

**ABSTRACT**

Malware is malicious software (e.g. viruses, worms, trojan horses, and spyware) that damages or performs harmful actions on computer systems . In this Internet-age, many malware attacks happen that pose serious security threats to financial institutions and everyday users.In order to combat the proliferation of malware, new strategies are essential to quickly identify and classify malware samples so that their behavior can be analyzed.

Machine learning approaches are becoming popular for classifying malware, however, most of the existing machine learning methods for malware classification use shallow learning algorithms (e.g. SVM). Recently, Convolutional Neural Networks (CNN), a deep learning approach, have shown superior performance compared to traditional learning algorithms, especially in tasks such as image classification.By converting malware binaries to grayscale images we can subsequently train a CNN for classification.

** TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Contents** | **Page No** |
|  |  |  |
| **1** | **INTRODUCTION** | 5 |
|  |  |  |
| 1.1 | Introduction | 5 |
|  |  |  |
| 1.2 | Objectives | 5 |
|  |  |  |
| **2** | **REQUIREMENTS AND ANALYSIS** | 7 |
|  |  |  |
| 2.1 | Existing System | 7 |
|  |  |  |
| 2.2 | Proposed System | 7 |
|  |  |  |
| 2.3 | Feasibility Study | 8 |
|  |  |  |
| 2.3.1 | Technical Feasibility | 9 |
|  |  |  |
| 2.3.2 | Financial Feasibility | 9 |
|  |  |  |
| 2.3.3 | Operational Feasibility | 9 |
|  |  |  |
| 2.3.4 | Behavioural Feasibility | 10 |
|  |  |  |
| 2.4 | Planning And Scheduling | 10 |
|  |  |  |
| **3** | **SYSTEM SPECIFICATION** | 11 |
|  |  |  |
| 3.1 | Software/Hardware specification | 11 |
|  |  |  |
| 3.2 | Functional specifications | 11 |
|  |  |  |
| 3.3 | Tools and platform used | 11 |
|  |  |  |
| 3.4 | Non-functional requirements | 12 |
|  |  |  |
| 3.4.1 | Performance requirements | 12 |
|  |  |  |
| 3.4.2 | Quality requirements | 12 |
|  |  |  |
| 3.5 | Tools and platform used | 13 |
|  |  |  |
| 3.5.1 | Deep Learning | 13 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 3.5.2 | Machine Learning | 13 |
|  |  |  |
| 3.5.3 | Streamlit | 14 |
|  |  |  |
| **4** | **PROBLEM DEFINITION** | 20 |
|  |  |  |
| **5** | **SYSTEM SPECIFICATION** | 21 |
|  |  |  |
| 5.1 | Using Word2Vec disassembler tool | 21 |
|  |  |  |
| 5.2 | Using CapsNet | 21 |
|  |  |  |
| 5.3 | Using SVM classifier | 22 |
|  |  |  |
| **6** | **DESIGN AND SPECIFICATION** | 23 |
|  |  |  |
| 6.1 | Overall Design | 23 |
|  |  |  |
| 6.2 | Developing a model | 25 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 6.3 | Screenshots of UI | 33 |
|  |  |  |
| **7** | **CODING** | 36 |
|  |  |  |
| 7.1 | Algorithm | 36 |
|  |  |  |
| 7.2 | Sample code | 37 |
|  |  |  |
| **8** | **TESTING AND IMPLEMENTATION** | 40 |
|  |  |  |
| 8.1 | Testing Various types of testing used | 40 |
|  |  |  |
| **9** | **DEPLOYMENT** | 44 |
|  |  |  |
| 9.1 | Deployment in streamlit cloud | 44 |
|  |  |  |
| **10** | **RESULTS AND DISCUSSIONS** | 45 |
|  |  |  |
| 10.1 | Advantages and Limitations | 43 |
|  |  |  |
| **11** | **CONCLUSION AND FUTURE SCOPE** | 46 |
|  |  |  |
| **12** | **BIBLIOGRAPHY** | 47 |



**Chapter 1 : INTRODUCTION**

**1.1 Introduction**

Malware is unwanted software that harms or disrupts computer systems, such as viruses, worms, trojan horses, and spyware. Malware attacks are common in this Internet age, posing major security risks to businesses and regular people alike. Figure.malware statistics displays data about malware over the previous ten years. It is obvious that the overall number of malware incidents has dramatically increased over time. For instance, Symantec reported that in 2016, more than 357 million unique malware variants were discovered.

**1.2 Objective**

The extensive use of obfuscation techniques by malware developers is one of the primary causes of this enormous volume of malware samples, which means that malicious files from the same malware family (i.e. similar code and common origin) are constantly modified and/or obfuscated.It is crucial to develop reliable malware classification approaches that are tolerant of variations of malware files that belong to the same family in order to keep up with the rapid growth of malware. I suggest a deep learning architecture for malware categorization in order to achieve this goal.

Most new viruses are variations of existing malware, according to prior study on the clas- sification of malware samples, which indicates that malware samples often belong to a family that exhibits common behaviours. As a result, the idea of developing a mechanism that can effectively categorise malware based on its family, regardless of whether it is a variety, seems extremely beneficial and a way to deal with the virus’s explosive expansion.



Figure 1.1: malware statistics

Compared to conventional methodologies, I analyse and categorise malware using a totally different way. To solve this issue, we employ a Convolutional Neural Network (CNN), a deep learning architecture. Deep learning has recently provided state-of-the-art performance for a variety of tasks in numerous domains, including bioinformatics, computer vision, speech recogni- tion, and natural language processing. CNNs have excelled particularly well in issues involving pictures. Motivated by this result, we convert the malware classification issue into a CNN-based picture classification issue.

**Chapter 2 : REQUIREMENT ANALYSIS**

**2.1 Existing System**

Software that “deliberately fulfills the harmful intent of an attacker” is referred to as malicious software or malware . These are intended to gain access to computer systems and network resources, disturb computer operations, and gather personal information without taking the consent of system’s owner, thus creating a menace to the availability of the internet, integrity of its hosts, and the privacy of its users. Malwares come in wide range of variations like Virus, Worm, Trojan-horse, Rootkit, Backdoor, Botnet, Spyware, Adware etc. These classes of malwares are not mutually exclusive meaning thereby that a particular malware may reveal the characteristics of multiple classes at the same time.

Existing system of classification is not purely based on the deep learning, and not at all based on the detection based on the greyscale images. The current system detect and classify malware based on the ode and what are the accessibility of that code. Its purely based on the code.

Classification of malware is mainly seen on antivirus softwares where the code detect what kind of malware it is and removing it from our computers. It takes so much time for finding or detect any malicious software or piece of code in it of any. Normal method of finding and classifying the malware is slower and consumes more resources.

**2.2 Proposed System**

**2.2.1 Purpose**

In order to combat the proliferation of malware, new strategies are essential to quickly identify and classify malware samples so that their behavior can be analyzed. Convolutional Neural Networks (CNN), a deep learning approach, have shown superior performance compared to traditional learning algorithms, especially in tasks such as image classification. By converting malware binaries to gray scale images we can subsequently train a CNN for classification.

**2.2.2 Overall Description**

Previous research on malware classification suggests that malware samples typically fall into a family that shares common behaviors, i.e. most new malware are variants of existing ones. Hence, the prospect of building a method that can efficiently classify malware based on its family irrespective of being a variant, seems especially fruitful and a means of dealing with the rapid growth of malware. In this paper, we take a completely different approach to analyze and classify malware compared with traditional methods. Here use a Convolutional Neural Network (CNN), a deep learning architecture, to tackle this problem. Recently, deep learning has produced state-of-the-art performance for various tasks in many fields such as natural language processing, computer vision, speech recognition, and bioinformatics. However, the capabilities for applying CNNs has not been well explored in many other fields. One field that may benefit significantly by advances in deep learning is cyber security. With the recent success of deep learning (especially CNNs) in various classification problems, I believe it is possible to classify malware with higher accuracy than any of the shallow learning algorithms such as Support Vector Machines (SVM). In particular, CNNs have been very successful in problems dealing with images. Motivated by this success, I translate malware classification problem into an image classification problem to be addressed using CNN by converting malware file to gray scale images.

**2.3 Feasibility Study**

During the system analysis, a feasibility study of the proposed system was carried out to see whether it was beneficial to the users. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product. While evaluating the existing system, many advantages and disadvantages were raised. Problematic areas are identified and information is collected. A feasibility study is to determine whether the proposed system is technically, economically and behaviourally feasible in all aspects.

The fundamental objective of a feasibility study is to assess potential applications and recommend the most viable and attractive product for development. The proposed system is regarded as financially viable if there is no loss for the organization. To find the optimum system that satisfies performance demands, a feasibility study is conducted. The analysis of the issue and gathering of all pertinent data pertaining to the product, including the various data items that would be input to the system, the processing that must be done on these data, the output data that the system must produce, as well as various constraints on the behavior of the system, are all part of the feasibility study activity.

During the feasibility analysis for this project, the following three primary areas of interest were considered.

➢ Technical Feasibility.

➢ Financial Feasibility.

➢ Operational Feasibility.

➢ Behavioral Feasibility

**2.3.1 Technical Feasibility**

Technical feasibility is the most important of all types of feasibility analysis. This study is carried out to check whether the system is technically feasible or not. It is to check the technical requirements. This study will lead a system to have high demand for the available technical resources. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

According to the feasibility analysis procedure, the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, and inputs are identified. While considering the problems of the existing system, it is sufficient to implement the new system. The proposed system can be implemented to solve issues in the existing system. It includes the evaluation of and how it meets the proposed system. This system uses streamlit (python library for creating webpage) as front-end technology, Python back-end technology and no framework is used.

**2.3.2 Financial Feasibility**

Economic analysis is performed to assess the potential system's effectiveness. The process, sometimes referred to as cost/benefit analysis, is comparing the predicted savings and benefits of a proposed system to those of the current one. In this approach, little to no money is required. The system is therefore financially viable. The firm's ability to create the system, whether its advantages should significantly outweigh its expenses, and whether the project has higher priority and profitability than other initiatives that might employ the same resources are some of the considerations related to the project's viability. There is no issue here.

**2.3.3 Operational Feasibility**

Economic analysis is performed to assess the potential system's effectiveness. The process, sometimes referred to as cost/benefit analysis, is comparing the predicted savings and benefits of a proposed system to those of the current one. In this approach, little to no money is required. The system is therefore financially viable. The firm's ability to create the system, whether its advantages should significantly outweigh its expenses, and whether the project has higher priority and profitability than other initiatives that might employ the same resources are some of the considerations related to the project's viability. There is no issue here.

**2.3.4 Behavioral Feasibility**

The response of the users to the suggested system should be estimated. Before implementing the suggested system, we need to study how users would behave. The system is deemed practical since this package is both economically and functionally practicable. It assesses and gauges user behavior or attitude regarding the creation of new systems. It aids in deciding whether the system needs extra work to inform, retrain, transfer, and change employee job status on new business practices.

**2.4 Planning And Scheduling**

The goal of the project plan is to outline every approach, process, and methodology that will be used to guarantee the timely delivery of the software that complies with the project's resource constraints. This will involve assessing and auditing the software projects and activities to ensure that they adhere to the relevant policies and standards, and reporting the findings to the software project and other appropriate managers.

**Chapter 3 : SYSTEM SPECIFICATION**

**3.1 Software/Hardware specification for development, implementation**

**3.1.1 Hardware Requirements**

* Processor : Intel Core i5
* Storage : 1 TB Hard Disk space • Memory : 8 GB RAM

**3.1.2 Software Requirements**

* Operating System : Linux/Windows
* Platform : Python
* Librarie used : tensorflow, pandas, matplotlib, numpy, sklearn,

**3.3 Functional Requirements**

The functional requirements includes all the activities or processes that should be achieved by the proposed system. It includes

* tensorflow: TensorFlow is an open source toolkit for numerical computation and large- scale machine learning developed by the Google Brain team. TensorFlow combines a va- riety of machine learning and deep learning (also known as neural networking) models and algorithms into a single paradigm. It makes use of Python to create a user-friendly front-end API for developing applications using the framework, which is then executed in high-performance C++.
* keras: Keras is an open-source software library for artificial neural networks with a Python interface. The TensorFlow library is accessed using Keras. Keras supported a variety of backends up until version 2.3, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. Only TensorFlow is supported as of version 2.4. It is user-friendly, modular, and expandable, with the goal of allowing quick experimentation with deep neural net- works. It was created as part of the ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) research project, and Fran ̧cois Chollet, a Google engineer, is the principal author and maintainer. Chollet is also the creator of the deep neural network model XCeption.
* pandas:Pandas is a widely used open source Python library for data science, data analysis, and machine learning activities. It is based on the Numpy library, which supports multi- dimensional arrays. Pandas, as one of the most popular data wrangling programmes, is normally included in every Python distribution, from those that come with your operating system to commercial vendor versions like ActiveState’s ActivePython.

**3.4 Non Functional Requirements**

**3.4.1 Performance Requirements**

* Accuracy : Accuracy in functioning and the nature of user-friendly should be maintained by the system.
* Speed : The system must be capable of offering speed.
* Low cost: This system is very cheap to implement and is also user-friendly.
* Less Time consuming: It uses very less time comparing to the existing system .
* User Friendly: This proposed system is highly user friendly they enables to create a good environment.

**3.4.2 Quality Requirements**

* Scalability : The software will meet all of the functional requirements.
* Maintainability : The system should be maintainable. It should keep backups to atone for  
     
  system failures, and should log its activities periodically.
* Reliability : The acceptable threshold for down-time should be large as possible. i.e. mean time between failures should be large as possible. And if the system is broken, time required to get the system backup again should be minimum.
* Availability: This system is easily available as the core equiments in building the sofware is easily obtained.
* High- Functionality: This system is highly functional in all environment since, They are highly adaptable.

**3.5 Tools And Platform Used**

**3.5.1 Deep Learning**

Deep learning is a subset of [machine learning](https://www.ibm.com/in-en/topics/machine-learning), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many [artificial intelligence (AI)](https://www.ibm.com/in-en/topics/artificial-intelligence) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

**3.5.2 Machine Learning**

In the real world, we are surrounded by people who are able to learn from their experiences thanks to their capacity for learning, and we also have computers or other robots that carry out our orders. But can a machine learn from past facts or experiences the same way a human does? So now the role of machine learning is revealed.

According to some, machine learning is a branch of artificial intelligence that focuses primarily on creating algorithms that enable a computer to independently learn from data and previous experiences. Arthur Samuel coined the phrase "machine learning" in 1959. In a nutshell, machine learning allows a machine to predict events without being explicitly programmed, automatically learn from data, and improve performance over time.

Machine learning algorithms create a mathematical model that aids in making predictions or judgments without being explicitly programmed with the use of sample historical data, sometimes referred to as training data. For the purpose of developing predictive models, machine learning combines statistics and computer science. Algorithms that learn from previous data are created or used in machine learning. The level of performance will increase as we supply more information.

**3.5.3 Python 3.10**

Python is an object-oriented, dynamically semantic, high-level, interpreted programming language. Rapid Application Development, as well as use as a scripting or glue language to bring existing components together, find its high-level built-in data structures, coupled with dynamic type and dynamic binding, to be particularly appealing. Python's straightforward syntax promotes readability, which lowers the expense of software maintenance. Python's support for modules and packages promotes the modularity of programmes and the reuse of code. Both the comprehensive standard library and the Python interpreter are freely distributable and are accessible in source or binary form for all popular systems. Python frequently inspires programmers to use it because of the productivity boost it offers. The edittext debug cycle lacks a compilation step because it is incredibly fast. Since a flaw or incorrect input will never result in a segmentation fault, debugging Python scripts is simple. An exception is instead raised when a mistake is found by the interpreter. When the programme misses the exception, the interpreter outputs a stack trace. It is possible to step through the code one line at a time, set breakpoints, check local and global variables, evaluate arbitrary expressions, and more with a source level debugger. Python's own debugger, a testament to Python's introspective power, was created in Python. Contrarily, adding a few print statements to the source code is frequently the fastest way to debug a programme because it allows for a short edit-test-debug cycle.

**3.5.4 Streamlit**

An open-source Python framework called Streamlit is used to create web applications for machine learning and data science. Using Streamlit, we can quickly design and launch web applications. You can use Streamlit to create apps in the same manner that you create Python code. Working on the interactive cycle of coding and watching outcomes on the web app is made simple by Streamlit.

#### Installing Streamlit

1. Make sure you have python installed in your system

2. Use the following command to install streamlit,

*pip install streamlit*

The flow of Streamlit while developing a web app

#### Running a streamlit app

First, you create a python script with streamlit commands and execute the script.

#### **Development flow**

The top-right corner of the app indicates whether to relaunch the application or not if the source code of the streamlit Python script changes. To always rerun when the source script changes, you may also choose the "Always rerun" option.

This greatly facilitates our development process because any modifications you make to your web app will take effect right away. Working with streamlit is seamless thanks to the loop between coding and seeing outcomes immediately.

#### 

#### **Data flow**

You can use Streamlit to create apps in the same manner that you create Python code. The data flow of the streamlit is separate; whenever your code is changed or something on the screen needs to be updated, the streamlit replays your Python script completely from top to bottom. This occurs when the source code is altered or when the user interacts with the widgets, such as a choose box or drop-down box.

#### **Displaying the data**

Streamlit provides you with many methods to display various types of data like arrays, tables, and data frames.

* To write a string simply use, st.write(“Your string”)
* To display a data frame use, st.dataframe method

#### **Widgets**

Streamlit offers a number of widgets, including st.selectbox, st.checkbox, st.slider, and others. If a user increases or decreases the widget, the code is rerun by the streamlit from top to bottom and the current state of the widget is assigned to the variable. During the initial run, the web app would print the string "0 squared is 0."

**3.5.4 Jupyter Notebook**

The most recent web-based interactive development environment for code, data, and notebooks is JupyterLab. Users can configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning using the interface's flexibility. A modular structure encourages expansions to increase and improve functionality.

**Notebook document**

The Jupyter Notebook App creates documents called notebook documents (or "notebooks," all lower case), which contain both computer code (like Python) and rich text components (including paragraphs, equations, diagrams, links, etc.). Both executable documents that can be used to do data analysis as well as human-readable documents holding the analysis description and the results (figures, tables, etc.) are considered notebook documents.

**Jupyter notebook app**

A server-client programme called the Jupyter Notebook App enables editing and running notebook papers from a web browser. The Jupyter Notebook App can be deployed on a remote server and accessible via the internet, or it can be run locally on a desktop without the need for an internet connection (as detailed in this paper).

The Jupyter Notebook App contains a "Dashboard" (Notebook Dashboard), a "control panel" exposing local files and letting them open notebook documents or shutting down their kernels, in addition to displaying, editing, and executing notebook documents.

**3.5.5 Libraries use**

**Numpy**

An open source library for Python called NumPy supports programming in the fields of mathematics, science, engineering, and data science. A fantastic library for performing mathematical and statistical computations is NumPy. For matrix multiplication and multi-dimensional arrays, it works flawlessly. In essence, an image is a regular NumPy array with pixels for the data points. Therefore, you can change the pixel values in a picture by using fundamental NumPy operations like slicing, masking, and clever indexing. The equivalent of arrays in Python are lists, although they take a long time to execute. The goal of NumPy is to offer array objects that are up to 50 times faster than conventional Python lists. The NumPy array object is known as ndarray and offers a number of helpful methods.

**Pandas**

Pandas is a Python open source library that is mostly utilized for machine learning and data science jobs. It is constructed on top of the multi-dimensional array support provided by the Numpy package. Being one of the most widely used data wrangling packages, Pandas integrates well with a variety of other data science modules within the Python ecosystem and is frequently available in all Python distributions, including those that come with your operating system and those sold by commercial vendors like ActiveState's ActivePython.

A Pandas data structure called a data frame is a two-dimensional, labeled data structure that resembles a spreadsheet's columns and rows or a SQL table. A DataFrame's columns can hold a variety of data kinds. The "loc" and "iloc" functions are part of the Pandas Data Frame syntax, and examples are data frame.loc[] and data frame.iloc[]. Both functions, "loc" for access by labels and "iloc" for access by position, or numerical indices, are used to access rows and/or columns.

**Matplotlib**

For use with Python and its NumPy numerical mathematics extension, Matplotlib is a plotting library. It offers an object-oriented API for embedding plots into programmes that make use of all-purpose GUI toolkits, such as Tkinter, wxPython, Qt, or GTK. Although its use is discouraged, there is a procedural "pylab" interface built on a state machine (similar to OpenGL) that is intended to closely mimic that of MATLAB. Matplotlib is a library used by SciPy.

**Seaborn**

Python has a module named Seaborn for creating statistical visuals. It is based on matplotlib and tightly integrated with pandas data structures. Seaborn aids in data exploration and comprehension. With its dataset-oriented, declarative API, you can concentrate on the meaning of the various plot parts rather than the specifics of how to render them. When managing pandas dataframes, Seaborn is more at ease. Python provides stunning graphics using simple sets of techniques. On top of the fundamental Python visualization package Matplotlib, Seaborn is constructed.However, Seaborn comes with some very important features .

➢ Built in themes for styling matplotlib graphics

➢ Visualizing univariate and bivariate data

➢ Fitting in and visualizing linear regression models

**3.6 DATASET USED**

This dataset has a total of 9,339 malware samples that are represented as grayscale images. Each malware sample in the dataset belongs to one of the 30 malware families/classes. Also, the number of samples belonging to a malware family vary across the dataset. In our experiments, we randomly select 90 % of malware samples in a family for training and the remaining 10 % for testing. At the end, we have 8,394 malware samples for training and 945 samples for testing.

**Chapter 4 : PROBLEM DEFINITION**

Software that “deliberately fulfills the harmful intent of an attacker” is referred to as ma- licious software or malware. These are intended to gain access to computer systems and network resources, disturb computer operations, and gather personal information without taking the con- sent of system’s owner, thus creating a menace to the availability of the internet, integrity of its hosts, and the privacy of its users. Malwares come in wide range of variations like Virus, Worm, Trojan-horse, Rootkit, Backdoor, Botnet, Spyware, Adware etc. These classes of malwares are not mutually exclusive meaning thereby that a particular malware may reveal the characteristics of multiple classes at the same time. Malware is one of the most terrible and major security threats facing the Internet today.

According to a survey, conducted by FireEye in June 2013, 47 % of the organizations ex- perienced malware security incidents/network breaches in the past one year. The malwares are continuously growing in volume (growing threat land-scape), variety (innovative malicious meth- ods) and velocity (fluidity of threats) . These are evolving, becom-ing more sophisticated and using new ways to target computers and mobile devices. McAfee catalogs over 100,000 new malware samples every day means about 69 new threats every minute or about one threat per second. With the increase in readily available and sophisticated tools, the new generation cyber threats/attacks are becoming more targeted, persistent and unknown. The advanced malwares are targeted, unknown, stealthy, personalized and zero day as compared to the traditional mal- wares which were broad, known, open and one time. Once inside, they hide, replicate and disable host protections. After getting installed, they call their command and control servers for further instructions, which could be to steal data, infect other machines, and allow reconnaissance

Attackers exploit vulnerabilities in web services, browsers and operating systems, or use so- cial engineering techniques to make users run the malicious code in order to spread malwares. Malware authors use obfuscation techniques like dead code insertion, register reassignment, sub- routine reordering, instruction substitution, code transposition, and code integration to evade detection by traditional defenses like firewalls, antivirus and gateways which typically use sig- nature based techniques and are unable to detect the previously unseen malicious executables . Commercial antivirus vendors are not able to offer immediate protection for zero day malwares as they need to classify these to create their signatures.

**Chapter 5 : SYSTEM SPECIFICATION**

The classification of malware is a vital problem since ages. As a part of my literature review I went through various papers and presentations on this topic. The quick summary of my findings are specified in this chapter.

**5.1 Using Word2Vec disassembler tool**

Word2vec disassembler tool is used for vector classification. After feeding the Word2Vec algorithm with the data, it will learn a vector representation for each word. This by itself, however, is still not enough to be used as features for text classification as each record in our data is a document not a word.Perhaps the biggest problem with word2vec is the inability to handle unknown or out-of-vocabulary (OOV) words. If your model hasn’t encountered a word before, it will have no idea how to interpret it or how to build a vector for it. You are then forced to use a random vector, which is far from ideal.

**5.2 Using 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 assumption that CapsNet is a weak learner increases the performance of a single CapsNet for two different wellknown malware datasets, which are highly imbalanced.

In machine learning, support vector machines (SVMs, also support vector networks) are super- vised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

**5.3 Using SVM classifier**

In machine learning, support vector machines (SVMs, also support vector networks) are super- vised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

**Chapter 6 : DESIGN AND SPECIFICATION**

The proposed system is used to classify malware automatically by using a pre-trained model. The model is trained using CNN with 10 layer architecture.

**6.1 Overall Design**

The suggested system is built on a client-server model. That is, there is a client and a server component to the malware classification system. The user enters the gray scale image that will be assessed into the client component. The client receives the assessed result after passing the input to the server. The client side is constructed with Streamlit and Python, while the server side is built with Python.

**6.1.1 System Design**

The system is accessible over the internet. The user provides input via a web page, which is then sent to a python application running on the server side. On the input data, the server ap- plication performs activities such as preprocessing and feature extraction. The input is evaluated using the pre-trained model based on the results of these steps.

The model was built ulilising combined data from Malimg dataset and grey scale converted Microsoft big 2015 malware dataset. we then used this combined dataset to create a classifier for this assignment.

For classification a deep convolutional neural network (CNN) architecture for malware classi- fication is used, which is generic in nature, unlike traditional methods. Existing techniques that achieve high accuracy are often tailored for a specific dataset. In contrast, the proposed approach is data independent and learns the discriminative representation from the data itself rather than depending on hand-crafted feature descriptors.

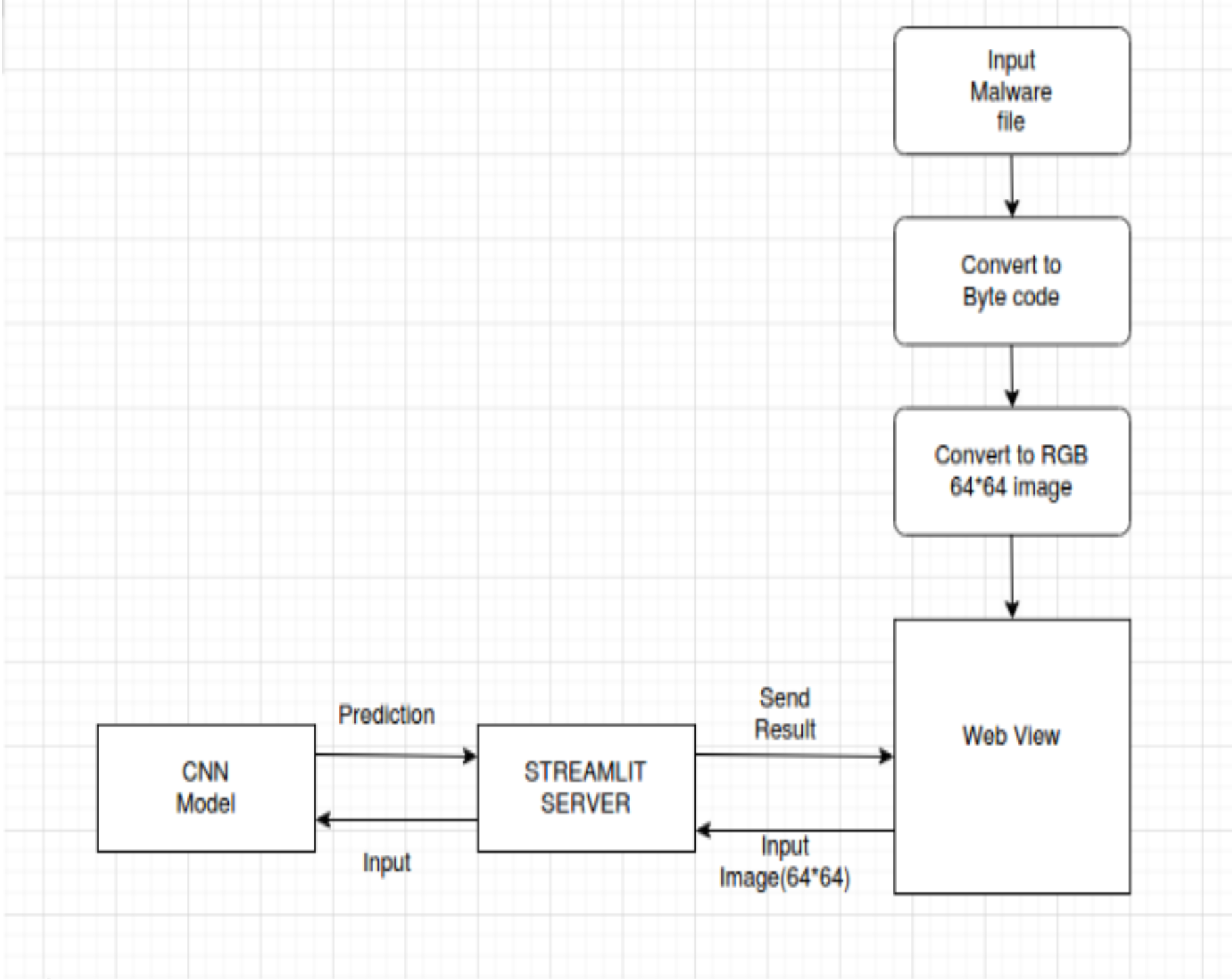


Figure 5.1: Flow diagram

**6.1.2 Methodology**

This project is divided into Three sections. The first step is to convert the Microsoft big2015 dataset malware file to gray scale image , the second is to build the CNN model by using the combined malimg and big2015 dataset and the third step is to build a UI that will interact with the pre-trained model.

**6.2 Developing the Model**

Preprocessing, the algorithm, and feature extraction are the primary processes in the model.

**6.2.1 Data Pre-processing**

* Malimg Dataset:  
     
  This dataset has a total of 9,339 malware samples that are represented as grayscale images. Each malware sample in the dataset belongs to one of the 25 malware families/classes. Also, the number of samples belonging to a malware family vary across the dataset. In our experiments, we randomly select 90 % of malware samples in a family for training and the remaining 10 % for testing. At the end, we have 8,394 malware samples for training and 945 samples for testing.

 malimg dataset

* Microsoft big2015 malware dataset: In 2015, Microsoft hosted a Kaggle competition for   
  malware classification . In this challenge, Microsoft released a huge dataset (almost half a terabyte when uncompressed) consisting of 21,741 malware samples. This dataset is divided in two parts, 10,868 samples for training and the other 10,873 samples for testing. Each malware sample belongs to one of 9 different malware families. Like the Malimg dataset, the distribution of malware samples over classes in the training data is not uniform and the number of malware samples of some families significantly outnumbers the samples of other families. There are two files that represent each malware sample, .bytes file that contains the raw hexadecimal representation of the file’s binary content with the executable headers removed and .asm file that contains the disassembled code extracted by the IDA disassembler tool. In our experiments, we only use the .bytes files to generate the malware images.

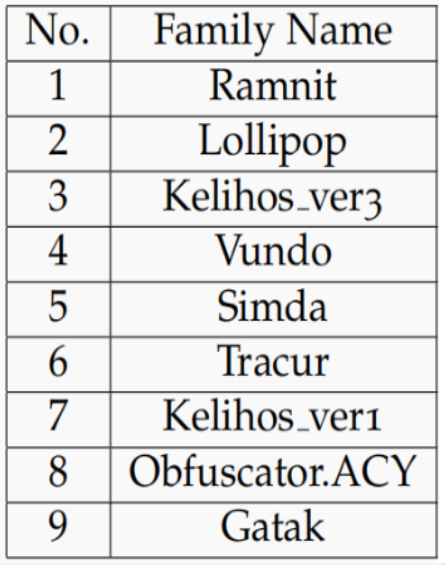


Figure 5.3: microsoft dataset

A given malware binary file is first read in a vector of 8-bits unsigned integers. Next, the binary value of each component of this vector is converted to its equivalent decimal value (e.g. the decimal value for [00000000] in binary is [0] and for [11111111] is [255]) which is then saved in a new decimal vector representative of the malware sample. At last, the resulting decimal vector is reshaped to a 2D matrix and visualized as a gray scale image.

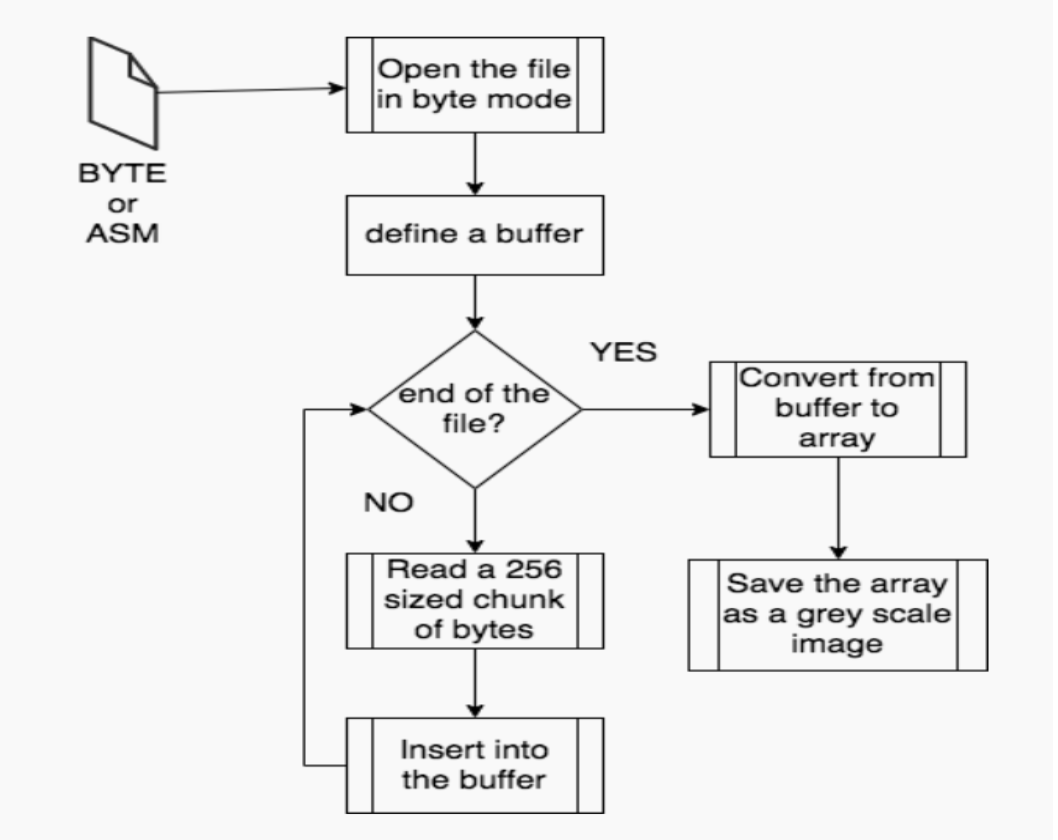
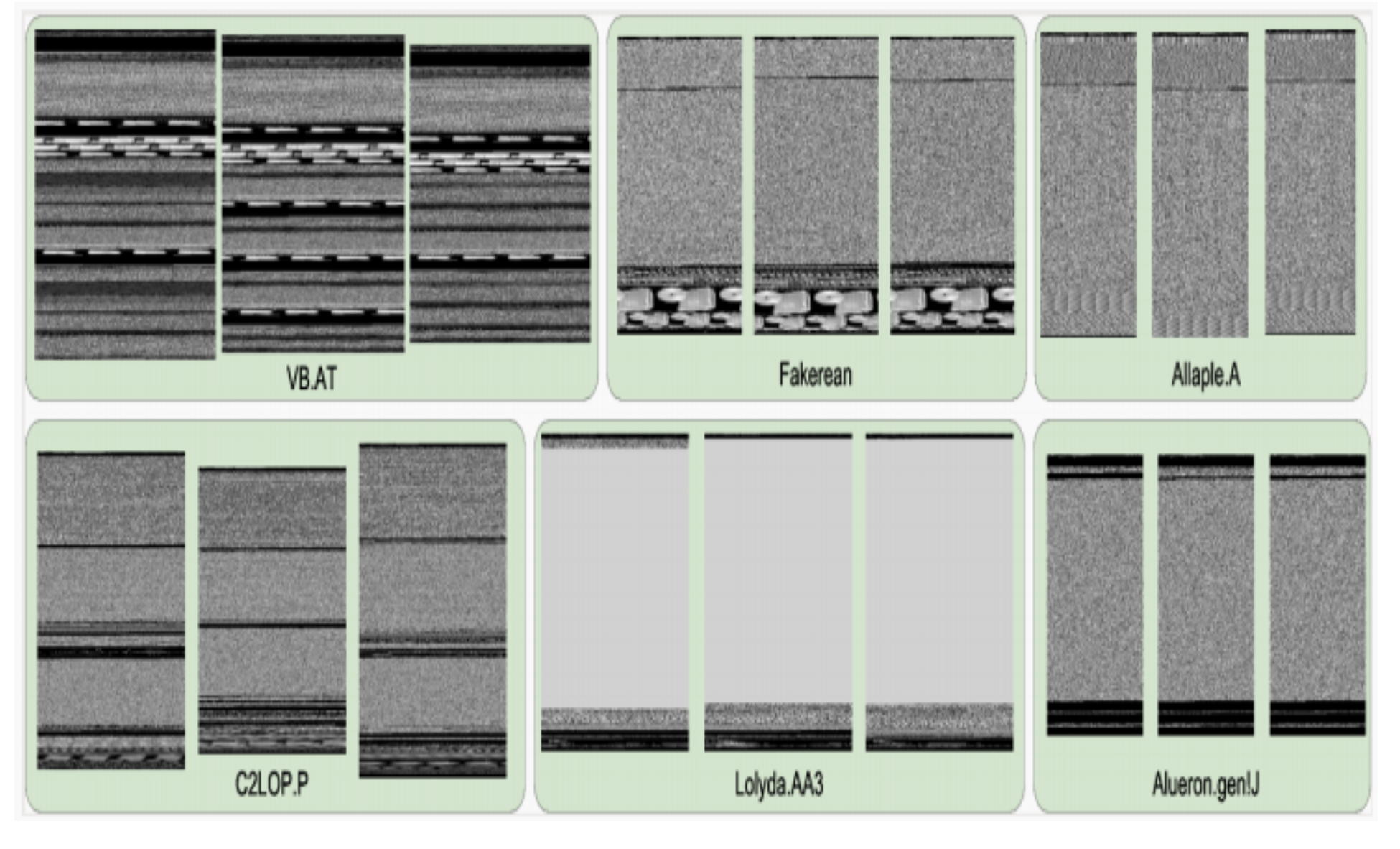


Figure 5.4: Gray scale image conversion

Selecting width and height of the 2D matrix (i.e. the spatial resolution of the image) mainly depends on the malware binary file size. Malware variants belonging to same family usually have similar texture (i.e. visual appearance). below figure provides some examples that support this observation.

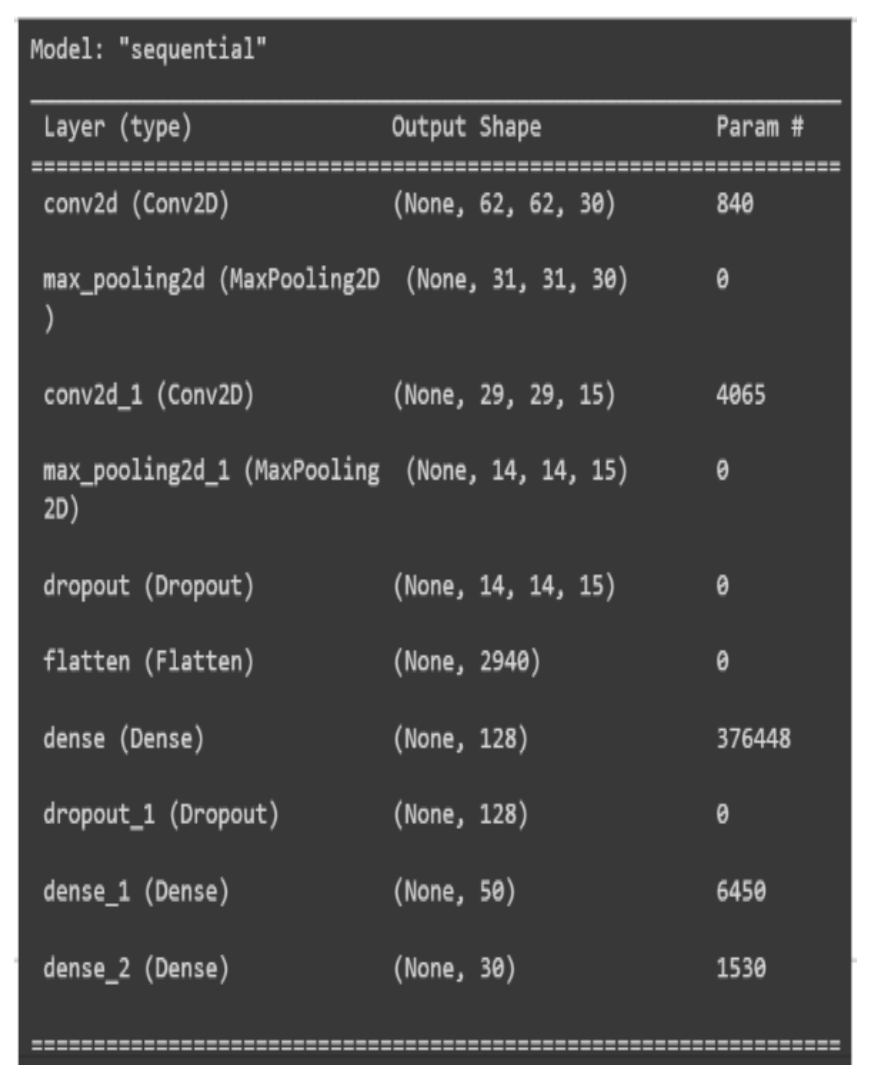
 Figure 5.5: same family malware

**6.2.2 Classification**

CNN model is used for the classification. A Convolution Neural Network (CNN) is a feed- forward neural network that is biologically inspired, specifically by the organization of animal visual cortex . CNN is the current state-of-the-art neural network architecture for image classifi- cation problem. CNN is comprised of neurons with learnable weights and biases. CNNs mainly consist of the following three components :

* Convolutional layers: These layers apply a certain number of convolution operations (linear filtering) to the image in sequence. Typically these filters extract edge, color, and shape information from the input image. Basically the filters operate on subregions of an image and perform computation such that it produces a single value as output for each subre- gion. The output of this layer is typically forwarded to a non-linear function (called ReLU activation).
* Pooling layers: This layer is responsible for downsampling (i.e. reducing the spatial res- olution of the input layers) the data produced from convolution layers so that processing time can be reduced, and so that computational resources can handle the scale of the data. This is due to that fact that as a result of pooling, the number of learnable parameters is reduced in the subsequent layers of the network. Max pooling is a commonly used pooling technique that keeps the maximum value in a region (e.g. 2x2 non-overlapping regions of data) and discards the remaining values.
* Fully connected layers: This layer performs classification on the output generated from convolution layers and pooling layers. Every neuron in this layer is connected to every neuron present in the previous layer. This type of layer is typically followed by a Dropout layer that improves the generalization capability of the model by preventing over-fitting which is commonly occurring problem in deep learning domain.

For this project a CNN model consist of 10 layers is used for building the model.

 Figure 5.6: model overview

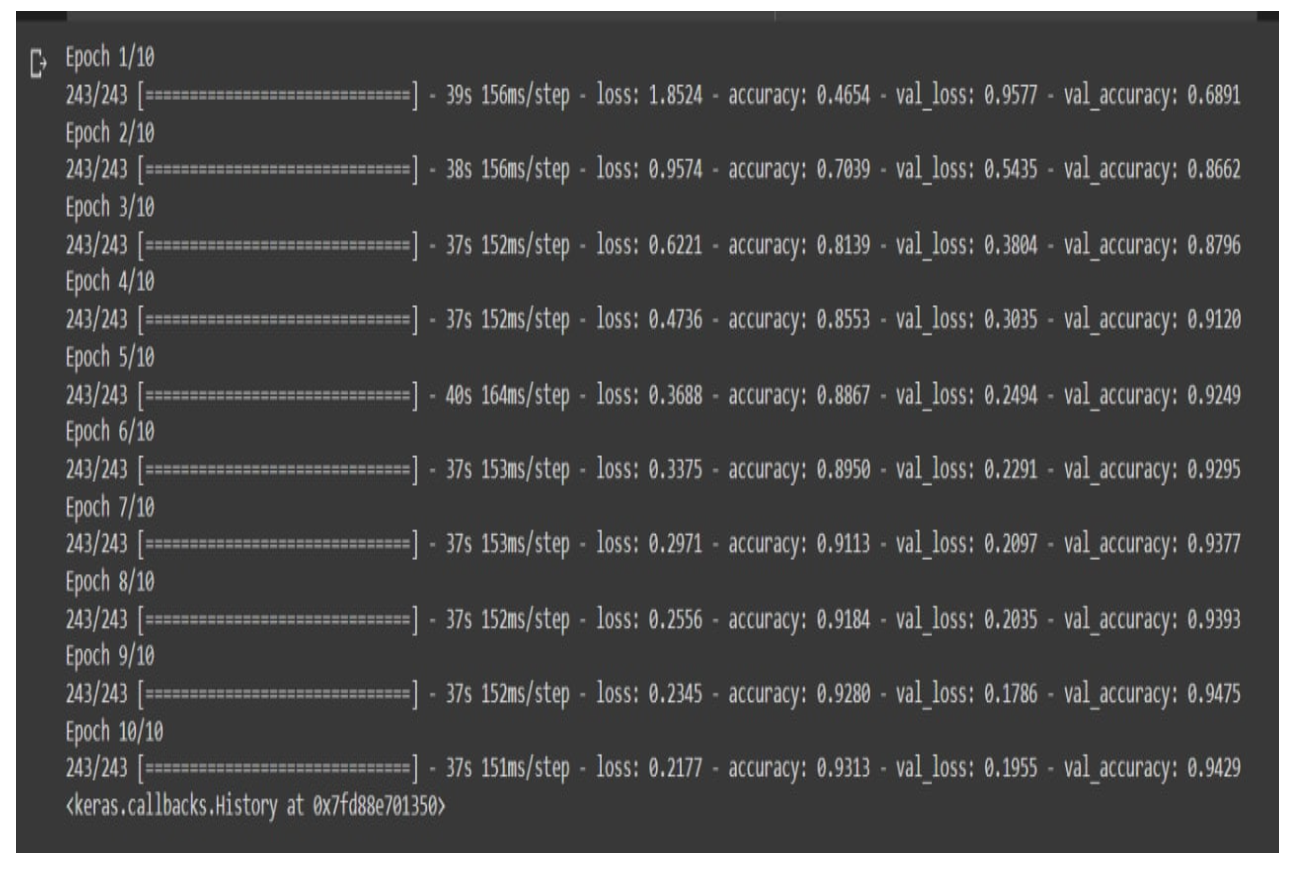
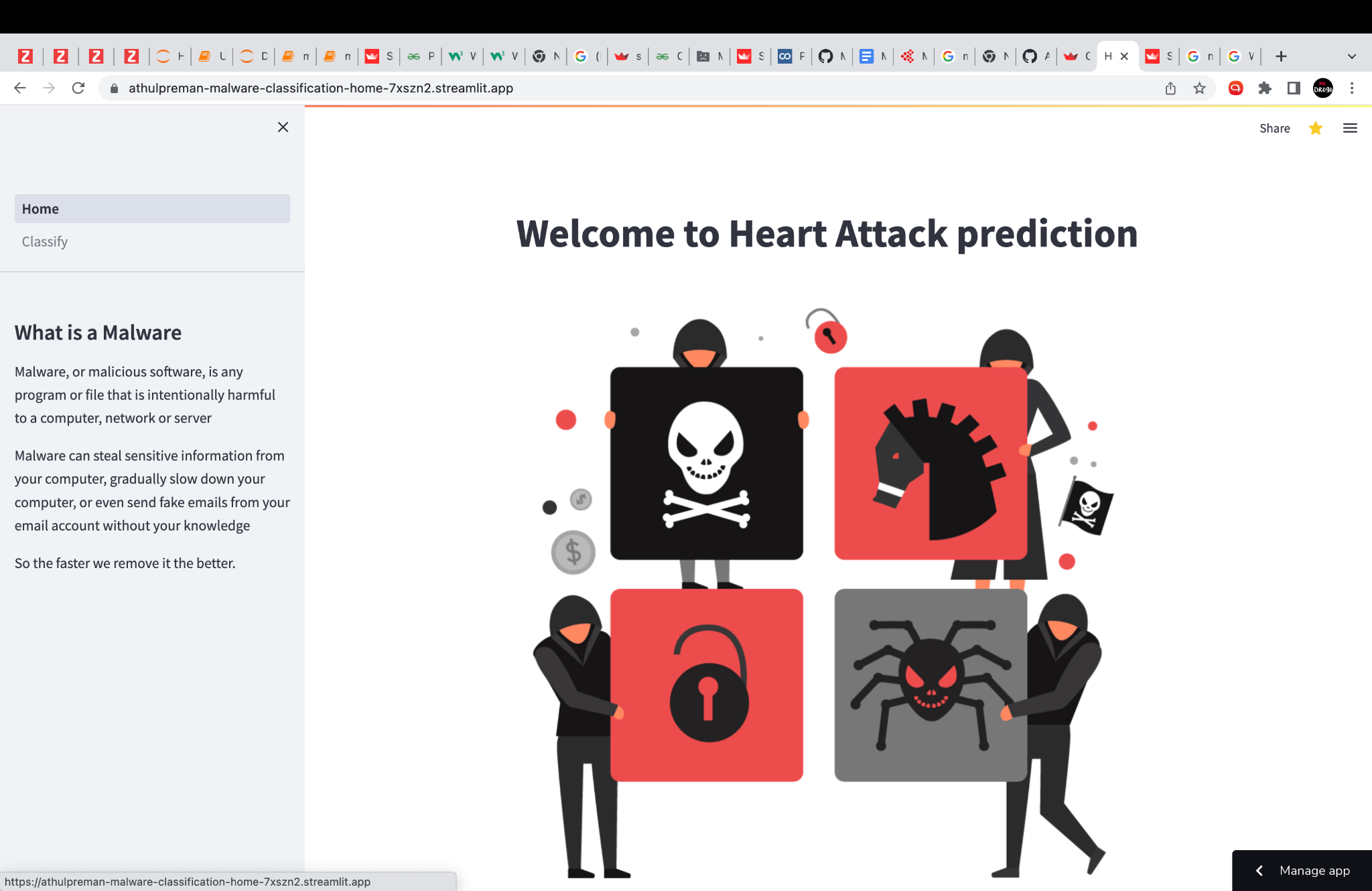


Figure 5.7: model accuracy

**6.3 Screenshots of UI**



5.3 Screenshots of Home page

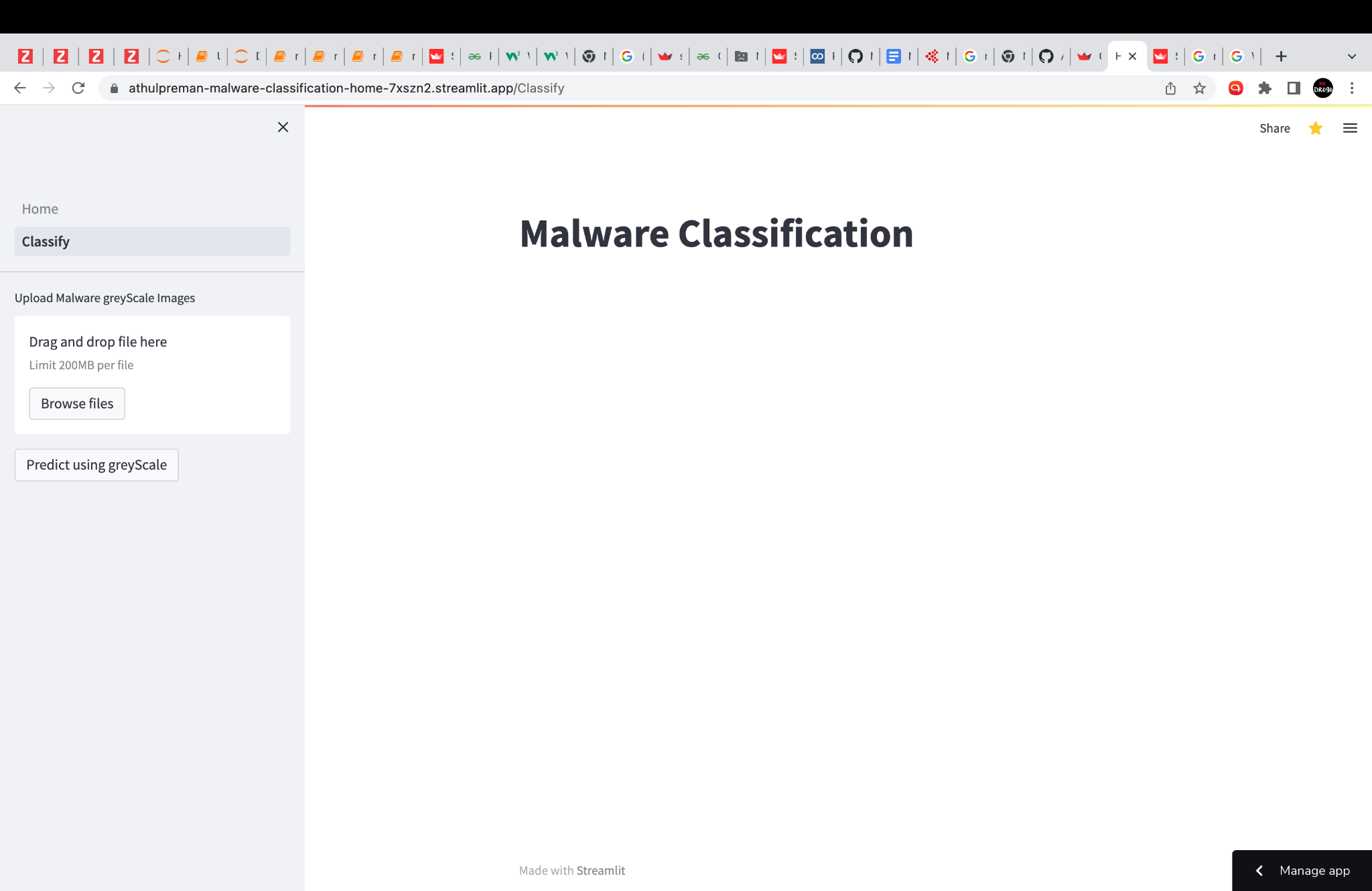


Figure 5.8: UI for prediction

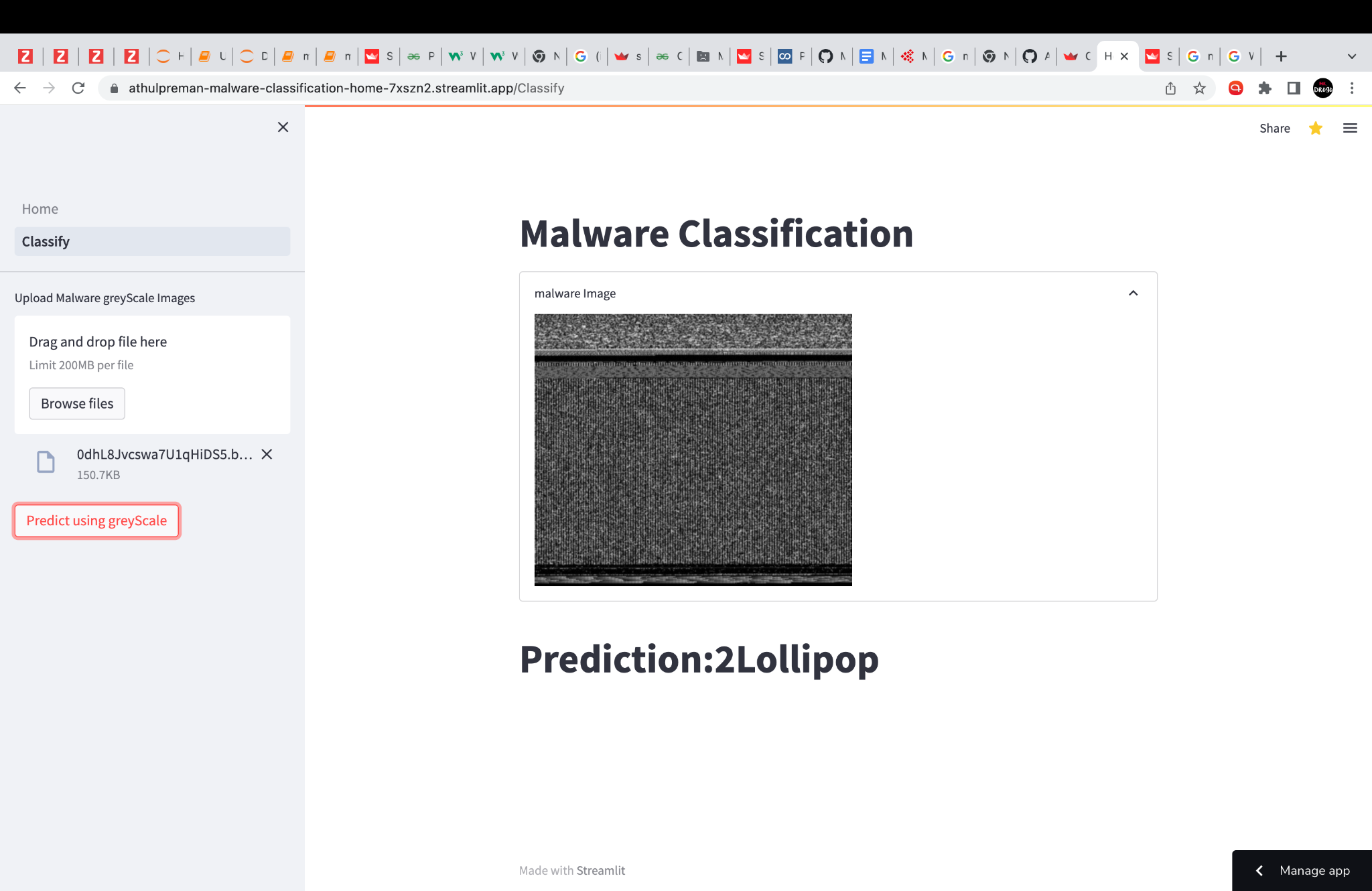


Figure 5.9: UI for result

**Chapter 7 : CODING**

**7.1 Algorithms**

**Algorithm 1 Algorithm for Creating the model:**

1: Start  
  
2: Divide the data set into two parts: training and testing. The majority of the dataset is utilised for training, while the remaining 20 % is used for testing to improve the results.  
   
3: The preprocessing stage uses the training dataset, and the preprocessed data is then utilised to extract the features. The input image’s characteristics are acquired.  
   
4: The retrieved attributes are then analysed to see which ones have the most impact on the outcomes. The model is then built using attributes that are highly dependent on the outcome.  
   
5: With the specified characteristics, the model is built using CNN with 10 layer architecture.

6: The generated model is fed the testing dataset, and the results are recorded.  
  
7: Toassessthemodel’sefficiency,theresultsofthetestingdatasetassessedusingthedeveloped model are compared to the actual values of the testing dataset. To evaluate the model, many statistical metrics like as accuracy, precision and recall score can be utilised.  
   
8: The developed model is fine-tuned in order to increase its efficiency.  
   
9: Stop

**Algorithm 2 Algorithm for web Application**

1: Start  
   
2: Through the user interface, read the user’s input gray scale image.  
   
3: The value from the web page is provided to the server application when the button is clicked.  
   
4: Access the input gray scale image from the server application and conduct the preprocessing  
   
 operations on it.

5: The model’s results are used to assess the outcome, which is then sent to the web page.  
   
6: The results are shown in the online application.  
   
7: Stop

**7.2 Sample code**

"""

@author: Athul P

"""

import streamlit as st

import tensorflow as tf

import cv2

from math import log

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

st.write("""# Malware Classification""" )

upload\_file = st.sidebar.file\_uploader("Upload Malware greyScale Images")

Generate\_pred=st.sidebar.button("Predict using greyScale")

#upload\_file2 = st.sidebar.file\_uploader("Upload bytecode Images")

#Generate\_pred2=st.sidebar.button("Predict using bytecode")

def convertAndSave(array,name):

print('Processing '+name)

if array.shape[1]!=16: #If not hexadecimal

assert(False)

b=int((array.shape[0]\*16)\*\*(0.5))

b=2\*\*(int(log(b)/log(2))+1)

a=int(array.shape[0]\*16/b)

array=array[:a\*b//16,:]

array=np.reshape(array,(a,b))

im = Image.fromarray(np.uint8(array))

im.save(name+'.png', "PNG")

return im

model=tf.keras.models.load\_model('mal.h5')

def import\_n\_pred(image\_data, model):

image\_data = np.asarray(image\_data)

#print(image\_data.shape)

img64=cv2.resize(image\_data,(64,64))

#print(img64.shape)

ar=np.asarray([img64])

#print(ar.shape)

pred=model.predict(ar)

size = (64,64)

#image = ImageOps.fit(image\_data, size)

#image = ImageOps.grayscale(image)

#img = np.asarray(image)

#reshape=img[np.newaxis,...]

#print(reshape)

#pred = model.predict(reshape)

return pred

if Generate\_pred:

image=Image.open(upload\_file).convert("RGB")

# print(image[...,:3].shape)

# image=cv2.imread(image,cv2.IMREAD\_COLOR)

with st.expander('malware Image', expanded = True):

st.image(image, width=350)#, use\_column\_width=True

pred=import\_n\_pred(image, model)

labels = ['1Ramnit','2Lollipop','3Kelihos\_ver3','8Obfuscator','9Gatak','Adialer.C','Agent.FYI','Allaple.A','Allaple.L','Alueron.gen!J','Autorun.K','C2LOP.P',

'C2LOP.gen!g','Dialplatform.B','Dontovo.A','Fakerean','Instantaccess','Lolyda.AA1','Lolyda.AA2','Lolyda.AA3','Lolyda.AT','Malex.gen!J','Obfuscator.AD',

'Rbot!gen','Skintrim.N','Swizzor.gen!E','Swizzor.gen!I','VB.AT','Wintrim.BX','Yuner.A']

st.title("Prediction:{0}".format(labels[np.argmax(pred)]))

**Chapter 8 : TESTING AND IMPLEMENTATION**

**8.1 Testing and various types of testing used.**

The next stage is to examine if the real results match the experimental findings when a piece of software has been written. This practise is known as testing. Its goal is to guarantee that the final product is flawless. The basic purpose of testing is to find errors and missing operations by executing the software. It also guarantees that the developer achieves all of the project’s goals. The purpose of testing is to uncover ways to enhance the efficiency, usability, and accuracy of the software that has been created, as well as to detect defects in it. Its objective is to evaluate the functionality, specifications, and performance of a software application.

Tests are conducted on the constructed software, and the results are compared to the expected documentation. When there are too many errors, debugging is used. After debugging, the software is retested to guarantee that it is error-free. The major testing methodologies utilised on this project are unit testing, integration testing, and system testing. Unit testing aims to test all of the software’s individual components. It guarantees that all software components are working properly. Integration testing is used to determine if the combined distinct elements functioned as planned. It helps us spot any problems that may arise when the units are combined.

During system testing, the entire software is checked to ensure that it fits all of the re- quirements. The tables below show the testing technique used during the development of the ”Malware Classification with deep learning” project. This section details the several measures made to guarantee that the project was error-free.

**8.1.1 Unit Testing**

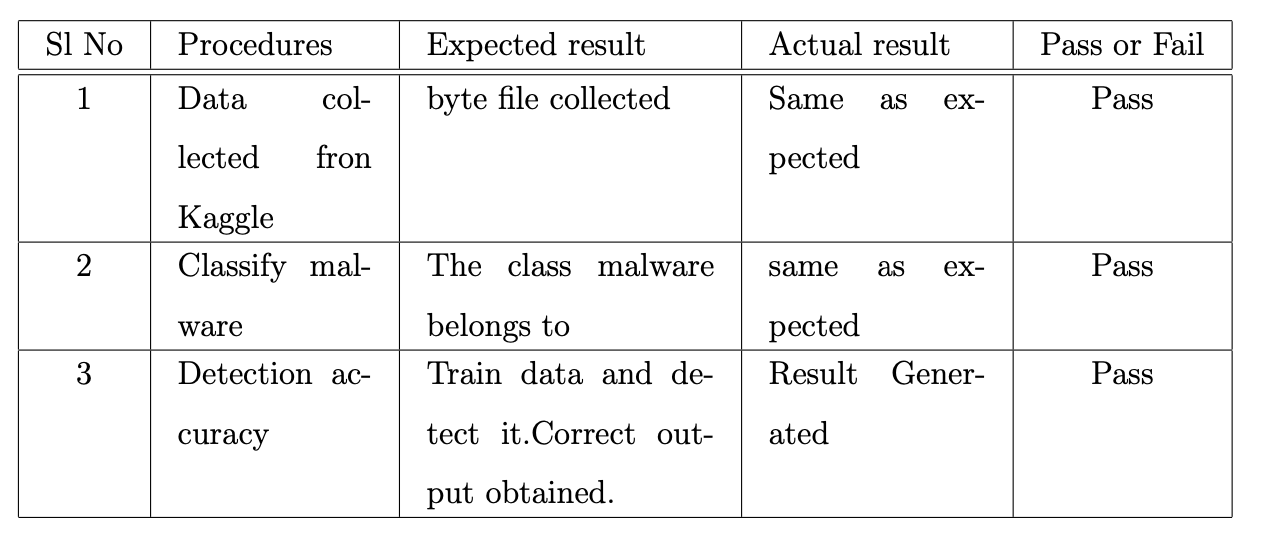
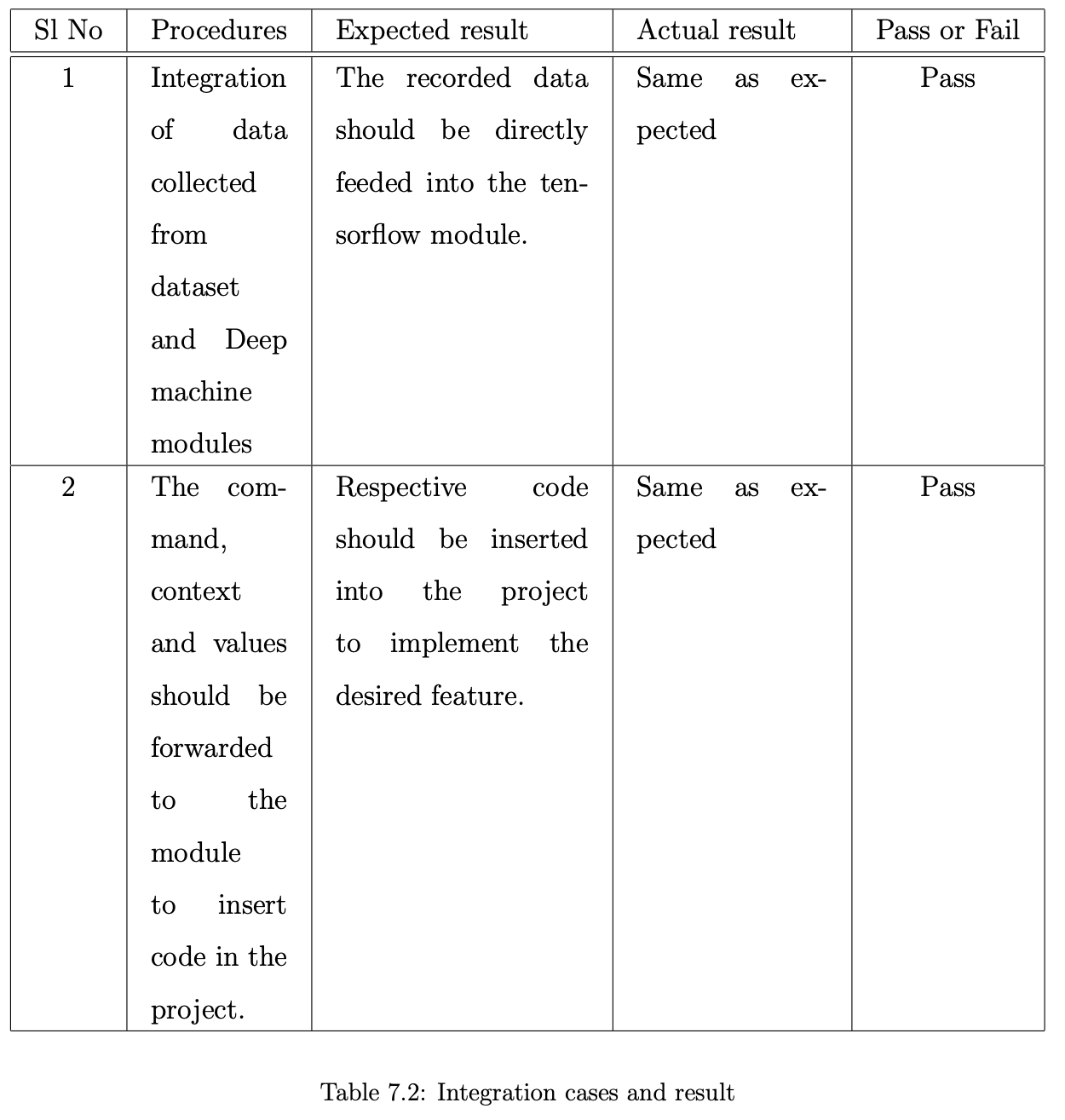
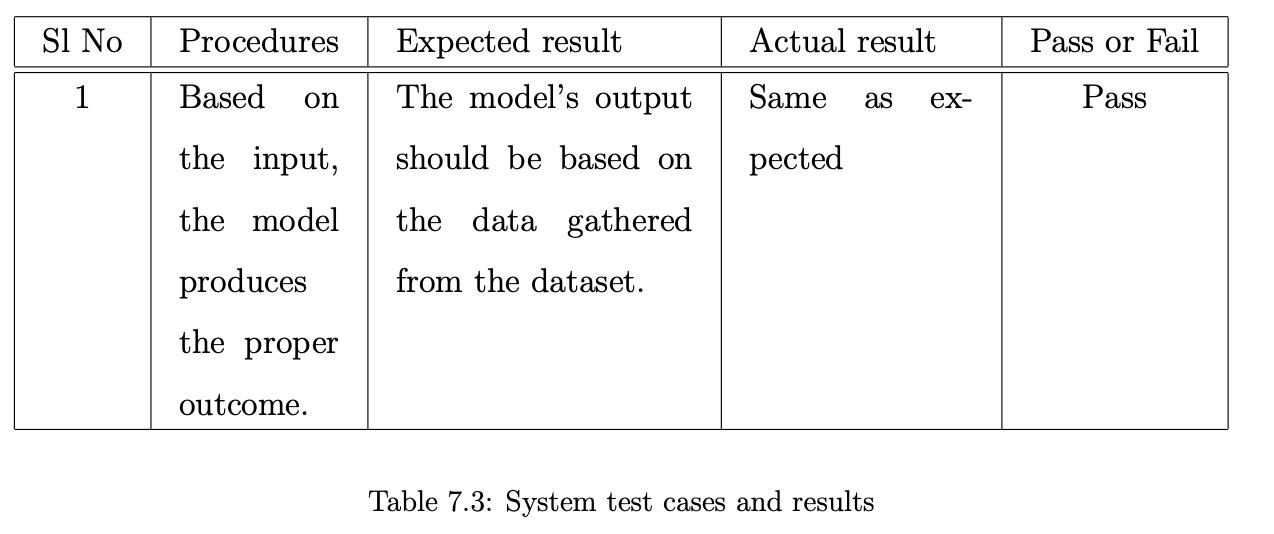


Table 8.1: Unit test cases and results





**Chapter 9 : DEPLOYMENT**

**9.1 Deployment in Streamlit Cloud**

Streamlit's cloud is an open and free platform for the community to deploy, discover, and share Streamlitapps and code with each other. The process of deploying applications in streamlits cloud is very easy. All we need to do is to upload the full project in the github and connect the github repository with the the streamlits cloud and deploy it.

We will get the link to access our project. We can paste the link in the browser to access the deployed project through the internet. We can also set the accessibility of the project as public or private according to our need.

**Chapter 10 : RESULTS AND DISCUSSION**

The major purpose of the study was to classify malware using a CNN model. Furthermore, the system appears to perform all of the functions as expected.

**10.1 Advantages and Limitations**

The proposed method classify the malware using deep learning method. The proposed method offers more advantages than the present system. The suggested method will help you save a significant amount of time. This strategy, like all others, has its own set of flaws. They are, however, insignificant in compared to the advantages, and they can be overcome in the future.

**10.1.1 Advantages**

• More features of malware can be studies because it uses a deep learning method. • It is easy to add new malware family to the model.

**10.1.2 Limitations**

* This project can only used for malware classification.
* Currently only 30 classes is classified . we can add more dangerous malware classes in the dataset.
* With powerful machine, model like vgg19 can be used to train the dataset
* We can also add feature like malware detection.

**Chapter 11 : CONCLUSION AND FUTURE SCOPE**

**11.1 Conclusion and Scope**

Malware is increasingly posing a serious security threat to computer systems. It is essential to analyze the behavior of malware and categorize samples so that robust programs to prevent malware attacks can be developed. Towards this endeavor, I proposed a deep convolutional neural network (CNN) architecture for malware classification. I first convert malware samples to grayscale images and then train a CNN for classification. Experimental results on malware classification datasets shows the effectiveness of my proposed method.

By using powerful deep learning architecture , classification accuracy can be increased. Wide range of powerful deep learning models are availabe now. With the help of powerful system having higher end graphics unit , these model can used .

**Chapter 12 : BIBLIOGRAPHY**

[1] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, ”Mal- ware Classification with Deep Convolutional Neural Networks,” 2018 9th IFIP Interna- tional Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328749.  
   
[2] O ̈. Aslan and A. A. Yilmaz, ”A New Malware Classification Framework Based on Deep Learning Algorithms,” in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/AC- CESS.2021.3089586.

* [3] Vasan, Danish, et al. ”IMCFN: Image-based malware classification using fine-tuned convo- lutional neural network architecture.” Computer Networks 171 (2020): 107138.