Text

Description automatically generated with medium confidence Logo, company name

Description automatically generated

**Informatics Institute of Technology**

in collaboration with

**University of Westminster, UK**

**GOPIT**

**5COSC009C: Software Development Group Project**

Module Leader-Mr. Banu Athuraliya

**BENG(Hons)Software Engineering/BSc (Hons)Computer Science**

Department of Computing

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Team Caffeine Brains**   |  |  |  | | --- | --- | --- | | **Name** | **IIT Number** | **UOW Number** | | H. B. D. S. Hewawasam | 2019944 | W1790957 | | I. C. Ilamperuma | 2019921 | W1790901 | | H. R. S. Hasaranga | 2019929 | W1790951 | | M. M. Mandara | 2019950 | W1790906 | | R. S. W. B. C. Ranathunga | 2019923 | W1790948 | | W. S. H. Miniruwan | 2019928 | W1790950 | |

Contents

[1.7 Resource Requirements 3](#_Toc69034797)

[1.7.1 Hardware Requirements 3](#_Toc69034798)

[1.7.2 Software Requirements 3](#_Toc69034799)

[1.7.3 Technology Stack 4](#_Toc69034800)

[Chapter 06 – Implementation 6](#_Toc69034801)

[6.1 Chapter Overview 6](#_Toc69034802)

[6.2 Overview of the prototype 6](#_Toc69034803)

[6.3 Technology Selections 6](#_Toc69034804)

[6.3.1 Language Selection 6](#_Toc69034805)

[6.3.2 Libraries / Frameworks selection 7](#_Toc69034806)

[6.3.3 Summary of Components in GOPIT 7](#_Toc69034807)

[6.4 Dataset 7](#_Toc69034808)

[6.5 Implementation of the features of the prototype in the backend 8](#_Toc69034809)

[6.5.4 Testing and Model using review. 8](#_Toc69034810)

[6.5.4 Categorization of issue 8](#_Toc69034811)

[6.6 Implementation of Graphical User Interface (GUI) 8](#_Toc69034812)

[6.6.1 Implementation of First Window (App Loading window) 8](#_Toc69034813)

[6.7 Combining of the backend and Frontend. 8](#_Toc69034814)

[6.7 Chapter Summary 8](#_Toc69034815)

[Reference 8](#_Toc69034816)

# 1.7 Resource Requirements

The following section details the resource requirements for GoPit application.

## 1.7.1 Hardware Requirements

|  |  |
| --- | --- |
| **Device** | **Specifications** |
| Huawei Y7p | Android 9.0 (Pie), EMUI 9, Kirin 710F (12 nm) chipset, Octa-core (4x2.2 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53), 4GB Ram, 64GB Internal Memory |
| Samsung Galaxy A51 | Android 11, One UI version 3.0, Exynox 9611 (10nm) chipset, Octa-core (4x2.3 GHz Cortex-A73 & 4x1.7 GHz-Cortex-A53), 6GB Ram, 128GB Internal Memory |
| Oppo F9 Model CPH1823 | Android 10, ColorOS 7.0, Mediatek MT6771 Helio P60 (12nm) chipset, Octa-core (4x2.0 GHz Cortex-A73 & 4x2.0 GHz Cortex-A53), 4GB Ram, 64 GB Internal Memory |
| Dialog 4G router/ Bell 4G router (Internet Connection) – Fast Internet connection | LTE model: tozed ZLP P28 |

## 1.7.2 Software Requirements

From the initial research, it was identified that the following languages, IDEs, Other software, APIs and Libraries will be needed for the successful completion of the project.

|  |  |
| --- | --- |
| **Languages** | |
| Python (V3.9) | For machine learning backend |
| Java (V11) | For main backend programming |
| C++ (C ++ 20 ) | For secondary backend programming |
| **IDEs and Other Software** | |
| Anaconda (V2020.07) | Python package management system |
| Jupyter Notebook (V6.1.6) | For programming backend while documenting |
| PyCharm (V2020.3.2) | For programming with Python with OOP |
| Teachable Machine | For create machine learning model |
| Android Studio (V4.1) | For programming with Java and develop the project |
| StarUML (V3.2.2) | For creating UML and Use case diagrams |
| Microsoft Office Word (V2020) | For making the reports and other documentations |
| https://www.gantt.com/ | For making the Gantt chart diagrams |
| Adobe Photoshop (CC 2017) | For editing and creating images and wireframes |
| Google Drive | For managing the documents and data in the cloud |
| Git (V2.30.0) | For version controlling |
| Jira(V8.16.8.0) | For project management system |
| Microsoft Teams | For corporate among the group |
| **APIs, Libraries and Frameworks** | |
| Pandas (V0.20.3) | For data analysis |
| Tensorflow(V2.4.1) | Machine Learning library with pre-built algorithms |
| Pickle (V3) | For serialization of data |

## 1.7.3 Technology Stack

The technology stack is a set of tools and frameworks that are used in the development of a software product. In software development, the technology stack consists of a very specific set of components that work together to create a functioning web application. (da-14.com, n.d.)

Technology stack in mobile application development can be divided into four areas. They are,

1. **Frontend Development**

This is the interface on which users interact with the mobile application.

1. **Backend Development**

Tools and software needed to create the underlying processing on the server.

1. **Development Platform**

A consolidated platform that provides the necessary libraries and interfaces to build the application.

1. **Additional (Supporting)**

Various tools and technologies that improve the security, flexibility, and performance of the application.

Technology stack for any mobile application depends on the platform that application is targeting. There are four types of technology stacks in mobile application development. They are **Android app technology stack, IOS app technology stack, Hybrid app technology stack** and **Cross-platform app technology stack.**

Among those technology stack types, **Android app technology stack** is chosen for the development of the GoPit application.

**Front-end, or Client-side**

The front-end components enable the user’s interaction with the application. GoTit application can be developed more user friendly with following components.

* **Fragments** are used to pass information between app screens. each screen of the app is associated with one or more fragments. Fragment represents a reusable portion of the app UI.
* **Layout Editor** is used to define and modify the layout either by coding XML or using the interactive visual editor.
* **XML drawables** are used to describe shapes, state, transitions of the application. it is used to decorate the application.

**Java** is the default language of the **Android Studio IDE**. Android OS is Linux kernel and it handles the memory management and the networking requirements for the device. Java is portable and it is capable of running on the ARM, X86 or MIPS processors. Java is an object-oriented language with much flexibility; it increases the code reusability and the security of the application. Most enterprise level mobile applications are developed using Java and there are a huge amount of supportive communities to get some help for development. In the opinion of many experts in mobile application development, Java is the most suitable language for android application developing.

**Back-end, or Server-side**

The back-end part of the development stack includes more components, and there are many more options to choose from here. Back-end works behind the scenes and is invisible to users, but it forms the engine that drives the application and implements its business logic. All back-end tech stacks include the following main parts:

* **Operating system** in which the development is done.
* Web server processing requests from the browser and returning the corresponding content.
* **Database** storing the app data.
* **Programming language** used to create the app code.

# Chapter 06 – Implementation

## 6.1 Chapter Overview

In this chapter overview before the implementation overview must mention that we changed our project scope. Because of the lack of proper data set and the use of data science does not happen much.

Before changed the scope we informed that issue to our module leader as instructed by the tutor.

In this implementation chapter will focus on the selected programming languages, libraries and frameworks and technologies that were used for implementing the GoPit app. And will discuss how one by one the features developed in the GoPit app with relevant code snipped and screenshots.

In addition, the challenges faced in implementing features and the solutions used are discussed in this chapter.

## 6.2 Overview of the prototype

* In the main backend, an image that is uploaded or captured at that moment is inserted into the system, analyzed according to the data set included in the system.
* After analyzing the data system will and divide them according to categories and display the result in the display result window.
* Mainly in the frontend has been developed by using android studio, in their it developed by using xml.

## 6.3 Technology Selections

After creating all the programming languages that can be used to create the GoPit app, java was selected as the main language. GoPit’s technology selection is divided in to two parts and discussed here.

Those are language selection and libraries and framework selection.

### 6.3.1 Language Selection

Java Language was selected for the following reasons,

* Java based on Object Oriented Programming language and It is easy to reuse the code.
* GoPit app is a mobile application. And mostly android development and java development is similar.
* It can edit easy.
* Configurations, Its high memory, high performance.
* Java is an excellent multifunctional IDE.
* It is easy to track the errors because it is statistically built language.
* It has capacity to perform multitask.
* If JVM is installed on the device, then the security is enhanced. (FITA,2018)
* Easy to find tutorials and good support for machine learning.
* Team members have good knowledge of java programming language and have a huge community to get the support.

Kotlin and Python languages are selected as secondary languages. Python language is most related to the machine learning implementations and It gives a good support to machine learning. Therefore, python language was selected for the machine learning part of GoPit project implementations.

Kotlin language was selected to create the real time camera in this GoPit app and Kotlin gives for good support for machine learning.

### 6.3.2 Libraries / Frameworks and Tools selections

#### 6.3.2.1 TensorFlow

TensorFlow is a Google’s open-source AI based framework used for training the machine learning model. To train the machine learning model on GoPit implementation selected TensorFlow. When use android studio IDE to develop GoPit application, can use TensorFlow as a plugin.

#### 6.3.2.2 TFLite Model Maker

#### 6.3.2.3 Junit

Junit is an open-source testing framework for java programmers. For Implementations of the GoPit app mainly will use java programming language. So, for the testing of the GoPit application it is easy to use Junit framework.

#### 6.3.2.4 Matplotlib

Matplotlib is a python library that is used for plotting graphs and visualize the result. In this GoPit project it is used for visualizing the backend result.

#### 6.3.2.5 android. Core

#### 6.3.2.6 android. Camarax

#### 6.3.2.7 NumPy

NumPy is a python library. That provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. (NumPy Documentation — NumPy v1.20 Manual, 2021)

NumPy is used for back-End implementation of GoPit project.

#### 6.3.2.8 Teachable Machine

### 6.3.3 Summary of Components in GoPit

### 

|  |  |  |
| --- | --- | --- |
| Requirement | Feature | Completion |
| 01 | Upload photo from gallery | Done |
| 02 | Gallery access | Done |
| 03 | Take a photo | Done |
| 04 | Access to Phone camera | Done |
| 05 | Get the photo to the system that took by camera | Done |
| 06 | Real time camara scan | Done |
| 07 | Show correct result | Done |

## 6.4 Implementation of the data science component

GoPiT application is using image classification data science component to achieve its main requirements. User must have the ability to classify the garbage of the restaurant by using GoPiT application.

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision.   
*(PDF) Image classification using Deep learning*. Available from: <https://www.researchgate.net/publication/325116934_Image_classification_using_Deep_learning> [accessed Apr 11, 2021].

### 6.4.1 Data Preparation and Pre-Processing

A garbage classification dataset is created by Mostafa Mohamed and released it under garbage on Kaggle was initially used for GoPit project. Data set includes 15,550 images from 12 different classes of Garbage.

GoPit app implements based on restaurant domain. So, for GoPit app do not need all the data include in the garbage classification dataset. So, had to clean the original data set as required by the GoPit app.

The data set created after cleaning the original data set contains only 8384 images and six garbage classes. Those are Glass, Hazardous, Metal, Plastic, Organic and Recyclable. The original images were size 255 x 255 pixels, and those images were used as it is because the size of those images are much suitable for processing.

While these images are converted into features, the brighter pixels will have more weightage than the normal pixels and number of estimators should always be greater to get a better picture. Pujari (2015)]

Following example are belongs to each class in the GoPit application.

A picture containing text

Description automatically generatedA picture containing text

Description automatically generated

Glass Hazardous Metal

A picture containing calendar

Description automatically generatedA picture containing drinking water, bottle

Description automatically generatedA picture containing snack food, potato

Description automatically generated

Organic Plastic Recyclable

#### Challenges Encountered and Solutions

* Having lot of irrelevant images in dataset for GoPiT application – Clean the dataset and remove unnecessary images.
* Hard to clean image dataset – There is no such a method to clean image dataset, it had to clean manually.

### 6.4.2 Modelling

After Data preparation and pre-processing is done, the next step is to train the data science component of GoPit application. TensorFlow is the framework which is use in the model of GoPit application. Google Teachable machine is used to implement the data science component (image classification model) of the project.

#### 6.4.2.1 Teachable Machine image classification model

Teachable machine is a web-based GUI tool for creating custom machine learning classification models. It introduced in 2017 by Google.

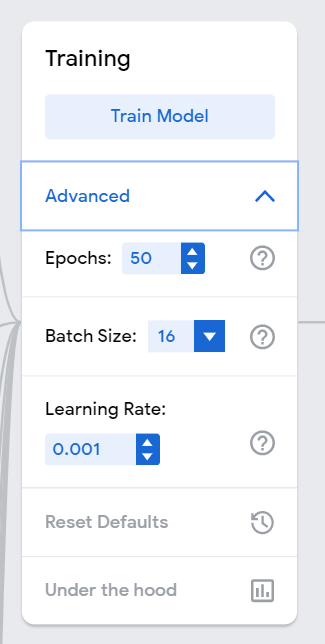
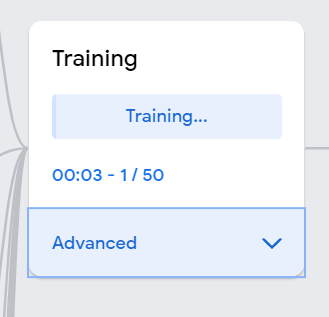
There are 3 types of models can be created with latest version of teachable machine, those are image classification model, audio recognition model and pose recognition model. Among those models, image classification model is used to train the machine learning model in GoPiT application.

Teachable machine is very user-friendly tool. Anyone can build a machine learning model without any coding. It created for help students, teachers, designers, and others learn about ML by creating and using their own classification models (Adedeji and Wang, 2019)

There are few steps to create an image classification model with Teachable machine. First step is to create the classes of the model which belongs to the image classification model and upload the data into those classes. There are six classes in GoPiT application such as Glass, Hazardous, Metal, Organic, Plastic and Recyclable. Those classes were created in Teachable machine and uploaded relevant images in the dataset to each class as follow **Figure #** and **Figure #.**

### 

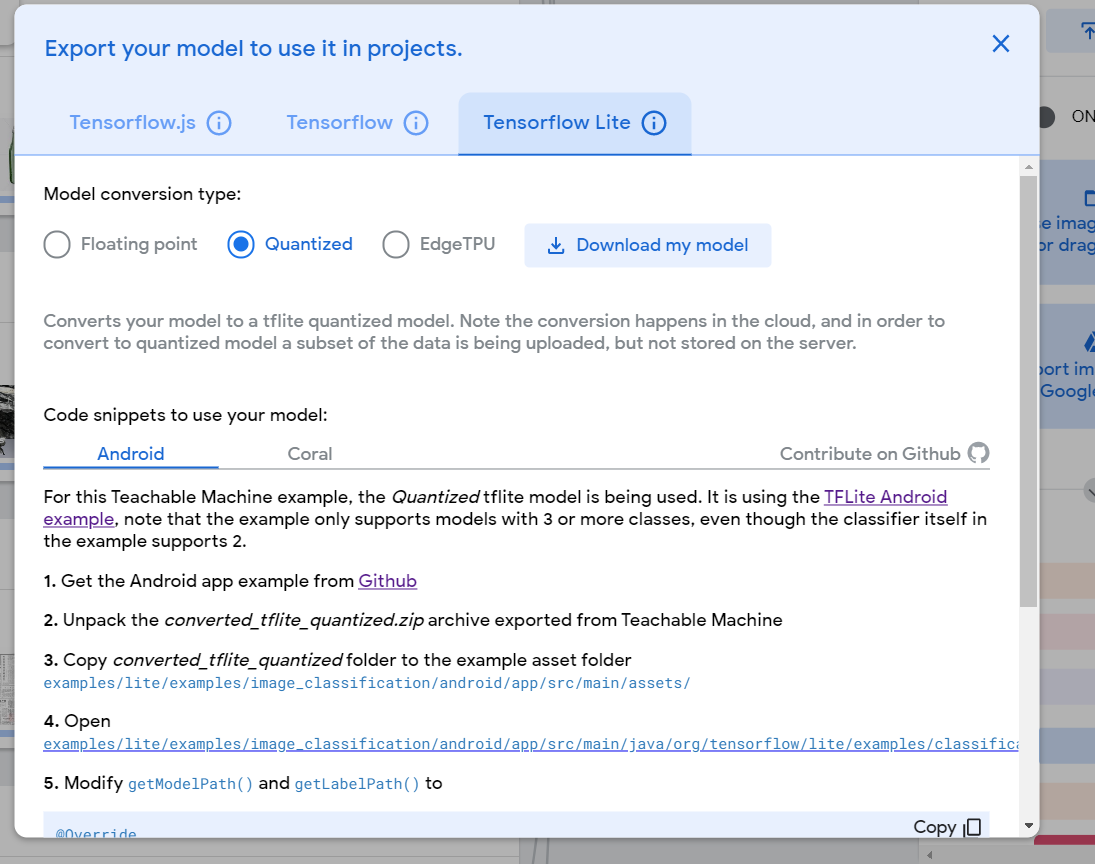
The next step is to train the image classification model. There are some options such as Batch Size, Epochs and Learning Rate that can be changed before the training began. Batch Size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters and the number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. (Brownlee, 2021) The learning rate is the amount that the weights are updated during training, it is configurable hyperparameter used in the training of neural network that has small positive value. (Brownlee, 2021) Image classification model of GoPiT application is trained with 50 epochs, Batch Size of 16 and 0.001 Learning rate. Following **figure #** and **figure #** represents the training period of the image classification model.

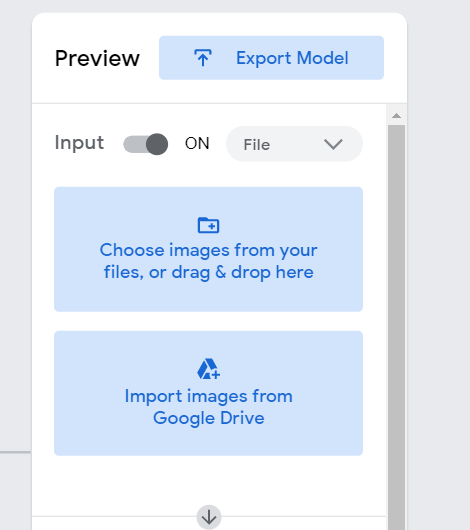


After training the model next step is to export the model. There are 3 different ways to export the model from Teachable machine, they are Tensorflow.js, Tensorflow and Tensorflow Lite. Tensorflow.js type is compatible with web applications that means the models exported from that can be used within web application. Tensorflow models are compatible with python. Then Tensorflow Lite models are compatible with mobile applications. GoPiT application is also mobile application then the image classification model of the application is also exported from Tensorflow Lite form.

There are 3 different versions in exporting the Tensorflow Lite model they are Floating point, Quantized and EdgeTPU. Floating point is the model exported with float16 or float32. Quantized models execute some or all of the operations on tensors with integers rather than floating point values. This allows for more compact model representation and the use of high performance vectorized operations on many hardware platforms. EdgeTPU is used in coral platform. Image classification model of GoPiT application is exported from quantized version of Tensorflow Lite. There are few reasons to use quantized model in GoPiT application.

* Accuracy is higher than floating point model.
* Compatible with many devices.
* High performance in any device.

**Figure #** and **Figure #** are displays the exporting period of image classification model with teachable machine.



It downloads 2 files after exporting the model from Teachable machine. They are Label.txt file and Model.tflite file. Label text file includes the labels of classes in the image classification model and tflite file is the model file. This model gets the inputs as a TensorBuffers and it includes process(TensorBuffer) method to get the result. Following is the description of this model.

NewModel model = NewModel.newInstance(context);

// Creates inputs for reference.

TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3}, DataType.UINT8);

inputFeature0.loadBuffer(byteBuffer);

// Runs model inference and gets result.

NewModel.Outputs outputs = model.process(inputFeature0);

TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

// Releases model resources if no longer used.

model.close();

Real time camera scan is functional requirement of GoPiT application. To achieve this requirement, the data science component should be able to take a inputs as TensorImages because real time camera inputs cannot be converted in to TensorBuffers easily. Behalf of that after using this teachable machine image classification model in GoPiT application, realized the accuracy of the model is little bit low. Because of those reasons there was need to empower the project with another image classification model as a secondary data science component of GoPiT application. Google collab and TFLite-model-maker are used to create this secondary image classification model of GoPiT application.

#### 6.4.2.2 Image Classification Model created with TFLite-Model-Maker

The TFLite Model Maker library simplifies the process of adapting and converting a TensorFlow neural-network model to particular input data when deploying this model for on-device ML applications. (tflite-model-maker, 2021)

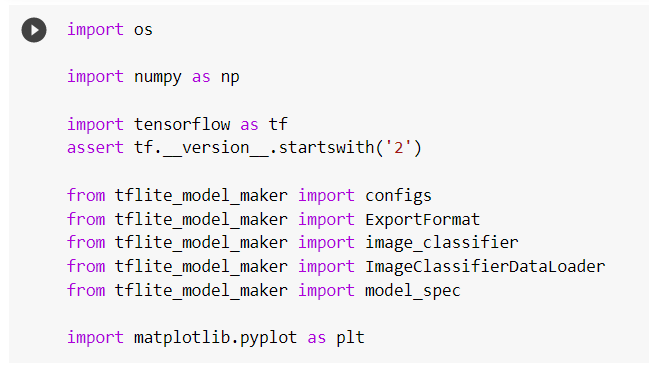
TFLite -Model-Maker two use cases, those are image classification and text classification. GoPiT application needs the image classification model because of that TFLite Model Maker used to create a image classification model by following steps.

For work with TFLite model maker, requires a Python development environment such as Google colab, Jupytor Notebook. Google colab is used for implementing the image classification model of GoPiT application.

First step is to install the TFLite-Model-Maker in the virtual memory of Google colab. **Figure #** below represents the installing of TFLite-Model-Maker.



After the installation, next step is to import the python files included in tflite-model-maker and some libraries such as Numpy, Matplotlib, Tensorflow which is required in training and evaluating criteria of image classification model.



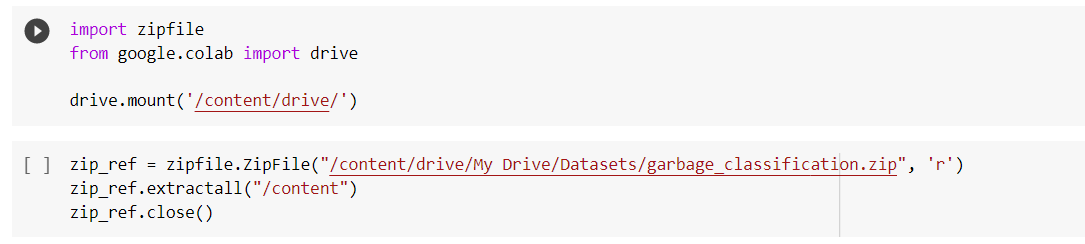
Import numpy for store data.

Import Tesorflow for create model.

Import python files in tflite-model maker.

Import matplotlib for data visualization.

Next step is to download the dataset into virtual storage in google calab. Before this step, the zip version of garbage dataset is uploaded into drive folder.

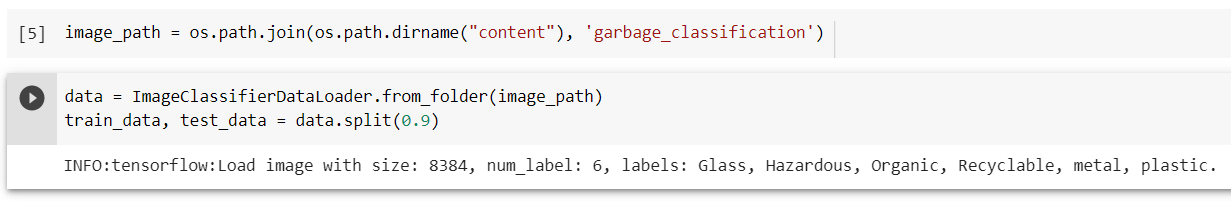


Mount the google drive into virtual storage.

Extracting all files in garbagr\_classification.zip into content folder in virtual storage.

Now the images in dataset split into two parts. One part is for train the model and other one is to test the model in evaluating period. 90% from data took as a train dataset and remaining 10% is too as test dataset.

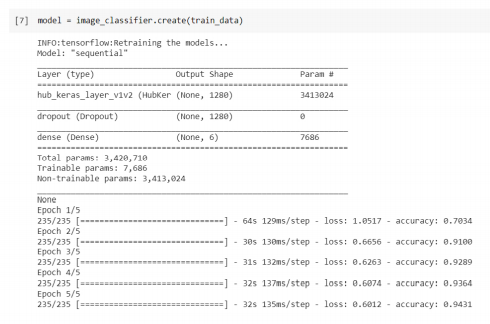
Get the path of dataset into image\_path variable.



Get all data of dataset into data variable.

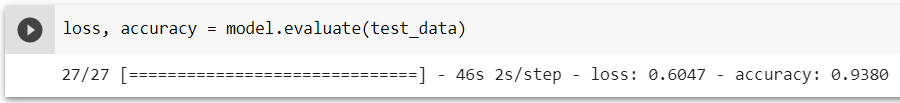
Split data into train\_data and test\_data, 90% for train\_data and 10% for test\_data

After partitioning the data, model is trained from train\_data dataset. This training is done with create method which is inbuild method of image\_classiffier python class in tflite-model-maker. This training is done with 5 epochs and after the training is finished the accuracy is shows as 94.31%.



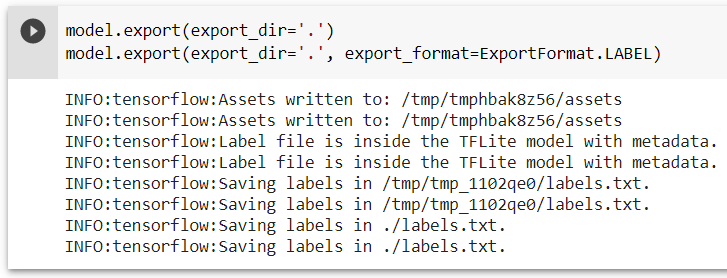
Accuracy of the model at 1st epoch and 5th (final) epoch.

Now the image classification model is trained successfully. Next step is to evaluate the model with test dataset which is split previously. There is method call evaluate in model class of tflite-model-maker, it is used to evaluate the model after training.



Accuracy of the model

Final step is exporting the model. After the export it download both Labels text file and model tflite file. Following is the exporting method of this.



The Labels file includes the labels of classes in this model and the model.tflite file is includes the image classification model. This model gets inputs as TensorImages and it includes the process(TensorImage) method for get the result. Following is the java cade of those methods and constructor of this model.

GCModel model = GCModel.newInstance(context);

// Creates inputs for reference.

TensorImage image = TensorImage.fromBitmap(bitmap);

// Runs model inference and gets result.

GCModel.Outputs outputs = model.process(image);

List<Category> probability = outputs.getProbabilityAsCategoryList();

// Releases model resources if no longer used.

model.close();

Because of this model gets the inputs as TensorImages, it can be used in Real time camera feature in GoPiT application. In addition, accuracy of this model is 93.8% and it is very positive percentage for the image classifications. Then Both of the models, which is created of Teachable machine and which is created using tflite-model-maker are used in the GoPiT application to achieve the functional requirements and non-functional requirements as well.

### 6.5.4 Testing and Model using review.

#### Challenges Encountered and Solutions

### 6.5.4 Categorization of issue

#### Challenges Encountered and Solutions

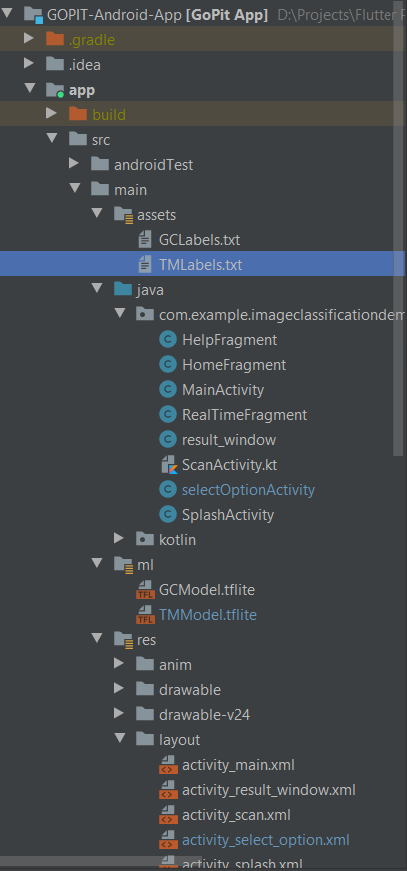
## 6.6 Implementation of Graphical User Interface (GUI)

#### Challenges Encountered and Solutions

### 6.6.1 Implementation of First Window (App Loading window)

## 6.7 Combining of the backend and Frontend.

As discussed in chapter 6.5, two TensorFlow Lite models were used for this project. There are two tflite model files and two text files includes the labels of the classes as a backend component. Finally, the backend and frontend components of the GoPiT project were combined by adding those text files to assets directory in the Android studio project and adding those tflite model files to ml directory in the android studio project. Figure # and figure # is showing the final file structure of the GoPiT project.

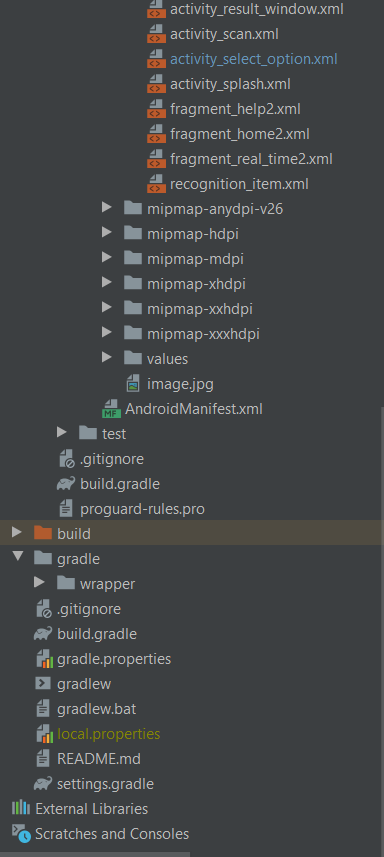


Controllers (Java & Kotlin files to control XML files)

Exported TFLite Models

Views (XML Files)

Exported Label files of TFLite files.



Views (XML Files)

Unit Tests

Build.gradle file

## 

Following code display the way that model which exported from Google Teachable machine, connecting to the GoPiT project.

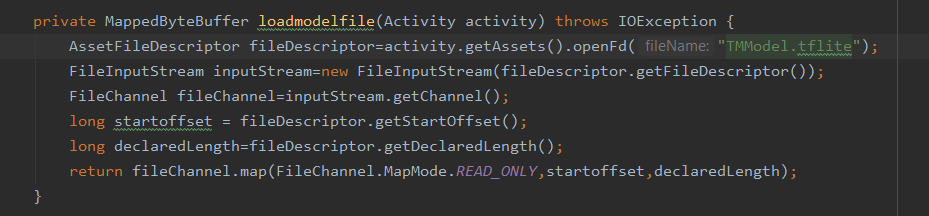
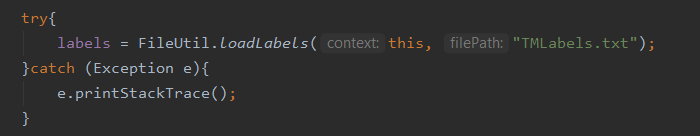
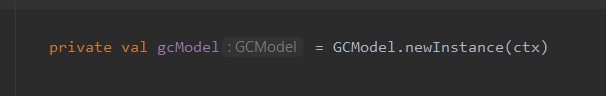


Figure # displays the way that how to connect to the Label text file (which exported from the Google Teachable machine) to the project.



By creating the new instance of the model which trained by using Google Collab, after that it connected to the GoPiT project. Following code displays that incident.



To the implementation of the GoPiT project, as a version control system, group members used the GitHub. created repository is free public repository. And the screenshots of the commits displayed in the Appendix #.

## 6.7 Chapter Summary

**Functional Requirements**

In the below two charts depicts the functional and Non-functional requirements of the system along with their priority levels.

* Critical - The requirements that are critically needed for successful completion of the project
* Medium - The requirements that are needed for a value-added completion of the project
* Low - The requirements that are needed for extra value for the project. Not mandatory

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Requirement List** | **Priority Level** | **Description** |
| FR1 | Upload images which are stored in device storage. | Critical | The application should be able to upload garbage images which are already stored in mobile device storage |
| FR2 | Real-Time camera feature | Critical | The system should be able to turn on mobile camera and analyze real time images by showing their garbage classes with percentages. |
| FR3 | Display selected image from device storage in Image view Window | Critical | After selecting image from device, it should be automatically display in image view on application. |
| FR4 | Analyze given image and show relevant garbage classification class | Critical | Should be able to display classification class name of garbage as a text after analyzing. |
| FR5 | Show relevant garbage bin | Medium | The application automatically shown up relevant garbage bin according to the given image. |
| FR6 | Show help window | Low | Should be able to show initial steps which are wanting to be followed. |

**Non – Functional Requirements**

The following table explain the Non – Functional requirements of the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Requirement List** | **Priority Level** | **Description** |
| NF1 | Accuracy | Critical | The system should have a high level of accuracy when identifying garbage classification classes regarding to images. |
| NF2 | Reliability | Critical | The system should be reliable for hotels and the other users to classify garbages. |
| NF3 | Performance | Critical | The application should be well-performed without any considerable lagging issue and slowness |
| NF4 | Usability | Medium | The system should have mobile friendly ,proper navigation and user-friendly interfaces. |
| NF5 | Extensibility | Critical | The system should be able to analyze the image and identify the correct category of garbage |

**Testing Functional Requirements**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Feature tested** | **Test case description** | **Test case condition** | **Expected result** | **Actual result** | **Status** |
| 1.0 | FR1 | Upload images which are stored in device storage. | Uploading at 60 jpg,jpeg,png,gif file formats with | Upload at least 50 images | Jpg,jpeg,png are successfully uploaded.  Gif file format successfully uploaded but it is shown in freeze format. | Pass  (91.66%) |
| 2.0 | FR2 | Real-Time camera feature | Pointed real time camera into 60 different objects which belongs to the classes in the GoPit App (Glass, Hazardous, Metal, Organic, Plastic, Recyclable) | Identify the objects with above 90% of accuracy. | 40 objects were identified with 70 - 80 percentage of accuracy and 15 objects were identified with above 90 percentage. Remaining 5 objects were identified with below 60 percent. | Pass  (91.66%) |
| 3.0 | FR3 | Capture image from device camera | Capture 20 images from device camera | Capture at least 15 images | Captured 20 images | Pass  (100%) |
| 4.0 | FR4 | Analyze given image and show relevant garbage classification class | Analyzed 60 images (10 images from each class) | Identify and display relevant garbage class at least 50 images. | Successfully analyzed and displayed relevant garbage class  for 52 images.  7 images from metal class and 9 images from each and every other class. | Pass  (86.67%) |
| 5.0 | FR5 | Show relevant garbage bin | Input 60 images (10 images from each class) | Display relevant garbage bin for at least 50 images. | Successfully displayed relevant garbage bin for all 60 images. | Pass  (100%) |
| 6.0 | FR6 | Show help window |  |  |  | Pass |

## Reference

Kaggle.com. 2021. *Kaggle: Your Machine Learning and Data Science Community*. [online] Available at: <https://www.kaggle.com/> [Accessed 1 April 2021].

FITA. 2021. *Why is Java highly suitable for mobile application?* [online] Available at: <https://www.fita.in/java-highly-suitable-mobile-application/> [Accessed 8 March 2021].

TensorFlow. 2021. *TensorFlow*. [online] Available at: <https://www.tensorflow.org/> [Accessed 14 March 2021].

Numpy.org. 2021. *NumPy Documentation — NumPy v1.20 Manual*. [online] Available at: <https://numpy.org/doc/stable/contents.html#> [Accessed 1 April 2021].

krishna, M. and Gopala Rao, M., 2018. *Image classification using Deep learning*. [online] Available at: <https://www.researchgate.net/publication/325116934\_Image\_classification\_using\_Deep\_learning> [Accessed 11 April 2021].

*(PDF) Image classification using Deep learning*. Available from: <https://www.researchgate.net/publication/325116934_Image_classification_using_Deep_learning> [accessed Apr 11, 2021].

Adedeji, O. and Wang, Z., 2019. *Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network*. [online] ScienceDirect. Available at: <http://www.sciencedirect.com> [Accessed 3 April 2021].

Pujari, A. (2015). Retail product image classication, p. 14

Brownlee, J., 2021. *Difference Between a Batch and an Epoch in a Neural Network*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20batch%20size%20is%20a%20hyperparameter%20that%20defines%20the%20number,updating%20the%20internal%20model%20parameters.&text=When%20the%20batch%20size%20is,called%20mini%2Dbatch%20gradient%20descent.> [Accessed 8 April 2021].

Brownlee, J., 2021. *Understand the Impact of Learning Rate on Neural Network Performance*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:text=Specifically%2C%20the%20learning%20rate%20is,is%20adapted%20to%20the%20problem.&text=It%20may%20be%20the%20most%20important%20hyperparameter%20for%20the%20model.> [Accessed 8 April 2021].

PyPI. 2021. *tflite-model-maker*. [online] Available at: <https://pypi.org/project/tflite-model-maker/> [Accessed 12 April 2021].