Fall 2023

Image-Based Classification of Malware using t-SNE Images

Vincent Stowbunenko

Image-Based Classification of Malware using t-SNE Images

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Vincent Stowbunenko

December 2023

The Designated Project Committee Approves the Project Titled

Image-Based Classification of Malware using t-SNE Images

by

Vincent Stowbunenko

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2023

Dr. Fabio Di Troia	Department of Computer Science
Dr. Mark Stamp	Department of Computer Science
Dr. William Andreopoulos	Department of Computer Science

ABSTRACT

Image-Based Classification of Malware using t-SNE Images

by Vincent Stowbunenko

This Master's project proposes a novel technique for classifying malware using image-based methods. The approach involves generating t-SNE images from the EMBER dataset, which contains one million samples of both malware and benign files, each represented by over 2,000 features. The t-SNE technique is well-suited for capturing intricate patterns in complex datasets because it effectively maintains the local structure. These t-SNE images are then used as inputs to train two lightweight image classification models, SqueezeNet and MobileNet. Additionally, to provide a benchmark for comparison, a non-image classification model using LightGBM is also explored.

As part of the investigation, the project compares two different normalization techniques, norm-1 and norm-2, applied to the feature vectors before converting them into t-SNE images. This comparison allows for a thorough understanding of how normalization affects the results.

Acknowledging the issue presented by excessive CPU memory usage throughout the training phase, the project embraces a practical stance. The training dataset gets partitioned into three distinct batches, facilitating consecutive training sessions on each individual batch. This tactic adeptly tackles memory limitations, thereby guaranteeing the attainability of model training.

The outcomes demonstrate remarkable accuracy scores of 0.914 for SqueezeNet and 0.944 for MobileNet. While these results showcase the promise of image-oriented methodologies in enhancing the identification and categorization of malicious software, it is important to note that the incumbent benchmark method,

LightGBM, still maintains a superior performance with an AUC value of 0.996. Given the absence of significant computational advantages with image-based methods, the recommendation at this time is to continue utilizing LightGBM as the preferred method for malware detection. However, this study provides valuable insights into the potential of image-based approaches and sets the stage for further exploration and refinement in future research.

Index Terms: Malware classification, Image-based methods, t-SNE images, EMBER dataset, Feature vectors, SqueezeNet, MobileNet LightGBM, Normalization techniques, norm-1, norm-2, Memory overload, Training process, Malware detection, Comparative analysis.

ACKNOWLEDGMENTS

My heartfelt gratitude to all who supported my Master's project: Prof. Fabio Di Troia, my advisor, for guidance and dedication; Prof. Mark Stamp and Prof. William Andreopoulos for invaluable insights; the faculty and staff at San Jose State University; my family and friends for unwavering support; and anonymous contributors for enriching my research. Thank you all!

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

Introduction

In recent times, the rise of malicious software has become a critical concern for digital security, necessitating the development of innovative methods for precise and effective malware detection and classification. In response to this challenge, this Master's project explores an effective new approach for malware classification, drawing from the pioneering work of the team behind DeepInsight [1].

DeepInsight introduced a methodology to convert non-image data into t-distributed stochastic neighbor embedding (t-SNE) [2] images for convolutional neural network architectures, thus revolutionizing the application of image-based models for robust classification. The t-SNE method is ideal for unraveling patterns in complex datasets due to its ability to preserve the local structure. Building upon this foundation, this project demonstrates a novel technique for classifying malware, using t-SNE images generated from the well-established EMBER dataset [3].

t-SNE is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map. It was first introduced by van der Maaten and Hinton, has emerged as a powerful visualization technique for high-dimensional data. The method preserves local structure, making it particularly suitable for capturing underlying patterns in complex datasets. Expanding on the concept of t-SNE, the DeepInsight methodology proposed by Sharma et al. [1] transforms non-image data into t-SNE images, leveraging the strengths of CNN architectures for classification tasks. This work serves as a significant source of inspiration for the present project, which utilizes t-SNE images to classify malware.

The EMBER dataset, known as the Elastic Malware Benchmark for Empowering Researchers, serves as the cornerstone of this Master's project. EMBER offers

a comprehensive collection of both malware and benign files, meticulously selected for research and evaluation in the malware detection and classification field. This dataset comprises a vast collection of malware and benign files, each represented by more than 2,000 features.

Following the footsteps of the DeepInsight team, the impact of two normalization techniques, norm-1 and norm-2, applied to the feature vectors before t-SNE image conversion is also investigated. This comparative study allows for a comprehensive understanding of the effects of normalization techniques [1] on the classification results and offers valuable insights into their relevance and significance in the context of malware detection.

Furthermore, to establish a benchmark for performance evaluation, a non-image classification model based on LightGBM [4] is examined in parallel . This comparative analysis further illustrates the strengths and limitations of the proposed image-based technique and helps contextualize its effectiveness.

This project also addresses the challenge of memory overload during the training of complex image classification models. To efficiently manage memory, the training dataset is partitioned into batches, each containing a significant subset of t-SNE images. To expedite data loading and prevent delays, these batches are further divided into sub-batches, utilizing parallel loading across multiple CPU cores. This approach not only facilitates model training but also optimizes memory usage, ensuring effective training even with memory constraints.

The primary focus of this comprehensive report is to present a detailed methodology, thorough experimental results, and an extensive comparative analysis of the performance of various image-based classification models for malware detection. The findings illustrate the efficacy and potential of the proposed approach, i.e., leveraging t-SNE images and convolutional neural networks, to bolster

malware detection and classification efforts.

Building upon the innovative work of the DeepInsight team and integrating their discoveries on normalization techniques, this Master's project offers a promising solution to the ever-evolving landscape of malware threats. The fusion of image-based techniques with sophisticated classification models combined with an meticulous resource management and optimization techniques for handling memory overload results in a method that opens up new avenues for enhanced and efficient malware detection.

1.1 Background and Motivation

The ever-increasing advancement of technology and the widespread use of the internet have resulted in a sharp rise in cyber threats, especially in the form of malware. Malicious software poses a serious risk to individuals, organizations, and governments, causing financial losses, data breaches, and disruptions to critical services. As malware becomes more sophisticated and elusive, there is an urgent need for innovative and effective techniques to detect and categorize these threats.

Traditionally, malware detection has relied on feature-based approaches, where numerical feature vectors are extracted from various attributes and behaviors of files. While these methods have achieved some success, they often struggle to cope with the growing volume and diversity of malware samples. Additionally, feature-based techniques may not fully capture the intricate patterns and relationships within the data, limiting their ability to distinguish subtle yet crucial differences between malware and benign files.

The emergence of image-based methods, inspired by the success of convolutional neural networks (CNNs) in computer vision tasks, has shown promising results across various domains. By converting non-image data into visual represen-

tations, like t-SNE images, and feeding them into CNN architectures, researchers have demonstrated significant improvements in accuracy and robustness for certain classification tasks.

This Master's project is motivated by the desire to apply image-based methods to the realm of malware detection and classification. By generating t-SNE images from the feature-rich EMBER dataset, which encompasses a diverse collection of malware and benign files, the project aims to explore the potential of visual patterns in discerning between malicious and non-malicious files.

Moreover, the comparison of two normalization techniques, norm-1 and norm-2, inspired by the work of the team behind DeepInsight, adds a crucial dimension to the investigation. Understanding how normalization impacts the classification results can provide insights into optimizing the image generation process and enhancing the overall performance of the classification models.

The ultimate objective of this study is to contribute to the advancement of cybersecurity by developing a novel image-based approach that complements and potentially surpasses traditional feature-based methods. By harnessing the power of convolutional neural networks and visual representations of malware samples, this project seeks to strengthen malware detection efforts and improve the resilience of digital systems against cyber threats.

Through this exploration of image-based malware classification, the project aspires to pave the way for a more sophisticated and efficient defense mechanism, ensuring a safer digital environment for individuals and organizations alike. By addressing the limitations of existing techniques and building upon the work of the DeepInsight team, this research holds the promise of making significant contributions to the field of cybersecurity and furthering our understanding of malware detection in the age of advanced cyber threats.

1.2 Objectives

The primary goal of this Master's project is to develop and evaluate a new image-based method for classifying malware using t-SNE representations. This approach aims to improve the accuracy and efficiency of malware detection. To achieve this objective, the project is guided by the following specific goals:

1. **Generate t-SNE Images**: The first step involves creating t-SNE images from the EMBER dataset, which contains one million samples of malware and benign files represented by over 2,000 features. These t-SNE images will visually represent the high-dimensional feature spaces, capturing patterns and relationships within the data.

2. **Train Image Classification Models**: In this step, two lightweight image recognition models, SqueezeNet and MobileNet, will be employed. These models will categorize the t-SNE images into two distinct groups: malware and benign. The training approach will heavily emphasize the tuning of hyperparameters to achieve the highest possible accuracy levels.

3. **Evaluate Normalization Techniques**: The project will investigate and compare the impact of two normalization techniques, norm-1 and norm-2, on the performance of the image-based classification models. By applying both normalization approaches to the feature vectors before generating t-SNE images, the research aims to identify the most effective normalization strategy.

4. **Benchmark Non-Image Classification Model**: Creating a point of reference for comparison involves utilizing a pre-trained non-image classification model with LightGBM. Evaluating this model will occur using the original

feature vectors. It's important to note that while the significance of this step is acknowledged, the project's true focus revolves around training and fine-tuning image-based classification models. Assessing the success of the proposed methodology will heavily rely on the subsequent evaluation of these image-based models.

5. **Handle Memory Limitations**: One of the key challenges addressed by the project is the problem of memory overload that often arises during the training process of image-based classification models. To effectively tackle this issue, the training dataset will be partitioned into three distinct batches. By adopting this strategy, the training can be carried out sequentially on each batch, thereby optimizing the utilization of CPU memory. This ensures the smooth and successful training of complex image-based models without the hindrance of memory limitations.

6. **In-depth Comparative Evaluation**: Conducting a comprehensive comparative evaluation will measure the effectiveness of the selected image-based models in contrast to a non-image model for malware detection. The evaluation will cover a variety of metrics, such as accuracy, precision, recall, F1-score, and other pertinent indicators. The analysis aims to reveal the relative strengths and limitations of the chosen methodologies. This includes the two image-based models and the non-image model utilized in this project.

7. **Highlight Efficacy and Potential**: The project will emphasize the efficacy and potential of the proposed image-based approach in enhancing malware detection efforts. Insights into the advantages of leveraging visual representations and convolutional neural networks for malware classification will be discussed, along with implications for cybersecurity and digital defense.

8. **Contribution to Knowledge**: By presenting new findings and insights on image-based malware classification, this project seeks to contribute to the existing body of knowledge in the field of cybersecurity. It aims to bridge the gap between traditional feature-based methods and cutting-edge image-based techniques, paving the way for future research in malware detection.

In general, this Master's project aims to establish the viability and effectiveness of the proposed image-based approach for malware classification. The outcomes of the project have the potential to advance the state-of-the-art in cybersecurity, enhancing the ability to detect and mitigate the ever-evolving threats posed by malware in the digital landscape.

1.3 Scope of the Project

The main focus of this Master's project is to develop and assess an image-based approach for the binary classification of files into two classes: benign and malware. The project aims to utilize the EMBER dataset, which contains diverse samples of malware and benign files, to distinguish between these two categories.

While the project is specifically geared towards binary classification, it does not delve into identifying individual malware families or analyzing fine-grained variations within each class. Instead, its primary objective is to provide a comprehensive analysis of the proposed image-based approach's effectiveness in discerning malicious files from non-malicious ones.

To achieve its goals, the project will undertake the following steps:

1. Generate t-SNE images from the EMBER dataset to represent the files' high-dimensional feature spaces.

2. Train SqueezeNet and MobileNet image classification models to categorize t-SNE images into malware and benign classes.

3. Investigate and compare the impact of two normalization techniques, norm-1 and norm-2, on the classification performance of the image-based models.

4. Evaluate the image-based classification models against a non-image classification model using LightGBM as a reference for performance assessment.

5. Implement a memory management strategy by splitting the training dataset into three batches and using one batch at a time to prevent memory overload.

6. Conduct a comprehensive comparative analysis of the image-based and non-image models using standard evaluation metrics, including accuracy, precision, recall, and F1-score.

7. Discuss the potential implications of the image-based approach for bolstering malware detection efforts, with a specific focus on its relevance to cybersecurity applications.

The project will be conducted within a specified timeframe and adhere to hardware and software constraints to ensure feasibility. Although the scope is confined to binary classification, the insights gained from this research may lay the groundwork for future investigations into more advanced classification tasks, such as malware family identification and other sub-classifications within the broader malware category.

By accomplishing this scope, the project aims to contribute significantly to the field of cybersecurity by advancing image-based methods for malware detection and classification and providing valuable insights into the efficacy of such approaches in real-world scenarios.

CHAPTER 2

Literature Review

In recent times, the proliferation of malware and its ever-changing nature have sparked extensive research efforts in the field of malware detection and classification. This section presents a comprehensive literature review, delving into key studies and advancements in several critical areas that lay the groundwork for this Master's project.

2.1 Malware Detection and Classification

Numerous studies have explored various techniques for detecting and classifying malware. Traditional approaches rely on numerical feature vectors to represent files, capturing their attributes and behaviors. Notable works by Rieck et al. [5], Kolter et al. [6], and Gibert et al. [7] demonstrate the effectiveness of machine learning algorithms in classifying malware based on these extracted features. However, with the rapid increase in malware variants and the limitations of feature-based methods in handling high-dimensional data, researchers have sought alternative approaches.

2.1.1 Non-Machine Learning Approaches

In the early days of malware analysis, non-ML approaches dominated the field. These methods primarily relied on signature-based detection [8], where known malware samples were matched against predefined signatures or patterns. While signature-based methods were effective in identifying known malware strains, they struggled with zero-day attacks and new, previously unseen malware variants. Additionally, heuristic and rule-based methods were employed to identify suspicious behaviors, such as code injection or system file modification.

Another non-ML approach involved the use of sandboxing and dynamic analysis, with one prominent example being the Cuckoo sandboxing system [9]. Sand-

boxing isolated suspicious files or processes in controlled environments to observe their behavior. This approach helped in understanding the actions of malware but often required significant computational resources and lacked the ability to handle polymorphic or metamorphic malware effectively.

2.1.2 Transition to Machine Learning

The limitations of non-ML approaches, particularly their struggle with new and evolving malware, prompted a transition to machine learning-based methods. Machine learning techniques offered the advantage of adaptability and scalability, making them well-suited for the dynamic nature of malware. Feature-based methods, like those employed by Rieck et al. [5], relied on engineered features extracted from malware samples. These features encompassed file attributes, behavior analysis, and code analysis, among others.

Machine learning models, such as decision trees, support vector machines, and random forests, were trained on these feature vectors to distinguish between benign and malicious files. While these approaches demonstrated considerable success, they also encountered challenges related to feature engineering and the dimensionality of feature spaces.

The shift towards deep learning and neural networks introduced more automated feature extraction and representation learning. Models like convolutional neural networks (CNNs) [10] and recurrent neural networks (RNNs) [11] gained prominence in extracting meaningful features from raw data. This advancement significantly improved the ability to classify malware variants based on complex patterns, leading to the development of image-based and sequence-based classifiers.

The transition to machine learning approaches in malware detection has proven effective in addressing the dynamic nature of malware. These methods

offer the flexibility to adapt to new threats and learn from evolving datasets, making them a valuable asset in the ongoing battle against malware.

2.2 Static vs Dynamic Features in Malware Analysis

Malware analysis research often employs two distinct types of features: static and dynamic. Static features focus on extracting information directly from the binary file without executing it, while dynamic features involve running the binary in a controlled environment to monitor its behavior. Both approaches have their advantages and limitations.

Static analysis is computationally efficient and less resource-intensive as it doesn't require the execution of potentially harmful code. Features such as API calls, file imports, and opcode sequences can be extracted from the binary file's structure. This approach offers quick results and is suitable for large-scale analysis.

In contrast, dynamic analysis provides insights into the actual behavior of malware during execution. It can reveal hidden or obfuscated functionalities that static analysis might miss. However, dynamic analysis demands significant computational resources, is time-consuming, and may not capture all malware behaviors if they are triggered conditionally.

For this project, static features are utilized due to their computational efficiency and scalability. The goal is to process a large dataset of Windows Portable Executable files efficiently. Static features, including byte sequences and opcode patterns, are extracted directly from the files' structures, making them suitable for the task at hand.

Several researchers, such as Ijaz et al. [12] and Shalaginov et al. [13], have demonstrated the effectiveness of static features in malware classification, especially when combined with machine learning algorithms. The focus on static fea-

tures aligns with the project's objectives of efficiently classifying a substantial dataset of malware samples while ensuring scalability and feasibility.

2.3 Open Benchmark Datasets for Malware Detection

The development and evaluation of machine learning models for malware detection greatly benefit from open benchmark datasets. These datasets provide standardized, well-labeled collections of malware and benign files that facilitate robust model training and testing. In the context of statically detecting malicious Windows portable executable (PE) files, several open benchmark datasets have emerged as valuable resources for researchers and practitioners alike.

One such dataset is the Malware Information Sharing Platform (MISP) [14]. MISP offers a diverse set of PE files, categorized into multiple families, making it suitable for both binary classification tasks and more granular malware family classification. This dataset's comprehensiveness aids in training models to detect a wide range of malware variants and their unique characteristics.

The EMBER dataset [3] is another noteworthy resource. EMBER stands out for its substantial size, containing a vast collection of PE files and an extensive set of handcrafted features. These features serve as a valuable foundation for feature engineering experiments, enabling researchers to explore various attributes that contribute to effective malware detection.

Furthermore, the Microsoft Malware Classification Challenge (BIG 2015) dataset [15] presents a well-defined challenge in classifying PE files into specific malware families. This dataset offers a curated set of files and associated metadata, making it a suitable choice for evaluating model performance in real-world scenarios.

The presence of these benchmark datasets ensures a standardized and con-

sistent evaluation of machine learning models for malware detection. Researchers can leverage these datasets to train and assess the effectiveness of their models, fostering innovation and advancing the field of malware detection.

This section introduces the significance of open benchmark datasets in malware detection and establishes the context for utilizing the EMBER dataset in this project. It highlights the importance of having access to diverse and well-structured datasets to train models capable of detecting a wide range of malicious PE files effectively.

2.4 Feature Engineering in Malware Detection

Effective feature engineering is a critical aspect of malware detection and classification. Feature engineering involves selecting or creating the most informative attributes or characteristics from raw data that will aid in the accurate classification of malware samples. It plays a pivotal role in traditional machine learning-based malware detection methods.

Feature engineering techniques have evolved over the years, with researchers exploring a wide range of features, including static and dynamic attributes of files and their behaviors. Static features encompass file size, file type, entropy, API call sequences, and more. Dynamic features involve monitoring the execution of files and observing their runtime behaviors, such as system calls and memory accesses.

Notable works by Zhang et al. [16], Carrier et al. [17], and Li et al. [18] have delved into the creation of effective feature sets for malware classification. These studies demonstrate the importance of feature engineering in achieving high accuracy in malware detection.

Feature engineering is a fundamental step in traditional malware detection approaches, but it also poses challenges. The feature space can be high-dimensional

and noisy, making it crucial to select relevant features and apply dimensionality reduction techniques when necessary. Moreover, feature engineering may struggle to adapt to the ever-evolving landscape of malware, as new variants constantly emerge.

This section sets the stage for understanding the transition from traditional feature-based malware detection to image-based classification methods. While feature engineering remains relevant, this project explores the potential of leveraging t-SNE images, which capture essential feature representations in a more adaptable and robust manner, ultimately contributing to the evolution of malware detection techniques.

2.5 Dimensionality Reduction in Malware Analysis

The use of dimensionality reduction techniques, such as t-distributed stochastic neighbor embedding (t-SNE), principal component analysis (PCA), and uniform manifold approximation and project (UMAP), has gained traction in malware analysis research. These techniques are employed to extract meaningful features from malware files and enable effective classification.

In the paper "Reliable Malware Analysis and Detection using Topology Data Analysis" by Tidjon et al. [19], t-SNE, PCA, and UMAP are applied to extract features from malware files for reliable analysis and detection. Their work demonstrates the efficacy of dimensionality reduction in enhancing the accuracy of malware classification.

However, this Master's project explores the novel approach of converting features from malware files into t-SNE images first employed by the DeepInsight team [1] and using them to train image classification models. This expansion of the usage of t-SNE, from feature extraction to image classification, represents a

unique contribution to the field of malware analysis, offering the opportunity to explore new perspectives on the potential of dimensionality reduction techniques in image-based classification.

2.6 Normalization Techniques

In their study, Sharma et al. [1] conducted an extensive investigation into the effects of various normalization techniques applied to t-SNE images within the realm of image classification tasks. The primary focus of their research was to evaluate how these normalization methods influenced the validation error of image classifiers. Their findings highlighted the crucial role of normalization in preprocessing image data for machine learning models, offering valuable insights.

Inspired by Sharma et al.'s findings, this project integrates and extends their work to investigate how the performance of image-based malware classifiers is influenced by two particular normalization techniques: norm-1 and norm-2. By implementing these techniques on t-SNE images derived from the EMBER dataset, the goal is to develop a more profound comprehension of the influence of normalization on the accuracy and robustness of image-based malware detection models.

This comparative analysis of normalization techniques adds significant knowledge to the field of image-based classification, particularly in the context of malware detection. The insights gained from this exploration will play a crucial role in guiding the methodology and decision-making process throughout the project.

2.7 Benchmark Models in Machine Learning

Benchmark models play a pivotal role in the evaluation and comparison of novel machine learning approaches. They serve as established reference points, allowing researchers to gauge the performance of their models against well-established baselines. In the realm of machine learning, benchmark models are

typically recognized for their strong performance on various datasets and tasks.

For the specific task of classifying Windows portable executable (PE) files as malicious or benign, benchmark models offer valuable insights into the state of the art in the field of static malware detection. These models are often designed to handle non-image data efficiently, making them suitable for comparison with image-based classification models.

Several benchmark models have been proposed and utilized in research on static malware detection, including Random Forests [20], Gradient Boosting [21], and Support Vector Machines (SVMs) [22]. Each of these models has unique strengths and applications.

Random Forests and Gradient Boosting are ensemble methods that combine the predictions of multiple decision trees, offering robust classification performance. SVMs, on the other hand, are known for their effectiveness in high-dimensional spaces and their ability to handle complex decision boundaries.

However, in this project, Light Gradient Boosting Machine (LightGBM), developed by Ke et al. [4], is selected as the reference model for several compelling reasons. LightGBM is a high-performance gradient boosting framework that excels in non-image data classification tasks. Its efficiency, scalability, and effectiveness in handling large datasets have made it a popular choice in various machine learning applications.

One primary advantage of LightGBM is its exceptional speed and memory efficiency. It utilizes a histogram-based approach to construct decision trees, resulting in faster training times compared to other gradient boosting implementations. This efficiency is particularly valuable when working with large-scale datasets, such as those encountered in malware classification.

The choice of LightGBM as the reference model allows for a fair and com-

prehensive comparison between image-based classification models and a well-established, efficient, and effective non-image model. By including LightGBM in the comparative analysis, researchers can assess whether image-based methods offer advantages in terms of accuracy, precision, recall, and F1-score compared to a state-of-the-art non-image model.

This approach enables the identification of scenarios where image-based methods may provide significant improvements in malware detection accuracy while acknowledging the capabilities of traditional non-image models.

In summary, benchmark models are indispensable tools in machine learning research, enabling robust assessments of novel techniques and fostering innovation within the field. LightGBM, with its speed, efficiency, and effectiveness, serves as an ideal reference model for this project's comparative analysis.

2.8 Notable Models for Image Classification

In the realm of CNN architectures for computer vision, this literature review highlights the utilization of four widely recognized image classification models in the current project. These models, known for their efficiency and performance, are well-suited for the task of malware classification using t-SNE images, emphasizing adaptability and fine-tuning for this specific challenge.

The next four subsections delve into the details of these models: SqueezeNet, MobileNet, ResNet, and Inception. Each of these models brings its unique strengths and architectural innovations to the field of image classification, making them valuable candidates for this Master's project.

2.8.1 SqueezeNet

SqueezeNet, presented by Iandola et al. in their paper "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size" [23], stands out

as an efficient convolutional neural network (CNN) architecture explicitly crafted to achieve high accuracy in image classification while significantly reducing the number of parameters. This innovation results in an up to 50 times reduction in parameters compared to the bulky AlexNet, all while maintaining competitive performance.

At the heart of SqueezeNet lies the innovative "fire module," which cleverly combines 1x1 and 3x3 convolutions, effectively reducing the model's size and overall computational cost. This unique architectural element contributes to SqueezeNet's exceptional efficiency and its ability to operate smoothly even on resource-constrained devices.

SqueezeNet exhibits proficiency in image classification and its impact resonates across various domains. It inspires the development of efficient neural network architectures, thereby revolutionizing the approach to deep learning in terms of both compactness and accuracy.

2.8.2 MobileNet

MobileNet, a groundbreaking innovation by Howard et al. [24], emerges as a standout in the realm of image classification models, lauded for its streamlined design that makes it an excellent fit for scenarios with limited resources. The core idea behind this design is the use of depth-wise separable convolutions, a technique that significantly reduces the number of parameters and computational workload, all while maintaining a competitive level of accuracy. The essence of MobileNet's efficiency lies in its ability to find a harmonious balance between the model's size and its performance, positioning it as an attractive choice for real-time applications and devices with constrained processing power.

2.8.3 ResNet

The ResNet architecture introduced by He et al. [25] revolutionized the field of deep learning with its deep residual networks. These networks, with their shortcut connections, addressed the vanishing/exploding gradient problem and enabled training of exceptionally deep neural networks. ResNet has been a cornerstone in various computer vision tasks, including image classification and object detection.

2.8.4 Inception

The Inception architecture, often referred to as GoogLeNet [26], developed by Szegedy et al., is known for its novel inception modules. These modules employ multiple filter sizes and concatenate their outputs to capture multi-scale features efficiently. Inception networks have achieved top-tier performance in various image classification competitions and remain influential in the design of modern CNNs.

2.8.5 Selection of SqueezeNet and MobileNet

SqueezeNet and MobileNet were chosen for this project due to their efficiency, adaptability, and fine-tuning capabilities. Their architecture allows for resource-efficient processing, making them ideal for the task of malware classification using t-SNE images. Additionally, their prevalence in the research community provides readily available resources and pre-trained models, streamlining the development process.

By thoroughly examining and utilizing the existing literature in these key areas, this Master's project has established a strong theoretical foundation for the methodology and experimental analysis. The insights garnered from this literature review were the essential foundational elements leading to the contributions and advancements made by this Master's project.

CHAPTER 3

Methodology

This methodology chapter functions as a roadmap for the processes involved in designing, implementing, and evaluating the proposed approach for classifying malware based on images. Each section explores distinct facets of the methodology, offering the essential background and understanding to grasp the experimental setup and the results that subsequent chapters will present.

The chapter first presents an overview of the dataset utilized for experimentation. Subsequently, it outlines the preprocessing steps that were utilized to ready the data for the training of the classification models. The discussion then moves on to cover the selection and configuration of the image classification models—specifically, SqueezeNet and MobileNet. The emphasis lies in the adaptation and refinement of these models for the binary classification mission involving malware detection.

In preprocessing, a comparative study of two normalization techniques, norm-1 and norm-2, is carried out on the feature vectors before generating t-SNE images. The rationale behind each normalization approach is explored, and the impact on the classification results is analyzed.

Addressing the challenge of memory overload during the training phase of image-based models involved implementing a computational resource management strategy. The strategy included splitting the training dataset into smaller batches. Training the models on one batch at a time effectively managed the limitations of CPU memory. This approach enabled the maintenance of the same level of accuracy and performance, while also reducing the burden on the CPU memory.

The evaluation metrics used to assess the performance of all classification models are thoroughly discussed, encompassing accuracy, precision, recall, and

F1-score, ensuring a comprehensive analysis of model effectiveness.

3.1 Dataset Description - EMBER

The EMBER dataset consists of one million samples, with each represented by over 2,000 features extracted through static analysis. These features encompass eight groups of raw attributes, including parsed features, format-agnostic histograms, and string counts. The usage of static analysis ensures a thorough representation of the files' characteristics without the need for dynamic analysis.

To ensure consistent evaluation and benchmarking, EMBER is divided into separate training and testing sets. The training set contains 800,000 samples, while the remaining 200,000 form the testing set. Notably, out of the 800,000 training samples, 200,000 are unlabeled and used solely for t-SNE image generation, allowing for an unsupervised approach to visual representation.

Within the training set, the remaining 600,000 labeled samples are further divided into a training-validation split with a ratio of 9:1. This division facilitates fine-tuning of image classifiers, enabling iterative model improvement while ensuring unbiased evaluations on the validation set.

EMBER encompasses a wide range of malware samples, including trojans, worms, viruses, ransomware, and more. The inclusion of diverse malware families ensures the dataset's challenge and relevance when evaluating malware detection models. Similarly, the benign files represent a diverse array of legitimate applications, covering various software types and categories.

To prevent data leakage and maintain fair evaluations, the dataset authors have taken measures to ensure the integrity of the training and testing sets. This includes balancing the sets with equal numbers of malware and benign files, avoiding bias towards any specific class.

The meticulous curation of this dataset, coupled with its large size, positions EMBER as a widely adopted benchmark in the field of malware research. By employing this dataset for experiments, this Master's project harmonizes its results with the broader research community, fostering direct comparisons and reproducibility of findings.

Leveraging the EMBER dataset as the primary data source allows for a comprehensive evaluation of the proposed image-based malware classification approach. The conversion of feature vectors into t-SNE images captures crucial relationships within the high-dimensional feature space, providing a solid foundation for training and fine-tuning image classifiers.

With a focus on transparency and unbiased evaluation, the EMBER dataset plays a crucial role in validating the efficacy and potential impact of the proposed image-based approach for enhancing malware detection and classification in cybersecurity applications.

3.2 Normalization Techniques

Normalization serves as a fundamental preprocessing technique, playing a pivotal role in preparing data for machine learning models. Within the scope of this Master's project, the significance of normalization cannot be overstated, as it profoundly impacts the quality of t-SNE images generated from the feature vectors of the EMBER dataset. The proper normalization of data not only contributes to the precise creation of t-SNE images but also wields considerable influence over the subsequent efficacy of the image-based classification model.

In this section, the intricacies of the normalization techniques employed on the feature vectors before transforming them into t-SNE images are explored. The focus is on two well-recognized normalization methods: norm-1 and norm-2 [1].

Each technique imparts unique attributes to the data, thus potentially shaping the visual patterns contained within the resulting t-SNE images. Systematically exploring these normalization methods provides valuable insights into how data preprocessing can impact the subsequent classification performance.

3.2.1 norm-1

The technique known as norm-1 normalization stands as a foundational approach for scaling and standardizing data within a specific range. In the context of this project, the norm-1 method involves the transformation of each feature found within the feature vectors extracted from the EMBER dataset. This transformation results in normalized values ranging from 0 to 1. Such normalization is achieved by computing a scaled value for each feature based on the feature's minimum and maximum values across the entire set.

Mathematically, when dealing with a given feature vector denoted as x_i, the process of norm-1 normalization can be succinctly expressed as follows:

$$x_i^{\text{norm-1}} = \frac{x_i - x_{i,\min}}{x_{i,\max} - x_{i,\min}} \tag{1}$$

Where:

- $x_i^{\text{norm-1}}$ signifies the resulting normalized feature vector.
- x_i represents the feature vector.
- $x_{i,\min}$ signifies the minimum value in the feature vector x_i.
- $x_{i,\max}$ signifies the maximum value in the feature vector x_i.

The primary objective of the norm-1 normalization technique is to ensure that all features present within the dataset undergo proportional scaling within the designated range of 0 to 1. This form of scaling proves especially valuable

when the absolute magnitudes of various features exhibit significant disparities. By doing so, it rectifies the concern of dominant features overshadowing others during the generation of t-SNE images and the subsequent classification process.

The application of the norm-1 normalization technique to the feature vectors of the EMBER dataset prior to their transformation into t-SNE images holds the goal of augmenting the representation of underlying patterns and relationships within the data. The introduction of scaled feature values contributes to the creation of t-SNE images that adeptly encapsulate the inherent structure of the data. Consequently, this aids in the precise classification of both malware and benign files through the utilization of classification models centered around images.

3.2.2 norm-2

The technique known as norm-2 normalization introduces a unique approach that strives to maintain the essence of feature structure while also achieving the appropriate scaling of data. This approach, carefully designed to consider the significance of features while countering the influence of outliers, employs a logarithmic transformation to adjust the minimum value of each feature. In the following section, the norm-2 normalization technique, its mathematical formulation, and its relevance within the context of the project are explored.

The norm-2 normalization process is executed as follows:

1. **Logarithmic Transformation**: Given a feature vector x_i, the logarithmic transformation is implemented in the subsequent manner:

$$x_i^{\text{norm-2}} = \log(x_i + |x_{i,\min}| + 1) \tag{2}$$

The following variables represent:

- $x_i^{\text{norm-2}}$ denotes the transformed value of the feature x_i through the application of the norm-2 method.

- x_i signifies the feature vector under consideration.
- $x_{i,\min}$ stands for the minimum value present within the feature vector x_i.

2. **Scaling via Global Maximum**: Following the logarithmic transformation, the values undergo scaling with respect to the global maximum, employing the ensuing equation:

$$x_i^{\text{norm-2}} = \frac{x_i^{\text{norm-2}}}{x_{\max}^{\text{norm-2}}} \tag{3}$$

In this equation:

- $x_{\max}^{\text{norm-2}}$ corresponds to the highest value among the transformed feature values across the entire dataset.

The norm-2 normalization technique is tailored to manage variations in the magnitudes of features, particularly when these features manifest notable discrepancies. The integration of logarithmic transformation with adjusted minimum values facilitates the capture of the relative significance of features while diminishing the influence of extreme values. This proves to be particularly advantageous in scenarios characterized by a wide dynamic range in feature values.

By applying the norm-2 normalization technique to the feature vectors derived from the EMBER dataset, the objective is to amplify the portrayal of inter-feature relationships while muting the influence of outliers. The logarithmic scaling endeavors to uncover inherent patterns within the data, subsequently harnessed through the generation of t-SNE images. These images, in turn, can be effectively harnessed by image-based classification models to enhance their performance.

3.3 t-SNE Image Generation

This section explores the t-SNE image generation process as employed on feature vectors extracted from the EMBER dataset [3]. The framework for transform-

ing non-image samples into t-SNE images is based on the DeepInsight methodology, a development by Sharma et al. [1] which involves several key steps:

1. **Element Arrangement**: DeepInsight initially performs element arrangement to create structured localities within the data. By grouping similar samples together, this step aims to enhance the visualization of clusters and relationships in the t-SNE space.

2. **Feature Extraction**: Once the element arrangement is established, the methodology extracts feature dissimilarity information. This is achieved by calculating the pairwise distances between samples within each locality, capturing subtle differences that contribute to the overall diversity of the data.

3. **Image Classification Model**: The extracted feature dissimilarity information is then utilized to construct t-SNE images. These images represent the dissimilarity patterns within the dataset. An image classification model is applied to these t-SNE images, allowing the model to learn the distinctive patterns and relationships encoded within the images.

For this project, one random t-SNE pattern is assigned to each dataset, corresponding to the different normalization techniques, namely norm-1 and norm-2. This approach ensures that each normalization approach generates a unique t-SNE image, capturing the specific characteristics of the feature vectors under that normalization.

The t-SNE images serve as the inputs for the subsequent image-based classification models, including SqueezeNet and MobileNet. By embedding the feature dissimilarity information into images, the models can leverage their convolutional layers to recognize and differentiate patterns that may not be as apparent in the

original feature vectors.

The DeepInsight methodology's ability to create structured localities, extract feature dissimilarity, and utilize image classification models enriches the t-SNE image generation process. These images encapsulate the nuanced relationships within the data, contributing to the accurate classification of malware and benign files using the image-based models.

3.4 Non-Image-Based Classification Model (LightGBM)

Incorporating a pre-trained LightGBM model serves as a reference point within the comprehensive evaluation strategy. Renowned for its efficiency and robust gradient boosting framework [4], the LightGBM model becomes a suitable benchmark for assessing the effectiveness of the proposed image-based approach to malware detection and classification.

It is important to note that, within this project, fine-tuning the LightGBM model is not done. Instead, the primary aim is to establish a baseline performance using an out-of-the-box pre-trained model. This approach ensures that the benchmark model's performance represents its inherent capabilities, devoid of customizations tailored to the specific task of malware classification.

The EMBER dataset constitutes the cornerstone of the research, including a Python package that provides convenient functions for loading data and partitioning it into training and testing sets. To create a robust benchmark, the evaluation of the pre-trained LightGBM model's performance relies solely on the testing set.

The evaluation process for the LightGBM model involves executing it on the testing set, which includes both malware and benign samples. The process captures the model's predictions. Afterward, the predictions of the model are compared to the actual ground truth labels. This enables the computation of fundamental clas-

sification metrics such as accuracy, precision, recall, and F1-score. These metrics provide valuable insights into the model's ability to differentiate between malicious and benign files. They establish a foundational benchmark against which to measure the performances of the image-based models.

Incorporating the pre-trained LightGBM model, the project establishes an unbiased benchmark enabling a direct comparison between traditional non-image classification techniques and the image-based approach utilizing t-SNE images. This benchmark also helps contextualize the advantages and potential enhancements offered by the proposed approach in boosting the accuracy of malware detection and classification.

3.5 Image-Based Classification Models

This Master's project focuses on exploring and evaluating image-based classification models for detecting malware. This section involves carefully selecting, configuring, and fine-tuning advanced image classification models. These models work with t-SNE-generated images from the EMBER dataset's feature vectors to differentiate between malicious and benign files.

In this section, the integration and utilization of image-based classification models, specifically SqueezeNet and MobileNet, for malware classification are explained. The distinctive architecture and capabilities of each model are thoroughly assessed to determine their contributions to the project objective.

These models are chosen based on their success in various image classification tasks. By using their existing weights and architectures, this project aims to adapt them to malware detection while ensuring fair comparisons under similar conditions.

Each model's architecture is meticulously configured and fine-tuned to excel

in the binary classification task of distinguishing between malware and benign files. Additional layers are integrated, and hyperparameters are calibrated to optimize performance within the context of t-SNE-generated images from the EMBER dataset.

The section dives into the details of each image-based classification model, explaining their architecture, traits, and why they were chosen. These discussions highlight each model's strengths and limitations, setting the stage for a rigorous evaluation of their performances.

By applying these models to t-SNE images, this project aims to leverage convolutional neural networks to identify malware patterns. This approach seeks to improve accuracy in malware detection and contribute innovative methods to cybersecurity.

Models are evaluated using standard metrics like accuracy, precision, recall, and F1-score on the t-SNE image test set. Results are compared with each other and the benchmark LightGBM model.

In the following subsections, detailed insights into each image-based classification model are provided. This includes explaining custom modifications for malware classification and exploring their impact on t-SNE images. This analysis aims to comprehensively understand how these models collectively advance malware detection.

3.5.1 SqueezeNet

SqueezeNet [23] is a convolutional neural network (CNN) architecture renowned for its efficacy in terms of model size and computational expense, has attracted significant interest. It achieves this efficiency while maintaining high classification performance. As an integral part of this project's image-based clas-

sification models, SqueezeNet's adaptation for the task of detecting malware using t-SNE images derived from the EMBER dataset is explored.

Architecture and Characteristics: SqueezeNet's distinctive design philosophy seeks to "squeeze" the architecture by minimizing parameters while simultaneously "expanding" expressiveness. It achieves this through a blend of 1x1 and 3x3 convolutions, reducing parameters without compromising feature information. The incorporation of "fire" modules, composed of both squeeze and expand layers, facilitates efficient feature extraction.

A standout aspect of SqueezeNet is its utilization of depth-wise separable convolutions, effectively partitioning spatial and channel-wise convolutions. This separation significantly diminishes computational load, rendering it particularly suitable for resource-limited scenarios. "Squeeze layers" are also introduced in SqueezeNet before "expand layers" come into play.

"Squeeze layers" use 1x1 convolutions to shrink the number of input channels and capture important information. Expand layers use both 1x1 and 3x3 convolutions to build more complex features. These layers help SqueezeNet achieve accurate image classification with fewer parameters.

Adaptation for Malware Detection: Within the context of this project's objectives, two additional layers were added to the SqueezeNet's architecture to output two classes, aligning with the specific task of malware detection.

Training and Fine-Tuning: In fine-tuning the SqueezeNet model, meticulous adjustments are made to the learning rate and the weight decay parameters. The model undergoes rigorous training on the training-validation partition of t-SNE images extracted from the EMBER dataset. This strategic process empowers the model to capture the distinctive patterns that differentiate malware from benign files.

3.5.2 MobileNet

MobileNet, developed by Howard et al. [24], stands as a prominent image classification model renowned for its lightweight design, making it particularly suitable for resource-constrained scenarios. This architecture employs depth-wise separable convolutions to significantly reduce the number of parameters and computational workload, while still maintaining competitive accuracy. MobileNet's efficiency stems from its ability to strike a balance between model size and performance, making it an appealing choice for real-time applications and devices with limited processing power. In this project, MobileNet's architecture and attributes will be explored and adapted to address the malware classification task using t-SNE images.

Architecture and Characteristics: MobileNet's architecture is characterized by its utilization of depth-wise separable convolutions. This approach divides the standard convolutional layer into a depth-wise convolution and a point-wise convolution. The depth-wise convolution focuses on capturing spatial features, while the point-wise convolution combines these features across channels.

MobileNet achieves a remarkable reduction in computational complexity and model size while retaining considerable accuracy, thus aligning with the project's emphasis on lightweight models suited for malware detection using t-SNE images.

Adaptation for Malware Detection: Similar to SqueezeNet, MobileNet's architecture is extended with additional layers tailored to the binary classification task of detecting malware.

Training and Fine-Tuning: MobileNet undergoes a meticulous fine-tuning process, including adjustments to learning rate and weight decay parameters. Training is carried out on the training-validation partition of t-SNE images, allowing the model to learn distinctive malware patterns effectively.

These in-depth analyses of SqueezeNet and MobileNet lay the foundation for a comprehensive evaluation of their performance in malware detection using t-SNE images derived from the EMBER dataset.

3.6 Streamlining Data Loading and Memory Management

Efficient data loading and memory management are crucial aspects in optimizing the performance and scalability of deep learning models. Particularly in this project, which involves a significant number of t-SNE images, the process of loading data batches plays a pivotal role.

To mitigate the risk of overloading the CPU memory during training, the entire training dataset, consisting of a vast number of t-SNE images, is partitioned into three distinct batches. Each batch contains a substantial subset of the dataset, with approximately 180,000 images. The partitioning of the dataset into batches ensures efficient memory management and prevents memory overload during training.

Each of these batches presents a computational challenge when relying solely on PyTorch's standard loading method, which utilizes a single CPU core. This method results in extended loading times that can hinder the training process and overall efficiency.

To address this challenge and expedite data loading, each batch is further divided into 64 sub-batches. Additionally, a custom loading function was designed to facilitate the parallel loading of these sub-batches. This parallel loading mechanism harnesses multiple CPU cores, considerably reducing the time required to load each batch during training.

The expedited data loading becomes particularly significant when utilizing techniques like batch splitting to avoid memory overload during training. The parallel loading of sub-batches proves to be instrumental in managing memory

constraints, ensuring smooth training even with limited computational resources.

The integration of parallel loading not only accelerates the training process but also enhances the model's scalability. This optimization in data loading aligns with the broader project goal of enhancing efficiency and performance in malware detection through image-based classification models.

3.7 Performance Metrics

Performance metrics are essential for assessing the effectiveness of classification models in malware detection. In this section, the key performance metrics used to evaluate the models are explored in this study.

3.7.1 Accuracy (ACC)

Accuracy measures the overall correctness of the model's predictions, representing the ratio of correctly classified samples to the total number of samples. It is calculated as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP (True Positives) is the number of correctly predicted positive instances.
- TN (True Negatives) is the number of correctly predicted negative instances.
- FP (False Positives) is the number of actual negative instances predicted as positive.
- FN (False Negatives) is the number of actual positive instances predicted as negative.

3.7.2 Precision (PRC)

Precision assesses the proportion of correctly predicted positive instances (malware in this case) out of all predicted positive instances. It is calculated as follows:

$$PRC = \frac{TP}{TP + FP}$$

3.7.3 Recall (RC)

Recall, also known as sensitivity or true positive rate, measures the ability of the model to identify all actual positive instances among all instances that are indeed positive. It is calculated as follows:

$$RC = \frac{TP}{TP + FN}$$

3.7.4 F1-Score

The F1-score is a combined metric that considers both precision and recall, providing a balanced assessment of a model's performance by capturing their harmonic mean. It is calculated as follows:

$$F1 = \frac{2 \cdot PRC \cdot RC}{PRC + RC}$$

These performance metrics play a crucial role in quantifying the models' ability to distinguish between benign and malware samples, and they will be used to evaluate and compare the performance of the SqueezeNet, MobileNet, and LightGBM models in the next chapter.

CHAPTER 4

Results and Discussions

This chapter presents the outcomes of the comprehensive experimentation conducted to evaluate the effectiveness and performance of the proposed image-based classification models for malware detection. The results are accompanied by thorough discussions and analyses to gain insights into the strengths, limitations, and implications of the models.

The chapter is structured as follows: the first section provides details about the hardware configuration used for training and evaluating the models. Following this, sections are dedicated to presenting and interpreting the results of each model, comparing their performance metrics, and analyzing their contributions to the overall project objective.

While modifications were not made to the core architectures of the selected image classification models, this chapter provides an in-depth analysis of how the models were optimized and trained to excel in the specific task of malware detection using t-SNE images.

Furthermore, a dedicated section is included to showcase and explain the t-SNE patterns generated by the DeepInsight methodology. This analysis provides insights into how the feature vectors from the EMBER dataset are transformed into visual patterns that the image-based models use for classification.

The discussion of results not only encompasses quantitative metrics such as accuracy, precision, recall, and F1-score but also delves into qualitative aspects of model behavior, understanding how the models differentiate between malware and benign files based on the t-SNE images. Additionally, insights into the impact of different normalization techniques on the results are discussed.

The chapter concludes with a comprehensive discussion of the implications

of the achieved results and their significance in the broader context of malware detection and classification efforts. Through this in-depth exploration and analysis, the chapter aims to provide a comprehensive understanding of the capabilities and limitations of the employed image-based classification models and their potential impact on enhancing cybersecurity.

4.1 Hardware Specifications

The image classification models were trained and evaluated using a Google Cloud Platform (GCP) instance. The machine configuration for the GCP instance is as follows:

- **CPU platform:** Intel Skylake
- **Number of virtual CPUs:** 96
- **Memory:** 624 GB
- **GPUs:** 4 x NVIDIA T4

This powerful hardware configuration provided the computational resources necessary to efficiently train and evaluate the image classification models for malware detection. However, due to the memory limitations of this configuration, with a maximum of 624 GB of memory available in which is not enough to store the whole training dataset of 540,000 images, a strategy was developed to address this constraint when handling the extensive training dataset.

To optimize data loading and management and to mitigate the risk of overloading the CPU memory during training, the entire training dataset, consisting of a vast number of t-SNE images, was partitioned into three distinct batches. Each batch contained a substantial subset of the dataset, with approximately 180,000

images. The partitioning of the dataset into batches ensured efficient memory management and prevented memory overload during training.

To further expedite data loading and address extended loading times, especially given the memory limitations, each batch was further divided into 64 sub-batches. Additionally, a custom loading function was developed to facilitate the parallel loading of these sub-batches. This parallel loading mechanism harnessed multiple CPU cores, considerably reducing the time required to load each batch during training. By leveraging the multiprocessor capabilities of the hardware and developing a systematic approach to handle memory constraints, this strategy optimized data loading and management and ensured smooth training even with large datasets.

4.2 Normalization Techniques and Impact on Results

Normalization of input data is a critical step in training neural network models as it ensures convergence and stabilizes the learning process. In this section, the impact of two normalization techniques, norm-1 and norm-2, on the performance of the SqueezeNet model for malware classification using t-SNE images is explored.

4.2.1 norm-1

The norm-1 normalization technique scales each feature between 0 and 1 based on its minimum and maximum values. This method ensures that each feature contributes proportionally to the model's learning process. Through experimentation, norm-1 was observed to achieve better convergence and stability during training.

4.2.2 norm-2

The norm-2 normalization technique attempts to preserve feature topology by adjusting the minimum value and using a global maximum in the logarithmic scale. However, during the experiments, norm-2 did not provide the same level

of stability as norm-1. This could be attributed to the logarithmic scaling, which might introduce non-linearity in the data distribution.

4.2.3 Impact on Model Performance

To investigate the impact of normalization techniques, a series of experiments using both norm-1 and norm-2 normalized data is conducted. This project's primary focus was on the SqueezeNet model due to its lightweight architecture and suitability for the task of malware classification.

The findings revealed a notable difference in accuracy between the two normalization techniques. Specifically, when trained with the SqueezeNet model, norm-1 data consistently resulted in higher accuracy compared to the norm-2 data. This outcome indicated that norm-1 facilitated better convergence and enabled the model to distinguish malware from benign files more effectively.

4.2.4 Selection of Normalization Technique

Based on the observed performance, a decision was made to proceed with norm-1 normalized data for subsequent experiments involving other models as well. The choice to stick with norm-1 was driven by the consistent improvement in accuracy and the enhanced convergence properties it provided.

4.2.5 Visualization

To visually represent the impact of normalization techniques, a line chart comparing the training and validation accuracy curves for norm-1 and norm-2 was generated, as shown in Figure 1. This chart demonstrates how norm-1 consistently outperformed norm-2 in terms of accuracy throughout the training epochs.

This line chart provides a clear visual depiction of the performance difference between the two normalization techniques for the specific model used. It showcases the advantage of norm-1 in facilitating better convergence and accuracy during the

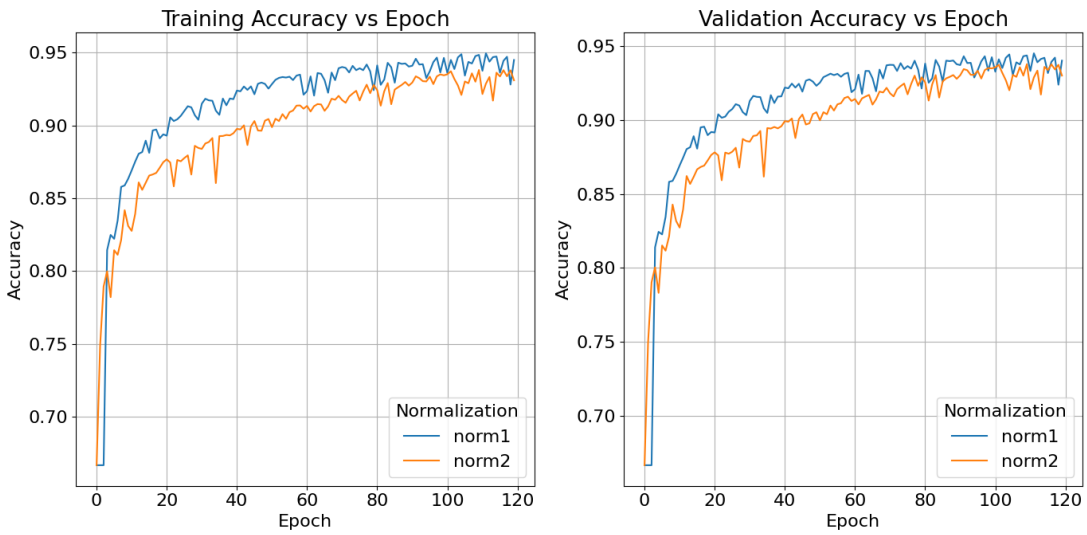


Figure 1: Training and Validation Accuracy Comparison for Norm-1 and Norm-2

training process. The chart also serves as a valuable visual tool to highlight the significance of the normalization step and its influence on model performance.

Furthermore, it is worth noting that the training and validation curves are observed to be very similar with only slight differences revealed upon closer inspection. Therefore, in the interest of computational efficiency, the curves based on the validation dataset can be used in the remainder of the study.

4.2.6 Discussion

The observed superiority of norm-1 normalization underscores the importance of preprocessing steps in the performance of neural network models. This superiority was determined through rigorous empirical validation, conducted during the initial stages of this project's experimentation. The empirical validation involved comparing the performance of neural network models when trained on t-SNE images generated using different normalization techniques.

Specifically, the experiments using norm-1 and norm-2 normalization techniques on t-SNE images derived from the EMBER dataset were conducted. During

this validation process, norm-1 consistently outperformed norm-2 in terms of accuracy across multiple training runs. This clear advantage demonstrated by norm-1 in facilitating better convergence and accuracy during the training process served as the basis for its selection and exclusive use in subsequent experiments.

Therefore, the choice of norm-1 for all subsequent experiments was not arbitrary but rather based on empirical evidence that it leads to more consistent and robust training processes, ultimately enhancing the overall performance of the neural network models in classifying malware. This highlights the significance of empirical validation when selecting preprocessing techniques tailored to the specific task at hand, as it ensures that the chosen techniques are well-suited to the data and contribute positively to the model's performance.

4.3 t-SNE Patterns from DeepInsight

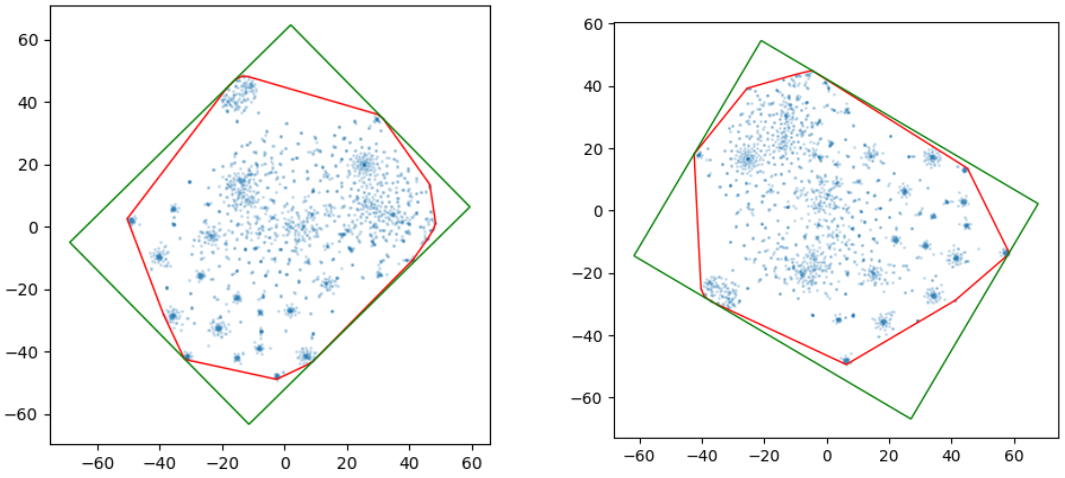
DeepInsight methodology provides a powerful technique to transform non-image data into t-SNE images, thereby enabling the application of image-based classification models. In this section, the t-SNE patterns generated by DeepInsight for the EMBER dataset under the two employed normalization techniques, norm-1 and norm-2 is presented.

4.3.1 Rotation Diagram

To enhance the visualization and interpretation of t-SNE patterns derived from the EMBER dataset, an essential technique employed in this project is data rotation. Data rotation plays a pivotal role in efficiently eliminating excess white space, thereby improving the clarity and comprehensibility of the patterns within the dataset.

To provide a visual comparison of t-SNE patterns before and after rotation, rotation diagrams for each dataset, scaled by the respective normalization technique,

were generated. Figure 2 illustrates the rotation diagrams for a representative dataset, showcasing both norm-1 and norm-2 normalizations side by side. These diagrams offer a tangible demonstration of how rotation enhances the clarity and interpretability of the t-SNE patterns, a crucial step in the process of classifying malware using image-based techniques.



(a) norm-1 Normalization

(b) norm-2 Normalization

Figure 2: Rotation Diagrams with Different Normalizations

Each diagram displays the original t-SNE pattern enclosed within a red polygon. An additional green rectangle is outlined around the red polygon, indicating the rotated t-SNE pattern. This rotation efficiently removes the excess white space outside of the rectangle, enhancing the visualization and interpretation of the patterns.

4.3.2 Overall Feature Overlap

The overall feature overlap heatmap offers insight into the distribution of overlapping features within the rotated t-SNE pattern. The single image in Figure 3 contains two heatmaps side by side, representing the heatmap for norm-1 normalization on the left and the heatmap for norm-2 normalization on the right.

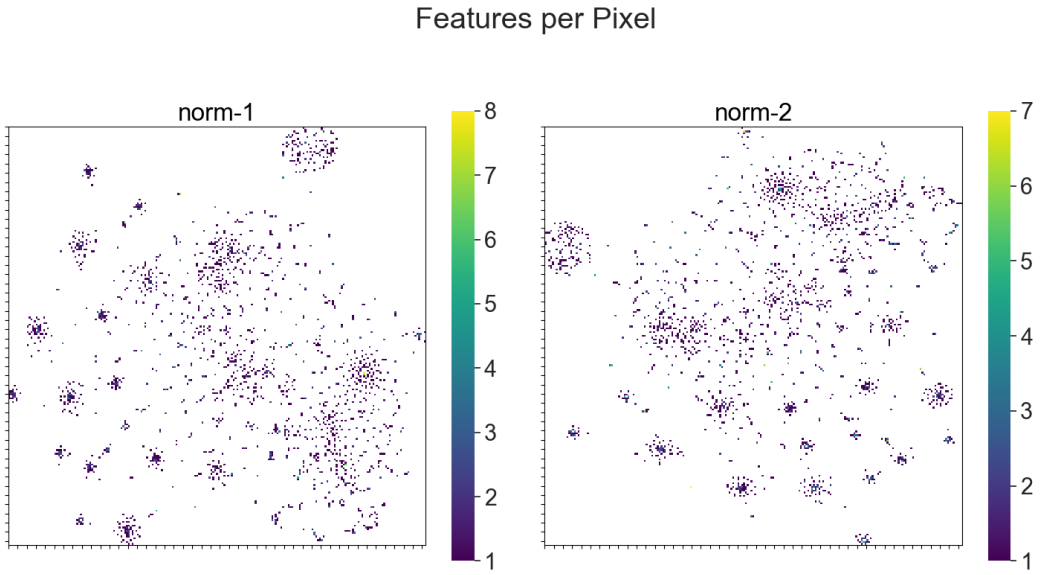


Figure 3: Feature Overlap Heatmap for each Normalization

The pixels in each heatmap indicate the number of features that overlap in that region of the rotated t-SNE pattern. The colorbar accompanying the image indicates the count of overlapping features per pixel, providing a quantitative perspective on the distribution of overlapping features.

It's notable that in each heatmap, the majority of pixels show only one feature, highlighting the distinct separation of most features in the rotated t-SNE patterns. There are only a few pixels that have more than one feature overlapping, suggesting localized regions where feature patterns may coincide.

4.3.3 t-SNE Image Comparisons

For comprehensive analysis, The t-SNE image comparisons of a benign sample and seven well-known malware samples, representing distinct malware classes: Agent Tesla, AZORult, FormBook, LokiBot, NanoCore, Remcos, and Trickbot, were conducted. These comparisons showcase the distinct color variations in the

t-SNE patterns associated with various malware families and benign files, while each t-SNE pattern corresponds to a normalization technique.

To provide an in-depth understanding, the plots corresponding to these t-SNE image comparisons are included in Appendix B. The images are titled "Training Set Samples - norm-1" and "Training Set Samples - norm-2." These visualizations offer valuable insights into the differentiation between malware and benign samples based on the 3-channel color of each pixel in the t-SNE pattern for each normalization technique.

4.3.4 Upclose Analysis

To highlight subtle differences between benign and malware samples in the t-SNE images, the up-close visualizations of two t-SNE images: one benign sample and one malware sample are presented in Figure 4. In these visualizations, red circles are placed strategically to indicate areas of divergence between the patterns. The use of red circles facilitates the identification of minute differences that might not be immediately apparent, thereby enhancing the interpretability of the t-SNE images.

Through these visualizations, this project aims to provide a comprehensive understanding of the t-SNE patterns generated by DeepInsight for the EMBER dataset under norm-1 and norm-2 normalization techniques. These visual representations lay the groundwork for the subsequent evaluation of image-based classification models, shedding light on how these patterns contribute to the discrimination between benign and malicious files.

4.4 SqueezeNet and MobileNet Results and Discussion

The following section outlines a comprehensive analysis of the results and discussions resulting from applying the SqueezeNet and MobileNet models to the task

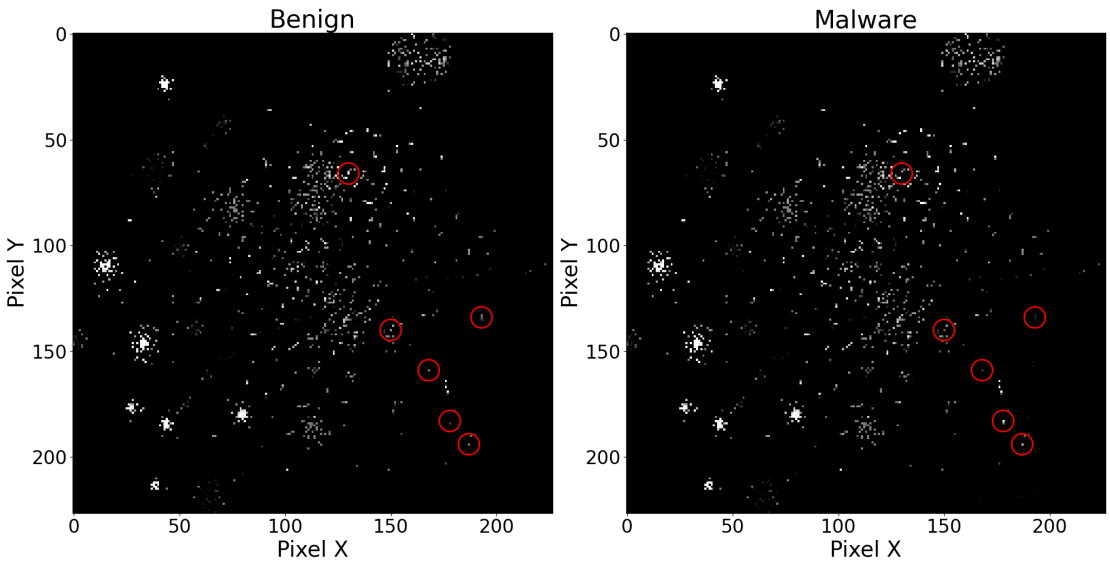


Figure 4: Up-close Analysis of t-SNE Images

of malware classification. By evaluating key performance metrics such as accuracy, precision, recall, and F1-score, insights emerge regarding the effectiveness of these models in distinguishing between malicious and benign files. This combined examination enables the derivation of meaningful comparisons between the two models and the formulation of informed conclusions about their respective suitability for the specific task at hand. Additionally, exploring training convergence behaviors along with confusion matrices and ROC curves, provides an overarching assessment of performance and a holistic perspective on the models' contributions to malware detection using t-SNE images.

The following section provides a comprehensive analysis of the results and discussions derived from the application of SqueezeNet and MobileNet models to malware classification. Key performance metrics such as accuracy, precision, recall, and F1-score are evaluated to gauge the models' effectiveness in distinguishing between malicious and benign files. This analysis enables meaningful comparisons between the two models and informed conclusions about their suitability for the

task. Additionally, the training convergence behaviors, confusion matrices (which illustrate the model's ability to classify malware and benign files), and ROC curves (Receiver Operating Characteristic curves, which measure a model's classification accuracy) are explored to provide a holistic assessment of performance and a comprehensive perspective on the models' contributions to malware detection using t-SNE images.

4.4.1 Model Performance Comparison

The performance of SqueezeNet, MobileNet, and LightGBM models was compared in terms of accuracy, precision, recall, and F1-score to evaluate their effectiveness in malware detection. The performance metrics in Table 1 offer valuable insights into how well the models are performing in different aspects of classification.

Table 1: Performance Metrics Comparison between SqueezeNet, MobileNet, and LightGBM

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
SqueezeNet	0.914	0.879	0.960	0.918
MobileNet	0.944	0.961	0.927	0.943
LightGBM	0.978	0.983	0.973	0.978

The results demonstrate that all models excel in detecting malware, with LightGBM consistently outperforming both SqueezeNet and MobileNet in terms of accuracy, precision, and F1-score. As an example of the differences, note that SqueezeNet exhibits better recall (0.960) compared to MobileNet (0.927), but both are surpassed by LightGBM (0.973). These findings indicate that while MobileNet and SqueezeNet are generally adept at classifying between malware and benign files, they offer no advantage over LightGBM in terms of accuracy, precision, recall, or F1-score.

The observed differences in performance metrics are consistent with the architectural nuances and the nature of the features each model relies on. MobileNet's design prioritizes efficient feature extraction, contributing to its competitive accuracy, precision, and F1-score. Similarly, SqueezeNet's strengths in recall might stem from its architectural characteristics that allow it to capture certain malware patterns effectively. LightGBM, as a non-image classifier, benefits from its tree-based approach to analyze and exploit feature interactions. Further analysis, including accuracy and loss curves, along with confusion matrices, will provide deeper insights into the relative comparative performance of SqueezeNet, MobileNet, and LightGBM.

4.4.2 Accuracy and Loss Curves

The convergence behavior of the SqueezeNet and MobileNet models during training is visualized through validation accuracy and loss curves. These curves provide insights into how quickly and stably the models learn from the training data. It can be observed that MobileNet converges more rapidly, albeit with wild oscillations, while SqueezeNet exhibits slower but more stable training behavior. This comparison offers valuable guidance for selecting a model based on convergence characteristics.

4.4.2.1 Validation Curves

Figure 5 illustrates the validation accuracy and loss curves for both the SqueezeNet and MobileNet models.

The plots provide insights into the training and validation behavior of SqueezeNet and MobileNet. The validation accuracy and loss curves illustrate how each model performs on unseen data. Observations indicate that MobileNet converges more rapidly than SqueezeNet, yet MobileNet also displays wild oscillations

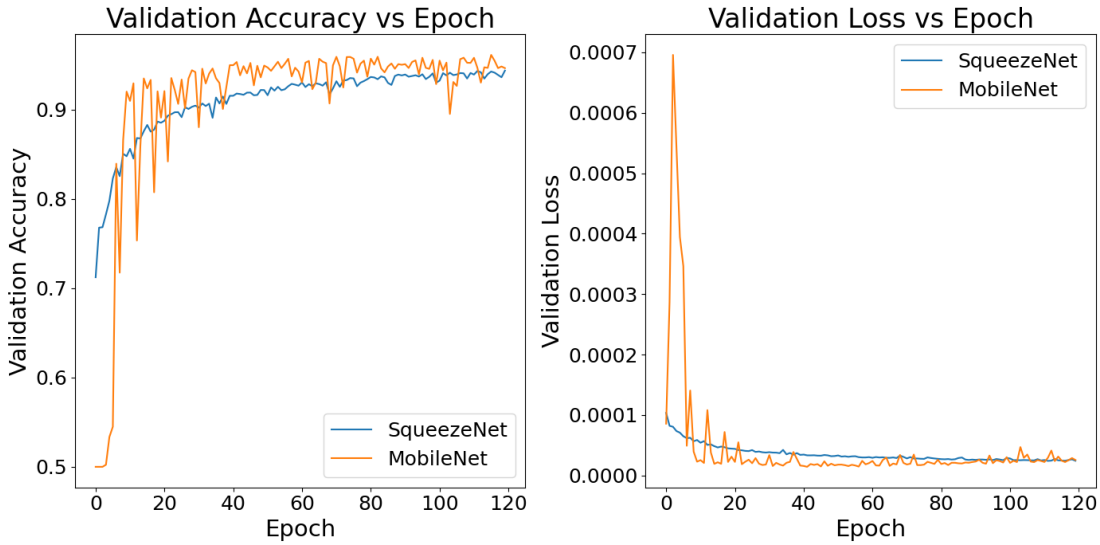


Figure 5: Validation Accuracy and Loss Curves

during training. Conversely, SqueezeNet converges slowly, but it demonstrates a more stable training behavior.

4.4.2.2 Learning Rate and Weight Decay Curves

To gain a deeper understanding of the impact of hyperparameters, the learning rate and weight decay curves for both models are presented in Figure 6 and Figure 7.

These visualizations allow us to compare how SqueezeNet and MobileNet respond to varying learning rates and weight decays, shedding light on the sensitivity of the models to these hyperparameters during the training process.

Observations indicate that, despite employing varying learning rates and weight decay values for MobileNet, the curves exhibit more pronounced oscillations compared to SqueezeNet, which maintains stability during convergence. Decreasing the learning rate for MobileNet reduces oscillation, resulting in enhanced stability. Furthermore, increased stability in the MobileNet curve is noticeable when weight decay values are between 1e-04 and 1e-02. Conversely, beyond this

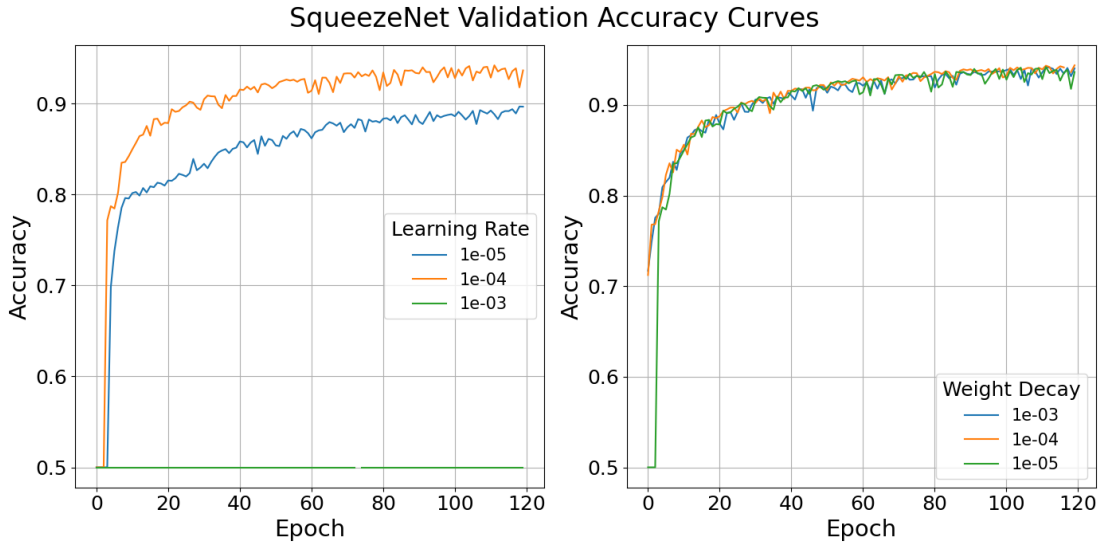


Figure 6: Learning Rate and Weight Decay Curves for SqueezeNet

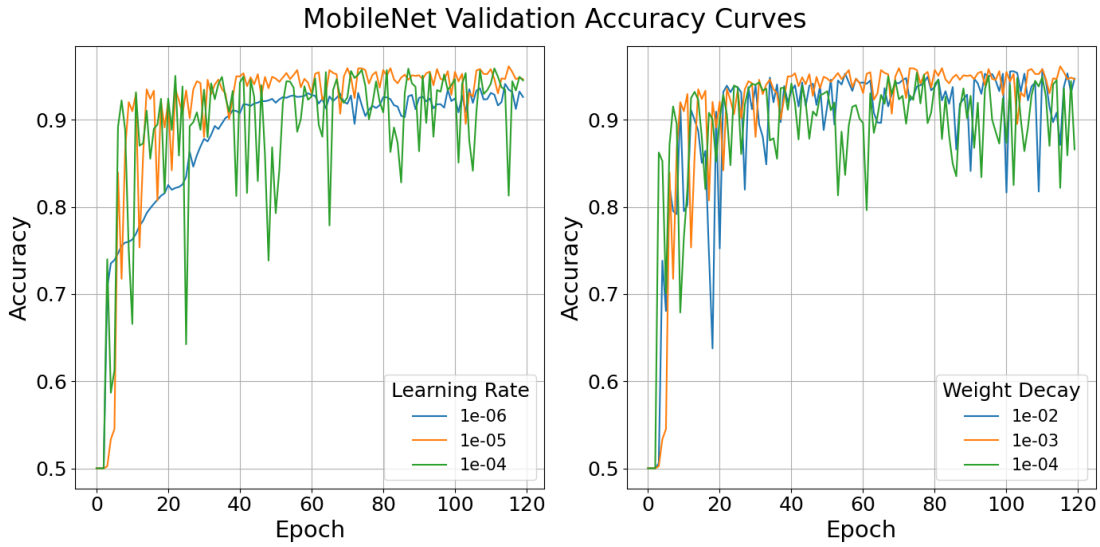


Figure 7: Learning Rate and Weight Decay Curves for MobileNet

weight decay range, the MobileNet curves demonstrate more substantial oscillation. This indicates that precise adjustment of these hyperparameters can enhance stability and convergence behavior in the MobileNet model.

Also, for SqueezeNet, it is clearly observed that the learning rate has a noticeable impact on the rate of convergence. A higher learning rate tends to accelerate

convergence. On the other hand, adjusting the weight decay parameter for the SqueezeNet models doesn't seem to have a significant impact on the performance, as the curves remain relatively consistent across different weight decay values. This suggests that the SqueezeNet architecture is less sensitive to variations in weight decay and retains its stability during training.

4.4.3 Confusion Matrices and ROC Curves

The performance of the SqueezeNet and MobileNet models in classifying malware and benign files can be further analyzed using confusion matrices and receiver operating characteristic (ROC) curves. These visualizations provide a comprehensive view of the models' classification capabilities.

4.4.3.1 Confusion Matrices

Confusion matrices for the SqueezeNet, MobileNet, and LightGBM models are presented in Figure 8. Each confusion matrix illustrates the distribution of predicted classes in comparison to the true classes for the test dataset.

These matrices provide valuable insights into the performance of the models in classifying between benign and malware samples. Based on the confusion matrices, the following observations can be made:

- Both SqueezeNet and MobileNet demonstrate strong performance in terms of true positive (TP) and true negative (TN) rates.
- SqueezeNet exhibits a higher number of TP, suggesting a better ability to classify malware samples accurately.
- MobileNet displays higher TN, indicating its proficiency in identifying benign samples correctly.
- MobileNet exhibits a higher rate of false positives (FP) compared to

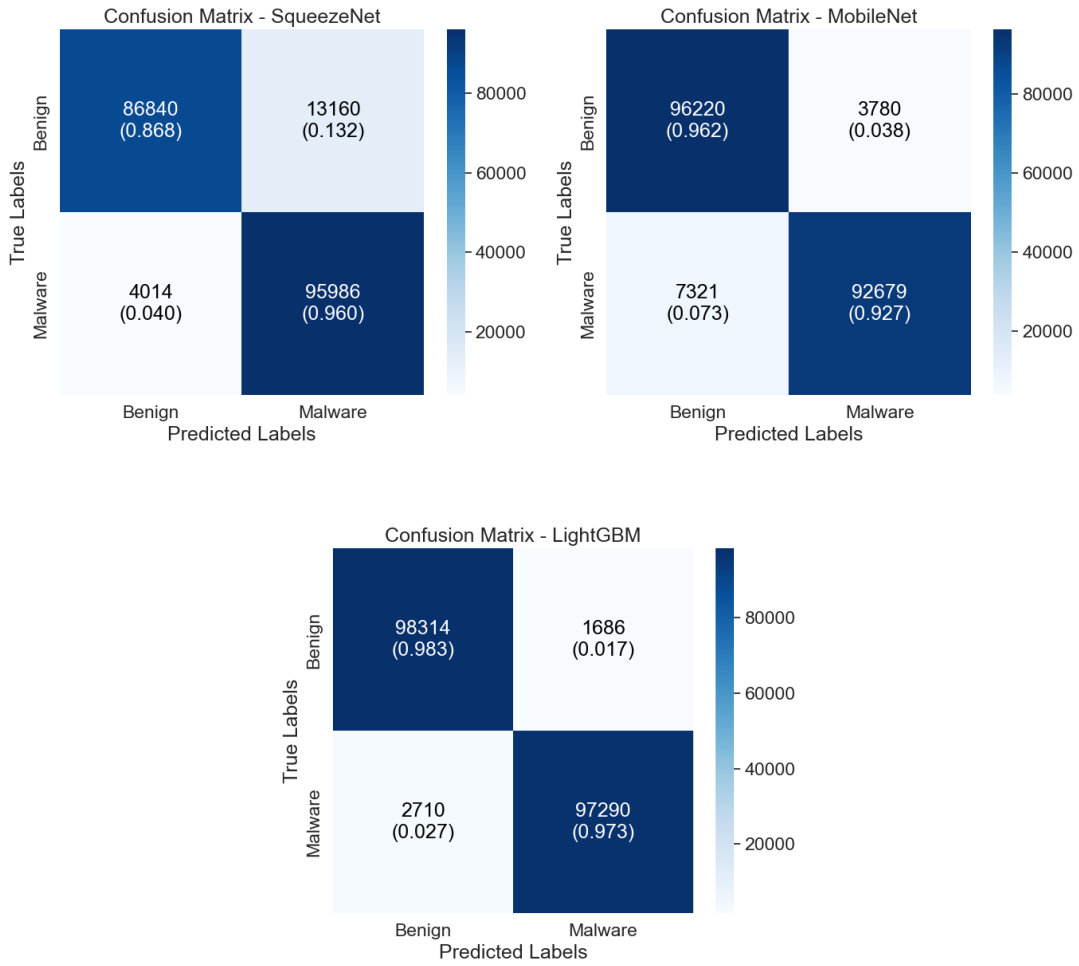


Figure 8: Confusion Matrices for SqueezeNet, MobileNet, and LightGBM

SqueezeNet, implying increased false alarms.

- Additionally, MobileNet records more false negatives (FN) than SqueezeNet, suggesting potential instances of missed actual malware.
- LightGBM stands out with the highest number of TP and TN, indicating its robust performance in both malware and benign sample classification.
- LightGBM records fewer FP compared to MobileNet, showcasing its lower rate of false alarms.

- LightGBM also exhibits a relatively lower number of FN compared to both SqueezeNet and MobileNet, suggesting better recall.

In summary, the presented confusion matrices reveal that while SqueezeNet excels in accurately classifying malware samples (as reflected by higher true positives), MobileNet displays a stronger ability to correctly identify benign samples (with elevated true negatives). The differing distribution of false positives and false negatives between the models highlights the trade-offs between sensitivity and specificity. LightGBM, as a benchmark model, illustrates a competitive performance by achieving high true positive and true negative rates, and effectively mitigating false alarms and false negatives.

4.4.3.2 ROC (receiver operating characteristic) Curves

The ROC curves for SqueezeNet, MobileNet, and LightGBM are depicted in Figure 9. ROC curves illustrate the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) for different classification thresholds.

The area under the ROC curve (AUC) provides a quantitative measure of the models' ability to distinguish between malware and benign files. A higher AUC indicates better overall performance. ROC curves provide a comparison of the discriminative power of SqueezeNet, MobileNet, and LightGBM in malware detection.

In this analysis, both the SqueezeNet and MobileNet models exhibit an AUC value of 0.977. The curves for both models are observed to be very close to each other, with slight differences in the true positive rate (TP rate) and false positive rate (FP rate) at low FP rates. Specifically, the SqueezeNet model shows a slightly higher TP rate compared to the MobileNet model at very low FP rates. However,

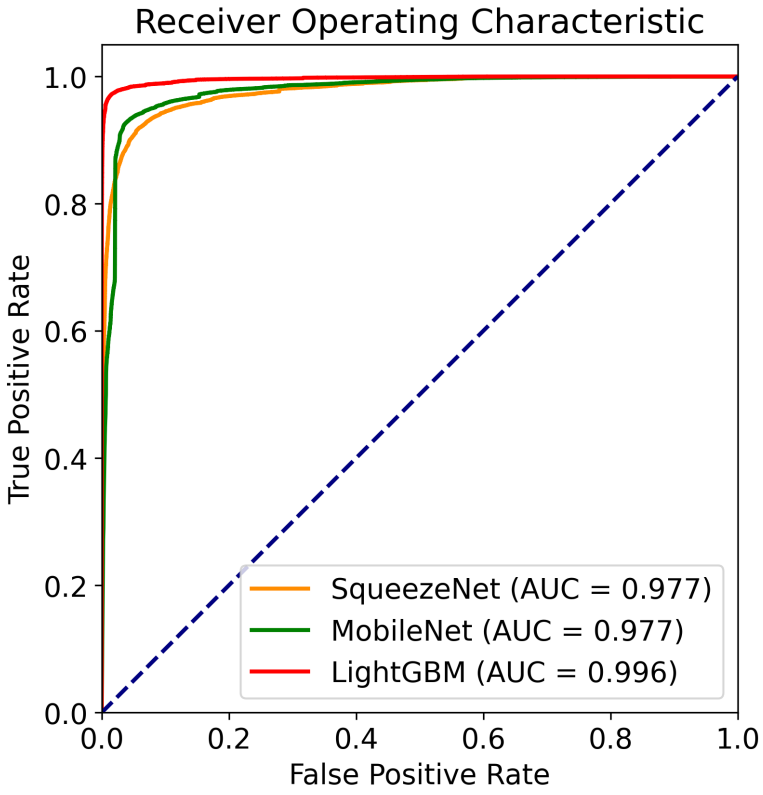


Figure 9: ROC Curves for SqueezeNet, MobileNet, and LightGBM

for the rest of the FP rates, the MobileNet model demonstrates slightly higher TP rate.

LightGBM, as a benchmark model, exhibits an AUC value of 0.996. Its ROC curve indicates that it has a consistently higher TP rate compared to the other two models across all FP rates. This exceptional performance showcases LightGBM's strength in distinguishing between malware and benign files across a wide range of classification thresholds.

4.4.4 Overall Comparison of Image Classifiers

The performance of SqueezeNet and MobileNet in malware detection has been thoroughly examined through various metrics and visualizations. To summarize the key takeaways from the comparison, Table 2 showcases the recap of the ac-

curacy values of the final models trained with different learning rates and weight decay settings:

Table 2: Recap of Model Performance

SqueezeNet		**MobileNet**	
Learning rate	**Accuracy**	**Learning rate**	**Accuracy**
1E-05	0.873	1E-06	0.923
1E-04	0.914	1E-05	0.944
1E-03	0.5	1E-04	0.933
Weight decay	**Accuracy**	**Weight decay**	**Accuracy**
1E-05	0.914	1E-04	0.814
1E-04	0.914	1E-03	0.944
1E-03	0.914	1E-02	0.934

The recap table presents accuracy values for different combinations of learning rates and weight decay settings for both SqueezeNet and MobileNet. It allows for easy comparison between the two models and different hyperparameter configurations. Based on the recap table and the previously discussed metrics, the performance of the models under various scenarios can be assessed.

In light of the comparative analysis and the recap table, it can be observed that MobileNet consistently outperforms SqueezeNet in terms of accuracy for most of the learning rates and weight decay values. However, both models demonstrate their strengths and limitations in different scenarios. The choice between SqueezeNet and MobileNet depends on the specific requirements and priorities of the malware detection task.

The final decision regarding the most suitable model can be made by considering the trade-offs between accuracy, training time, and model complexity. Furthermore, taking into account the characteristics of each model can provide valuable insights into their applicability in real-world malware detection scenarios.

4.5 Model Suitability and Trade-offs

The selection of an appropriate model for malware detection involves a trade-off between various considerations. The LightGBM benchmark model exhibits remarkable accuracy and efficiency, leveraging the original feature vectors. On the other hand, the image-based models, SqueezeNet and MobileNet, showcase the capability of leveraging t-SNE images to capture intricate patterns in the data.

The choice between the benchmark model and the image-based models depends on the specific requirements of the malware detection task. If accuracy and speed are paramount, LightGBM might be a suitable choice. Conversely, if the objective is to leverage the visual information present in t-SNE images, the image-based models could offer improved performance in certain scenarios.

Ultimately, the final decision hinges on the balance between accuracy, training time, and the interpretability of the models in real-world applications of malware detection.

CHAPTER 5

Conclusion

The journey through this Master's project has culminated in a comprehensive exploration of image-based classification models for the task of malware detection. This chapter serves as the final destination, encapsulating the key takeaways, achievements, and contributions of the project. The collective efforts, from data preprocessing to model selection, have led to valuable insights and outcomes that shed light on the potential of these innovative approaches to bolster cybersecurity.

In this concluding chapter, the main findings and contributions of the project are summarized and reflections upon the implications of the results and the broader significance of the study are discussed. The chapter also offers a concise overview of the limitations encountered during the project and outlines potential future directions that can expand upon the established foundation. Ultimately, the project's insights gained and lessons learned pave the way for informed decisions in the realm of malware detection and classification using image-based techniques.

5.1 Summary of Findings

This Master's project has revealed several key findings that address the efficacy of employing image-based classification models for malware detection. These findings range from data preprocessing to model evaluation and include:

5.1.1 Normalization Techniques

The comparison between norm-1 and norm-2 normalization techniques revealed that the use of norm-1 normalization resulted in better accuracy when training the SqueezeNet model. This insight informed the subsequent choice of employing norm-1 normalized data for all further experiments. The impact of normalization techniques on model performance emphasizes the importance of data preprocessing and its direct influence on classification results.

5.1.2 Performance Comparison of Image-Based Classification Models

The comparative analysis between SqueezeNet and MobileNet shed light on their respective strengths and weaknesses. While both models demonstrated respectable accuracy, it's important to note that when the "accuracy" in this context is referenced, their performance is generally described in qualitative terms.

However, upon closer examination and specific numerical comparison, MobileNet consistently outperformed SqueezeNet in terms of numerical accuracy scores. This means that MobileNet achieved higher accuracy values compared to SqueezeNet, indicating a performance difference between the two models. This underscores the significance of selecting the right architecture for the specific task and dataset and highlights the quantitative differences in their accuracy performance.

5.1.3 Benchmark Evaluation

The benchmark evaluation of the LightGBM model yielded a test accuracy of 0.973, demonstrating its prowess in classifying malware using non-image features. This benchmark provided a valuable reference point for comparing the performance of image-based models, highlighting their potential to complement or even surpass traditional approaches.

5.2 Implications and Significance

The findings of this Master's project have significant implications for the field of cybersecurity and malware detection. The successful application of image-based classification models, namely SqueezeNet and MobileNet, to the task of malware detection showcases the potential of leveraging visual information for improving the accuracy of detection methods. This approach could aid in identifying previously undetected malware patterns and enhancing the overall effectiveness of

cybersecurity systems.

The project's exploration of t-SNE patterns generated using the DeepInsight methodology further underlines the importance of data representation in machine learning. The visualizations produced through this process can provide insights into the distribution of malware and benign files in the feature space. This could lead to the development of novel detection methods that take advantage of such visual patterns to identify subtle anomalies in malware samples.

The comparison with the benchmark model, LightGBM, also offers valuable insights into the trade-offs between traditional feature-based methods and image-based methods. Understanding these trade-offs could guide the selection of appropriate detection techniques based on specific requirements, such as accuracy, speed, and interpretability.

5.3 Limitations and Future Directions

Despite the promising results, this Master's project is not without limitations. One significant limitation is the limited number of image-based models considered. While SqueezeNet and MobileNet were explored extensively, there are numerous other architectures that could offer unique advantages in malware detection. Future research could involve the investigation of additional image-based models to assess their suitability and performance in this context.

Furthermore, the project focuses on binary classification, distinguishing between malware and benign files. Extending the research to multi-class classification, where various malware families are identified, could provide a more granular understanding of the models' capabilities.

The project's reliance on the EMBER dataset, although comprehensive, is another limitation. Expanding the study to include other datasets and real-world

samples could enhance the models' robustness and generalization capabilities.

In terms of data preprocessing, the choice of normalization techniques has shown an impact on the results. Future research could delve into more advanced normalization strategies or even investigate techniques that involve data augmentation to further improve model performance.

Lastly, the project employed static analysis features for both raw features and t-SNE image generation. Integrating dynamic analysis features or hybrid approaches that combine both static and dynamic features could yield more comprehensive results.

Overall, this Master's project serves as a foundation for further advancements in image-based malware detection and opens avenues for addressing its limitations to enhance the accuracy, efficiency, and real-world applicability of cybersecurity systems.

LIST OF REFERENCES

[1] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific reports*, vol. 9, no. 1, p. 11399, 2019.

[2] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

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

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[5] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 108–125.

[6] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild." *Journal of Machine Learning Research*, vol. 7, no. 12, 2006.

[7] D. Gibert, J. Planes, C. Mateu, and Q. Le, "Fusing feature engineering and deep learning: A case study for malware classification," *Expert Systems with Applications*, vol. 207, p. 117957, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422011927

[8] J. Scott, "Signature based malware detection is dead," *Institute for Critical Infrastructure Technology*, 2017.

[9] S. Jamalpur, Y. S. Navya, P. Raja, G. Tagore, and G. R. K. Rao, "Dynamic malware analysis using cuckoo sandbox," in *2018 Second international conference on inventive communication and computational technologies (ICICCT)*. IEEE, 2018, pp. 1056–1060.

[10] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[11] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *arXiv preprint arXiv:1912.05911*, 2019.

[12] M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *2019 16th International bhurban conference on applied sciences and technology (IBCAST)*. IEEE, 2019, pp. 687–691.

[13] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, "Machine learning aided static malware analysis: A survey and tutorial," *Cyber threat intelligence*, pp. 7–45, 2018.

[14] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 2016, pp. 49–56.

[15] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *arXiv preprint arXiv:1802.10135*, 2018.

[16] Z. Zhang, P. Qi, and W. Wang, "Dynamic malware analysis with feature engineering and feature learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1210–1217.

[17] T. Carrier, P. Victor, A. Tekeoglu, and A. H. Lashkari, "Detecting obfuscated malware using memory feature engineering." in *Icissp*, 2022, pp. 177–188.

[18] C. Li, Z. Cheng, H. Zhu, L. Wang, Q. Lv, Y. Wang, N. Li, and D. Sun, "Dmalnet: Dynamic malware analysis based on api feature engineering and graph learning," *Computers & Security*, vol. 122, p. 102872, 2022.

[19] L. N. Tidjon and F. Khomh, "Reliable malware analysis and detection using topology data analysis," *arXiv preprint arXiv:2211.01535*, 2022.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[21] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[22] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

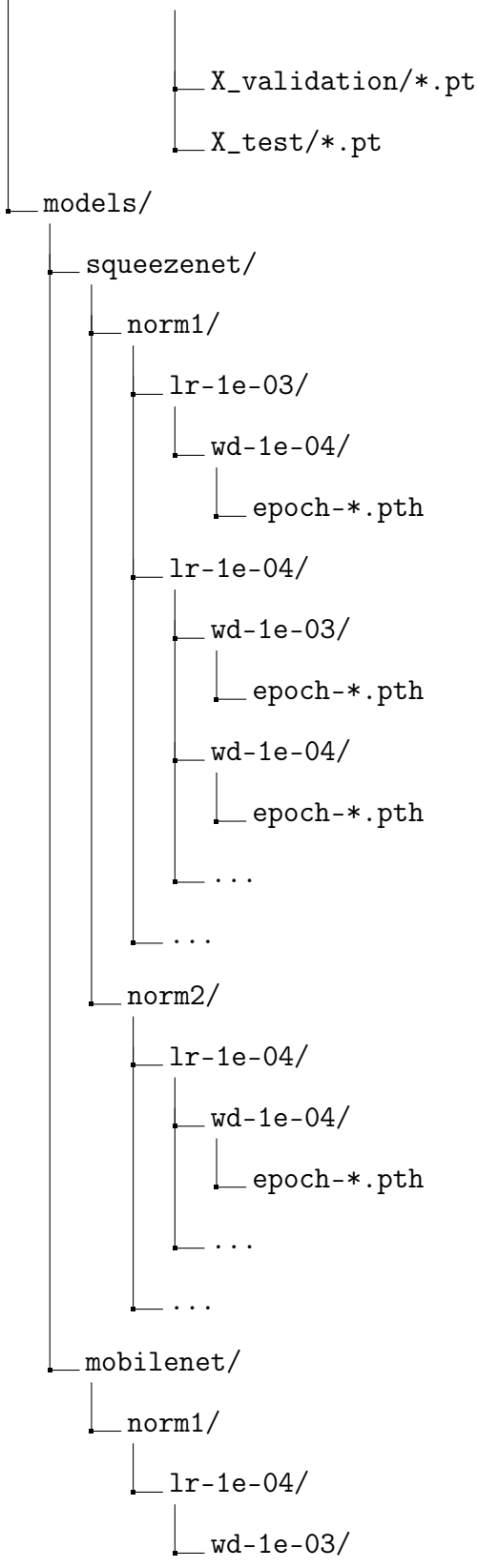
[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

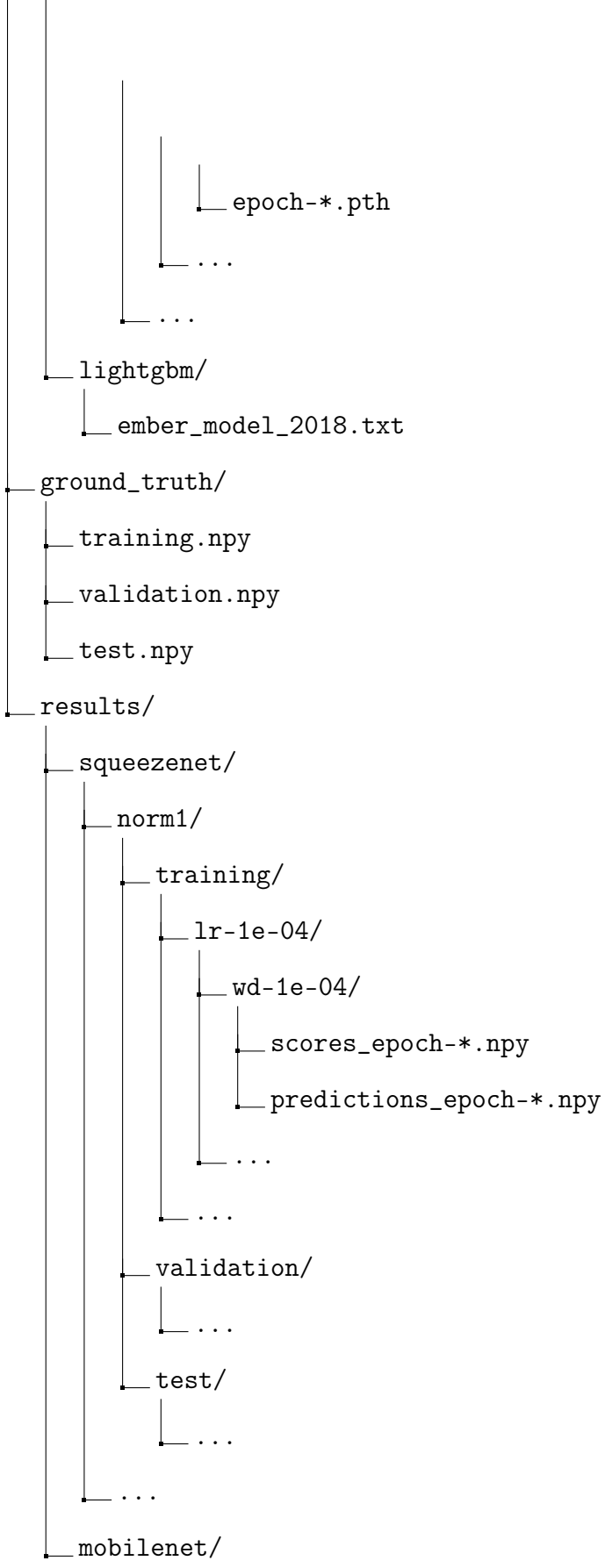
APPENDIX A

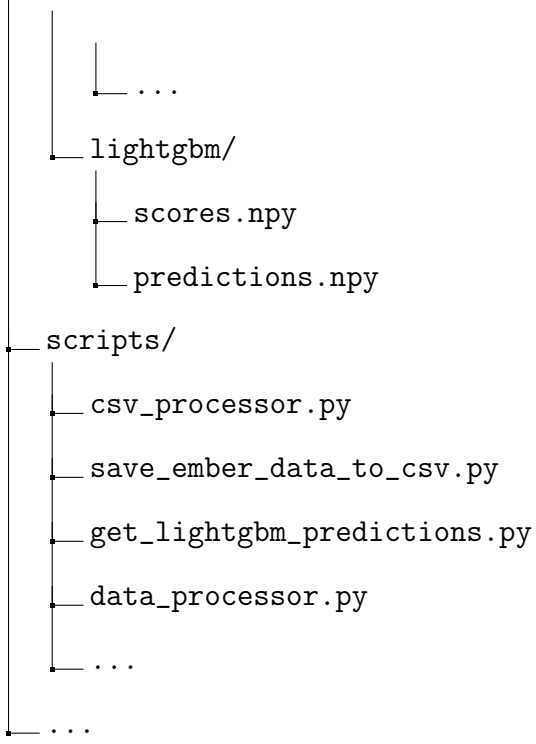
Experiment Setup

The codebase for conducting the experiments was organized to maintain a clear and modular structure. The project directory was structured as follows:

```
project_root/
└── data/
    ├── ember2018/
    │   ├── X_train.csv
    │   ├── X_test.csv
    │   ├── y_train.csv
    │   └── y_test.csv
    └── tsne_images/
        ├── ndarrays/
        │   ├── norm1/
        │   │   ├── X_train_batch_*.npy
        │   │   └── X_test_batch_*.npy
        │   └── norm2/
        │       ├── X_train_batch_*.npy
        │       └── X_test_batch_*.npy
        └── tensors/
            ├── norm1/
            │   ├── X_train_batch_*/*.pt
            │   ├── X_validation/*.pt
            │   └── X_test/*.pt
            └── norm2/
                └── X_train_batch_*/*.pt
```


X_validation/*.pt
X_test/*.pt
models/
squeezenet/
norm1/
lr-1e-03/
wd-1e-04/
epoch-*.pth
lr-1e-04/
wd-1e-03/
epoch-*.pth
wd-1e-04/
epoch-*.pth
...
...
norm2/
lr-1e-04/
wd-1e-04/
epoch-*.pth
...
...
mobilenet/
norm1/
lr-1e-04/
wd-1e-03/


epoch-*.pth
...
...
lightgbm/
ember_model_2018.txt
ground_truth/
training.npy
validation.npy
test.npy
results/
squeezenet/
norm1/
training/
lr-1e-04/
wd-1e-04/
scores_epoch-*.npy
predictions_epoch-*.npy
...
...
validation/
...
test/
...
...
mobilenet/



The experiments were conducted using the Python programming language. The following libraries and frameworks were utilized:

The Python programming language was the foundation for the entire project. In addition to PyTorch for training image-classification models (SqueezeNet and MobileNet), the following libraries and frameworks played crucial roles:

- EMBER: Used for dataset loading and preparation, easing the process of accessing and partitioning the EMBER dataset.
- LightGBM: Used to implement the benchmark non-image classification model.
- pyDeepInsight: Employed for generating t-SNE images, providing insights into the distribution of features within the dataset.
- NumPy and Pandas: Utilized for data manipulation and management, crucial for efficient data processing.

- Matplotlib and Seaborn: Employed for data visualization, enabling the creation of various plots, curves, and matrices.
- Scikit-learn: Utilized for computing metrics, generating confusion matrices, and assisting with model evaluation.

This modular and flexible codebase structure allowed for straightforward experimentation with different models, hyperparameters, and normalization techniques. The directory organization ensured clarity in managing data, models, results, and scripts, contributing to efficient experimentation and comprehensive analysis.

APPENDIX B

Sample t-SNE Images

Additional t-SNE image comparisons for a comprehensive analysis of the distinct color variations in the t-SNE patterns associated with various malware families and benign files are provided in this appendix. These visualizations offer valuable insights into the differentiation between malware and benign samples based on the 3-channel color of each pixel in the t-SNE pattern for each normalization technique.

These additional visualizations further emphasize the role of normalization techniques in shaping the t-SNE patterns and contributing to the accuracy of the image-based classification models.

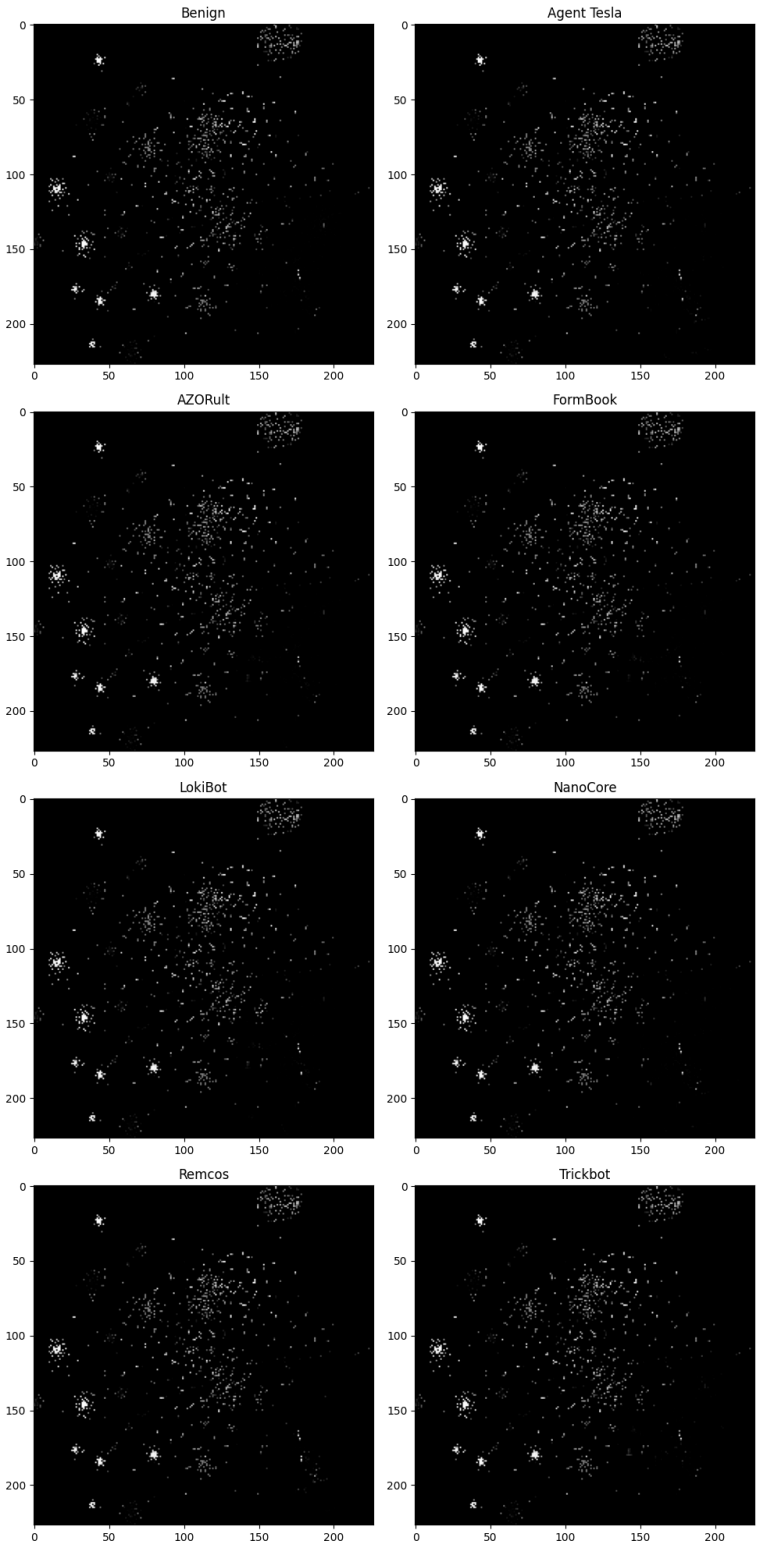


Figure B.10: Training Set Samples - norm-1

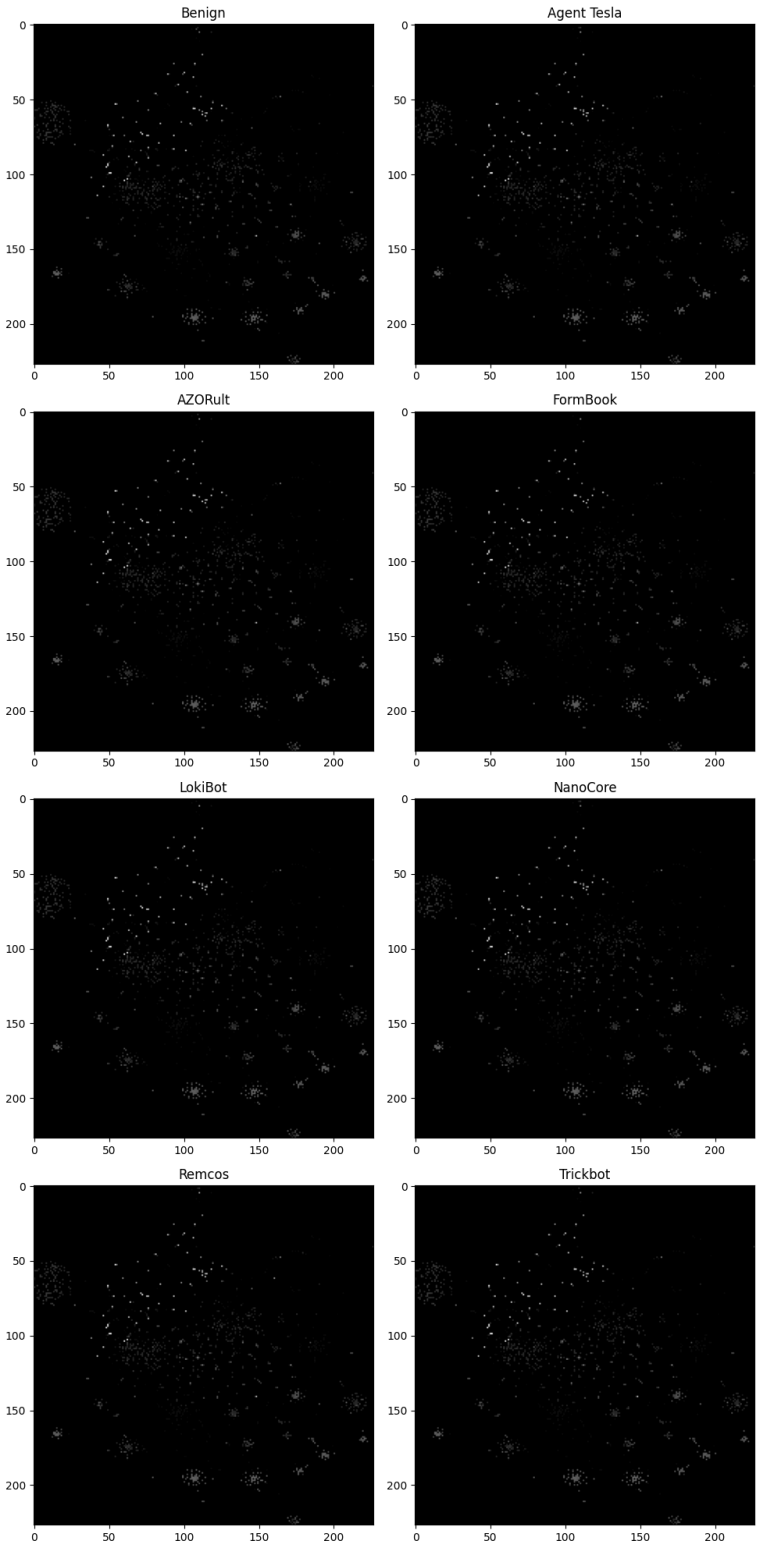


Figure B.11: Training Set Samples - norm-2