
KADİR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
PROGRAM OF MANAGEMENT INFORMATION SYSTEMS

# RANDOM CAPSULE NETWORK (CAPSNET) FOREST MODEL FOR IMBALANCED MALWARE TYPE CLASSIFICATION TASK

AYKUT ÇAYIR

PH.D. THESIS

Submitted to the School of Graduate Studies of
Kadir Has University in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Management Information Systems

İSTANBUL, JULY, 2021

DECLARATION OF RESEARCH ETHICS /
METHODS OF DISSEMINATION

I, AYKUT ÇAYIR, hereby declare that;

- this Ph.D. thesis is my own original work and that due references have been appropriately provided on all supporting literature and resources;
- this Ph.D. thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- I have followed *Kadir Has University Academic Ethics Principles prepared in accordance with The Council of Higher Education's Ethical Conduct Principles.*

In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations.

Furthermore, both printed and electronic copies of my work will be kept in Kadir Has Information Center under the following condition as indicated :

✓ The full content of my thesis will be accessible only within the campus of Kadir Has University.

AYKUT ÇAYIR
28.07.2021

KADİR HAS UNIVERSITY

SCHOOL OF GRADUATE STUDIES

# ACCEPTANCE AND APPROVAL

This work entitled RANDOM CAPSULE NETWORK (CAPSNET) FOREST MODEL FOR IMBALANCED MALWARE TYPE CLASSIFICATION TASK prepared by AYKUT ÇAYIR has been judged to be successful at the defense exam on 28.07.2021 and accepted by our jury as Ph.D. thesis.

APPROVED BY:

Prof. Dr. Hasan Dağ (Advisor) .....
Kadir Has University

Prof. Dr. Mustafa Bağrıyanık .....
Istanbul Technical University

Prof. Dr. Berk Canberk .....
Istanbul Technical University

Assist. Prof. Dr. E. Fatih Yetkin .....
Kadir Has University

Assist. Prof. Dr. Oğuzhan Ceylan .....
Kadir Has University

I certify that the above signatures belong to the faculty members named above.

.....

Prof. Dr. Timur Aydemir

Dean of School of Graduate Studies

DATE OF APPROVAL: 28.07.2021

# TABLE OF CONTENTS

RANDOM CAPSULE NETWORK (CAPSNET) FOREST MODEL FOR IMBALANCED MALWARE TYPE CLASSIFICATION TASK

# ABSTRACT

Behavior of malware varies depending the malware types, which affect the strategies of the system protection software. Many malware classification models, empowered by machine and/or deep learning, achieve superior accuracy for predicting malware types. Machine learning-based models need to do heavy feature engineering work, which affects the performance of the models greatly. On the other hand, deep learning-based models require less effort in feature engineering when compared to that of the machine learning-based models. However, traditional deep learning architectures' components, such as max and average pooling, cause architecture to be more complex and the models to be more sensitive to data. The capsule network architectures, on the other hand, reduce the aforementioned complexities by eliminating the pooling components. Additionally, capsule network architectures based models are less sensitive to data, unlike the classical convolutional neural network architectures. This thesis proposes an ensemble capsule network model based on the bootstrap aggregating technique. The proposed method is tested on two widely used, highly imbalanced datasets (Malimg and BIG2015), for which the-state-of-the-art results are well-known and can be used for comparison purposes. The proposed model achieves the highest F-Score, which is 0.9820, for the BIG2015 dataset and F-Score, which is 0.9661, for the Malimg dataset. Our model also reaches the-state-of-the-art, using 99.7% lower the number of trainable parameters than the best model in the literature.

**Keywords:** Capsule networks, Malware, Ensemble model, Deep learning, Machine learning.

DENGESİZ SINIF DAĞILIMINA SAHİP KÖTÜ AMAÇLI YAZILIM SINIFLANDIRMA GÖREVİ İÇİN RASSAL KAPSÜL AĞI (KAPSAĞ) ORMAN MODELİ

# ÖZET

Kötü amaçlı yazılımın davranışı, sistem koruma yazılımının stratejilerini etkileyen kötü amaçlı yazılım türlerine bağlı olarak değişir. Yapay ve/veya derin öğrenme ile güçlendirilmiş birçok kötü amaçlı yazılım sınıflandırma modeli, kötü amaçlı yazılım türlerini tahmin etmek için üstün doğruluklar elde eder. Yapay öğrenme tabanlı modeller performanslarını büyük ölçüde etkileyen ağır öznitelik mühendisliği çalışmalarına ihtiyaç duyarlar. Öte yandan, derin öğrenme tabanlı modeller, yapay öğrenme tabanlı modellere kıyasla öznitelik mühendisliğine daha az ihtiyaç duyarlar. Bununla birlikte, geleneksel derin öğrenme mimarilerinin maksimum ve ortalama havuzlama gibi bileşenleri, mimarinin daha karmaşık olmasına ve modellerin verilere daha duyarlı olmasına neden olur. Kapsül ağ mimarileri ise havuzlama bileşenlerini ortadan kaldırarak yukarıda bahsedilen karmaşıklıkları azaltır. Ek olarak, kapsül ağ mimarisi tabanlı modeller, klasik evrişimli sinir ağı mimarilerinin aksine verilere daha az duyarlıdır. Bu tez, rastgele örnekleme toplama tekniğine dayalı bir topluluk kapsül ağı modeli önermektedir. Önerilen yöntem, en son sonuçlarının iyi bilindiği ve karşılaştırma amacıyla kullanılabilecek, yaygın olarak kullanılan, oldukça dengesiz iki veri kümesi (Malimg ve BIG2015) üzerinde test edilmiştir. Önerilen model, BIG2015 veri kümesi için 0.9820 olan en yüksek F-Skoruna ve Malimg veri kümesi için 0.9661 olan F-Skoruna ulaşmaktadır. Modelimiz aynı zamanda literatürdeki en iyi modele göre $\%99,7$ daha az eğitilebilir parametre kullanarak en son teknolojiye ulaşmaktadır.

**Anahtar Sözcükler:** Kapsül ağlar, Kötü amaçlı yazılım, Topluluk model, Derin öğrenme, Yapay öğrenme.

# ACKNOWLEDGEMENTS

I would like to express my deep sense of thanks and gratitude to my supervisor Prof. Dr. Hasan Dağ, Director of Center for Cybersecurity and Critical Infrastructure Protection, Kadir Has University, for his contributions and mentorship to my thesis. His scholarly advice and scientific approach have helped me to a great extent to accomplish this task.

I owe a great sense of gratitude to Işıl Yenidoğan Dağ, Department of Management Information Systems, Kadir Has University for her technical supports and her valuable contributions to my academic life.

I thank Uğur Ünal, my colleague at Cybersecurity and Critical Infrastructure Protection, Kadir Has University for his worthy contributions to implement the user interface of the WARNING application.

I would like to thank my best friend Mustafa Alp Çolakoğlu for his support during my thesis when I was bored.

I am incredibly thankful to my parents, Cihat and Zuhal Çayır, and my elderly brother İbrahim Çayır for their lifelong supports, suggestions, and guidance for my life.

The last but not the least, it is my privilege to thank my wife, Mrs. Hilal Aydın Çayır, for her extremely valuable support during my thesis study.

To my parents and my lovely wife

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS/ABBREVIATIONS

| | |
|---|---|
| $\ell_m$ | Margin loss |
| $\ell_r$ | Reconstructed loss |
| $\lambda$ | Constant value which is 0.5 |
| $L$ | Average loss |
| $L_c$ | Loss for class or capsule $c$ |
| $m$ | Constant value which is 0.9 |
| $N$ | Number of samples |
| $s_i$ | Input vector of $i_{th}$ capsule |
| $v_i$ | Output vector of $i_{th}$ capsule |
| $x_c$ | Real sample for class or capsule $c$ |
| $\hat{x}_c$ | Reconstructed sample for class or capsule $c$ |
| $y_c$ | Ground truth for class or capsule $c$ |
| $\hat{y}_c$ | Prediction for class or capsule $c$ |
| API | Application Programming Interface |
| APK | Extension for Android Application Package |
| ASM | Extension or type of assembly files |
| AUC-ROC | Area under curve - Received operating curve |
| BERT | Bidirectional encoder representations from transformers |
| BiGRU | Bidirectional gated recurrent unit |
| BiLSTM | Bidirectional long short term memory |
| BIG2015 | Microsoft malware dataset published in 2015 |
| BYTE | Extension or type of byte files |
| CapsNet | Capsule network |
| CNN | Convolutional neural network |
| CPU | Central processing unit |
| DoS | Denial of service |
| FN | False negative |
| FP | False positive |
| GAN | Generative adversarial network |

| | |
|---|---|
| GIST | Global image features |
| GLCM | Grey level co-occurrence matrix |
| GPU | Graphical processing unit |
| IMCEC | Image-based malware classification using ensemble of CNN architectures |
| IMCFN | Image-based malware classification using fine-tuned convolutional neural network architecture |
| LSTM | Long short term memory |
| Malimg | The dataset from grey scale malware images |
| Malware | Malicious Software |
| MVC | Model-view-controller |
| Opcode | Operation code |
| PE | Portable Executable |
| RCNF | Random capsule network forest |
| ReLU | Rectified linear unit |
| ROC | Received operating curve |
| SFTA | Segmentation-based fractal texture analysis |
| SHA256 | Secure hash algorithm 256 |
| TP | True positive |
| VGG16 | Visual Geometry Group's Deep Learning Model With 16 Layers |

# 1. INTRODUCTION

Malicious software (malware) type classification is as important as malware detection problem because system protection software make their strategies based on malware family types. Malware families have different behaviors and effects on a computer system. Each malware family uses different resources, files, ports, and other components of operating systems. For example, malware in an online banking systems aim to perform fraud, steal private information of users, and use different spreading behaviors (Etaher et al. 2015, Azab et al. 2016). In addition to this, due to the trends in technology, new malware types occur almost daily. Thus, most of the computers, smartphones, wireless sensor networks, and other digital systems are vulnerable to new malware and denial of service (DoS) attacks (Benzaid et al. 2016). In this case, many zero-day attacks are performed (Alazab et al. 2013). The raising of the number of malware makes using the big data techniques crucial for malware analysis (Tang et al. 2017).

Malware type classification is the most common problem in the cybersecurity domain, because strategies of protection systems vary with respect to malware family type. Malware type classification problem is broadly dealt with in three different ways: static, dynamic, and image-based (Nataraj, Yegneswaran, Porras & Zhang 2011, Abijah Roseline et al. 2019, Ni et al. 2018). This thesis focuses on the image-based malware family classification problem. However, malware family type classification is an imbalanced task, so this makes many models unsuccessful at predicting the rare classes. To this end, two imbalanced datasets are used and the results are compared to the other models in the literature.

Image-based malware classification is a broad research and application area. At the

same time, deep learning drives the computer vision and image processing researches to this field. Many deep convolutional neural networks have proven their success in image processing. CapsNet is the most important deep convolutional neural architecture that removes pooling to avoid losing the spatial features of images. This is the power of CapsNet comparing to classical CNNs. Therefore, the number of applications of CapsNet is increasing in image processing. The main motivation of this thesis is to design a simple and accurate classifier for imbalanced malware type classification problem using bagging (Breiman 1996*a*, 2001) and CapsNet architecture. This thesis also presents a detailed comparison of the proposed model with the other models in the literature.

This thesis proposes a new model named **R**andom **C**apsule **N**etwork **F**orest (RCNF) based on bootstrap aggregation (bagging) ensemble technique and capsule network (CapsNet) (Breiman 1996*a*, Sabour et al. 2017). The main motive behind the proposed method is to reduce the variance of different CapsNet models (as weak learners) using bagging. In this perspective, the main contributions of this thesis can be listed as follows:

- The thesis introduces the first application of CapsNet in the field of malware type classification. Although image-based malware classification is a broad research and application area, there is no research and application of CapsNet in the literature to our knowledge.
- The thesis uses the first ensemble model of CapsNet. The key idea of creating an ensemble of CapsNet is assuming a single CapsNet model as a weak classifier like a decision tree model. In this way, an ensemble model of CapsNet can be easily created using bootstrap aggregating. The main assumption that CapsNet is a weak learner increases the performance of a single CapsNet for two different well-known malware datasets, which are highly imbalanced.
- The proposed model uses simple architecture engineering, which means finding a better neural network structure instead of complex convolutional neural network architectures and domain-specific feature engineering techniques. In

addition to this, CapsNet does not require the usage of transfer learning, and the model is easily trained from scratch. Because of that, the created network and its ensemble version have reasonably lower number of parameters. The proposed model obtains F-Score and accuracy, which are close to the-state-of-the-art results, using 99.7% lower the number of trainable parameters than the best model in the literature.

- The proposed model is compared with the latest studies that use deep neural networks for image-based malware classification tasks. For a fair comparison, especially, the last studies using the Malimg and the BIG2015 datasets are chosen and compared with the proposed method.
- The proposed method is reproducible and broadly simpler than other complex deep neural network architectures regarding the number of trainable parameters.
- The thesis presents a concrete application, which is web-based and supports multi-users and multi-requests at the same time, starting from the experimental design of the proposed model.

The thesis is organized as follows. Section 2 presents a literature survey for CapsNet applications and previous malware analysis studies. The methodology of the thesis is described in Section 3, whereas Section 3.2 gives details of the inspiring model and the proposed model. In Section 4, the test results are discussed and many comparisons with related work published within the last couple of years are listed. Section 5 argues the limitations of the proposed model and its implementation for the application. Section 6 presents the papers that contribute to the thesis, and finally, Section 7 provides the concluding remarks.

# 2. RELATED WORK

Machine learning is the core discipline that is placed at the heart of the malware classification problem. Thanks to the methods in machine learning, there are efficient and novel approaches that can process datasets that are high dimensional and contain thousands of samples. However, traditional machine learning methods require powerful, complex, and intense feature engineering or pre-processing steps to transform the raw data into a valuable input for the model. For this reason, traditional machine learning models for malware classification focus on the-state-of-the-art feature engineering or pre-processing techniques in the research domain. On the other hand, unlike machine learning, deep learning models focus on creating different architectures that can process raw input from the dataset requiring less pre-processing than complex feature engineering methods. The tools designed for deep learning architectures allow the researchers to develop ensemble, hybrid, or multi-modal models quickly. Therefore, this section of the thesis investigates some important papers and studies, which are the latest in the malware family classification and detection research area. Subsection 2.1 lists some conventional machine learning approaches in malware classification. Subsection 2.2 presents a new approach based on graph convolutional neural networks, LSTM networks, and transformers using API calls from executable malware files to classify. Subsection 2.3 inspects the most common deep learning approaches such as convolutional neural networks and transfer learning based models for grey-scale malware images for malware classification. The latest hybrid, ensemble, and multi-modal machine and deep learning models are presented in Subsection 2.4. Imbalanced malware datasets are the most common in a malware classification problem, so Subsection 2.5 exhibits the latest papers in imbalanced data handling. To point out the lack of capsule network application for malware classification, Subsection 2.6 presents a brief review for mod-

ified and specialized capsule network architectures in some research problems such as computer vision, text classification, and time series analysis. Lastly, Subsection 2.7 summarizes the related work section with selected papers from subsections.

## 2.1 Traditional Machine Learning Methods

There are many different ways to represent malware files for machine learning-based identification (Alazab & Tang 2019). One of them is to extract features from the application programming interface (API) calls of malware. For example, Alazab (2015) proposes a framework to get features statically and dynamically from malware API calls. He uses similarity mining and machine learning to profile and classify malware. He obtains a 0.966 receiver-operating-curve (ROC) score in the malware dataset containing $66,703$ samples (Malign or Benign) with the k-nearest neighbors algorithm. Moreover, Azab et al. (2014) focus on grouping malware in the same variants using hashing techniques for malware binaries. They use two different Zeus datasets. The first dataset contains 856 binaries, and the second dataset contains 22 binaries. Each binary has a SHA256 value. They achieve a 0.999 F-Score using the k-nearest neighbors and SDHASH. Some of the essential conventional machine learning applications for malware classification are investigated in Subsection 2.4 in detail.

Some papers in the literature aim to find the best combinations of pre-processing or feature extraction methods and conventional machine learning algorithms, whose classification performance is highly dependent on hand-crafted feature extraction. Iadarola et al. (2020) give a brief assessment between feature extraction and classification methods for image-based malware family detection. This paper compares different feature vector extraction methods and especially traditional machine learning models. The paper uses feature extraction methods such as Gabor filter, color layout filter, auto color correlogram filter, and global image descriptor (GIST). The machine learning models used in the paper are the k-nearest neighbors classifier, random forest classifier, J48 classifier, and decision table classifier. They have eval-

uated all combinations with a dataset containing $20,748$ samples and 10 different families. Furthermore, they have built a random forest using GIST descriptors, and they have obtained the highest average accuracy, which is $96,9\%$. However, there are some missing parts in the paper. For example, there is no assessment of multi-layer perceptrons or deep learning models, and the number of models evaluated in the paper is not enough to find the best combination of feature extraction methods and models. On the other hand, they have pointed out the obfuscation is the most critical issue for malware family classifiers, and they express that they are planning to test the robustness of a malware classifier using adversarial learning approaches in the literature.

## 2.2 Transformer, Graph Convolutional and LSTM Neural Networks

One of the newest representation approaches is creating graph structure from API calls of malign and benign executable files or source codes. Moreover, this representation approach allows a new deep learning framework called graph convolutional networks to learn node or edge embeddings. Pei et al. (2020) introduce a new framework based on graph convolutional networks. They have combined a graph convolutional network and recurrent neural networks in the proposed model called AMalNet. The recurrent network part of this model learns word embeddings for word, character, and lexical features from permissions, application components, and suspicious API calls from Android APK files. For the graph convolutional part of the AMalNet model, they have created a graph representation for each malware from permissions, components, or APIs. The paper has used five different malware datasets, and two of them are multi-class datasets. The most interesting part of the paper is sustainability analysis, and this part presents a detailed table that shows the proactive behavior of the AMalNet for zero-day attacks. However, they have not specified the total number of trainable parameters of the AMalNet model, although they present execution time analysis for each dataset. For this reason, the model is not available to compare with other deep learning architectures in terms of the total number of trainable parameters.

Graph representation of malware files created from API calls can provide us to develop a behavioral malware classification model. Schranko de Oliveira & Sassi (2019) have collected a dataset that has goodware (or benign) and malware samples and contains API call sequences belong to each sample. The dataset has 42,797 malware samples and 1,079 goodware samples. The class distribution of the dataset shows that the dataset is highly imbalanced for a binary classification problem. They introduce a graph convolutional neural network and generate a behavioral graph representation for their dataset. Because of skewness between goodware and malware classes in the dataset, they evaluate their proposed model on two versions of the dataset: balanced and imbalanced versions. They also compare their proposed models to a dummy model that predicts all samples as malware, in F-score, AUC-ROC, precision, recall, and accuracy. According to their findings, their graph convolutional neural network model that learns behavioral graph representations has reached significant results for the binary malware classification problem.

Although graph convolutional neural networks are efficient and useful for graph-like datasets, it is possible to represent graphs as sequential features such as API calls and opcodes extracted from malware files. One of the papers based on this idea is done by Niu et al. (2020). They present a new deep learning model to classify Android malware samples using opcode features. In order to extract sequential opcodes from Android DEX files, they have parsed function calls graphs. They have built two different models: long short term memory-base (LSTM-based) model and support vector machine model. For the proposed support vector machine model, a skip-gram method is used to represent opcodes. For the proposed LSTM-based model, time distributed embedding layer is used to learn dense representations of opcodes. They have evaluated their proposed models with a dataset containing 1,796 Android application files, and this dataset has three distinct labels such as Trojan, Adware, and Normal. They report that the LSTM-based model achieves 97% accuracy, and the support vector machine model obtains 95% accuracy on their dataset. Furthermore, they have pointed out that their LSTM-based model has a 6M trainable parameter which is the minimum among the competitor models in their

paper. Some graph-like datasets for malware classification can be reformulated as a sequential data classification problem in this perspective.

Transformers are one of the groundbreaking models used in deep learning (Vaswani et al. 2017). Especially, transformers are very successful for natural language problems. On the other hand, there are many variations for transformers for different tasks. For example, malware classification from application programming interface calls, function calls, or opcodes can be formulated as a sequence-based classification problem like sentiment analysis or text classification. Therefore, transformers can be efficient models for text-based malware classification. Rahali & Akhloufi (2021) introduce an early example study that uses bidirectional encoder representations from transformers (BERT) for text-based malware classification. They have also tested LSTM, XLNet, RoBERTA, and DistilBERT models in their paper. They have evaluated these models with a dataset containing $12,000$ benign and $10,000$ malware samples. The malware samples in the dataset have been collected from eleven unique malware families, so one part of the dataset containing malware samples can be used to classify malware families. The evaluation part of the paper used four different metrics: accuracy, F-score, loss, and the Matthews Correlation Coefficient (MCC) used for binary classification with an imbalanced dataset. Their results in the experiments show that the BERT model obtains 97.61% accuracy for the binary classification and 91.02% accuracy for the malware family classification problem.

### 2.3 Convolutional Neural Networks and Transfer Learning

Another efficient way to feed machine learning algorithms to classify malware files is image-based representation. For example, Nataraj, Karthikeyan, Jacob & Manjunath (2011) convert malware files to greyscale images to represent malware. They extract GIST features from malware images, and then they classify malware family types using the euclidean k-nearest neighbors algorithm. They reach 0.98 classification accuracy for the dataset with $9,339$ samples and 25 malware families. Similarly,

Kancherla & Mukkamala (2013) use image-based malware representation to feed the support vector machine to classify malware files as malign or benign. They extract three different features, such as intensity-based, wavelet-based, and Gabor-based. Their dataset contains $15,000$ malign and $12,000$ benign samples. They attain a 0.979 ROC score. These studies utilize traditional machine learning algorithms, such as k-nearest neighbors and support vector machines. These algorithms require to extract good features from images to classify malware types with high performance.

After an impacting success of a deep convolutional neural network (CNN) on the Imagenet dataset, a new era started in computer vision (Krizhevsky et al. 2012). CNNs could classify images using raw pixel values without complex feature engineering methods. This success has also started the new age for image-based malware classification using deep neural architectures and variations such as combination, ensemble, or hybridization. The latest paper by Hemalatha et al. (2021) benefiting from the advantages of deep learning architectures provides an efficient DenseNet architecture-based malware detection model for binary malware images. Their model uses weighted loss to deal with imbalanced classification problems. They have tested the proposed model on five different malware datasets such as BIG2015, Malimg, MaleVis, and Malicia.

There are several methods to create greyscale malware images from values rather than byte pixel values of malware binary files in the literature. For example, Ni et al. (2018) create greyscale image files using SimHash bits of malware. They obtain 99.26% accuracy on the dataset containing $10,805$ samples using a CNN classifier.

Another work by Yuan et al. (2020) create malware Markov images from malware binary files. For this image representation, the pixel values of malware Markov images are byte transfer probabilities different from the actual value of bytes. They convert this byte transfer probabilities to $256 \times 256$ greyscale image for each malware binary file. Thus, these images can be directly used in a convolutional neural network

as input. They have proposed a five convolutional layered CNN to classify the BIG2015 and the DREBIN datasets. They have also emphasized that these datasets are highly imbalanced, and they are reporting recall, precision, and F1-score to evaluate the performance of their model. Finally, they present the classification performances of the proposed model for different proportions of training and testing of the BIG2015 and the DREBIN datasets.

One of the exciting papers by Xiao et al. (2020) introduces a combination of a convolutional neural network and support vector machine, which is a well-known classifier in machine learning. Their model called MalFCS is a static malware analysis method based on greyscale image representation of malware samples. They point out that the methods in traditional machine learning need much time, and their success depends on feature engineering methods that have been used. Therefore, they build a convolutional neural network to extract high-level features from raw pixel values of greyscale malware images. Their convolutional neural network structure contains thirteen sequential convolutional layers and one fully connected layer at the end. They train the convolutional neural network, and it learns the high-level features. Finally, they have transformed the grayscale malware images to input for support vector classifier using the pre-trained convolutional neural network. They have tested the MalFCS model on the BIG2015 and the Malimg datasets, and they have evaluated the model using F-score, kappa metric, and accuracy because they indicate these two datasets are highly imbalanced. They report 0.9991 F-score and 0.9967 kappa metric for 10-fold cross-validation the Malimg dataset and 100% accuracy for 10-fold cross-validation of the BIG2015 dataset.

Transfer learning is a crucial method to increase convolutional neural networks' generalization capacity for datasets that have relatively small sample sizes. Generally, transfer learning uses pre-trained models in two ways: feature extraction and fine-tuning. For the fine-tuning part, some shallow features are typical for different tasks. For this reason, low-level weights of some models are used to initialize the weights of new or proposed neural networks for new tasks like greyscale malware

image classification. Therefore, transfer learning positively affects the generalization performance of the proposed neural network because of weights initialization via a pre-trained network rather than random initialization. Zhao et al. (2020) proposes a convolutional neural network based on a Faster RCNN model combining transfer learning via fine-tuning. They have used the BIG2015 dataset to evaluate their model. However, they have removed the Simda, the rarest malware family in the dataset, because it has 42 samples. They have trained two different models. The first Faster RCNN model has been trained from scratch, and the second model has been trained with pre-trained weights of the Faster RCNN model on the ImageNet dataset. In order to show that the effect of the transfer learning approach on the malware classification task, they have compared these two models in terms of accuracy, detection rate, and false-positive rate. They report 92.8% accuracy, 95.6% detection rate, 6.8% false positive rate, and 85 minutes training time. Moreover, drawing a bounded box onto malware images according to their family types is one of the valuable features of this model. This model's ability provides valuable information on which part of the texture is essential for which malware family.

Vu et al. (2020) use a novel approach, called HIT4Mal, extracting features from raw files to get image representations of malware. The idea behind the paper is to feed a convolutional neural network with a more complex image transformation that makes the model robust to obfuscation. According to their paper, existing simple transformations have not considered color encoding and pixel rendering on the classifier's performance. HIT4Mal contains two phases. The first phase is to apply to PE files four different byte encoding techniques such as byte-class, grayscale, gradient, and entropy. In the second phase, these encoded representations are converted to malware images using combinations of four different byte layout methods following carriage return, wrap-around, Hcurve, and Hilbert. At the end of the HIT4Mal system, a convolutional neural network takes the images produced by the best byte encoding and byte layout methods as input set. They have tested the HIT4Mal model on the dataset containing $8,000$ malware and $8,000$ benign samples. The paper shows that the best model contains combinations of byte-class as byte encoding

method, Hilbert as byte layout method, and a convolutional neural network. This configuration of HIT4Mal achieves 93.01% accuracy, which is higher than the simple image transformations such as byte-class, greyscale, gradient, and entropy.

In transfer learning, using different pre-trained convolutional neural networks simultaneously to extract features from greyscale malware images provides to learn different representations for malware classification. This variety in representations can increase the generalization capacity of a model. Based on this idea, Aslan & YILMAZ (2021) leverage a deep learning model that uses two different pre-trained convolutional neural networks as feature extractors. Their proposed model has two branches to extract features from greyscale malware images. One of these branches is AlexNet, and the other is ResNet152. These pre-trained networks extract two different feature representations for a given greyscale malware image sample. After that, the model merges these two representations and redirects to three sequential fully connected layers whose dimensions are $4,096$, and each fully connected layer contains a ReLU activation function and batch normalization layer. The final layer is the softmax layer. They have evaluated the proposed model with the BIG2015, Malimg, and Malevis datasets. They have split these three datasets into three parts: 70% of samples for training sets, 10% of samples for validation sets, and 20% of samples for test sets. They have used accuracy, sensitivity, specificity, and F-score as performance metrics. Finally, the proposed model obtains 97.78% accuracy for the Malimg dataset, 94.88% accuracy for the BIG2015, and 96.5% accuracy for the Malevis dataset. One of the weaknesses of their system is that the system requires 30 hours for the training phase without GPU.

## 2.4 Hybrid, Ensemble, and Multi-modal Models

Ganaie et al. (2021) inspect ensemble deep learning models in different domains and some ensembling strategies in the literature. Their study points out some future studies and research directions on deep ensemble learning models. One of the most developing research domains that the study does not cover is malware classification

or detection, where ensemble machine or deep learning models are frequently used.

Abusitta et al. (2021) put forth a survey containing recent developments in malware detection and classification research area. They also propose a new malware classification taxonomy based on algorithms and features. In this taxonomy, the features branch is divided into two parts: feature extraction and feature type. Feature extraction contains three different parts such as static, dynamic, and hybrid. Feature types are listed, such as system calls, network, byte sequences, file system, printable strings, assembly codes, the dynamic linked library function calls, and functions control graphs. Besides, the algorithms part is divided into two parts: the signature based approach and the artificial intelligence based approach. The paper expresses that the hybrid method is applied to achieve higher success in malware classification by combining static and dynamic features. The paper also addresses some critical research gaps in the literature, such as robust solutions, collaborative solutions, and sustainable solutions.

Last but not least, they characterize each reviewed paper concerning both algorithms and features used and highlight their strengths and limitations. In this perspective, making hybrid or ensemble models in malware classification is the most common trick to increase the generalization capacity of artificial intelligence based approaches. Therefore, the thesis focuses on creating a novel ensemble of CapsNet based on bagging methods like random forest classifier with great inspiration.

There are various examples of ensemble methods for malware classification in the literature. For instance, Liu et al. (2017) combine features from gray-scale malware images, n-gram of opcodes, and function imports from malware source files. For this reason, this study is also a good example of a hybrid approach of image-based and sequence-based malware classification. They use a dataset, which has 9 different malware types, collected from three sources, and the dataset has more than $20,000$ malware samples. They propose a decision-making system for malware classification based on bootstrap aggregating ensemble method that uses classification methods

such as decision tree, K-nearest neighbor, random forest, gradient boosting classifier, naive Bayes, logistic regression, and support vector machine (with the polynomial kernel). They make an ensemble of these classifiers using majority voting. They report that their decision-making model has 98.9% for their test set, and the model can successfully classify new samples with 86.7% accuracy. Although their study is an important example of an ensemble model for malware classification, the model heavily depends on classical feature extraction methods because they use traditional machine learning models instead of deep learning architectures.

Pektaş & Acarman (2017) make an ensemble model for a hybrid version of static features such as permissions, hidden payloads, and dynamic features such as application programming interface calls, installed services, network connections extracted from Android application package files. They obtain 92% accuracy on their test set, but their sample size of the dataset is comparatively less than the BIG2015 and the Malimg. Their model also requires massive feature engineering because they ensemble models traditional machine learning models such as random forest, support vector machine, and logistic regression.

Fang et al. (2020) have conducted one of the newest studies that show an ensemble of different features or different models for Android malware types. They have used the Android malware dataset called AMD, which contains $24,553$ samples and 71 unique malware families. However, they have re-sampled from the original dataset, and the new dataset contains $3,000$ samples and 15 unique malware families. Their model takes three different feature sets extracted from raw files. The first is the texture feature set from RGB malware images using the GIST method. The second is the color feature set from RGB malware images using the color moments. The third is the text feature set from the DEX files using the simhash approach. They have fused those three feature sets using three kernelized support vector machine models, and they have taken the weighted average of the kernelized models. They report a 0.96 F-score on a holdout test set which is 20% of their dataset.

One of the latest studies for Android malware family classification with an ensemble model by Bakour & Ünver (2021) aims to develop a novel generic image-based malware classifier for Android malware families. In order to this, they have created five greyscale image datasets collected from Android malware sample sources. Each malware dataset contains $4,850$ greyscale malware images. They have extracted two types of feature sets for each dataset. The first type is local features, including scale-invariant feature transform, speeded-up robust features, ORB, and KAZE features. The second type is global features, including color histogram, Hu moments, and Haralick texture. They have trained six different machine learning models, including random forest, k-nearest neighbor, decision trees, bagging, Adaboost, and gradient boosting trees. In this way, they have hybridized an ensemble majority voting classifier using local and global features as voters. The proposed model has used stacking for global features and a bag of visual words for the local features. In addition to this, they also have tested residual networks and Inception-V3 networks on their datasets. Finally, they have compared the proposed model with the results of some the-state-of-the-art models in terms of classification accuracy, execution time, and generality. They have obtained 97.94% accuracy for their hybrid model, and they have attained on their datasets 96.34% and 95.12% classification accuracies using respectively ResNet and Inception-V3. They also report execution times of their model maximum 401seconds and minimum 342 seconds.

Making an ensemble model of traditional machine learning methods can reach the state-of-the-art results on some malware classification datasets with less feature dimension and running time. Verma et al. (2020) declare that building efficient models for imbalanced malware classification datasets is a great challenge. They also point out that the high feature dimensionality increases overhead for the machine or deep learning models. They aim to fill this gap by presenting a model for greyscale malware image representations for malware family classification. Their approach derives a novel combination of first-order and GLCM-based second-order statistical features for greyscale malware image representations. There are nineteen first-order features and eight second-order features extracted by them. These features are

stacked in a design matrix, and each row represents one malware sample in the dataset. They have obtained the state-of-the-art results using random forest, an ensemble model, in terms of F-score and accuracy on the Malimg dataset. They have reported 98.58% accuracy and 98.05% F-score for the Malimg dataset using only 35 features, which is the lowest dimension among four studies in their paper. Moreover, the feature extraction for this model takes 37 msec, and the training of the random forest that uses these features takes an average time of 0.001 sec.

Vinayakumar et al. (2019) also emphasize that the machine learning algorithms require extensive feature engineering. They construct a deep learning model to avoid performing extensive feature engineering, and they try to fill the gap for zero-day malware detection. In order to that, they have created train and test splits of public and private datasets collected in different timescales. They propose a novel deep learning architecture that uses static, dynamic, and image-based malware features. They combine convolutional layers with LSTM layers, and that architecture called DeepImageMalDetect has the less total number of trainable parameters among other models mentioned in their paper. One of the datasets in their paper is the Malimg dataset, and they obtain a 0.962 F-score on that dataset with DeepImageMalDetect. Last but not least, they present a holistic real-time malware analysis architecture that contains the DeepImageMalDetect model as a sub-deep neural network.

Azeez et al. (2021) present another essential example of an ensemble method that uses stacking for malware classification. Their stacked model contains two stages called base and final. The base stage contains a stacked ensemble of fully connected and one-dimensional convolutional neural networks, and they analyze 15 different machine learning classifiers for choosing the meta learner in the final stage. The results of the stacked ensemble model are obtained from a binary classification dataset which is a collection of Windows Portable Executable (PE) files. The dataset has 19,611 malware samples. They propose the best-stacked model with an ensemble of seven neural networks and the ExtraTrees classifier as the final stage classifier.

Korine & Hendler (2021) develop a distinctive novel approach, which aims to be a dataset-agnostic and platform-agnostic malware classifier. According to their study, their proposed system is also explainable at the same time. They have tested the proposed model on five different datasets to show the model is dataset-agnostic, which means the model gives high accuracy on any malware dataset. They have also shown that the model is platform-agnostic because it can classify malware files from Android and Windows operating systems. Their system called DAEMON contains five stages such as entropy threshold computation, family representative n-gram extraction, pairwise-separating feature selection, feature-vector computation, and random forest model generation. They have reached 99.72% accuracy, which is an excellent result for the dataset BIG2015, using the traditional machine learning model. However, this system requires dense and sequential feature-engineering steps from stages one to five, and the total running time for generating a DAEMON model is 16 hours. The RCNF model in the thesis has reached the-state-of-the-art results using lightweight pre-processing methods thanks to deep learning and CapsNet architecture.

Deep learning allows the researchers who study the malware classification problem to create novel models that can predict malware types from multi-modal feature sets without performing heavy feature engineering methods, unlike the ensemble and hybrid models that use machine learning methods mentioned before. For example, Gibert et al. (2020*a*) propose a multi-modal deep learning architecture for the BIG2015 dataset. They define three types of modalities from the raw files in the BIG2015 dataset. These are the list of Windows API function calls, the sequence of assembly language instructions from ASM files in the BIG2015 dataset, and the sequence of hexadecimal values from the BYTE files in the BIG2015 dataset. Their system called HYDRA contains three sub-neural networks as API-based component, byte-based component, and opcode-based component. A dense layer combines API-based and byte-based components in the intermediate fusion and classifier layer. Before the classification layer, another fully connected layer also combines the output of the previous dense layer with the opcode-based component, and the classifier

layer takes the output of the last dense layer as input. Thus, the model can predict malware types using a multi-modal structure. They report 98.71% accuracy and 0.9695 F-score without pre-training and modality dropout. In their paper, their highest results are 99.75% accuracy and 0.9954 F-score using pre-training and modality dropout.

Internet of things and future networks (e.g., 5G) need new security research because of the rapid improvements. One of the papers utilizing multi-modal deep learning architectures done by Dib et al. (2021) emphasizes to lack of sophisticated malware classification models for the internet of things and new generation networks. The paper benefits from features from strings and image-based representations of executable malware binaries to propose a novel multi-dimensional malware classifier using deep learning architectures. The proposed model contains two components: image-based and string-based. The image-based component has built a convolutional neural network that takes $128 \times 128$ greyscale malware image representations as input. The string-based component is a long short term neural network that takes string representations extracted from byte code files of malware samples as input. A fusion layer combines these two components following two sequential dense layers and a softmax layer called the classification layer. In order to evaluate their models, they have used a dataset containing more than $70,000$ recently detected malware samples from the internet of things platforms. They also offer a new label named unknown for newly detected malware samples which are the rarest in the dataset. Thus, the dataset seven unique malware family labels. Finally, they report 0.9978 accuracy and 0.9957 F-score for their multi-modal model.

Schranko de Oliveira & Sassi (2020) propose a novel ensemble malware classifier called CHIMERA for only Android platform, similar to DAEMON. CHIMERA has three sub-neural networks: CHIMERA-S, CHIMERA-R, and CHIMERA-D. Each sub-network learns from different representations; for example, CHIMERA-S is a fully connected neural network that uses static features from intents and permissions. CHIMERA-R is a convolutional neural network that uses greyscale images from DEX

files. Finally, CHIMERA-D is a transformer architecture that uses dynamic features from system call sequences. Thus, the CHIMERA system is an ensemble model of those three sub-networks via an intermediate fusion layer. In this perspective, the CHIMERA system is a hybrid malware classification model that can use static, dynamic, and image-based malware representations simultaneously. They have used a public benchmark dataset called Omnidroid, which contains $22,000$ malware and goodware samples. The dataset is balanced in terms of class distribution. In the paper, the CHIMERA system has the highest performance metrics for 10-fold cross-validation. Although its highest performance on a binary classification problem for the Android platform, there is no performance analysis of the CHIMERA system for family types of Android malware files.

This thesis also points that ASM and BYTE files can be more useful and distinguishable for malware type classification problems. Using ASM and BYTE files simultaneously for the BIG2015 dataset in the RCNF model is one of the unique ideas behind the thesis. In this perspective, this thesis is the earlier study that can process ASM and BYTE files at the same time to classify malware families. Li et al. (2021), the latest study citing our paper (Çayır et al. 2021), emphasizes the lack of studies that simultaneously process ASM and BYTE files to classify malware types. They have proposed a standard convolutional neural network architecture for the BIG2015 dataset. Arguably, they have removed all samples from the Simda family, which is the rarest and the most challenging family in the BIG2015, so they have reported their metrics for eight classes instead of nine.

The other work indicating feature fusion like this thesis by Zhu et al. (2021) introduces a new model that uses malware images created from byte codes and opcodes at the same time for malware homology determination. For this reason, their CNN architecture consists of dual convolutional branches in the feature extraction layer of the network. This study uses the BIG2015 dataset. They also investigate the effects of the size of the malware images on classification performance. According to their findings, the ideal image size is $64 \times 64$.

## 2.5 Imbalanced Dataset Handling Methods

Another aspect is to gather data for malware classification task. Malware datasets are imbalanced due to its nature. These are some examples to deal with imbalanced issues such as Ebenuwa et al. (2019) pointed imbalanced classification problem in binary classification. They inspected three different techniques, such as sampling-based, algorithm modifications, and cost-sensitive approaches. They proposed variance ranking feature selection techniques to get better results in imbalanced datasets for binary classification problems. Their findings indicate combinations of algorithms such as ensemble and bagging preferred solution for imbalanced datasets.

Data augmentation is a brilliant idea to increase the number of samples for training datasets containing images. Thanks to the data augmentation approach, many learning models can learn the rarest malware families. Source code obfuscation is one of the fundamental approaches for malware datasets. Marastoni et al. (2021) notice that augmenting the existing greyscale malware images is not helpful to classify the rarest families, and they propose a way to create augmented greyscale malware images using an obfuscator application from the source code. They have tested their proposed method on the BIG2015 and the Malimg datasets, similarly to the thesis. Although the proposed augmentation method is quite impressive, the method requires datasets containing source code files like BIG2015, and it is not applicable for existing greyscale malware images like the Malimg dataset.

Another data augmentation approach for imbalanced datasets is generating new samples using generative models or adding some noise to existing samples via statistical approaches. For example, Catak et al. (2021) suggest a new data augmentation method for three-channel malware image representations. Their system has three stages. The system converts malware files to RGB images for the first stage using decimal conversion, entropy conversion, and zeros. In the second stage, the system generates new three-channel malware images adding Gaussian, Poisson, and Laplace noises to existing three-channel malware images obtaining from the first

stage. Finally, they train a convolutional neural network using augmented and original three-channel malware images in the final stage of their proposed system. They have used the dataset collected by them, and they have published the dataset for public usage.

Some critical studies show that transfer learning and data augmentation efficiently overcome the imbalanced dataset problem in malware classification. For instance, Nisa et al. (2020) suggest a novel hybrid model for imbalanced malware classification. The proposed model contains data augmentation, image resizing, feature extraction, feature fusion, dimension reduction using principal component analysis and classification steps. In the data augmentation part, they express that the augmentation method must avoid modifying the texture topologies in malware classes because of using a method called segmentation-based fractal texture analysis (SFTA) in the feature extraction step. Thus, they have used rotation, scaling, and flipping for the data augmentation part of the proposed model. In the image resizing part, they have resized greyscale malware images to $128 \times 128$ for SFTA, $227 \times 227$ for AlexNet, and $229 \times 229$ for InceptionV3. In the feature extraction part, there are three different methods. One of them is SFTA, and it returns an 87 dimensional feature vector for each greyscale malware image. They have used two pre-trained convolutional neural networks AlexNet and InceptionV3. They have extracted a $4,096$ dimensional feature vector with AlexNet and extracted a $2,048$ dimensional feature vector with InceptionV3. They have combined these three feature vectors in the fusion step and obtained a $6,231$ dimensional fused feature vector. After that, the fused feature vector is reduced to $3,000$ dimensional feature space using principal component analysis in the dimension reduction part. Finally, they have used various machine learning classifiers such as support vector machines, k-nearest neighbors, and decision trees. The proposed model has only been evaluated on the Malimg dataset. They have reported 99.3% accuracy for the proposed model.

## 2.6 Capsule Networks

CapsNet, as a new CNN structure, has been implemented by Sabour et al. (2017), especially in the health domain (Jiménez-Sánchez et al. 2018), CapsNet have many applications in the literature. For instance, Afshar et al. (2018) use CapsNet for brain tumor classification problems like classification of breast cancer histology images of Iesmantas & Alzbutas (2018). Mobiny & Van Nguyen (2018) create a fast CapsNet architecture for lung cancer diagnosis. Another important application area of CapsNet is object segmentation. LaLonde & Bagci (2018) use CapsNet for object segmentation. Traditional CNN structures, on the other hand, are used in **G**enerative **A**dversarial **N**etworks, GANs. CapsNet is very useful to make GANs better by removing the weakest point of these CNNs (Jaiswal et al. 2018).

Although capsule networks are new convolutional neural network architectures, they significantly impact the computer vision area. However, there are other applications of capsule networks other than computer vision. For example, Ye et al. (2020) introduces a capsule network application called CapsLoc for indoor localization from received signal strength indicator values coming from wireless access points. They have collected $33,600$ data points using the wifi fingerprinting method. They have used mutual differences of received signal strength indicator values from different access points as data representations. They report a 0.68 meter average error comparing other conventional methods such as support vector machine, k-nearest neighbors, and convolutional neural networks.

Capsule networks with some modifications can outperform some models for sequential datasets for some tasks such as text classification, tag recommendation, and speech recognition. Lei et al. (2020) introduce a tag recommendation model by text classification. They have combined capsule networks with an attention layer, a novel and efficient deep learning technique for sequential representations like text in their tag recommendation model. The attention layer helps the capsule network to distill the important information from the input documents. They have tested the

proposed model with two different datasets called TPA and AG. The TPA dataset contains 18,464 academic articles and five unique tags. The AG dataset contains 127,600 news articles ad four unique tags. They have attained a 0.824 F-score for the TPA dataset and 0.923 F-score for the AG dataset.

Another exciting application of a modified capsule network on sequential representations is introduced by Dong et al. (2020). They propose a capsule network combined with BiLSTM layers for a sentiment analysis task. They have used three different datasets: the internet movie database (IMDB), movie review (MR), and Stanford sentiment tree (SST) dataset. The IMDB dataset contains 50,000 samples and two unique classes, such as positive and negative, and the average sentence length is 294. The MR dataset has 5,331 positive reviews, 5,331 negative reviews and the average sentence length is 20. The SST dataset contains 11,855 sentences and five different categories. In the SST dataset, the average sentence length is 19. They have obtained 81.47% accuracy for the MR dataset, 91.96% accuracy for the IMDB dataset, and 48.34% accuracy for the SST dataset. Their results show that using a capsule network with BiLSTM layers can reach the state-of-the-art results on these three datasets.

One of the interesting applications of the capsule network model for a sequential dataset is designed for remote sensor signals. For example, He et al. (2020) design a capsule network called PickCapsNet for automatic P-wave arrival picking in microseismic monitoring. They emphasize that existing semi-automatic P-wave arrival picking models use complex feature extraction from wave-formed datasets. However, PickCapsNet, their proposed model, is a highly scalable capsule network for P-wave arrival picking from a single waveform without complex feature extraction methods. They have used a dataset containing 1,736 records of microseismic waveforms from a copper mine located in China. All samples in the dataset are associated with manually determined P-wave arrival picks. The paper has split the original dataset into a training set containing 75% of the records and a test set containing 25% of the records. They compared PickCapsNet with three methods: Akaike information

criterion, short and long time average, and standard convolutional neural network. In order to evaluate the performance of the proposed model, they have used a signal-to-noise ratio. Their results show three critical findings. First, PickCapsNet is more accurate than others. Second, PickCapsNet has high accuracies under different signal-to-noise ratios. Finally, PickCapsNet is more stable and reliable than traditional mathematical models for P-wave arrival picking.

The snippet studies above show that CapsNet is a promising architecture against the traditional CNN. In the literature, although there are many applications of CapsNets, there is a missing and important area. This area is a computer and information security. This gap can be seen easily in the pre-print version of a survey about CapsNets (Patrick et al. 2019).

## 2.7 Summary of the Related Work

Malware classification and detection with the machine and deep learning is a broad and very dynamic research area. However, some selected papers summarized above show that image-based malware classification with traditional machine learning approaches is the most common technique. Thanks to the rapid developments in deep learning, many researchers have focused on building different neural network architecture, multi-modal models, ensemble models, or hybrid versions of different static, dynamic, and image-based malware features. In addition, researchers can apply many deep learning models to raw datasets, so using deep learning architectures reduces the dependency on complex and domain knowledge-based feature engineering methods.

Table 2.1 provides a summary of some of the selected papers from the related work for each topic we focus on and discuss through section 2. Also, it indicates the lack of studies on CapsNet and imbalanced dataset handling for malware family type classification task. To this end, this thesis aims to develop a malware classification model based on an ensemble of CapsNet architecture for imbalanced malware

datasets to fill the research gap in malware classification. Moreover, the latest publications citing our study (Çayır et al. 2021) show that CapsNet and its bagging ensemble version have great potential for image-based malware classification tasks (Hamza et al. 2021, Zhang et al. 2021).

**Table 2.1** Some Related works investigated by subject.

| **Paper** | **Malware Classification** | **Capsule Network** | **Imbalanced Dataset Handling** |
|---|---|---|---|
| Alazab (2015) | + | - | - |
| Azab et al. (2014) | + | - | - |
| Nataraj, Karthikeyan, Jacob & Manjunath (2011) | + | - | - |
| Kancherla & Mukkamala (2013) | + | - | - |
| Krizhevsky et al. (2012) | + | - | - |
| Ni et al. (2018) | + | - | - |
| Sabour et al. (2017) | - | + | - |
| Afshar et al. (2018) | - | + | - |
| Iesmantas & Alzbutas (2018) | - | + | - |
| Mobiny & Van Nguyen (2018) | - | + | - |
| LaLonde & Bagci (2018) | - | + | - |
| Jaiswal et al. (2018) | - | + | - |
| Patrick et al. (2019) | - | + | - |
| Ebenuwa et al. (2019) | - | - | + |

# 3. METHODOLOGY

There are many open research issues in malware classification. These issues can be listed such as class imbalance, concept drift, adversarial learning, interpretability (explainability) of the models, and public benchmarks (Gibert et al. 2020*b*). In this thesis, our model called RCNF focuses on the class imbalance issue. Thus, the base CapsNet and the proposed RCNF models have been tested on two very well-known malware datasets called Malimg and Microsoft Malware 2015 (BIG2015). These datasets are highly imbalanced in terms of the number of classes. This section describes these datasets.

## 3.1 Malware Datasets

### 3.1.1 Malimg

Nataraj et al. introduced a new malware family type classification approach based on visual analysis, converted binaries into greyscale images and they published these images as a new malware dataset called Malimg (Nataraj, Karthikeyan, Jacob & Manjunath 2011). This dataset has $9,339$ samples and 25 different classes. Table 3.1 presents the number of samples for each malware family. This distribution shows that the dataset is highly imbalanced.

Figure 3.1 shows the malware images created from the byte files. All images are single-channel and are resized to $224 \times 224$ for CapsNet architecture. This size is the largest value that can be processed in our computer system.

**Table 3.1** Sample distribution for each malware family.

| No. | Family Name | Number of Samples |
|---|---|---|
| 1 | Allaple.L | 1591 |
| 2 | Allaple.A | 2949 |
| 3 | Yuner.A | 800 |
| 4 | Lolyda.AA 1 | 213 |
| 5 | Lolyda.AA 2 | 184 |
| 6 | Lolyda.AA 3 | 123 |
| 7 | C2Lop.P | 146 |
| 8 | C2Lop.gen!g | 200 |
| 9 | Instantaccess | 431 |
| 10 | Swizzot.gen!I | 132 |
| 11 | Swizzor.gen!E | 128 |
| 12 | VB.AT | 408 |
| 13 | Fakerean | 381 |
| 14 | Alueron.gen!J | 198 |
| 15 | Malex.gen!J | 136 |
| 16 | Lolyda.AT | 159 |
| 17 | Adialer.C | 125 |
| 18 | Wintrim.BX | 97 |
| 19 | Dialplatform.B | 177 |
| 20 | Dontovo.A | 162 |
| 21 | Obfuscator.AD | 142 |
| 22 | Agent.FYI | 116 |
| 23 | Autorun.K | 106 |
| 24 | Rbot!gen | 158 |
| 25 | Skintrim.N | 80 |

(a) Adialer (b) Fakerean

**Figure 3.1** Malware image samples obtained from byte files using the algorithm described in (Nataraj, Karthikeyan, Jacob & Manjunath 2011) and shown in Figure 3.2

### 3.1.2 Microsoft Malware 2015 (BIG2015)

BIG2015 dataset has been released as a Kaggle competition (Ronen et al. 2018). Table 3.2 presents the sample distribution for each malware family in BIG2015 dataset. The distribution shows that the dataset is highly imbalanced and *Simda* is the toughest malware family to be predicted for the dataset. The dataset contains $10,868$ BYTE (bytes) files and $10,868$ ASM (assembly code) files and 9 different malware family types. BIG2015, unlike the Malimg dataset, contains raw files. As in the pre-processing approach on the Malimg dataset, we opened a file from the BIG2015 dataset in the byte mode, and then the file is read by 256 sized chunks till the end of the file. Finally, the buffer is converted to an array and the array is saved as a greyscale image into the file system. All required pre-processing steps are depicted in Fig 3.2. This method is the most common way to convert from malware files to images (Nataraj, Karthikeyan, Jacob & Manjunath 2011, Venkatraman & Alazab 2018).

Figure 3.3 depicts image representations created from the BYTE and ASM files of the same malware sample in *Ramnit* malware family. All images are single channel. All images are resized to $112 \times 112$ dimensions for our CapsNet architecture, because the architecture uses both BYTE and ASM image representations at the same time.

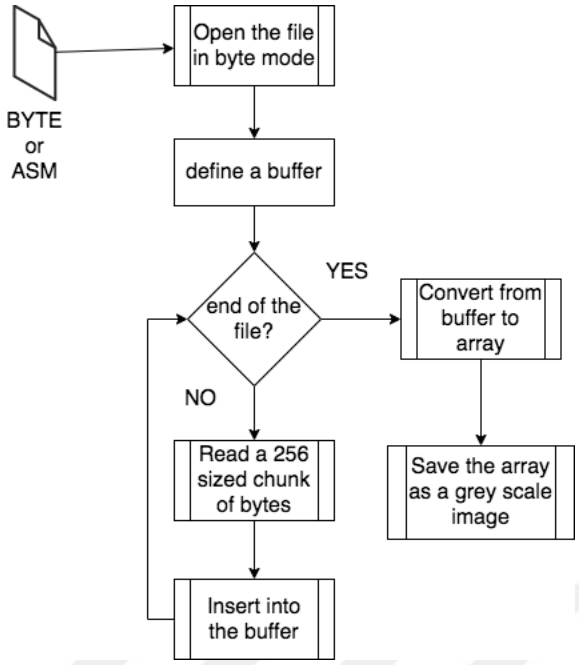


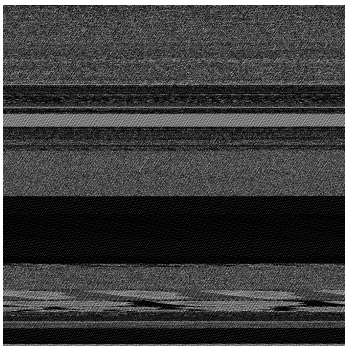

**Figure 3.2** Flowchart of the process of a flat-file to greyscale image for the BIG2015 dataset.

**Table 3.2** Number of samples for each malware family in BIG2015 dataset.

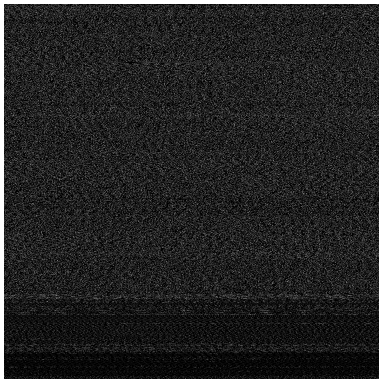
| No. | Family Name | Number of images |
|---|---|---|
| 1 | Ramnit | 1541 |
| 2 | Lollipop | 2478 |
| 3 | Kelihos_ver3 | 2942 |
| 4 | Vundo | 475 |
| 5 | Simda | 42 |
| 6 | Tracur | 751 |
| 7 | Kelihos_ver1 | 398 |
| 8 | Obfuscator.ACY | 1228 |
| 9 | Gatak | 1013 |

## 3.2 Models

In this section, general capsule networks, base CapsNet architecture for Malimg and base CapsNet architecture for BIG2015 are described. CapsNet architectures are different for both Malimg and BIG2015 Dataset.



(a) Ramnit image from BYTE file



(b) Ramnit image from ASM file

**Figure 3.3** BIG2015 image samples from BYTE and ASM files using the algorithm described in ((Nataraj, Karthikeyan, Jacob & Manjunath 2011)).

### 3.2.1 Capsule Networks

Capsule networks are special architectures of convolutional neural networks aiming to minimize information loss because of max pooling (Sabour et al. 2017). This method is the weakest point for preserving spatial information (Iesmantas & Alzbutas 2018). A CapsNet contains capsules similar to autoencoders (Krizhevsky & Hinton 2011, Sabour et al. 2017). Each capsule learns how to represent an instance for a given class. Therefore, each capsule creates a fixed-length feature vector to be input for a classifier layer without using max pooling layers in its internal structure. In this way, this capsule structure aims to preserve texture and spatial information with minimum loss. In addition to those, we chose CapsNet as a base estimator since the model reaches the-state-of-the-art results in some tasks with less the number of trainable parameters. Therefore, this case makes creating an ensemble of CapsNet easier than other deep convolutional neural networks containing millions of trainable parameters.

(Sabour et al. 2017) propose an efficient method to train CapsNet architectures. This method is called a dynamic routing algorithm, which uses a new non-linear activation function called squashing shown in (3.1). This equation emphasizes that short vectors are shrunk to almost zero and long vectors are shrunk to 1 (Sabour et al. 2017). In this equation, $v_i$ is the output of $i^{th}$ capsule and $s_i$ shows the total

input of this capsule.

$$v_i = \frac{\|s_i\|^2}{1+\|s_i\|^2} \frac{s_i}{\|s_i\|} \tag{3.1}$$

Visualizing the squash activation function described in (3.1) is hard because its input is a high dimensional vector. If the activation function can be thought of as a single variable function, as described in Marchisio et al. (2019), then the behavior of the function and its derivative can be visualized, as in Figure 3.4.

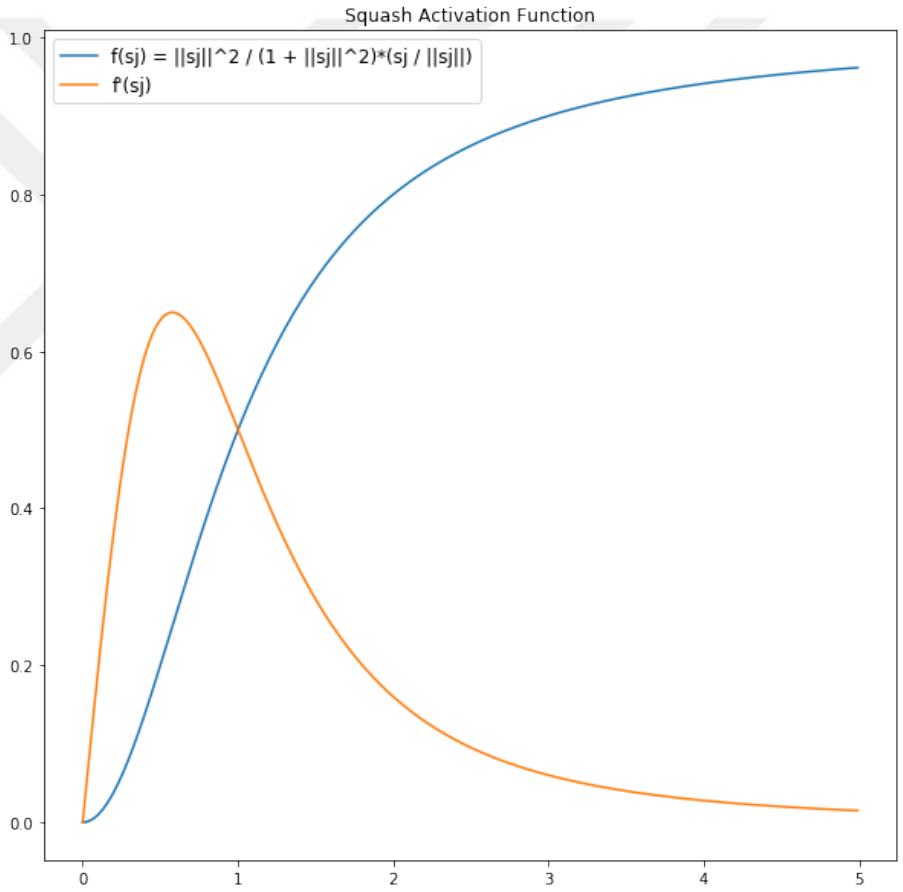



**Figure 3.4** Squashing activation function and its derivation in 2-D plane.

A basic CapsNet architecture contains two parts: the standard convolution blocks and the capsule layer as shown in Figure 3.5. A convolution block is made from a combination of convolution filters and ReLU activation function. At the end of the convolution block, obtained feature maps are reshaped and projected to $d-$

*dimensional* vector representation. This representation feeds each capsule in the capsule layer. Each capsule learns how to represent and reconstruct a given sample like an autoencoder (Krizhevsky & Hinton 2011) architecture. In order to learn how to reconstruct a malware sample, the capsule network will minimize reconstruction error in (3.2), where $x_c \in R^{d \times d}$ is the real sample in the capsule $c$ and $\hat{x}_c \in R^{d \times d}$ is the reconstructed sample by the same capsule $c$. These representations are used to calculate the class probabilities for the classification task.

$$\ell_r = (x_c - \hat{x}_c)^2 \tag{3.2}$$

Margin loss function is used for CapsNet. This function is similar to hinge loss (Rosasco et al. 2004). The equation (3.3) defines the margin loss $\ell_m$ for capsule $c$,

$$\ell_m = y_c \times (max(0, m - \hat{y}_c))^2 + \lambda \times (1 - y_c) \times (max(0, \hat{y}_c - (1 - m)))^2 \tag{3.3}$$

where $m = 0.9$, $\lambda = 0.5$, $y_c$ denotes actual class, and $\hat{y}_c$ represents the current prediction.

$$L_c = \ell_m + 0.0005 \times \ell_r \tag{3.4}$$

$$L = \frac{1}{N} \sum_{n=1}^{N} L_c \tag{3.5}$$

The mean of $L_c$ for each capsule gives the total loss in (3.4), where $L_c$ is sum of margin loss $\ell_m$ (as described in (3.3)) and reconstruction loss $\ell_r$ (as described in (3.2)). However, reconstruction loss is multiplied by 0.0005 to avoid suppressing the margin loss (Sabour et al. 2017). In order to minimize loss $L$ in (3.5), one can use the most applicable optimizer algorithm for CapsNet by Adam (Kingma & Ba

2014, Sabour et al. 2017). We have observed that our CapsNet architecture cannot converge to minimum loss value with optimizers other than Adam, unlike Chauhan et al. (2018). This is obviously an open issue for our CapsNet studies in the future.

In image-based malware family type classification problem, there are no complex patterns that are easily detected by classical convolutional neural networks. For this reason, the predictive model must recognize the pattern of pixel distribution of the image-based malware sample. On the other hand, CapsNet can learn pixel density distribution of each malware family. Thus, a CapsNet model can be easily trained from scratch for this problem, unlike CNNs. This is the most important advantage of using CapsNet as a base classifier in our proposed model.

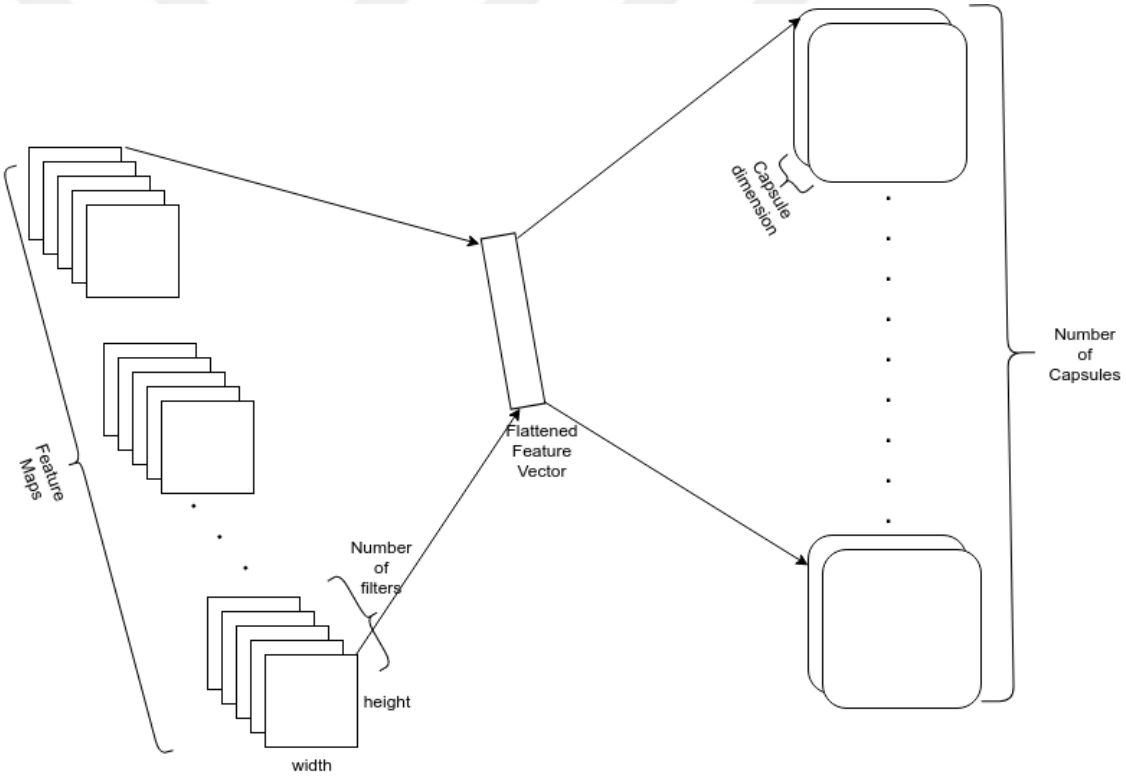



**Figure 3.5** Basic CapsNet architecture.

The main assumption of this thesis is that CapsNet architecture will be able to successfully classify malware family types using raw pixel values obtained from malware binary and assembly files. In addition to the main assumption, this thesis aims to increase CapsNet malware type classification architecture accuracy with the bagging

ensemble method.

### 3.2.2 Base Capsule Network Model for Malimg Dataset

Before creating an ensemble CapsNet model, the base CapsNet estimator must be built. This architecture depends on the dataset. Thus, base CapsNet estimator architecture has a single convolution line, as shown in Figure 3.6. The convolutional line contains two sequential blocks and each block contains two sequential convolutions and ReLU layers. The first two convolutional layers have $3 \times 3$ kernels and 32 filters. The second two convolutional layers have $3 \times 3$ kernels and 64 filters. Feature maps are reshaped to 128 feature vectors. At the end of the reshape step, there is a capsule layer containing 25 capsules; the dimension of each capsule is 8 and the routing iteration is 3 of the capsule layer. This is the tuned CapsNet architecture for the Malimg dataset depending on our experiments.

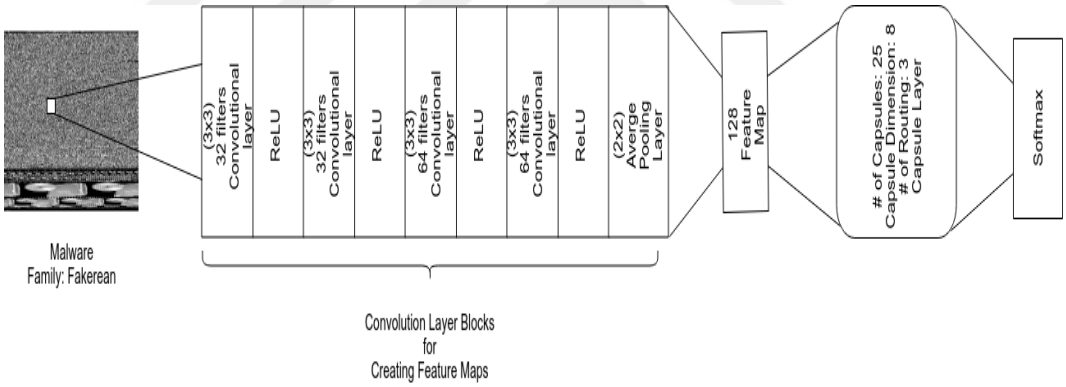



**Figure 3.6** CapsNet architecture for Malimg dataset.

### 3.2.3 Base Capsule Network Model for BIG2015 Dataset

The BIG2015 dataset has two different files for each sample. One of them is a binary file and the other is an assembly file. Thus, it is possible to design a CapsNet, which can be fed by two different image inputs at the same time. Figure 3.7 shows a CapsNet architecture, which has two exactly identical convolution lines. In this architecture, the first two sequential layers contain $3 \times 3$ kernels and 64 filters. The second two sequential layers contain $3 \times 3$ kernels and 128 filters.

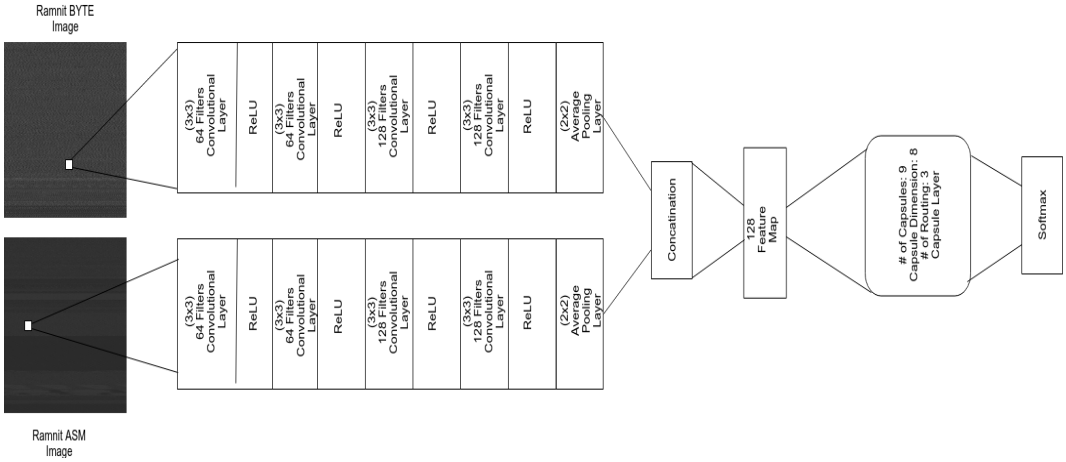



**Figure 3.7** CapsNet architecture for BIG2015 dataset.

Features extracted from the ASM and BYTE images are concatenated and the final feature vector is reshaped to a vector with length 128. For the next level, as an input, this feature vector feeds to a capsule layer containing 9 capsules. In this layer, the dimension of each capsule is 8 and the routing iteration is 3. This hyper-parameter set is optimal for the base CapsNet estimator for the BIG2015 dataset.

### 3.2.4 The Proposed Random CapsNet Forest Model for Imbalanced Datasets

Random CapsNet Forest (RCNF) is an ensemble model, which is inspired by the Random Forest algorithm (Breiman 2001). The basic idea behind RCNF is to assume identical CapsNet models as weak learners create different training sets for each model from the original training set using the bootstrap resampling technique, as shown in Algorithm 1. The training algorithm is a variant of bootstrap aggregating (also known as bagging) (Breiman 1996*a*) for CapsNet model and bagging reduces the variance of the model while increasing robustness of the model (Breiman 1996*b*). In this study, bagging is preferred to create an ensemble of CapsNet instead of boosting (Freund et al. 1996), because it is shown that boosting tends to overfit (Quinlan et al. 1996). During the training phase, each epoch updates the weights of the CapsNet. Therefore, the weight of the best model at the end of each epoch is saved according to the validation score to increase model performance and consistency against random weight initialization of the CapsNet.

**Algorithm 1** Random CapsNet Forest Training Algorithm

**procedure** TRAIN($base\_model, n\_estimators, trainset, valset, epochs$)
    **for** $i \leftarrow 1, n\_estimators$ **do**
        $bs\_trainset \leftarrow resample(trainset, replacement = True)$
        **for** $e \leftarrow 1, epochs$ **do**
            $base\_model.fit(bs\_trainset)$
            $val\_score \leftarrow get\_accuracy(base\_model, valset)$
            **if** $is\_best\_score(val\_score) == True$ **then**
                $save\_weights(base\_model)$

**Algorithm 2** Random CapsNet Forest Prediction Algorithm

**procedure** PREDICT($n\_estimators, testset, numclasses$)▷ Average Ensembling
    $total\_preds \leftarrow zeros\_like(testset.shape[0], numclasses)$
    **for** $i \leftarrow 1, n\_estimators$ **do**
        $model_i \leftarrow load\_model\_weights(i)$
        $total\_preds += model_i.predict(testset)$
    $preds \leftarrow total\_preds/n\_estimators$
    **return** argmax($preds$) ▷ The final predictions of CapsNet models

The prediction method is described in Algorithm 2. Each weight of CapsNet model is loaded and test samples are predicted by the model. Cumulative predicted probabilities are added onto *total_preds* variable and this step is known as average ensembling step. At the end of the estimation loop, the index of the highest probabilities is assigned as a predicted class.

# 4. EXPERIMENTS & RESULTS

CapsNet and RCNF ensemble models are tested on two different datasets called Malimg and BIG2015. The Malimg dataset has been divided into three parts: training, validation, and test sets. The training set has $7,004$ samples, the validation set has $1,167$ samples, and the test set has $1,167$ samples. BIG2015 has also been divided into three parts like the Malimg dataset. In the experiments of CapsNet and RCNF ensemble model for the BIG2015 dataset, the training set has $8,151$ samples, the validation set has $1,359$ samples and the test set has $1,358$ samples. The first experiment is made to obtain performance results of single base CapsNet estimators for each dataset. The second experiment is about the performance of the RCNF model. Model evaluation has been done in terms of accuracy, F-Score, and the number of parameters of deep neural nets. These performance metrics are defined as follows:

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4.1}$$

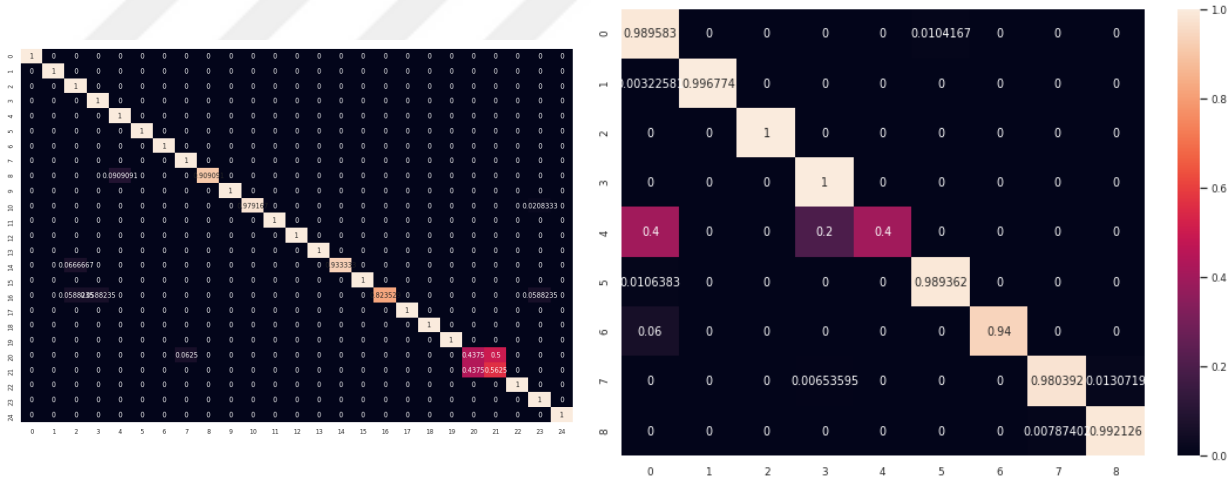
$$F\text{-}Score = \frac{2 \times TP}{2 \times TP + FN + FP} \tag{4.2}$$

where true positive (TP) and false positive (FP) are the numbers of instances correctly and wrongly classified as positive respectively. True negative (TN) and false negative (FN) are the number of instances correctly and wrongly classified as negative respectively. Accuracy is the ratio of the number of true predictions to all instances in the set as shown in (4.1). F-Score is shown as the set (4.2) in terms of true positives, false negatives, and false positives. Accuracy is not a correct performance metric for imbalanced datasets. On the other hand, papers compared in this work use accuracy and F-Score performance metrics to measure the success of their models. Thus, this thesis gives the experiment results in terms of accuracy and the

F-Score. Our main goal is that an ensemble of the CapsNet can increase F-score by virtue of decreasing FN and FP in (4.2).

Figure 4.1 shows confusion matrices for each test part of both datasets. Each confusion matrix (Figure 4.1a and 4.1b) implies that a model containing single CapsNet incorrectly predicts rare malware families in both datasets.

Figure 4.2 is the confusion matrix of RCNF containing 5 base CapsNet models. This confusion matrix shows the prediction accuracy of the model for each malware family type in the Malimg test set. Class 8, 10, 20, and 21 have been predicted wrongly by the RCNF model. On the other hand, the model has been very successful at correctly predicting other malware types in the test set. This confusion matrix also shows that RCNF is successful at correctly predicting rare malware types in the Malimg test set.




(a) Malimg test set (b) BIG2015 test set

**Figure 4.1** Confusion matrices of single CapsNet model for each test set.

In the second experiment, RCNF is tested on the BIG2015 dataset. Figure 4.3 shows the prediction results of RCNF containing 10 base CapsNet for BIG2015 dataset. Class 4 is the rarest malware type in the whole dataset. Training, validation, and test sets are stratified, so the class distribution is preserved for each partition. This result shows that RCNF can predict the rarest malware type pretty well. Class 0, 1,

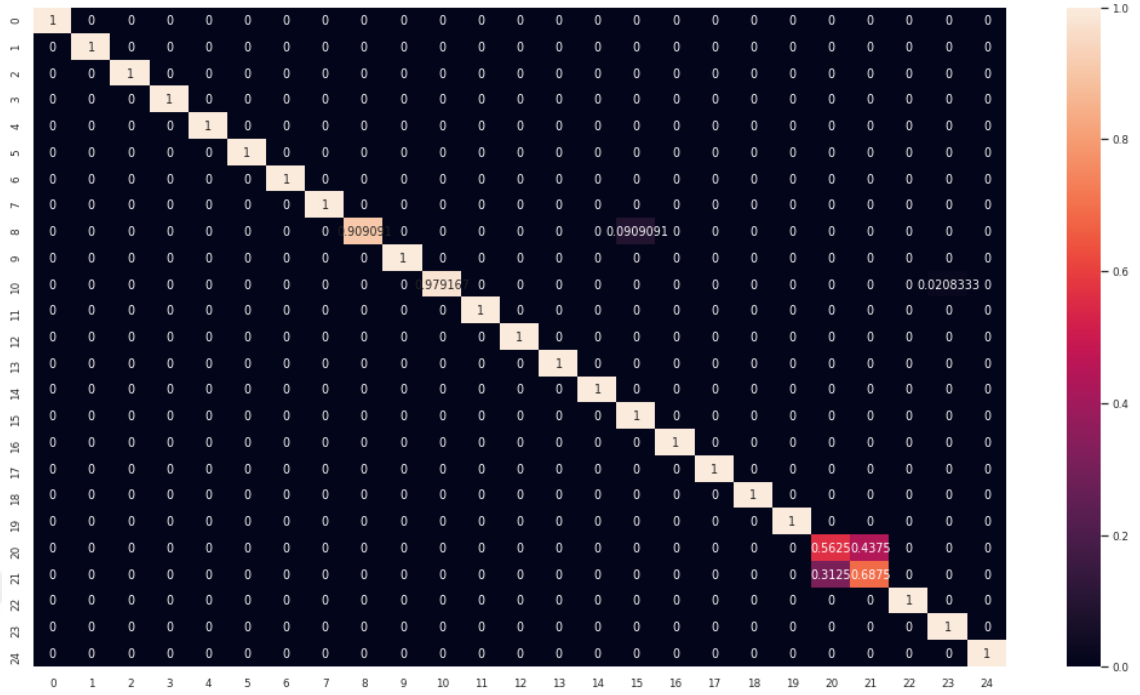


**Figure 4.2** Confusion matrix of 5-RCNF for Malimg test set.

2, and 6 are predicted perfectly by RCNF. If the performance of RCNF is compared with the performance of a single CapsNet model, it is easily seen that RCNF is better than a single CapsNet at predicting rare malware families for imbalanced datasets.

**Table 4.1** Comparison RCNF and other methods for Malimg test set performance.

| Model | Number of Parameters | F-Score | Accuracy |
|---|---|---|---|
| Yue (2017) | 20M | - | 0.9863 |
| Cui et al. (2018) | - | 0.9455 | 0.9450 |
| Venkatraman et al. (2019) | 212,885 | 0.916 | 0.963 |
| Vasan, Alazab, Wassan, Naeem, Safaei & Zheng (2020) | 134M | 0.9820 | 0.9827 |
| Vasan, Alazab, Wassan, Safaei & Zheng (2020) | 157M | 0.9948 | 0.9950 |
| **CapsNet for Malimg** | 90,592 | 0.9658 | 0.9863 |
| **RCNF for Malimg** | $5 \times 90,592$ | 0.9661 | 0.9872 |

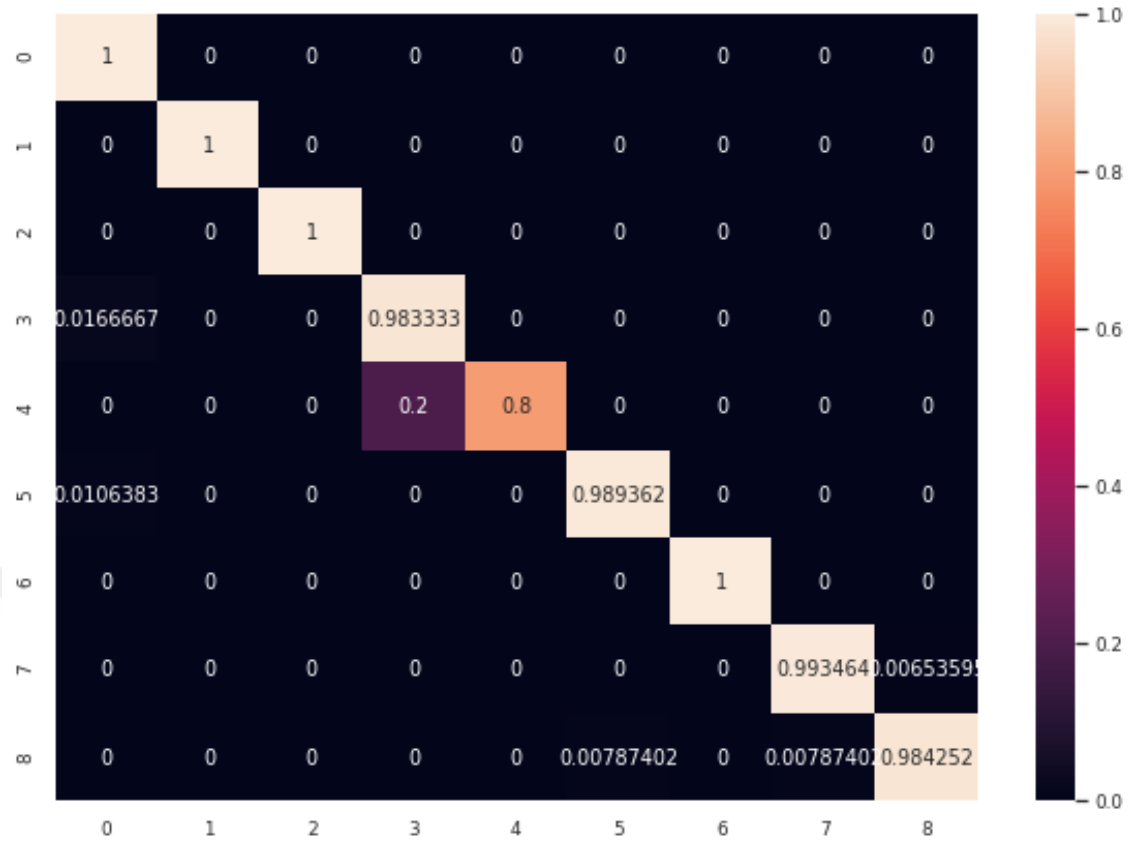



**Figure 4.3** Confusion matrix of 10-RCNF for BIG2015 test set.

Table 4.1 shows the test performance of the proposed models and others for the Malimg dataset. Yue (2017) uses a weighted loss function to handle the imbalanced class distribution problem in the Malimg dataset and also uses the transfer learning (Yosinski et al. 2014) method to classify malware family types. Due to the usage of transfer learning, the architecture has 20M parameters and the model is so huge. Cui et al. (2018) use classical machine learning methods such as K-Nearest Neighbor and support vector machines. They have trained these algorithms using GIST and GLCM features, which are feature engineering methods for images and they have applied resampling to the dataset to solve imbalanced dataset problems. RCNF does not use a weighted loss function or any sampling method to overcome the imbalanced dataset problem. Our results are higher than these two methods and the results also show that CapsNet and RCNF do not require any method for extra feature engineering in the Malimg dataset. A single CapsNet architecture for the Malimg dataset has $90,592$ trainable parameters and RCNF has $452,960$ trainable

**Table 4.2** Comparison RCNF and other methods for BIG2015 test set performance.

| Model | Number of Parameters | F-Score | Accuracy |
|---|---|---|---|
| Venkatraman et al. (2019) | $212,885$ | 0.725 | - |
| Cao et al. (2018) | - | - | 0.95 |
| Gibert et al. (2018) | - | 0.9813 | 0.9894 |
| Kreuk et al. (2018) | - | - | 0.9921 |
| Le et al. (2018) | $268,949$ | 0.9605 | 0.9820 |
| Chen (2018) | - | - | 0.9925 |
| Jung et al. (2018) | $148,489$ | - | 0.99 |
| Abijah Roseline et al. (2019) | - | - | 0.9914 |
| Zhao et al. (2019) | - | - | 0.929 |
| Khan et al. (2018) | - | - | 0.8836 |
| Safa et al. (2019) | - | - | 0.9931 |
| Kebede et al. (2017) | - | - | 0.9915 |
| Kim et al. (2018) | - | - | 0.9266 |
| Kim & Cho (2018) | - | 0.8936 | 0.9697 |
| Yan et al. (2018) | - | - | 0.9936 |
| Naeem et al. (2019) | - | 0.971 | 0.9840 |
| Jang et al. (2020) | - | - | 0.9965 |
| **CapsNet for BIG2015** | $527,232$ | 0.9779 | 0.9926 |
| **RCNF for BIG2015** | $10 \times 527,232$ | 0.9820 | 0.9956 |

parameters, so our proposed methods are reasonably smaller than Yue's method. Venkatraman et al. (2019) propose two different models called CNN BiLSTM and CNN BiGRU with two variants of these models, which are called cost-sensitive and

cost-insensitive. When CNN BiGRU reaches its own highest F-Score and accuracy on the Malimg dataset, its number of the trainable parameters is greater than RCNF and its scores are lower than RCNF. In other words, RCNF reaches the-state-of-the-art scores in terms of F-Score and accuracy with a lower parameter size.

IMCFN is an important deep convolutional network for the Malimg dataset (Vasan, Alazab, Wassan, Naeem, Safaei & Zheng 2020). When the network is trained from scratch without any data augmentation, it outperforms our RCNF implementation. IMCFN also has $126,727,705$ trainable parameters. When it is compared to RCNF, IMCFN is a huge convolutional network. On the other hand, RCNF is more accurate than IMCFN, but it has a lower F-Score than IMCFN. The most important advantage of RCNF is using a lower number of parameters, and this makes RCNF trainable in GPUs with low memory.

IMCEC is one of the most successful an ensemble model of CNN architectures for the Malimg dataset in terms of accuracy and F-Score (Vasan, Alazab, Wassan, Safaei & Zheng 2020). However, IMCEC is more complex than RCNF. IMCEC uses two different CNN architectures VGG16 (Simonyan & Zisserman 2014) and ResNet-50 (He et al. 2016) to use transfer learning. Although IMCEC is more accurate than RCNF in terms of accuracy and F-Score, IMCEC has a total of 157M parameters because of using these CNN networks. Unlike IMCEC, RCNF has $452,960$ trainable parameters and it is reasonably accurate as much as IMCEC.

Table 4.2 compares the test performance of the proposed models and others for the BIG2015 dataset. Venkatraman et al. (2019) also test the CNN BiGRU model on the BIG2015 dataset. In this case, the performance of the CNN BiGRU model is lower than RCNF, but its number of trainable parameters are reasonably lower than RCNF. Chen (2018) and Khan et al. (2018) use transfer learning architectures for the dataset. Test results of our proposed methods are better than those two models, but our results are very close to Chen (2018) in terms of accuracy. Our proposed models are better than Gibert et al. (2018) in terms of accuracy, but the F-Score

of RCNF is very close to this model. Our proposed models are better than models of Cao et al. (2018), Zhao et al. (2019), Kim et al. (2018), and Kim & Cho (2018). Jung et al. (2018) propose a reasonably smaller model than our models in terms of the number of parameters, but our model has a higher accuracy score than this model. Abijah Roseline et al. (2019), Safa et al. (2019), Kebede et al. (2017), and Yan et al. (2018) propose deep learning models whose accuracy scores are close to our proposed models. In addition to these, Jang et al. (2020) report five different accuracy scores for the BIG2015 dataset and they do not use F-Score despite the dataset is highly imbalanced. Their accuracy is the highest one for the dataset, but their network has two complex phases and uses GAN based data augmentation (they call this method malware obfuscator). On the other hand, RCNF does not use data augmentation, data resampling, transfer learning, and weighted loss functions for both datasets. In this perspective, RCNF is a simple version of an ensemble of CapsNet, and this simplicity highlights RCNF among its competitors.

For those tables, the last studies using Malimg and BIG2015 datasets are chosen. To compare them fairly, these models are drawn from image-based malware analysis studies. These results show that while RCNF reaches the-state-of-the-art scores in terms of F-Score and accuracy on the Malimg dataset with less trainable parameter size, it outperforms some of its competitors on the BIG2015 with larger size of parameters.

# 5. LIMITATIONS

There are several limitations to the RCNF model. The first limitation is the number estimators in the RCNF model. In this implementation, an RCNF model can contain up to 10 CapsNets because of hardware limitations. The second limitation is the training time. Training of an RCNF with 10 CapsNets for the BIG2015 dataset takes five hours (100 epochs for each CapsNet). This training time of the RCNF with 5 CapsNets model for the Malimg dataset is decreasing (100 epochs for each CapsNet). On the other hand, the RCNF can be easily parallelized to increase efficiency in the training phase. Each CapsNet can be trained on multiple GPUs. We will develop a distributed multi-GPU version of the RCNF as a future work. Last but not least, the third limitation is about the optimization algorithm in the CapsNet training phase. In the training phase of the CapsNet, the Adam optimizer is the only option. If other optimization options in the deep learning literature become applicable for the training phase of the CapsNet, then the training time will be more reduced on the GPU devices. Thus, this is an open problem for optimization techniques for the CapsNet architecture. On the other hand, our RCNF implementation, which can process BYTE and ASM files simultaneously for the BIG2015 dataset, is not feasible for a web application because the pre-processing step takes much time.

We implemented the RCNF using Tensorflow (version 1.5) (Abadi et al. 2016) and Keras (Chollet et al. 2015), Sklearn (Pedregosa et al. 2011), Numpy (Oliphant 2006) and Pandas (McKinney et al. 2010). All scripts were written in Python3. The configuration of the computer used in this study was 12GB GPU (GeForce GTX 1080 Ti) and Intel Core i9-9900K processor with 64 GB main memory for testing.

# 6. INCLUDED PAPERS AND CONTRIBUTIONS

In this part of the thesis, our papers contributing to the leading research on malware classifiers using capsule networks are listed with their abstracts and contributions. Paper in Section 6.1 is the starting point of researches on malware classification problems which is the main topic of this thesis, and the holdout validation strategy used in this paper has also been used in the experiments of this thesis. Paper in Section 6.2 has provided us a detailed literature review on imbalanced data handling methods and weaknesses of transfer learning approaches on domain-specific datasets rather than plain text, object, or natural images. Finally, the paper in Section 6.3 is the core article about the RCNF model and published in the journal titled Computers & Security, which is one of the most prestigious scientific journals in the cybersecurity research area.

## 6.1 Paper-I

Çayır, A., Ünal, U., Yenidoğan, I., & Dağ, H. (2019). Use Case Study: Data Science Application for Microsoft Malware Prediction Competition on Kaggle. Proceedings Book, 98. (Çayır et al. 2019)

### 6.1.1 Summary

Malware prediction is the most prominent area of cybersecurity domain. Malware prediction applications are leaned to be empowered by machine learning due to rapidly emerging intrusion attacks. In this perspective, defense systems aim to conjoin data science and cybersecurity. There are many platforms which provide public datasets and organize competitions for sector specific problems. For instance, Netflix

has organized a competition with $1 Million prize to develop a new recommendation system. Kaggle organizes many competitions for research, business and educational purposes. Microsoft has sponsored two important malware prediction competitions on Kaggle in 2015 and 2019. In this paper, we present our solution as a use case which is placed at 5th among 2,426 teams on Microsoft Malware Prediction 2019 dataset.

### 6.1.2 Contributions

The main contributions of this paper can be listed as follows:

- This paper is the first study on malware datasets related to this thesis. The paper has provided us to dive deeper into the literature on malware classification.
- This paper is the first study that we use the holdout validation technique used during the development of RCNF.
- The literature review of this paper has introduced two essential datasets, BIG2015 and Malimg, to us.

## 6.2 Paper-II

Demirkıran, F., Çayır, A., Ünal, U., & Dağ, H. (2020, September). Website category classification using fine-tuned BERT language model. In 2020 5th International Conference on Computer Science and Engineering (UBMK) (pp. 333-336). IEEE. (Demirkıran et al. 2020)

### 6.2.1 Summary

The contents on the Word Wide Web is expanding every second providing web users a rich content. However, this situation may cause web users harm rather than good due to its harmful or misleading information. The harmful contents can contain

text, audio, video, or image that can be about violence, adult contents, or any other harmful information. Especially young people may readily be affected with these harmful information psychologically. To prevent youth from these harmful contents, various web filtering techniques, such as keyword filtering, Uniform Resource Locator (URL) based filtering, Intelligent analysis, and semantic analysis, are used. We propose an algorithm that can classify websites, which may contain adult contents, with 67.81% (BERT) accuracy among 32 unique categories. We also show that a BERT model gives higher accuracy than both the Sequential and Functional API models when used for text classification.

### 6.2.2 Contributions

The main contributions of this paper can be listed as follows:

- This paper is the initial work that investigates a highly imbalanced dataset for a deep learning model.
- This paper uses the transfer learning approach for initializing the embedding layer using a pre-trained model called BERT and the transfer learning approach needs to learn similar patterns from the new dataset. Like text classification, transfer learning is useful in image classification tasks using models trained in similar image datasets. However, the transfer learning is not helpful for grey-scale malware images because all architectures used in image-based classification tasks have different patterns from the malware images. Because of this reason, the thesis relies on a model that can be easily trained from scratch and without using transfer learning for grey-scale malware images, such as CapsNet.

## 6.3 Paper-III

Çayır, A., Ünal, U., & Dağ, H. (2021). Random CapsNet forest model for imbalanced malware type classification task. Computers & Security, 102, 102133. (Çayır et al. 2021)

### 6.3.1 Summary

Behavior of malware varies depending the malware types, which affects the strategies of the system protection software. Many malware classification models, empowered by machine and/or deep learning, achieve superior accuracies for predicting malware types. Machine learning-based models need to do heavy feature engineering work, which affects the performance of the models greatly. On the other hand, deep learning-based models require less effort in feature engineering when compared to that of the machine learning-based models. However, traditional deep learning architectures' components, such as max and average pooling, cause architecture to be more complex and the models to be more sensitive to data. The capsule network architectures, on the other hand, reduce the aforementioned complexities by eliminating the pooling components. Additionally, capsule network architectures based models are less sensitive to data, unlike the classical convolutional neural network architectures. This paper proposes an ensemble capsule network model based on the bootstrap aggregating technique. The proposed method is tested on two widely used, highly imbalanced datasets (Malimg and BIG2015), for which the-state-of-the-art results are well-known and can be used for comparison purposes. The proposed model achieves the highest F-Score, which is 0.9820, for the BIG2015 dataset and F-Score, which is 0.9661, for the Malimg dataset. Our model also reaches the-state-of-the-art, using 99.7% lower the number of trainable parameters than the best model in the literature. This paper is the heart of the thesis, which describes complete details from the problem definition to the web-based cybersecurity application. For this reason, the thesis is an extended version of this paper.

### 6.3.2 Contributions

The main contributions of this paper can be listed as follows:

- The paper introduces the first application of CapsNet in the field of malware type classification. Although image-based malware classification is a broad research and application area, there is no research and application of CapsNet in the literature to our best knowledge.
- The paper uses the first ensemble model of CapsNet. The key idea of creating an ensemble of CapsNet is assuming a single CapsNet model as a weak classifier like a decision tree model. In this way, an ensemble model of CapsNet can be easily created using bootstrap aggregating. The main assumption that CapsNet is a weak learner increases the performance of a single CapsNet for two different well-known malware datasets, which are highly imbalanced.
- The proposed model uses simple architecture engineering instead of complex convolutional neural network architectures and domain-specific feature engineering techniques. In addition to this, CapsNet does not require the usage of transfer learning, and the model is easily trained from scratch. Because of that, the created network and its ensemble version have reasonably lower number of parameters. The proposed model obtains F-Score and accuracy, which are close to the-state-of-the-art results, using 99.7% lower the number of trainable parameters than the best model in the literature.
- The proposed model is compared with the latest studies that use deep neural networks for image-based malware classification tasks. For a fair comparison, especially, the last studies using the Malimg and the BIG2015 datasets are chosen and compared with the proposed method.
- The proposed method is reproducible and broadly simpler than other complex deep neural network architectures regarding the number of trainable parameters.

- The proposed model is used in the WARNING application for the static malware identification module of the project supported by The Scientific and Technological Research Council of Turkey under grant number 118E400.

# 7. CONCLUSION

## 7.1 Application

This work in the thesis is part of the research project titled "WARNING: A Defense-in-depth Cyber Intelligence Platform to Defend against Emerging Cyber Attacks," supported by The Scientific and Technological Research Council of Turkey under the grant number 118E400. This project aims to design a cybersecurity platform containing four different modules. Of which, the most important the is the malware detection and prediction module.

The malware detection and prediction module contain three different approaches such as static, dynamic, and hybrid malware analysis approaches (Raff & Nicholas 2020).The proposed model in the thesis is used for malware type classification application as the static approach. However, there are some differences in implementation details in the application part, unlike in the thesis. This part explains these differences and the architecture of the application. The first difference is that we have used the BIG2015 dataset for training the RCNF model for the application because we need to use raw files different from the Malimg dataset. The second difference is the size of the malware images. We resize images from the raw files to $32 \times 32$ and use ASM or BYTE files only, unlike the proposed model, because converting files to images proposed pre-processing part of the thesis takes much time at the user interface. Last but not least, the third significant difference is that we have developed two different RCNF models for BYTES and ASM for the application part.

To develop the application, we need to use Flask and Redis python libraries. Flask makes using model-view-controller (MVC) pattern in python easier and faster (Mu-

fid et al. 2019). Redis is an in-memory database, and it allows us to use worker abstraction for multi-request in web applications (Zhang et al. 2014).

As shown in Figure 7.1, the model in the MVC pattern for this application contains the name of the uploaded file as a key, predicted class value, types, probabilities coming from the prediction model, and hash values of the file using SHA256.

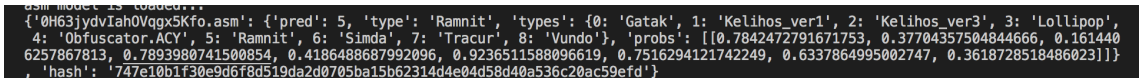



**Figure 7.1** Model structure of the application in MVC pattern.

The view part contains an upload tool and a button that is inactive before file uploading. The user can upload the raw file as an ASM, BYTE, or EXE format, as shown in Figure 7.2. The controller part of the MVC pattern starts with clicking on the button titled "Analyze File".

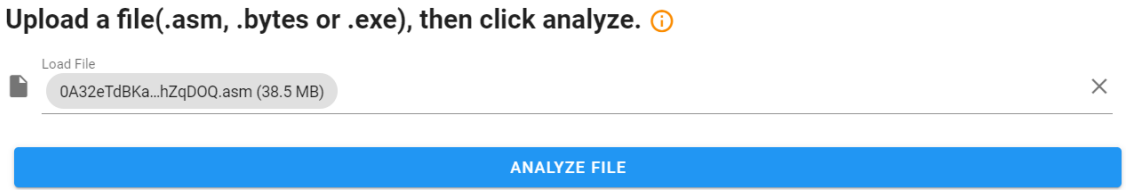



**Figure 7.2** Input view structure of the application in MVC pattern.

The button triggers a flow of a job that is scheduled by the Redis queue system as shown is Figure 7.3. Each flow will be wrapped by a structure called worker, which provides for handling multi-request for the application. Figure 7.4 represents that each request coming from the application port addressed 80 starts a worker wrapping a job flow directs to Redis queue system using the internal port addressed 6379.

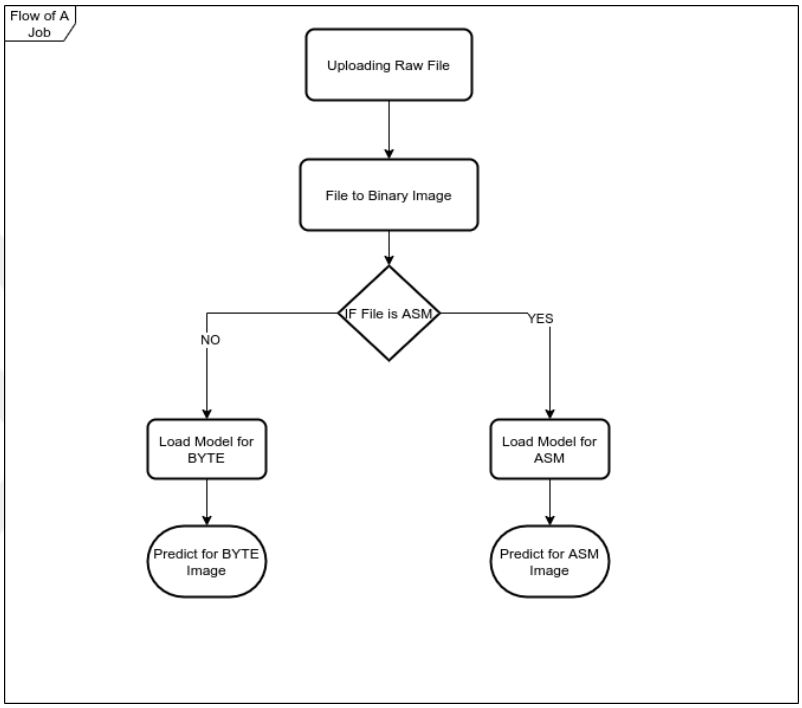



**Figure 7.3** A flow of a job for prediction in Redis queue system.

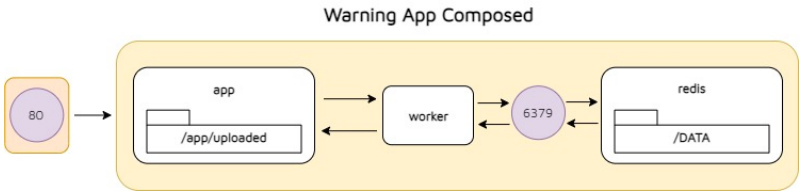



**Figure 7.4** WARNING worker structure for multi-requests.

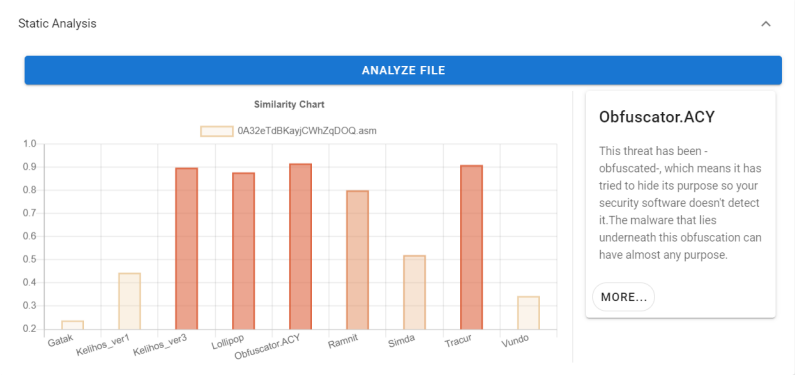



**Figure 7.5** Prediction results for an ASM file.

We use the un-normalized squashing activation results of class capsules for the user interface presenting prediction results. Figure 7.5 shows the final prediction results on the user interface of the WARNING application. If the user clicks on the button named "MORE...", the application redirects the user to a new page with more information about the final predicted malware family type.

To sum up, this thesis presents a simple, user-friendly, and multi-user-supported user interface that uses the proposed model. From the perspective of management information systems, the thesis explains all the steps from defining the problem in malware classification to a web application. Therefore, the web application for the WARNING project is one of the most valuable and the most concrete outputs of the thesis.

## 7.2 Summary and Future Work

This thesis introduces the first application of CapsNet on imbalanced malware family type classification task. Moreover, the first ensemble model of CapsNet called RCNF is introduced in this thesis. The proposed models do not require any complex feature engineering methods or architecture for deep networks. To show that, we used two different malware family type datasets: Malimg and BIG2015. These datasets are used for image-based malware classification. Our proposed models can utilize these datasets directly using raw pixel values.

Datasets in the thesis are highly imbalanced in terms of class distribution. CapsNet and RCNF do not use oversampling, under sampling, and weighted loss function during the training phase. Results show that CapsNet and RCNF are the best models least suffering from imbalanced class distribution among others in the literature.

Experiment results show that a single CapsNet model has good performance for both BIG2015 and Malimg datasets. However, we have assumed an ensemble model of CapsNet can help us to increase generalization performance and RCNF has better generalization performance results than a single CapsNet model as expected. In this point, results show that creating a bagging ensemble model CapsNet increases the performance on predicting rare malware classes. While single CapsNet can obtain 0.9779 F-Score for the BIG2015 dataset, an ensemble of 10 CapsNets achieves 0.9820 F-Score. We can observe the similar effects on the Malimg dataset. It is shown that bagging increases the performance of the CapsNet for imbalanced datasets and the ensemble model is more successful at predicting rare classes than a single CapsNet model due to ensembling.

Many models compared with the proposed model are complex and large in terms of the number of parameters. Some of them use data augmentation, weighted loss functions, different extra feature engineering methods, and pre-trained deep neural networks that have a large number of parameters. Our proposed model achieves the F-Score, which is 2.88% lower than the state-of-the-art result with reducing 99.7% the number of trainable parameters of the best model in the literature.

As for future work, we are planning to develop a distributed version of RCNF. On the other hand, working on open issues such that CapsNet's convergence strictly dependent on Adam optimizer and limitations of the number of estimators will be addressed.

# REFERENCES


Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016), Tensorflow: A system for large-scale machine learning, *in* '12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)', pp. 265–283.

Abijah Roseline, S. et al. (2019), Intelligent malware detection using deep dilated residual networks for cyber security, *in* 'Countering Cyber Attacks and Preserving the Integrity and Availability of Critical Systems', IGI Global, pp. 211–229.

Abusitta, A., Li, M. Q. & Fung, B. C. (2021), 'Malware classification and composition analysis: A survey of recent developments', *Journal of Information Security and Applications* **59**, 102828.

Afshar, P., Mohammadi, A. & Plataniotis, K. N. (2018), Brain tumor type classification via capsule networks, *in* '2018 25th IEEE International Conference on Image Processing (ICIP)', IEEE, pp. 3129–3133.

Alazab, M. (2015), 'Profiling and classifying the behavior of malicious codes', *Journal of Systems and Software* **100**, 91–102.

Alazab, M. & Tang, M. (2019), *Deep Learning Applications for Cyber Security*, Springer.

Alazab, M., Venkatraman, S., Watters, P. & Alazab, M. (2013), Information security governance: the art of detecting hidden malware, *in* 'IT security governance innovations: theory and research', IGI Global, pp. 293–315.

Aslan, Ö. & YILMAZ, A. A. (2021), 'A new malware classification framework based on deep learning algorithms', *IEEE Access* .

Azab, A., Alazab, M. & Aiash, M. (2016), Machine learning based botnet identification traffic, *in* '2016 IEEE Trustcom/BigDataSE/ISPA', IEEE, pp. 1788–1794.

Azab, A., Layton, R., Alazab, M. & Oliver, J. (2014), Mining malware to detect variants, *in* '2014 Fifth Cybercrime and Trustworthy Computing Conference', IEEE, pp. 44–53.

Azeez, N. A., Odufuwa, O. E., Misra, S., Oluranti, J. & Damaševičius, R. (2021), Windows pe malware detection using ensemble learning, *in* 'Informatics', Vol. 8, Multidisciplinary Digital Publishing Institute, p. 10.

Bakour, K. & Ünver, H. M. (2021), 'Visdroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques', *Neural Computing and Applications* **33**(8), 3133–3153.

Benzaid, C., Lounis, K., Al-Nemrat, A., Badache, N. & Alazab, M. (2016), 'Fast authentication in wireless sensor networks', *Future Generation Computer Systems* **55**, 362–375.

Breiman, L. (1996*a*), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

Breiman, L. (1996*b*), 'Bias, variance, and arcing classifiers'.

Breiman, L. (2001), 'Random forests', *Machine learning* **45**(1), 5–32.

Cao, D., Zhang, X., Ning, Z., Zhao, J., Xue, F. & Yang, Y. (2018), An efficient malicious code detection system based on convolutional neural networks, *in* 'Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence', ACM, pp. 86–89.

Catak, F. O., Ahmed, J., Sahinbas, K. & Khand, Z. H. (2021), 'Data augmentation based malware detection using convolutional neural networks', *PeerJ Computer Science* **7**, e346.

Çayır, A., Ünal, U. & Dağ, H. (2021), 'Random capsnet forest model for imbalanced malware type classification task', *Computers & Security* **102**, 102133.

Çayır, U., Yenidoğan, I. & Dağ, H. (2019), 'Use case study: Data science application for microsoft malware prediction competition on kaggle', *Proceedings Book* p. 98.

Chauhan, A., Babu, M., Kandru, N. & Lokegaonkar, S. (2018), 'Empirical study on convergence of capsule networks with various hyperparameters'.

Chen, L. (2018), 'Deep transfer learning for static malware classification', *arXiv preprint arXiv:1812.07606* .

Chollet, F. et al. (2015), 'keras'.

Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G.-g. & Chen, J. (2018), 'Detection of malicious code variants based on deep learning', *IEEE Transactions on Industrial Informatics* **14**(7), 3187–3196.

Demirkıran, F., Çayır, A., Ünal, U. & Dağ, H. (2020), Website category classification using fine-tuned bert language model, *in* '2020 5th International Conference on Computer Science and Engineering (UBMK)', IEEE, pp. 333–336.

Dib, M., Torabi, S., Bou-Harb, E. & Assi, C. (2021), 'A multi-dimensional deep learning framework for iot malware classification and family attribution', *IEEE Transactions on Network and Service Management* .

Dong, Y., Fu, Y., Wang, L., Chen, Y., Dong, Y. & Li, J. (2020), 'A sentiment analysis method of capsule network based on bilstm', *IEEE Access* **8**, 37014–37020.

Ebenuwa, S. H., Sharif, M. S., Alazab, M. & Al-Nemrat, A. (2019), 'Variance ranking attributes selection techniques for binary classification problem in imbalance data', *IEEE Access* **7**, 24649–24666.

Etaher, N., Weir, G. R. & Alazab, M. (2015), From zeus to zitmo: Trends in banking malware, *in* '2015 IEEE Trustcom/BigDataSE/ISPA', Vol. 1, IEEE, pp. 1386–1391.

Fang, Y., Gao, Y., Jing, F. & Zhang, L. (2020), 'Android malware familial classification based on dex file section features', *IEEE Access* **8**, 10614–10627.

Freund, Y., Schapire, R. E. et al. (1996), Experiments with a new boosting algorithm, *in* 'icml', Vol. 96, Citeseer, pp. 148–156.

Ganaie, M., Hu, M. et al. (2021), 'Ensemble deep learning: A review', *arXiv preprint arXiv:2104.02395* .

Gibert, D., Mateu, C. & Planes, J. (2018), An end-to-end deep learning architecture for classification of malware's binary content, *in* 'International Conference on Artificial Neural Networks', Springer, pp. 383–391.

Gibert, D., Mateu, C. & Planes, J. (2020*a*), 'Hydra: A multimodal deep learning framework for malware classification', *Computers & Security* **95**, 101873.

Gibert, D., Mateu, C. & Planes, J. (2020*b*), 'The rise of machine learning for detection and classification of malware: Research developments, trends and challenges', *Journal of Network and Computer Applications* p. 102526.

Hamza, A. A., Abdel-Halim, I. T., Sobh, M. A. & Bahaa-Eldin, A. M. (2021), 'A survey and taxonomy of program analysis for iot platforms', *Ain Shams Engineering Journal* .

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

He, Z., Peng, P., Wang, L. & Jiang, Y. (2020), 'Pickcapsnet: capsule network for automatic p-wave arrival picking', *IEEE Geoscience and Remote Sensing Letters* .

Hemalatha, J., Roseline, S. A., Geetha, S., Kadry, S. & Damaševičius, R. (2021), 'An efficient densenet-based deep learning model for malware detection', *Entropy* **23**(3), 344.

Iadarola, G., Martinelli, F., Mercaldo, F. & Santone, A. (2020), Image-based malware family detection: An assessment between feature extraction and classification techniques., *in* 'IoTBDS', pp. 499–506.

Iesmantas, T. & Alzbutas, R. (2018), Convolutional capsule network for classification of breast cancer histology images, *in* 'International Conference Image Analysis and Recognition', Springer, pp. 853–860.

Jaiswal, A., AbdAlmageed, W., Wu, Y. & Natarajan, P. (2018), Capsulegan: Generative adversarial capsule network, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 0–0.

Jang, S., Li, S. & Sung, Y. (2020), 'Fasttext-based local feature visualization algorithm for merged image-based malware classification framework for cyber security and cyber defense', *Mathematics* **8**(3), 460.

Jiménez-Sánchez, A., Albarqouni, S. & Mateus, D. (2018), Capsule networks against medical imaging data challenges, *in* 'Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis', Springer, pp. 150–160.

Jung, B., Kim, T. & Im, E. G. (2018), Malware classification using byte sequence information, *in* 'Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems', ACM, pp. 143–148.

Kancherla, K. & Mukkamala, S. (2013), Image visualization based malware detection, *in* '2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)', IEEE, pp. 40–44.

Kebede, T. M., Djaneye-Boundjou, O., Narayanan, B. N., Ralescu, A. & Kapp, D. (2017), Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset, *in* '2017 IEEE National Aerospace and Electronics Conference (NAECON)', IEEE, pp. 70–75.

Khan, R. U., Zhang, X. & Kumar, R. (2018), 'Analysis of resnet and googlenet models for malware detection', *Journal of Computer Virology and Hacking Techniques* pp. 1–9.

Kim, C. H., Kabanga, E. K. & Kang, S.-J. (2018), Classifying malware using convolutional gated neural network, *in* '2018 20th International Conference on Advanced Communication Technology (ICACT)', IEEE, pp. 40–44.

Kim, J.-Y. & Cho, S.-B. (2018), Detecting intrusive malware with a hybrid generative deep learning model, *in* 'International Conference on Intelligent Data Engineering and Automated Learning', Springer, pp. 499–507.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Korine, R. & Hendler, D. (2021), 'Daemon: Dataset/platform-agnostic explainable malware classification using multi-stage feature mining', *IEEE Access* .

Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B. & Keshet, J. (2018), 'Deceiving end-to-end deep learning malware detectors using adversarial examples', *arXiv preprint arXiv:1802.04528* .

Krizhevsky, A. & Hinton, G. E. (2011), Using very deep autoencoders for content-based image retrieval., *in* 'ESANN', Vol. 1, p. 2.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

LaLonde, R. & Bagci, U. (2018), 'Capsules for object segmentation', *arXiv preprint arXiv:1804.04241* .

Le, Q., Boydell, O., Mac Namee, B. & Scanlon, M. (2018), 'Deep learning at the shallow end: Malware classification for non-domain experts', *Digital Investigation* **26**, S118–S126.

Lei, K., Fu, Q., Yang, M. & Liang, Y. (2020), 'Tag recommendation by text classification with attention-based capsule network', *Neurocomputing* **391**, 65–73.

Li, L., Ding, Y., Li, B., Qiao, M. & Ye, B. (2021), 'Malware classification based on double byte feature encoding', *Alexandria Engineering Journal* .

Liu, L., Wang, B.-s., Yu, B. & Zhong, Q.-x. (2017), 'Automatic malware classification and new malware detection using machine learning', *Frontiers of Information Technology & Electronic Engineering* **18**(9), 1336–1347.

Marastoni, N., Giacobazzi, R. & Dalla Preda, M. (2021), 'Data augmentation and transfer learning to classify malware images in a deep learning context', *Journal of Computer Virology and Hacking Techniques* pp. 1–19.

Marchisio, A., Hanif, M. A. & Shafique, M. (2019), Capsacc: An efficient hardware accelerator for capsulenets with data reuse, *in* '2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)', IEEE, pp. 964–967.

McKinney, W. et al. (2010), Data structures for statistical computing in python, *in* 'Proceedings of the 9th Python in Science Conference', Vol. 445, Austin, TX, pp. 51–56.

Mobiny, A. & Van Nguyen, H. (2018), Fast capsnet for lung cancer screening, *in* 'International Conference on Medical Image Computing and Computer-Assisted Intervention', Springer, pp. 741–749.

Mufid, M. R., Basofi, A., Al Rasyid, M. U. H., Rochimansyah, I. F. et al. (2019), Design an mvc model using python for flask framework development, *in* '2019 International Electronics Symposium (IES)', IEEE, pp. 214–219.

Naeem, H., Guo, B., Naeem, M. R., Ullah, F., Aldabbas, H. & Javed, M. S. (2019), 'Identification of malicious code variants based on image visualization', *Computers & Electrical Engineering* **76**, 225–237.

Nataraj, L., Karthikeyan, S., Jacob, G. & Manjunath, B. (2011), Malware images: visualization and automatic classification, *in* 'Proceedings of the 8th international symposium on visualization for cyber security', ACM, p. 4.

Nataraj, L., Yegneswaran, V., Porras, P. & Zhang, J. (2011), A comparative assessment of malware classification using binary texture analysis and dynamic analysis, *in* 'Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence', ACM, pp. 21–30.

Ni, S., Qian, Q. & Zhang, R. (2018), 'Malware identification using visualization images and deep learning', *Computers & Security* **77**, 871–885.

Nisa, M., Shah, J. H., Kanwal, S., Raza, M., Khan, M. A., Damaševičius, R. & Blažauskas, T. (2020), 'Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features', *Applied Sciences* **10**(14), 4966.

Niu, W., Cao, R., Zhang, X., Ding, K., Zhang, K. & Li, T. (2020), 'Opcode-level function call graph based android malware classification using deep learning', *Sensors* **20**(13), 3645.

Oliphant, T. E. (2006), *A guide to NumPy*, Vol. 1, Trelgol Publishing USA.

Patrick, M. K., Adekoya, A. F., Mighty, A. A. & Edward, B. Y. (2019), 'Capsule networks–a survey', *Journal of King Saud University-Computer and Information Sciences* .

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Pei, X., Yu, L. & Tian, S. (2020), 'Amalnet: A deep learning framework based on graph convolutional networks for malware detection', *Computers & Security* **93**, 101792.

Pektaş, A. & Acarman, T. (2017), Ensemble machine learning approach for android malware classification using hybrid features, *in* 'International Conference on Computer Recognition Systems', Springer, pp. 191–200.

Quinlan, J. R. et al. (1996), Bagging, boosting, and c4. 5, *in* 'AAAI/IAAI, Vol. 1', pp. 725–730.

Raff, E. & Nicholas, C. (2020), 'A survey of machine learning methods and challenges for windows malware classification', *arXiv preprint arXiv:2006.09271* .

Rahali, A. & Akhloufi, M. A. (2021), 'Malbert: Using transformers for cybersecurity and malicious software detection', *arXiv preprint arXiv:2103.03806* .

Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E. & Ahmadi, M. (2018), 'Microsoft malware classification challenge', *arXiv preprint arXiv:1802.10135* .

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M. & Verri, A. (2004), 'Are loss functions all the same?', *Neural Computation* **16**(5), 1063–1076.

Sabour, S., Frosst, N. & Hinton, G. E. (2017), Dynamic routing between capsules, *in* 'Advances in neural information processing systems', pp. 3856–3866.

Safa, H., Nassar, M. & Al Orabi, W. A. R. (2019), Benchmarking convolutional and recurrent neural networks for malware classification, *in* '2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)', IEEE, pp. 561–566.

Schranko de Oliveira, A. & Sassi, R. J. (2019), 'Behavioral malware detection using deep graph convolutional neural networks'.

Schranko de Oliveira, A. & Sassi, R. J. (2020), 'Chimera: An android malware detection method based on multimodal deep learning and hybrid analysis'.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Tang, M., Alazab, M. & Luo, Y. (2017), 'Big data for cybersecurity: Vulnerability disclosure trends and dependencies', *IEEE Transactions on Big Data* .

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B. & Zheng, Q. (2020), 'Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture', *Computer Networks* **171**, 107138.

Vasan, D., Alazab, M., Wassan, S., Safaei, B. & Zheng, Q. (2020), 'Image-based malware classification using ensemble of cnn architectures (imcec)', *Computers & Security* p. 101748.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), 'Attention is all you need', *arXiv preprint arXiv:1706.03762* .

Venkatraman, S. & Alazab, M. (2018), 'Use of data visualisation for zero-day malware detection', *Security and Communication Networks* **2018**.

Venkatraman, S., Alazab, M. & Vinayakumar, R. (2019), 'A hybrid deep learning image-based analysis for effective malware detection', *Journal of Information Security and Applications* **47**, 377–389.

Verma, V., Muttoo, S. K. & Singh, V. (2020), 'Multiclass malware classification via first-and second-order texture statistics', *Computers & Security* **97**, 101895.

Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P. & Venkatraman, S. (2019), 'Robust intelligent malware detection using deep learning', *IEEE Access* **7**, 46717–46738.

Vu, D.-L., Nguyen, T.-K., Nguyen, T. V., Nguyen, T. N., Massacci, F. & Phung, P. H. (2020), 'Hit4mal: Hybrid image transformation for malware classification', *Transactions on Emerging Telecommunications Technologies* **31**(11), e3789.

Xiao, G., Li, J., Chen, Y. & Li, K. (2020), 'Malfcs: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks', *Journal of Parallel and Distributed Computing* **141**, 49–58.

Yan, J., Qi, Y. & Rao, Q. (2018), 'Detecting malware with an ensemble method based on deep neural network', *Security and Communication Networks* **2018**.

Ye, Q., Fan, X., Fang, G., Bie, H., Song, X. & Shankaran, R. (2020), Capsloc: A robust indoor localization system with wifi fingerprinting using capsule networks, *in* 'ICC 2020-2020 IEEE International Conference on Communications (ICC)', IEEE, pp. 1–6.

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. (2014), How transferable are features in deep neural networks?, *in* 'Advances in neural information processing systems', pp. 3320–3328.

Yuan, B., Wang, J., Liu, D., Guo, W., Wu, P. & Bao, X. (2020), 'Byte-level malware classification based on markov images and deep learning', *Computers & Security* **92**, 101740.

Yue, S. (2017), 'Imbalanced malware images classification: a cnn based approach', *arXiv preprint arXiv:1708.08042* .

Zhang, H., Tudor, B. M., Chen, G. & Ooi, B. C. (2014), 'Efficient in-memory data management: An analysis', *Proceedings of the VLDB Endowment* **7**(10), 833–836.

Zhang, X., Wu, K., Chen, Z. & Zhang, C. (2021), 'Malcaps: A capsule network based model for the malware classification', *Processes* **9**(6), 929.

Zhao, Y., Cui, W., Geng, S., Bo, B., Feng, Y. & Zhang, W. (2020), 'A malware detection method of code texture visualization based on an improved faster rcnn combining transfer learning', *IEEE Access* **8**, 166630–166641.

Zhao, Y., Xu, C., Bo, B. & Feng, Y. (2019), 'Maldeep: A deep learning classification framework against malware variants based on texture visualization', *Security and Communication Networks* **2019**.

Zhu, X., Huang, J., Wang, B. & Qi, C. (2021), 'Malware homology determination using visualized images and feature fusion', *PeerJ Computer Science* **7**, e494.

# CURRICULUM VITAE

**Personal Information**

Name Surname : AYKUT ÇAYIR

**Education**

Undergraduate Education : Computer Engineering, Kadir Has University, Istanbul, Turkey

Graduate Education : MSc. in Computer Engineering, Kadir Has University, Istanbul, Turkey

Foreign Language Skills : English

**Work Experience**

Companies and Dates :

RCE TECH March 2010 – July 2011

Kadir Has University (Teaching Assistant) September 2011 – June 2013

George Washington University (Visiting Researcher) May 2012 – October 2012

AGMLAB June 2013 – October 2013

Herevmarket October 2013 – March 2014

Kadir Has University (Teaching Assistant) September 2014 – June 2016

Huawei Technologies July 2016 – July 2018

Kadir Has University Center For Cybersecurity and Critical Infrastructure Protection July 2018 - Present