://arcticwolf.com/resources/blog/8-types-of-malware/

[11] K. Baker, "The 12 Most Common Types of Malware." Accessed: Jan. 27, 2024. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/malware/types-of-malware/

[12] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," 2005. [Online]. Available: http://lear.inrialpes.fr

[13] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.03426

[14] "How UMAP Works." Accessed: May 15, 2024. [Online]. Available: https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

[15] A. Coenen and A. Pearce, "Understanding UMAP." Accessed: May 17, 2024. [Online]. Available: https://pair-code.github.io/understanding-umap/

[16] "Basic UMAP Parameters." Accessed: Apr. 27, 2024. [Online]. Available: https://umap-learn.readthedocs.io/en/latest/parameters.html

[17] "UMAP for Supervised Dimension Reduction and Metric Learning." Accessed: May 15, 2024. [Online]. Available: https://umap-learn.readthedocs.io/en/latest/supervised.html

[18] J. Grus, "Data Science from Scratch SECOND EDITION First Principles with Python," 2019. Accessed: Apr. 22, 2024. [Online]. Available: http://oreilly.com

[19] N. A. Ahmed, "What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained." Accessed: Apr. 25, 2024. [Online]. Available: https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning

[20] scikit-learn, "Histogram of Oriented Gradients." Accessed: Apr. 29, 2024. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html