A

Major Project

On

IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING

(Submitted in partial fulfillment of the requirements for the award of degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

BY

Mohammed Abdul Haseeb(187R1A05G4)

Gardas Nithin Kumar(187R1A05E5)

Ersetpally Anudeep Goud(187R1A05E2)

Under the Guidance of

A. KIRAN KUMAR

(Assistant professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING**" being submitted by **MOHAMMED ABDUL HASEEB(187R1A05G4), GARDAS NITHIN KUMAR(187R1A05E5), EARSETPALLY ANUDEEP GOUD (187R1A05E2)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. A KIRAN KUMAR
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ____________________

ACKNOWLEGDEMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide. **Mr. A Kiran Kumar,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to **Sri. Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

MOHAMMED ABDUL HASEEB (187R1A05G4)
GARDAS NITHIN KUMAR (187R1A05E5)
EARSETPALLY ANUDEEP GOUD (187R1A05E2)

ABSTRACT

This project is titled as “Image Based Classification of Malware Using Deep Learning”. The number of malicious files detected every year are counted by millions. One of the main reasons for these high volumes of different files is the fact that, in order to evade detection, malware authors add mutation. This means that malicious files belonging to the same family, with the same malicious behavior, are constantly modified or obfuscated using several techniques, in such a way that they look like different files. In order to be effective in analyzing and classifying such large amounts of files, we need to be able to categorize them into groups and identify their respective families on the basis of their behavior. In this project, malicious software is visualized as images since its ability to capture minor changes while retaining the global structure helps to detect variations.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled as "Image Based Classification of Malware Using Deep Learning". This project provides facility to use malicious code and convert it into images and then classify them into respective classes of malware. This project uses CNN algorithm and Deep learning libraries to classify malwares.

1.2 PROJECT PURPOSE

This has been developed to facilitate the identification, retrieval of the items and information. System is built with manually exclusive features. In all cases system will specify object which are physical or on performance characteristics. They are used to give optimal distraction and other information. Data are used for identifying, accessing, storing and matching records. The data ensures that only one value of the code with a single meaning is correctly applied to give entity or attribute as described in various ways.

1.3 PROJECT FEATURES

The main features of this project are that the designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, we examine in detail the operations performed by the system and the relationships within and outside the system. One of the key questions is, "What must be done to solve the problem?". The system is viewed as a whole and the inputs are outlined. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

A detailed study of the process must be made by various techniques like Image processing, feature recognition etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

2.2 EXISTING SYSTEM

In the existing system, Signature-based detection, heuristic detection, behavior-based detection techniques are used to detect malwares. It takes longer time to identify the traits of malware as they search for specified bytes sequences into an object so that it can exceptionally a particular type of a malware. Annual reports from antivirus companies show that thousands of new malwares are created every single day. These new malwares become more sophisticated that they could no longer be detected by the traditional detection techniques.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Cannot detect zero-day or new malware since these malware signatures are not supposed to be listed into the signature database.
- Requires some part of the process has to be done manually.
- The computational complexity of malware is not clear, and the detection of malware problem is proved to be NP-complete.

In order to avoid all these limitations and increase working accuracy, the system has to be implemented efficiently.

2.3 PROPOSED SYSTEM

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides higher accuracy and efficiency. In contrast, the proposed system attempts to eliminate or reduce these difficulties up to an extent. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

System implementation is very simple. The system requires very few system resources and will operate in almost all configurations. It has got following features.

- It is much faster than existing system.
- Effective to detect new malware.
- Effective to detect different variants of the same malware.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. A system analysis must include a feasibility study of the proposed system. The purpose of this is to make sure that the company will not be burdened by the system. There are three key aspects to the feasibility study:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

Effort should be dedicated to the project, which gives the best return on investment at the earliest opportunity. One of the factors that affect the development of a new system is the cost of its development.

The following are some of the key financial questions asked during the preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

In this study, the technical feasibility of the system, that is, its technical requirements, is examined. An application developed should not be too demanding on the available technical resources. Developing the system should have modest requirements, as minimal or no changes are needed to implement it.

2.4.3 SOCIAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project is beneficial because it will satisfy the goals when it is developed and installed. Based on careful analysis of all social aspects, it is concluded that the project is socially feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS

A hardware interface specifies the characteristics of each interface between a software product and the hardware of a system. The following are some hardware requirements.

- Processor : Intel Dual Core i5
- Hard disk : 16 GB.
- RAM : 4 GB.

2.5.2 SOFTWARE REQUIREMENTS

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements:

- Operating system : Windows 7, 8, 10
- Languages : Python 3.7 with Sklearn, numpy, seaborn and keras
- Backend : Deep Learning
- IDE : Google Collaboratory Notebook

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for Image based classification of malware using Deep learning, starting from input to final prediction.

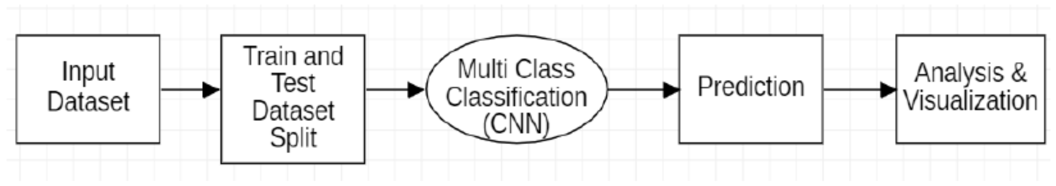

Input Dataset
Train and Test Dataset Split
Multi Class Classification (CNN)
Prediction
Analysis & Visualization

Figure 3.1: Project Architecture of Image based classification of malware using DL

3.2 MODULES DESCRIPTION

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

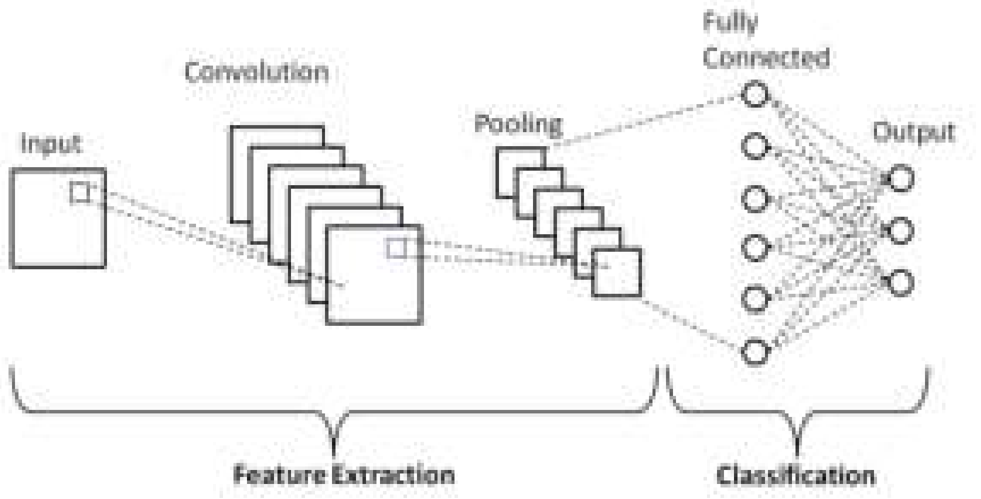


Figure 3.2: Convolutional Neural Network (CNN).

Convolution layer:

The convolution layer is the core building block of the CNN. It carries the main portion of the network’s computational load. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride.

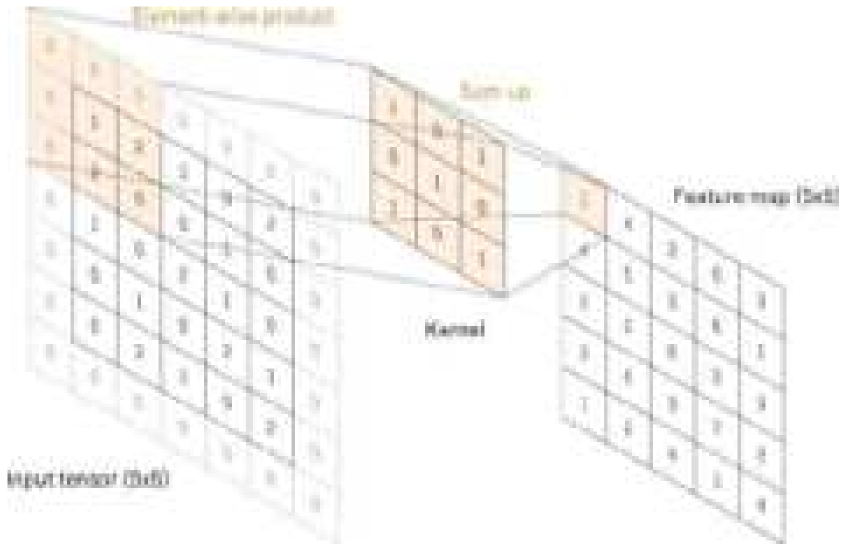


Figure 3.3: Convolutional Layer and Kernel.

Pooling layer:

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features.

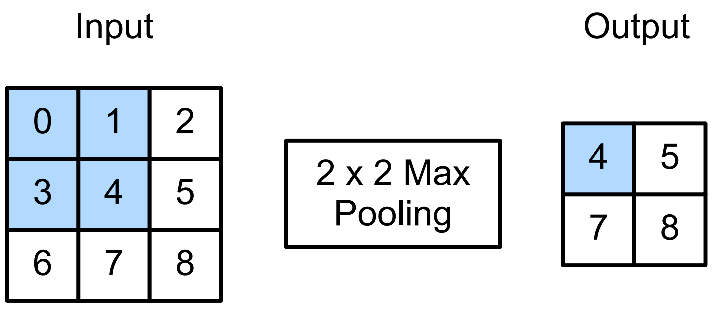


Figure 3.4: Pooling Layer (Max Pooling).

Fully connected layer:

Fully Connected Layer (also known as Hidden Layer) is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

We can add multiple such layers based on the depth to which we want to take our classification model. Note that this entirely depends on the training dataset. Output from the final hidden layer is sent to Softmax or Sigmoid function for probability distribution over final set of total number of classes.

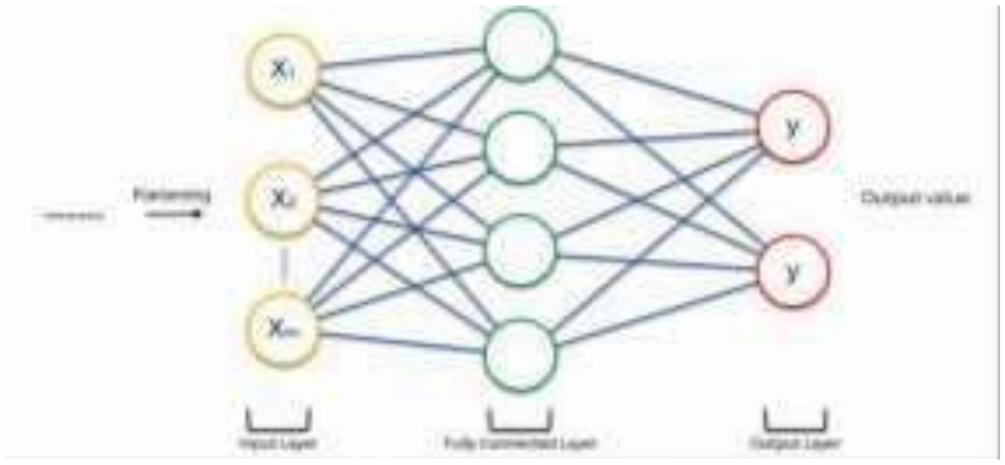


Figure 3.5: Fully Connected Layer.

Softmax Layer:

Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer.

Dropout Layer:

The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. We can apply a Dropout layer to the input vector, in which case it nullifies some of its features; but we can also apply it to a hidden layer, in which case it nullifies some hidden neurons.

Dropout layers are important in training CNNs because they prevent overfitting on the training data. If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner. This, in turn, would prevent the learning of features that appear only in later samples or batches.

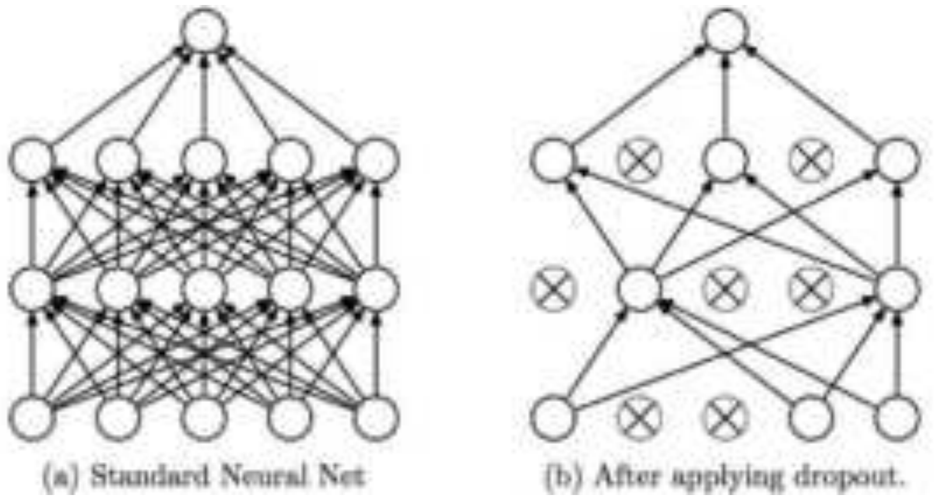
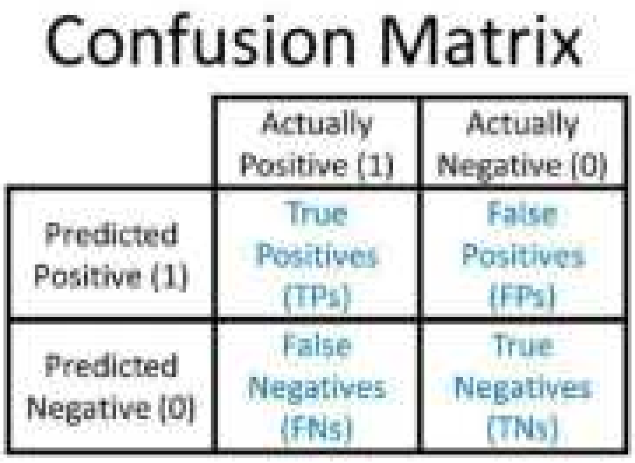


Figure 3.6: Dropout Layer.

Confusion matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 3.7: Confusion Matrix.

How to Calculate a Confusion Matrix?

1. You need a test dataset or a validation dataset with expected outcome values.
2. Make a prediction for each row in your test dataset.
3. From the expected outcomes and predictions count:
   - The number of correct predictions for each class.
   - The number of incorrect predictions for each class, organized by the class that was predicted.

These numbers are then organized into a table, or a matrix as follows:

- Expected down the side: Each row of the matrix corresponds to a predicted class.
- Predicted across the top: Each column of the matrix corresponds to an actual class.

The counts of correct and incorrect classification are then filled into the table. The total number of correct predictions for a class go into the expected row for that class value and the predicted column for that class value. In the same way, the total number of incorrect predictions for a class go into the expected row for that class value and the predicted column for that class value.

MODULE 1: PRE-PROCESSING DATASET

- Unzip the dataset: Unzipping is the act of extracting the files from a zipped single file or similar file archive. If the files in the package were also compressed (as they usually are), unzipping decompresses them.
- Data Augmentation: Data augmentation is the process of modifying, or “augmenting” a dataset with additional data. This additional data can be anything from images to text, and its use in machine learning algorithms helps improve their performance.

MODULE 2: TRAIN THE MODEL

- Train/Test split: The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms.
- Balance Class weight: Data are said to suffer the Class Imbalance Problem when the class distributions are highly imbalanced which leads to low predictive accuracy for the infrequent class.
- Compile and train model: Compile defines the loss function, the optimizer and the metrics. You need a compiled model to train (because training uses the loss function and the optimizer).

MODULE 3: ANALYSIS

- Finding Accuracy: It is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.
- Plotting Confusion matrix: A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

3.3 USE CASE DIAGRAM

In the use case diagram, we have one actor who is user/developer. The actor has right to compare the actual and predicted result of classification. These results can be clearly seen in confusion matrix.

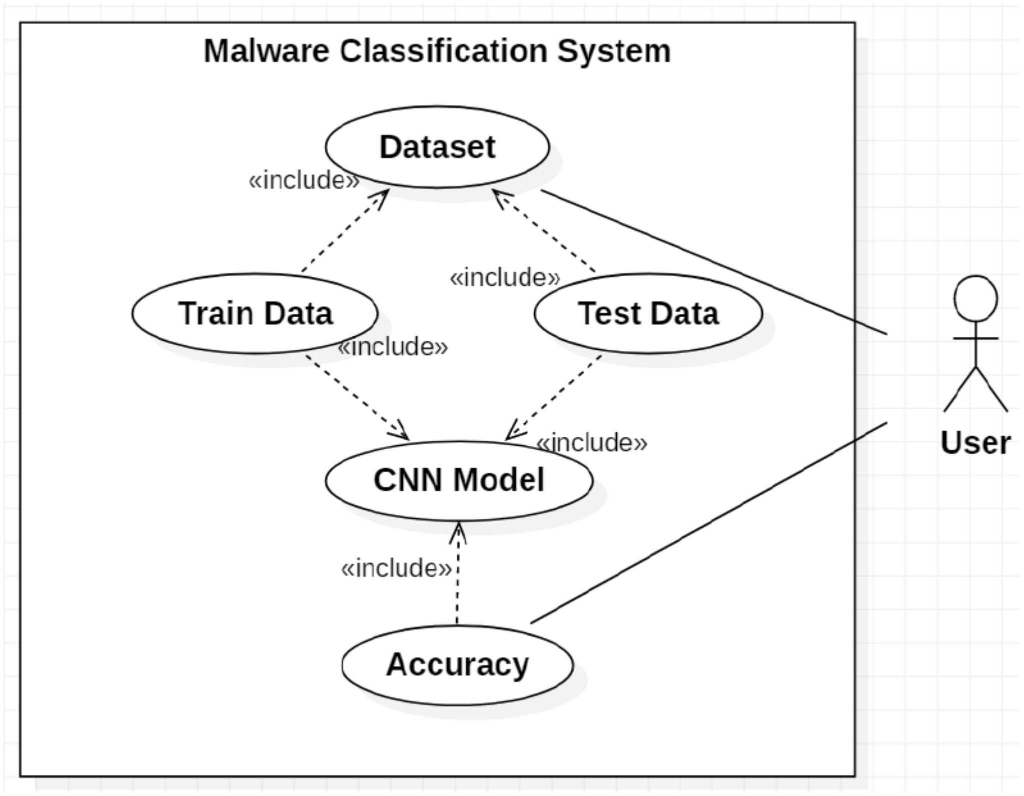


Figure 3.8: Use Case Diagram for Image based classification of malware using DL.

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

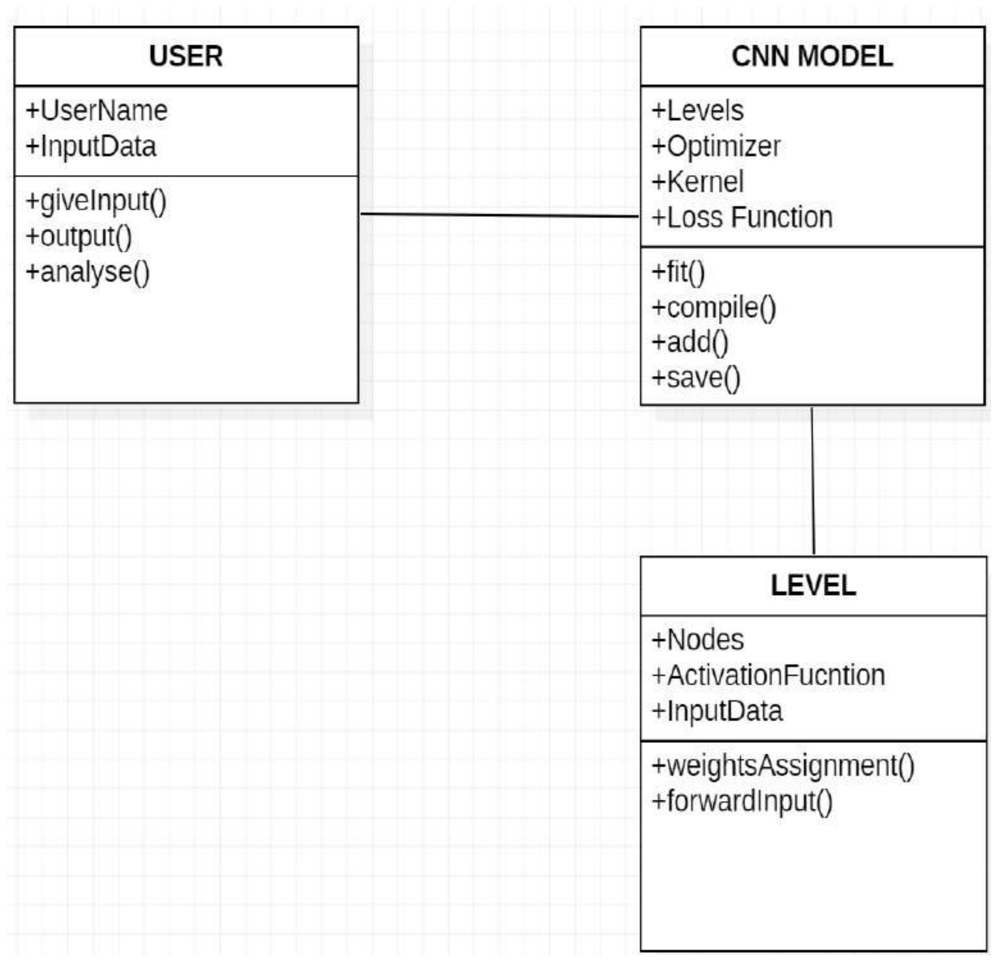


Figure 3.9: Class Diagram for Image based classification of malware using DL.

3.5 SEQUENCE DIAGRAM

Sequence diagram is a sequence of operations executed in our project.

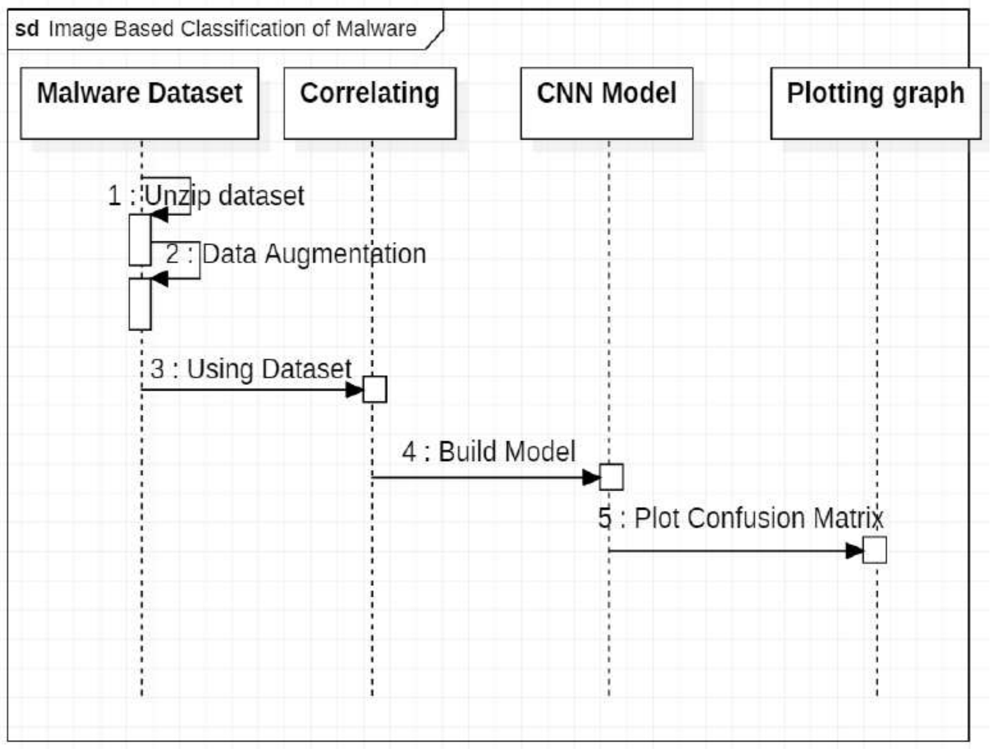


Figure 3.10: Sequence Diagram for Image based classification of malware using DL.

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

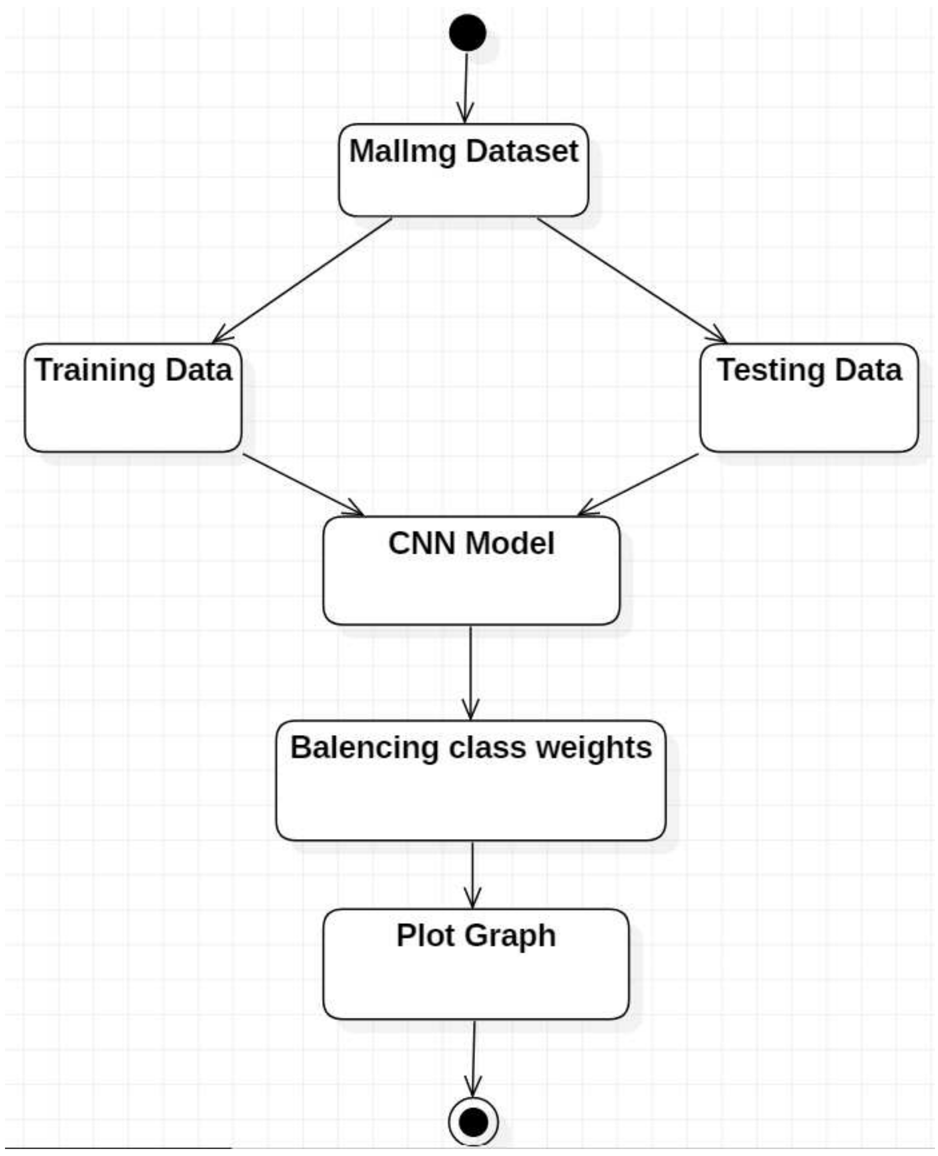


Figure 3.11: Activity Diagram for Image based classification of malware using DL.

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

MalwareClassification.ipnb:

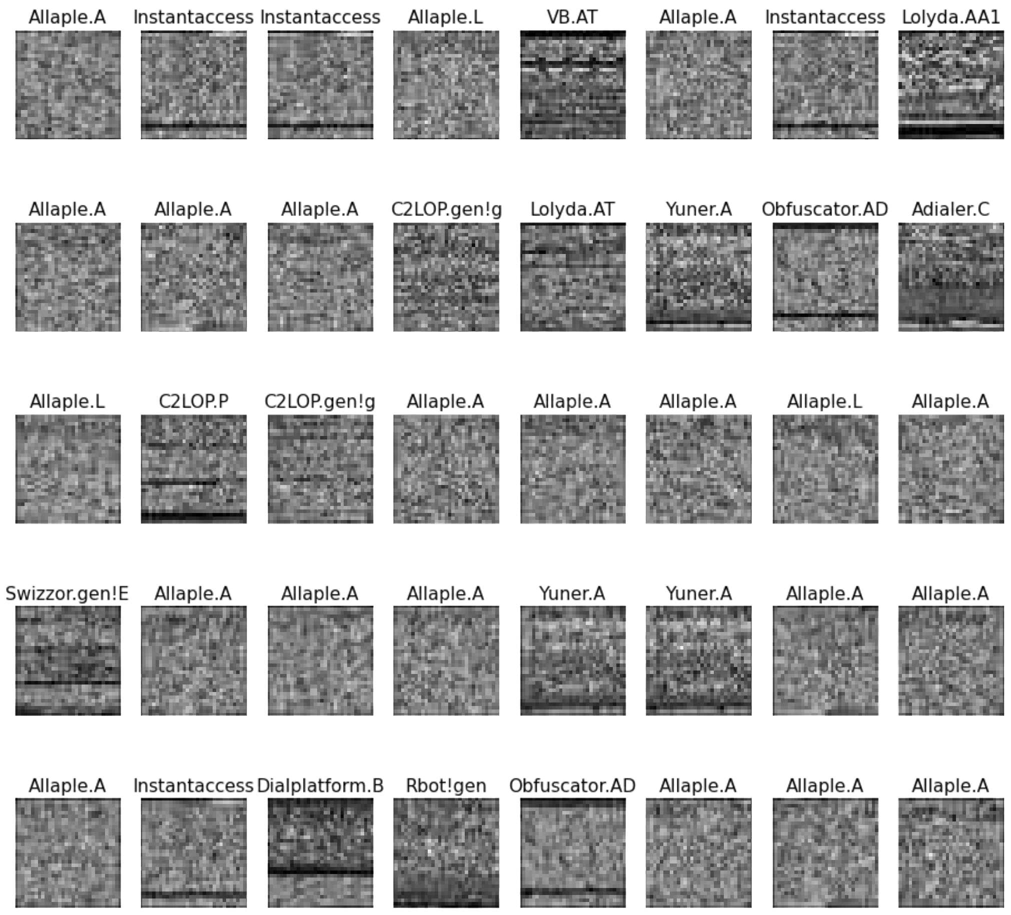
```
!pip install -q kaggle

import sys
import os
from math import log
import numpy as np
import scipy as sp
from PIL import Image
import matplotlib.pyplot as plt

from google.colab import files
files.upload()

! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/

! chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets download -d keerthicheepurupalli/malimg-dataset9010

!unzip malimg-dataset9010.zip

from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split

path_root = "/content/dataset_9010/dataset_9010/malimg_dataset/train"
batches = ImageDataGenerator().flow_from_directory(directory=path_root,
                                                   target_size=(64,64),
                                                   batch_size=10000)
batches.class_indices

imgs, labels = next(batches)
imgs.shape
labels.shape

def plots(ims, figsize=(20,30), rows=10, interp=False, titles=None):
    if type(ims[0]) is np.ndarray:
        ims = np.array(ims).astype(np.uint8)
        if (ims.shape[-1] != 3):
            ims = ims.transpose((0,2,3,1))
    f = plt.figure(figsize=figsize)
    cols = 10
    for i in range(0,50):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None:
            sp.set_title(list(batches.class_indices.keys())[np.argmax(titles[i])], fontsize=16)
        plt.imshow(ims[i], interpolation=None if interp else 'none')

plots(imgs, titles = labels)
classes = batches.class_indices.keys()
perc = (sum(labels)/labels.shape[0])*100
plt.xticks(rotation='vertical')
plt.bar(classes,perc)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(imgs/255.,labels, test_size=0.3)
X_train.shape
X_test.shape
y_train.shape
y_test.shape

import keras
from keras.models import Sequential, Input, Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization

num_classes = 25
def malware_model():
    Malware_model = Sequential()
    Malware_model.add(Conv2D(30, kernel_size=(3, 3),
                                                activation='relu',
                                                input_shape=(64,64,3)))

    Malware_model.add(MaxPooling2D(pool_size=(2, 2)))
    Malware_model.add(Conv2D(15, (3, 3), activation='relu'))
    Malware_model.add(MaxPooling2D(pool_size=(2, 2)))
    Malware_model.add(Dropout(0.25))
    Malware_model.add(Flatten())
    Malware_model.add(Dense(128, activation='relu'))
    Malware_model.add(Dropout(0.5))
    Malware_model.add(Dense(50, activation='relu'))
    Malware_model.add(Dense(num_classes, activation='softmax'))
    Malware_model.compile(loss='categorical_crossentropy', optimizer = 'adam',
                                                                        metrics=['accuracy'])
    return Malware_model

Malware_model = malware_model()

Malware_model.summary()
y_train
y_train_new = np.argmax(y_train, axis=1)
y_train_new
```

```
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight(class_weight='balanced',
                                                  classes = np.unique(y_train_new),
                                                  y = y_train_new)
                                                  class_weight_dict = dict(enumerate(class_weights))
class_weight_dict

Malware_model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20,
                                                  class_weight=class_weight_dict)

scores = Malware_model.evaluate(X_test, y_test)
print('Final CNN accuracy: ', scores[1])

import numpy as np
import pandas as pd
y_pred = np.argmax(Malware_model.predict(X_test), axis=-1)
y_pred
y_test2 = np.argmax(y_test, axis=1)
y_test2

from sklearn import metrics
c_matrix = metrics.confusion_matrix(y_test2, y_pred)

import seaborn as sns
def confusion_matrix(confusion_matrix, class_names, figsize = (10,7), fontsize=14):
    df_cm = pd.DataFrame(confusion_matrix, index=class_names, columns=class_names, )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")
    heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0,
                                                                 ha='right',
                                                                 fontsize=fontsize)
    heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=45,
                                                                 ha='right',
                                                                 fontsize=fontsize)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

class_names= batches.class_indices.keys()

import pandas as pd
confusion_matrix(c_matrix, class_names, figsize = (20,7), fontsize=14)
```

5. SCREENSHOT

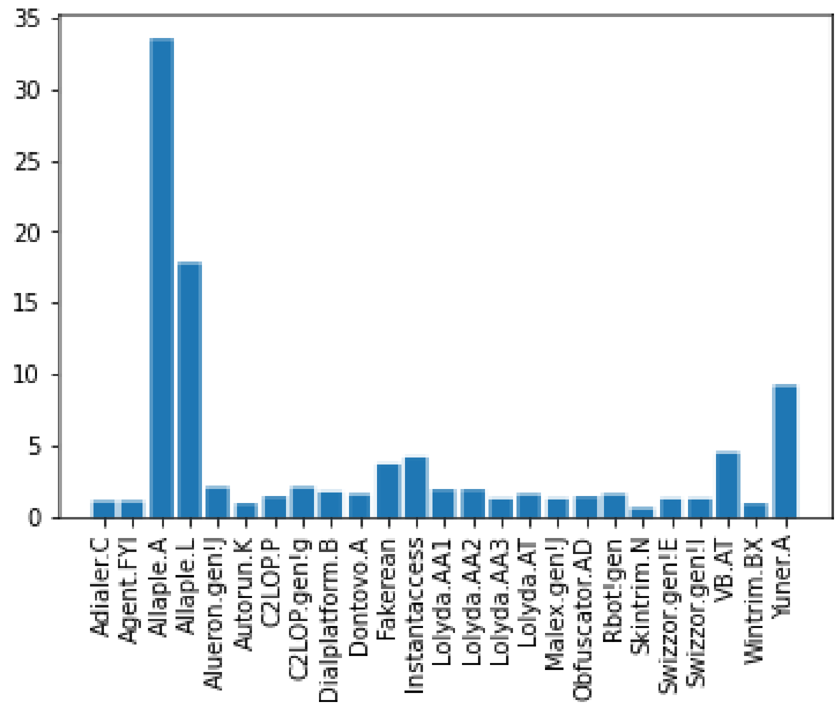
5. SCREENSHOT

5.1 SAMPLE DATASET



Screenshot 5.1: Sample Dataset of malimg-dataset9010.

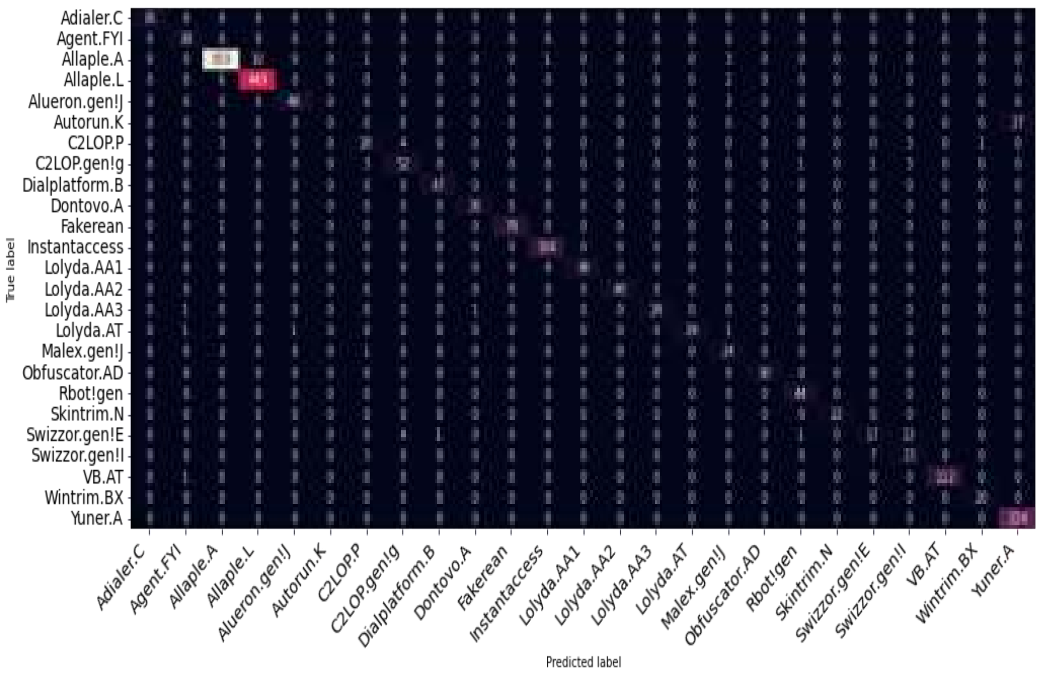
5.2 DATASET OVERVIEW



Screenshot 5.2: Dataset overview of malimg-dataset9010.

5.3 CONFUSION MATRIX

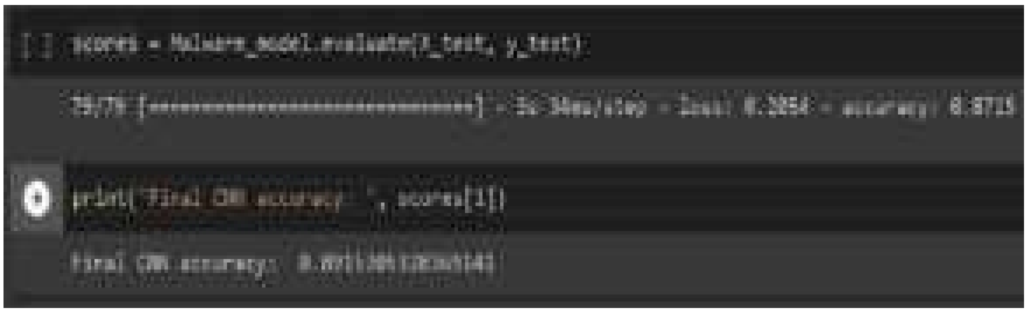
Confusion Matrix is used for result analysis, it plots a graph between actual and predicted output.



Screenshot 5.3: Confusion Matrix of malware model.

5.4 ACCURACY

The proposed model uses CNN multi-class classifier to achieves 89% accurate results with minimal rate of misclassification in less time and computational resources.



```
[ ] scores = Malware_model.evaluate(X_test, y_test)

    79/79 [==============================] - 5s 34ms/step - loss: 0.2054 - accuracy: 0.8715

print('Final CNN accuracy: ', scores[1])

    Final CNN accuracy:  0.87[illegible]
```

Screenshot 5.4: Recorded Accuracy of malware model.

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit tests having several test cases that ensure that your internal program logic works correctly and that your program inputs produce valid output. All decision branches and internal code flows need to be validated. This is a test of the individual software units of your application. This is done after the completion of a single unit before integration. This is knowledge about its design and is an invasive structural test. Unit tests run basic tests at the component level to test specific business processes, applications, and system configurations. Unit tests ensure that every path in a business process corresponds exactly to the documented specification and contains well-defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. The tests are event driven and are interested in the basic results of the screen or panel. Integration testing shows that the combination of components is correct and consistent, as indicated by the success of unit testing, even though the components are individually filled. Mixing tests are specially designed to identify problems that arise from composite parts.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems	: interfacing systems or procedures must be invoked.

The design and adjustment of performance appraisals focuses on special requirements, tasks, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

6.3.1 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1	Classification Test 1	To test classifier model	An Adialer.C image	Classified as Adialer.C
2	Classification Test 2	To test classifier model	An Agent.FYI image	Classified as Agent.FYI
3	Classification Test 3	To test classifier model	An Allaple.A image	Classified as Allaple.A
4	Classification Test 4	To test classifier model	An Allaple.L image	Classified as Allaple.L
5	Classification Test 5	To test classifier model	An Alueron.gen!J image	Classified as Alueron.gen!J
6	Classification Test 6	To test classifier model	An Autorun.K image	Wrong Classified as Allaple.A
7	Classification Test 7	To test classifier model	A C2LOP.P image	Wrong Classified as C2LOP.gen!g
8	Classification Test 8	To test classifier model	A C2LOP.gen!g image	Classified as C2LOP.gen!g
9	Classification Test 9	To test classifier model	A Dialplatform.B image	Classified as Dialplatform.B
10	Classification Test 10	To test classifier model	A Dontovo.A image	Classified as Dontovo.A

Test case ID	Test case name	Purpose	Input	Output
11	Classification Test 11	To test classifier model	A Fakerean image	Classified as Fakerean
12	Classification Test 12	To test classifier model	An Instantaccess image	Classified as Instantaccess
13	Classification Test 13	To test classifier model	A Lolyda.AA1 image	Classified as Lolyda.AA1
14	Classification Test 14	To test classifier model	A Lolyda.AA2 image	Classified as Lolyda.AA2
15	Classification Test 15	To test classifier model	A Lolyda.AA3 image	Classified as Lolyda.AA3
16	Classification Test 16	To test classifier model	A Lolyda.AT image	Classified as Lolyda.AT
17	Classification Test 17	To test classifier model	A Malex.gen!J image	Worng Classified as Lolyda.AA1
18	Classification Test 18	To test classifier model	An Obfuscator.AD image	Classified as Obfuscator.AD
19	Classification Test 19	To test classifier model	A Rbot!gen image	Classified as Rbot!gen
20	Classification Test 20	To test classifier model	A Skintrim.N image	Classified as Skintrim.N
21	Classification Test 21	To test classifier model	A Swizzor.gen!E image	Wrong Classified as Swizzor.gen!I
22	Classification Test 22	To test classifier model	A Swizzor.gen!I image	Wrong Classified as Swizzor.gen!E
23	Classification Test 23	To test classifier model	A VB.AT image	Classified as VB.AT
24	Classification Test 24	To test classifier model	A Wintrim.BX image	Classified as Wintrim.BX
25	Classification Test 25	To test classifier model	A Yuner.A image	Classified as Yuner.A

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

Malware is most commonly used for performing attacks by cyber attackers. To control the attackers from performing attacks, stealing the information, and harming computer systems, security specialists and antimalware software companies are constantly working on identifying new methods. The proposed model uses CNN multi-class classifier to achieves 89% accurate results with minimal rate of misclassification in less time and computational resources.

7.2 PROJECT FUTURE SCOPE

In future we can use other type of convolutional neural networks by downloading the modules directly into the project files. The proposed system is much faster and efficient to identify the malware. Hence it can be used in antivirus software to determine malicious files. This model can combinedly used with other models to detect malware. This model can grow if we can combine other datasets.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 GITHUB REPOSITORY LINK

https://github.com/haseeb09552/B.Tech-Major-project

https://github.com/Gardasnithinkumar/Majorproject

https://github.com/EAnudeepGoud/B.Tech-Major-Project

8.2 REFERENCES

[1] Gibert, D., Mateu, C., & Planes, J. (2020). HYDRA: A Multimodal Deep Learning Framework for Malware Classification. Computers & Security, 101873. doi:10.1016/j.cose.2020.101873

[2] Alsulami, B., & Mancoridis, S. (2018). Behavioral Malware Classification using Convolutional Recurrent Neural Networks. 2018 13th International Conference on Malicious and Unwanted Software (MALWARE). doi:10.1109/malware.2018.8659358

[3] Zhao, Jing & Basole, Samanvitha & Stamp, Mark. (2021). Malware Classification with GMM-HMM Models.

[4] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., & Iqbal, F. (2018). Malware Classification with Deep Convolutional Neural Networks. 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS). doi:10.1109/ntms.2018.8328749

[5] Dang, Dennis & Di Troia, Fabio & Stamp, Mark. (2021). Malware Classification Using Long Short-Term Memory Models

8.3 WEBSITES

[1] https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf

[2] https://docs.broadcom.com/docs/istr-24-2019-en

9. JOURNAL

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING

A. Kiran Kumar*1, Mohammed Abdul Haseeb*2, Gardas Nithin Kumar*3, Ersetpally Anudeep Goud*4

*1Professor, JNTUH, Department Of Computer Science And Engineering, CMR Technical Campus, Hyderabad, Telangana, India.

*2,3,4Student, JNTUH, Department Of Computer Science And Engineering, CMR Technical Campus, Hyderabad, Telangana, India.

ABSTRACT

Malware is a threat to people in the cyber world. It steals personal information and harms computer systems. Various developers and information security specialists around the globe continuously work on strategies for detecting malware. The number of malicious files detected every year are counted by millions. From the last few years, machine learning has been investigated by many researchers for malware classification. The existing solutions require more computing resources and are not efficient for datasets with large numbers of samples. One of the main reasons for these high volumes of different files is the fact that, in order to evade detection, malware authors add mutation. This means that malicious files belonging to the same family, with the same malicious behavior, are constantly modified or obfuscated using several techniques, in such a way that they look like different files. In order to be effective in analyzing and classifying such large amounts of files, we need to be able to categorize them into groups and identify their respective families on the basis of their behavior. In this project, malicious software is visualized as images since its ability to capture minor changes while retaining the global structure helps to detect variations.

Keywords: Convolutional Neural Network, Malware, Analysis, Research, Classification, Deep Learning.

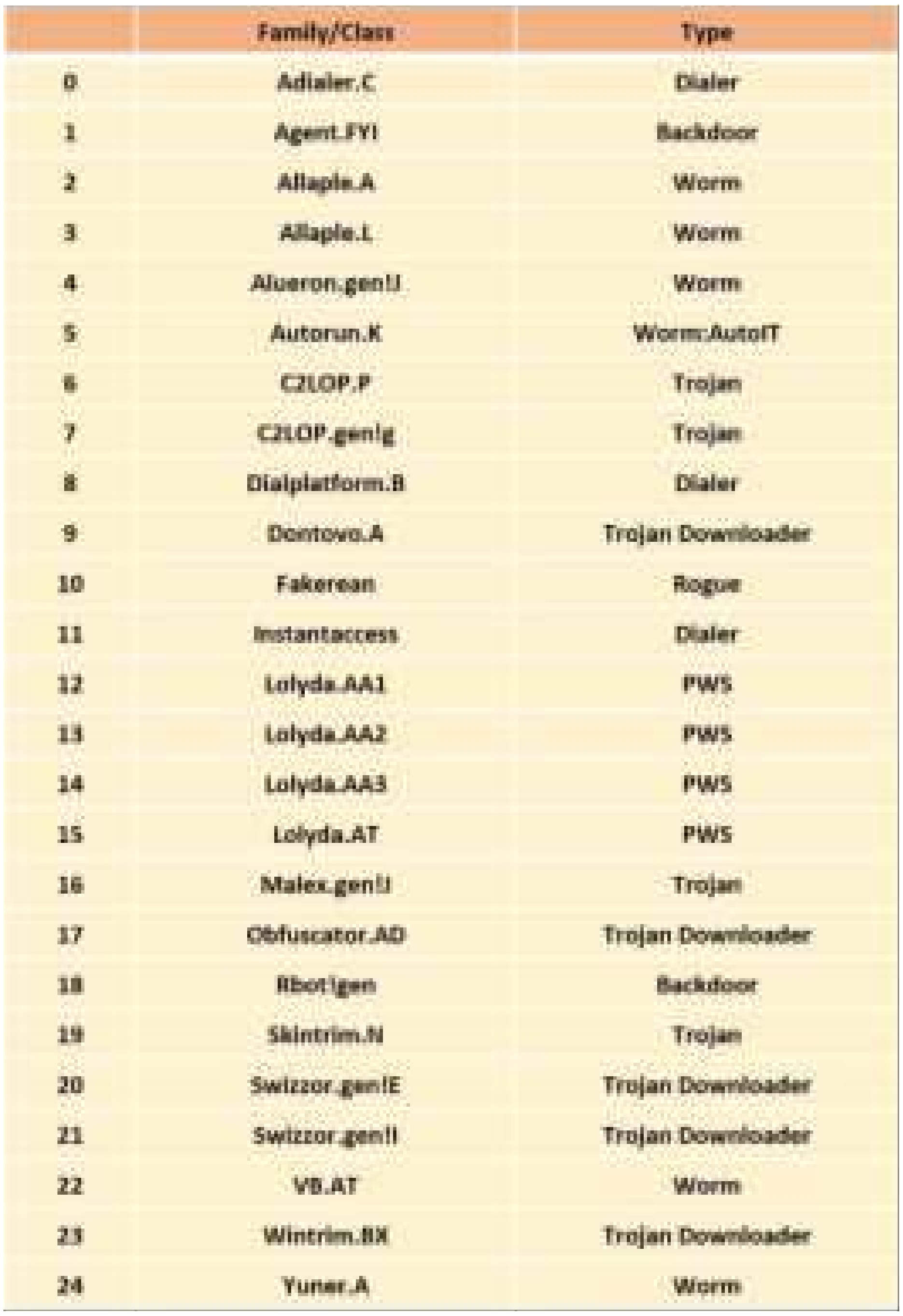
I. INTRODUCTION

Malicious code is a code that is a part of the software, system, or scripts, causing harm to the system. In the last few years, growth in Internet usage has grown significantly. With more Internet usage, there is a rise in criminal activities and hacking code injections. For doing such activities, criminals are using malicious codes to perform illegal activities on devices connected to the Internet. Creating malicious codes with online available automated tools is easy for attackers. Malware is divided into virus, worm, trojan etc. A virus needs an existing host program for it to cause harm. In general, viruses attempt to spread through programs/files on a single computer system. While worms spread through network connections to infect as many computer systems connected to the network as possible, and strengthened by the high reliance on the use of technology for humans today. A Trojan horse is a malware embedded by its designer in an application or system. Even though there are lots of security solutions available in the current market like antivirus, SSL certificate encryption, firewall protection, Signature-based detection, heuristic detection, behavior-based detection techniques, these security solutions only provide temporary protection due to their defensive mode. Hence, the new defensive solution must be updated consistently to ensure it continuously protects the information against malicious activities or software. It takes longer time to identify the traits of malware as they search for specified bytes sequences into an object so that it can exceptionally a particular type of a malware. Annual reports from antivirus companies show that thousands of new malwares are created every single day. These new malwares become more sophisticated that they could no longer be detected by the traditional detection techniques. This project provides facility to use malicious code and convert it into images and then classify them into respective classes of malware. This project uses CNN algorithm and Deep learning libraries to classify malwares. The system provides higher accuracy and efficiency. In contrast, the proposed system attempts to eliminate or reduce these difficulties up to an extent. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging.

II. METHODOLOGY

Dataset

The Dataset used in this project is "malimg_dataset9010". Dataset contains images of malware. It contains 8404 samples which belongs to 25 different families of malware.



	Family/Class	Type
0	Adialer.C	Dialer
1	Agent.FYI	Backdoor
2	Allaple.A	Worm
3	Allaple.L	Worm
4	Alueron.gen!J	Worm
5	Autorun.K	Worm:AutoIT
6	C2LOP.P	Trojan
7	C2LOP.gen!g	Trojan
8	Dialplatform.B	Dialer
9	Dontovo.A	Trojan Downloader
10	Fakerean	Rogue
11	Instantaccess	Dialer
12	Lolyda.AA1	PWS
13	Lolyda.AA2	PWS
14	Lolyda.AA3	PWS
15	Lolyda.AT	PWS
16	Malex.gen!J	Trojan
17	Obfuscator.AD	Trojan Downloader
18	Rbot!gen	Backdoor
19	Skintrim.N	Trojan
20	Swizzor.gen!E	Trojan Downloader
21	Swizzor.gen!I	Trojan Downloader
22	VB.AT	Worm
23	Wintrim.BX	Trojan Downloader
24	Yuner.A	Worm

Figure 1: Dataset overview

Convolutional Neural Network (CNN)

Deep learning is a sub-branch of the machine learning field, inspired by the structure of the brain. Deep learning techniques used in recent years continue to show an impressive performance. A convolutional neural network (CNN) is a class of deep neural networks used in image recognition problems. Convolutional neural network (CNN) is a powerful tool which is extensively utilized for image classification. The hierarchical structure and efficient feature extraction characteristics from an image make CNN a dynamic model for image classification. Coming to how CNN works, the images given as input must be recognized by computers and converted into a format that can be processed. For this reason, images are first converted to matrix format. The system determines which image belongs to which label based on the differences in images and therefore in matrices. CNN consists of three different layers that are a convolutional layer, pooling layer, and fully connected layer to perform these operations effectively.

Convolution Layer

Subheading should be Font Size- 10pt, Font Type- Cambria, justified. Convolutional layer is the base layer of CNN. It is responsible for determining the features of the pattern. In this layer, the input image is passed through a filter. The values resulting from filtering consist of the feature map. This layer applies some kernels that slide through the pattern to extract low- and high-level features in the pattern. The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. CNN is not limited to one Convolutional Layer. Conventionally, the first Conv-Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well.

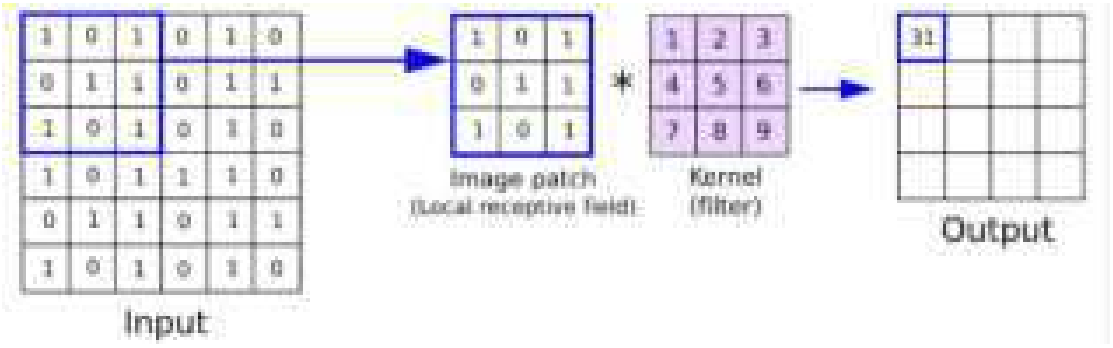


Figure 2: Convolution Layer

Pooling Layer

The second layer after the convolutional layer is the pooling layer. Pooling layer is usually applied to the created feature maps for reducing the number of feature maps and network parameters by applying corresponding mathematical computation. In this study, we used max-pooling. The max-pooling process selects only the maximum value by using the matrix size specified in each feature map, resulting in reduced output neurons. It is connected to the fully connected layer after global average pooling layer. The other intermediate layer used is the dropout layer. The main purpose of this layer is to prevent network over fitting and divergence.

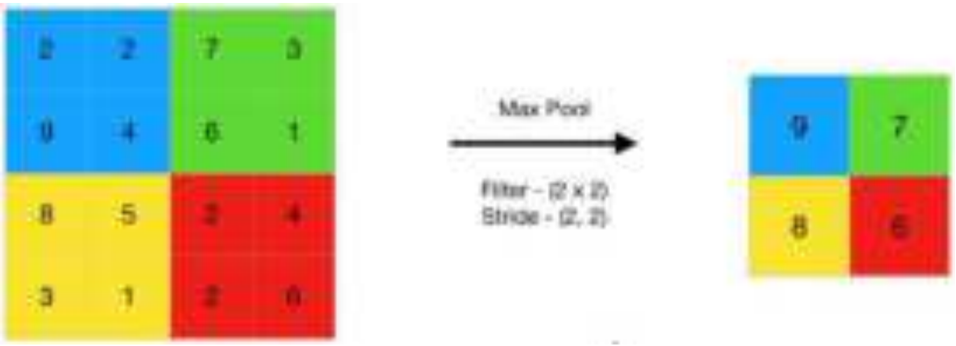


Figure 3: Max Pooling

Fully Connected Layer

The feature map matrix will be flattened into column vector (x1, x2, x3, …). The flattened output is fed to a feed-forward neural network and backpropagation is applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

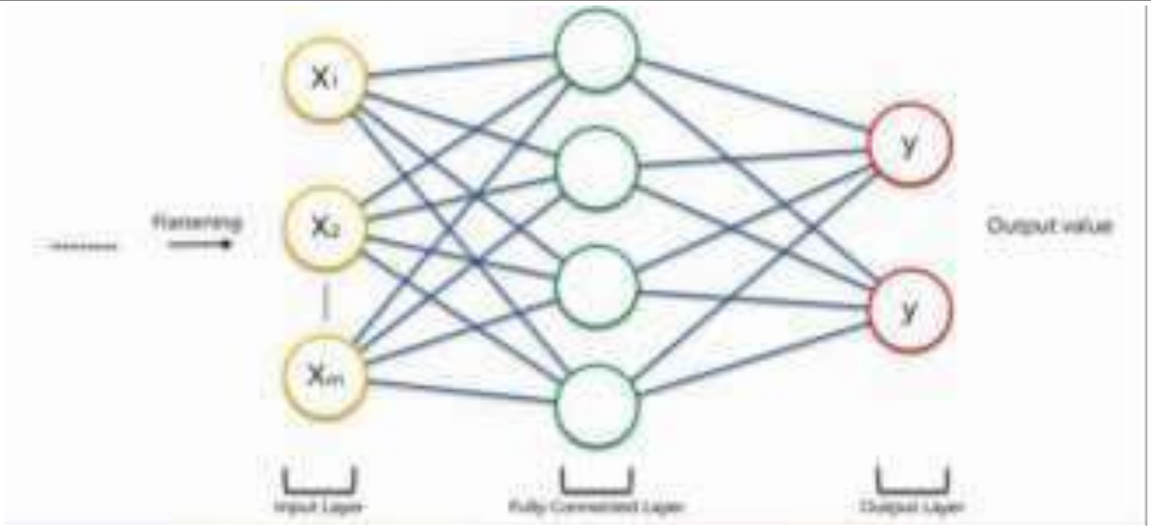


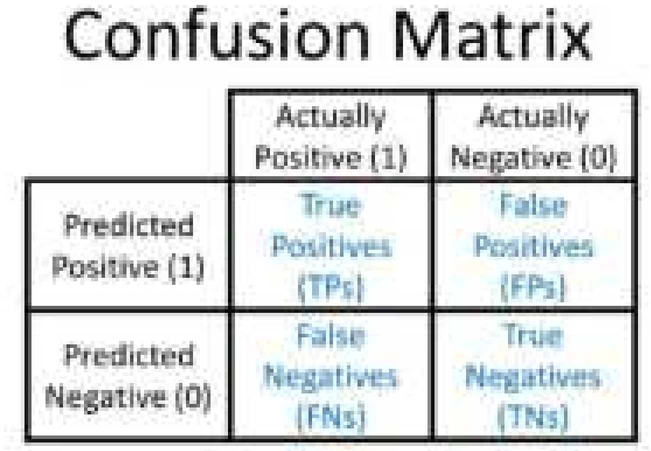
Figure 4: Fully Connected Layer

Softmax

Softmax Classifier takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying Softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying Softmax, each component will be in the interval (0, 1) and the components will add up to 1, so that they can be interpreted as probabilities.

Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 5: Confusion Matrix

III. MODELING AND ANALYSIS

CNN architecture

- Convolutional Layer : 30 filters, (3 * 3) kernel size
- Max Pooling Layer : (2 * 2) pool size
- Convolutional Layer : 15 filters, (3 * 3) kernel size
- Max Pooling Layer : (2 * 2) pool size
- DropOut Layer : Dropping 25% of neurons.
- Flatten Layer
- Dense/Fully Connected Layer : 128 neurons, Relu activation function
- DropOut Layer : Dropping 50% of neurons.

- Dense/Fully Connected Layer : 50 neurons, Softmax activation function
- Dense/Fully Connected Layer : num_class neurons, Softmax activation function

Pre-processing Dataset

- Unzip the dataset: Unzipping is the act of extracting the files from a zipped single file or similar file archive. If the files in the package were also compressed (as they usually are), unzipping decompresses them.
- Data Augmentation: Data augmentation is the process of modifying, or "augmenting" a dataset with additional data. This additional data can be anything from images to text, and its use in machine learning algorithms helps improve their performance.

Train the Model

- Train/Test split: The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms.
- Balance Class weight: Data are said to suffer the Class Imbalance Problem when the class distributions are highly imbalanced which leads to low predictive accuracy for the infrequent class.
- Compile and train model: Compile define the loss function, the optimizer and the metrics. You need a compiled model to train (because training uses the loss function and the optimizer).

Analysis

- Finding Accuracy: It is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.
- Plotting Confusion matrix: A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

IV. RESULTS AND DISCUSSION

From the confusion matrix we can observe that most of the malware families are correctly classified, but some families of malware are incorrectly classified such as Autorun.K and Swizzor.gen!l. Autorun.K is misclassified as Yuner.A probably due to underfitting whereas Swizzor.gen!l is misclassified as Swizzor.gen!E. We have achieved the accuracy of 87% with minimal rate of misclassification in less time and computational resources.

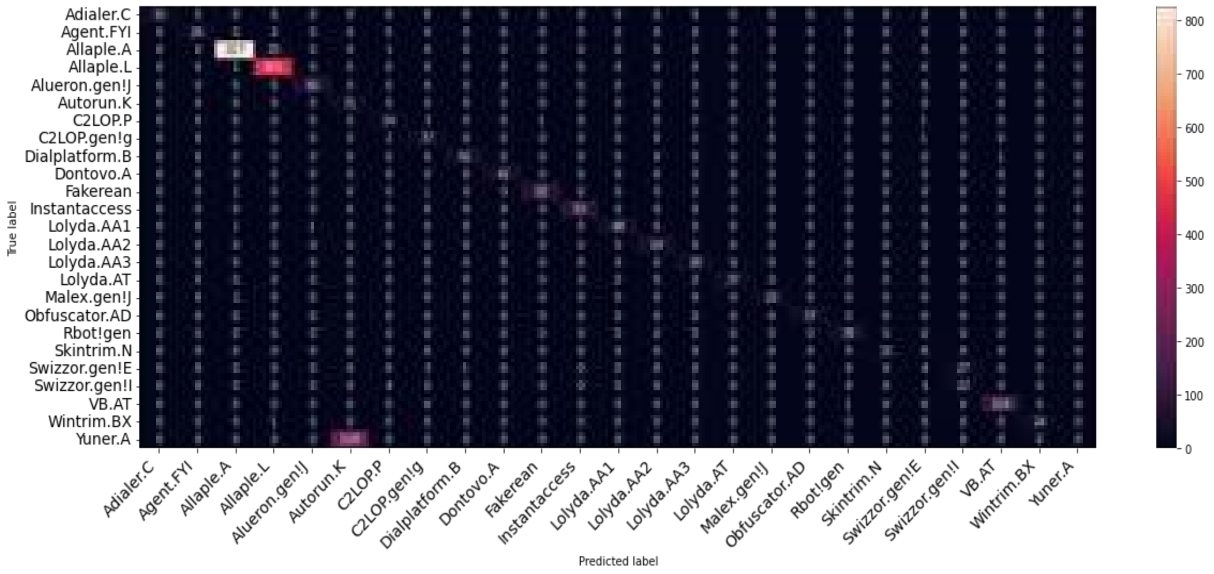


Figure 6: Observed Confusion Matrix

V. CONCLUSION

Malware is most commonly used for performing attacks by cyber attackers. To control the attackers from performing attacks, stealing the information, and harming computer systems, security specialists and antimalware software companies are constantly working on identifying new methods. The proposed model uses CNN multi-class classifier to achieves 89% accurate results with minimal rate of misclassification in less time and computational resources.

ACKNOWLEDGEMENTS

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide. **Mr. A Kiran Kumar**, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao**, **Dr. T. S. Mastan Rao**, **Mr. A. Uday Kiran**, **Mr. A. Kiran Kumar**, **Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. Ch. Gopal Reddy, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

VI. REFERENCES

[1] Gibert, D., Mateu, C., & Planes, J. (2020). HYDRA: A Multimodal Deep Learning Framework for Malware Classification. Computers & Security, 101873. doi:10.1016/j.cose.2020.101873

[2] Alsulami, B., & Mancoridis, S. (2018). Behavioral Malware Classification using Convolutional Recurrent Neural Networks. 2018 13th International Conference on Malicious and Unwanted Software (MALWARE). doi:10.1109/malware.2018.8659358

[3] Zhao, Jing & Basole, Samanvitha & Stamp, Mark. (2021). Malware Classification with GMM-HMM Models.

[4] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., & Iqbal, F. (2018). Malware Classification with Deep Convolutional Neural Networks. 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS). doi:10.1109/ntms.2018.8328749

[5] Dang, Dennis & Di Troia, Fabio & Stamp, Mark. (2021). Malware Classification Using Long Short-Term Memory Models.

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600106676

Date: 18/06/2022

Certificate of Publication

*This is to certify that author "**Mohammed Abdul Haseeb**" with paper ID "**IRJMETS40600106676**" has published a paper entitled "IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING"* ***in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022***

[illegible]

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

Google scholar issuu Academia.edu

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600106676 *Date: 18/06/2022*

Certificate of Publication

*This is to certify that author "**Gardas Nithin Kumar**" with paper ID "**IRJMETS40600106676**" has published a paper entitled "IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING"* ***in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022***

[illegible]

Editor in Chief

IRJMETS Impact Factor 6.752

International Research Journal of Modernization in Engineering Technology and Science

We Wish For Your Better Future

www.irjmets.com

Google scholar issuu Academia.edu

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600106676 Date: 18/06/2022

Certificate of Publication

*This is to certify that author “**Ersetpally Anudeep Goud**” with paper ID “**IRJMETS40600106676**” has published a paper entitled “IMAGE BASED CLASSIFICATION OF MALWARE USING DEEP LEARNING” **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022***

[illegible]

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com