REAL TIME IMAGE CLASSIFIER APP

A MINI PROJECT REPORT

Submitted by

JONATHAN THANGADURAI
311116205024
DILKHUSH MIHIRSEN D
311116205013

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

LOYOLA-ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

Mar 2019

BONAFIDE CERTIFICATE

Certified that the Mini project report on **"REAL TIME IMAGE CLASSIFIER APP"** is the bonafide work of "***JONATHAN THANGADURAI - 311116205024***" and "***DILKHUSH MIHIRSEN D- 311116205013***" who carried out the project work under my supervision.

SIGNATURE
Dr. R Juliana M.E., PhD.,
HOD
Department of Information Technology
LICET
Chennai-34

SIGNATURE
Ms. G Shobana M.Tech.,
PROJECT GUIDE
Assistant Professor/IT
LICET
Chennai-34

ACKNOWLEDGEMENT

First of all we thank The Lord Almighty for showering His blessings on us, which enabled us to complete this project successfully.

We express our sincere gratitude to our beloved and respected **Director, Rev**. **Dr. Alphonse Manickam SJ.** We wish to acknowledge with thanks to our college, **Dean of Studies, Rev Dr Justine SJ** for his support and encouragement.

We extend our sincere gratitude to **Dr L Antony Michael Raj**, **Principal**, for his motivation and support towards this project.

We express our sincere gratitude to the **Head of the Department** of Information Technology, **Dr R Juliana** for her constant support, motivation, encouragement and being a source of inspiration throughout this project and during the course of this semester.

Our sincere thanks to our **Project Guide Ms. G Shobana** for their valuable guidance in shaping this project.

We also take this opportunity to thank all the **Faculty and Non-teaching staff members** of Department of Information Technology for their constant support. Finally we thank each and every one who helped us to complete this project.

JONATHAN THANGADURAI

DILKHUSH MIHIRSEN D

TABLE OF CONTENTS

ABSTRACT

Recent advances in deep learning made tasks such as Image and speech recognition possible. Deep Learning is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. Deep learning excels in recognizing objects in images as it's implemented using 3 or more layers of artificial neural networks where each layer is responsible for extracting one or more feature of the image.A Neural Network is a computational model that works in a similar way to the neurons in the human brain. Each neuron takes an input, performs some operations then passes the output to the following neuron. Computers are able to perform computations on numbers and is unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand.

There are two common ways to do this in Image Processing:

1. Using Greyscale:

The image will be converted to greyscale (range of gray shades from white to black) the computer will assign each pixel a value based on how dark it is. All the numbers are put into an array and the computer does computations on that array.

2. Using RGB Values:

Colors could be represented as RGB values (a combination of red, green and blue ranging from 0 to 255). Computers could then extract the RGB value of each pixel and put the result in an array for interpretation.When the computer interprets a new image, it will convert the image to an array by using the same technique, which then compares the patterns of numbers against the already-known objects. The computer then allots confidence scores for each class. The class with the highest confidence score is usually the predicted one.

One of the most popular techniques used in improving the accuracy of image classification is Convolutional Neural Networks. Convolutional Neural Network is a special type Neural Networks that works in the same way of a regular neural network except that it has a convolution layer at the beginning

Instead of feeding the entire image as an array of numbers, the image is broken up into a number of tiles, the machine then tries to predict what each tile is. Finally, the computer tries to predict what's in the picture based on the prediction of all the tiles. This allows the computer to parallelize the operations and detect the object regardless of where it is located in the image.

1.INTRODUCTION:

1.1 Problem Statement:

Computers are able to perform computations on numbers and is unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand. Deep Learning is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. Deep learning excels in recognizing objects in images as it's implemented using 3 or more layers of artificial neural networks where each layer is responsible for extracting one or more feature of the image.A Neural Network is a computational model that works in a similar way to the neurons in the human brain. Each neuron takes an input, performs some operations then passes the output to the following neuron.

Our aim is to develop a machine learning model for recognizing images and to train it efficiently by feeding it datasets taken from various sources which offer labelled data. This model will then be fused into an android application. Ultimately, we obtain an application running on Android which will obtain a live feed from the camera and calculate the probability of what the image is when the camera is pointed at an object.

2. REQUIREMENTS

2.1 Functional Requirements

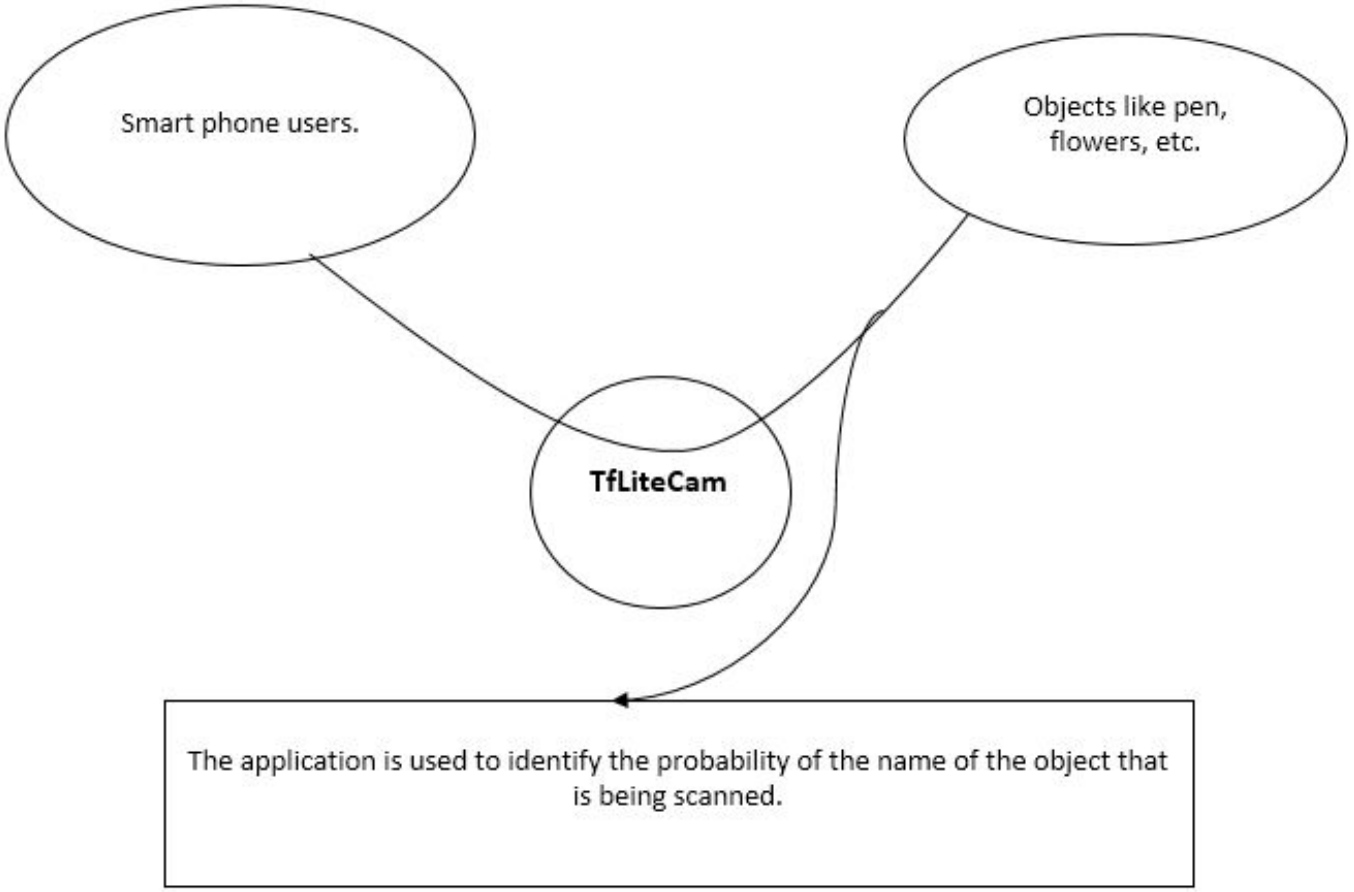
- Smartphone with camera
- Stand-alone computer
- Android Studio
- Tensorflow classification module
- Training dataset
- Training module
- Classified data
- Image probability module
- Wifi router
- Imagenet database

2.2 Non- Functional Requirements

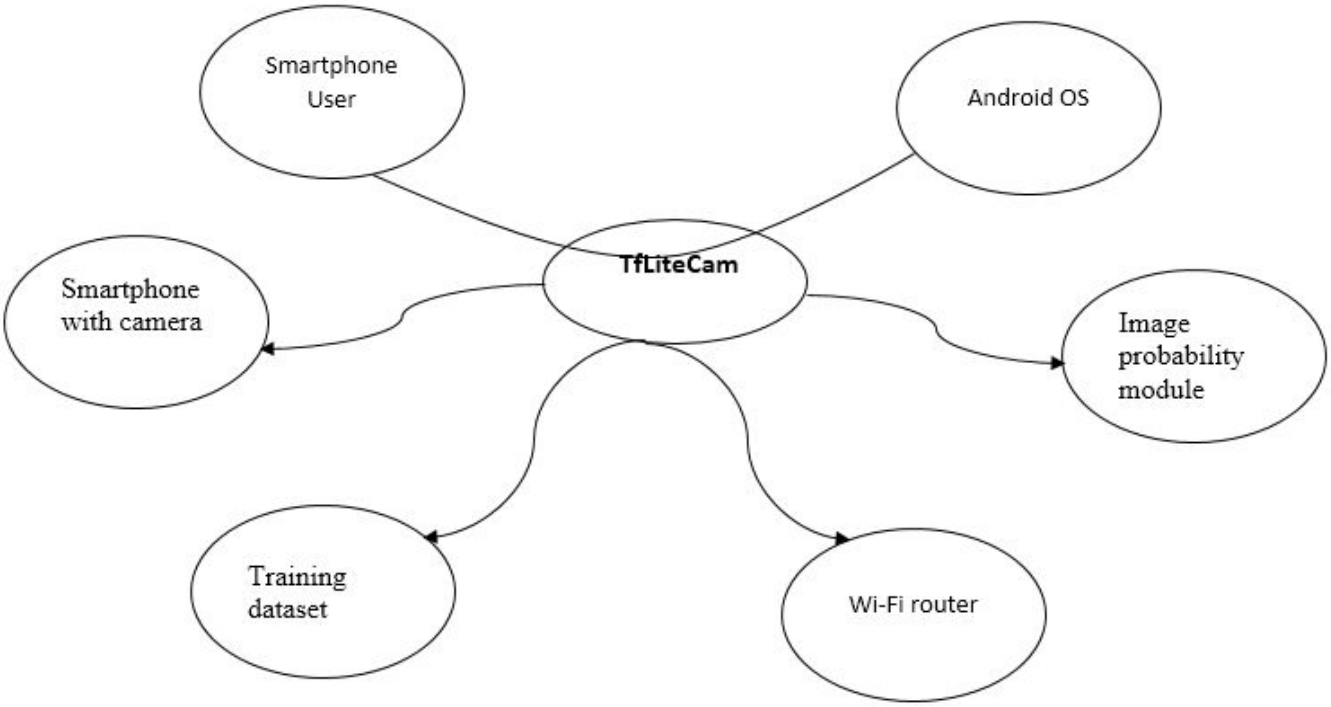
- Accessibility
- Consistency
- Reliability
- Mobility
- Modifiability
- Efficiency
- Accuracy

3. ANALYSIS

3.1 Bull diagram

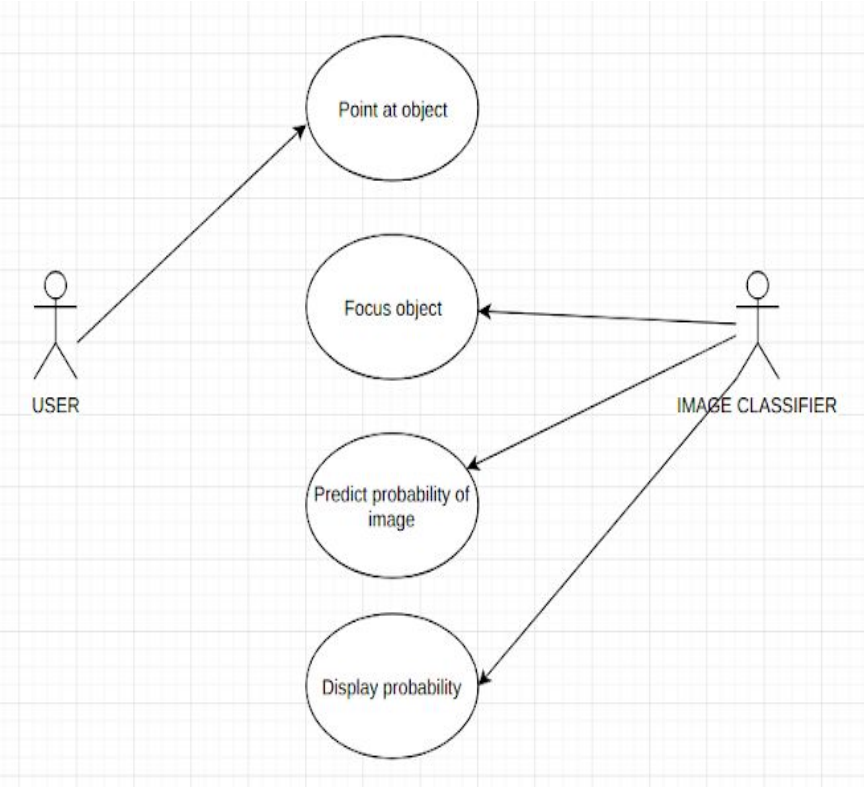


3.2 Octopus diagram

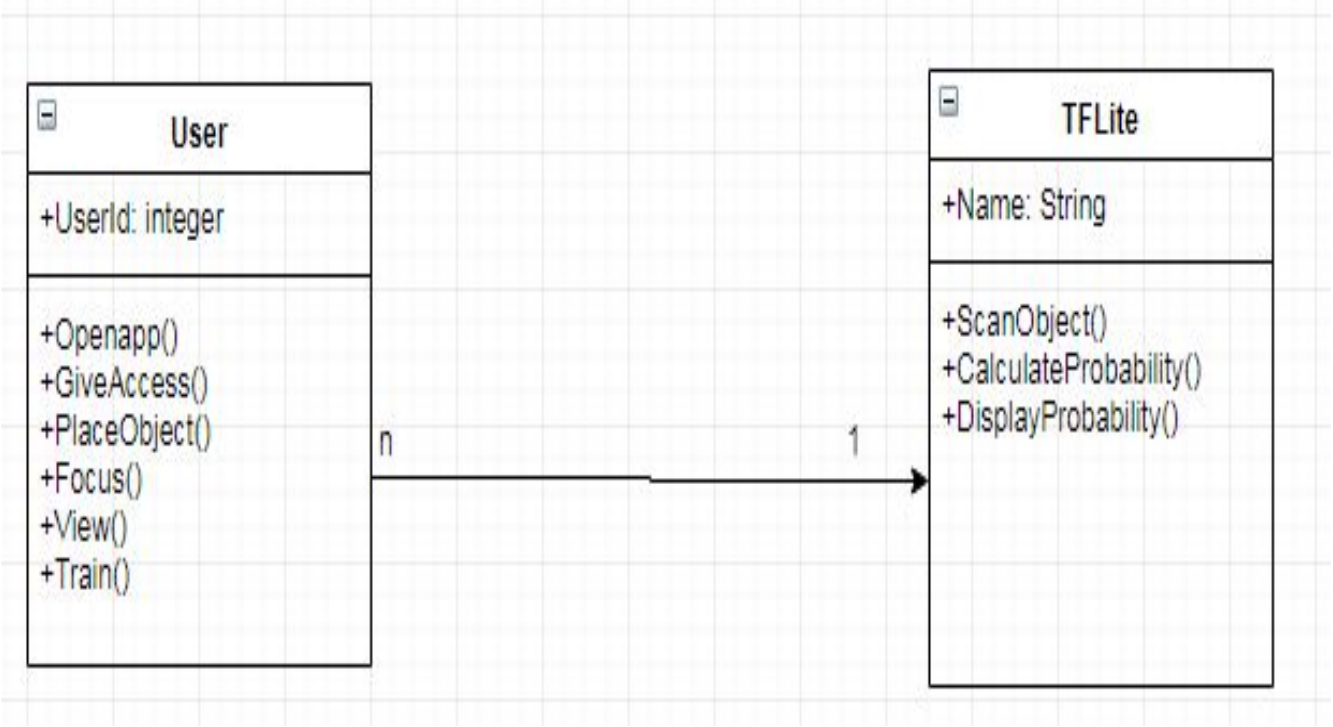


4. SYSTEM DESIGN – UML MODELING

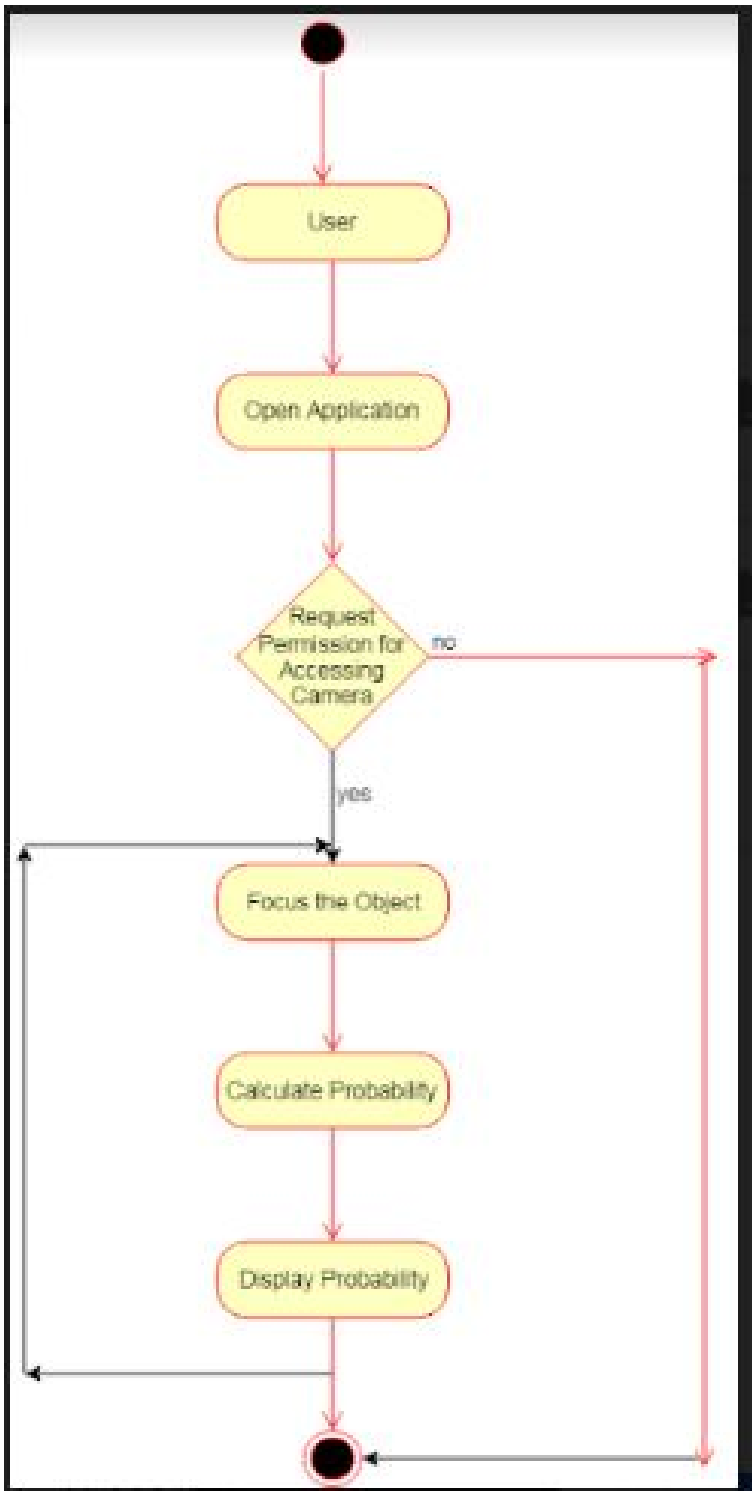
USE CASE DIAGRAM



CLASS DIAGRAM



ACTIVITY DIAGRAM



5. MODULE DESCRIPTION

Tensorflow module

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

Classifier module

Tensorflow comes with a pretrained module which is the classifier module. This module is capable of being trained by the user by feeding it with a dataset comprising of labelled data. The module identifies the object placed in front of the mobile camera and compared the image data with the existing image data obtained from the training dataset. Through extended training, the accuracy and efficiency of the application to classify the object is increased.

Training module

This module supports the Machine Learning paradigm of the Tensorflow framework. This module is capable of learning from a pre-trained dataset. That is , the module learns from enormous amounts of labelled data and is able to observe from the data and further learn from it.

6. CONCLUSION:

In conclusion, we have created an android application which makes use of the Tensorflow module. The module has been trained through the mobile app to identify images and classify them through probabilities. The training dataset was derived from imagenet and google images.

7. REFERENCES:

https://www.tensorflow.org/tutorials

http://www.image-net.org/

https://developers.google.com/

https://developer.android.com/

8. APPENDICES

8.1. CODE

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.tensorflow.demo">

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <application android:allowBackup="true"
        android:debuggable="true"
        android:label="@string/app_name"
        android:icon="@drawable/ic_launcher"
        android:theme="@style/MaterialTheme">

        <activity android:name="org.tensorflow.demo.ClassifierActivity"
                android:screenOrientation="portrait"
                android:label="@string/activity_name_classification">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
            </intent-filter>
```

```
</activity>

<activity android:name="org.tensorflow.demo.DetectorActivity"
        android:screenOrientation="portrait"
        android:label="@string/activity_name_detection">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="org.tensorflow.demo.StylizeActivity"
        android:screenOrientation="portrait"
        android:label="@string/activity_name_stylize">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="org.tensorflow.demo.SpeechActivity"
    android:screenOrientation="portrait"
    android:label="@string/activity_name_speech">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
            <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
        </intent-filter>
    </activity>
</application>


</manifest>
```

MainActivity

```
private const val REQUEST_PERMISSIONS = 1
private const val REQUEST_TAKE_PICTURE = 2

class MainActivity : AppCompatActivity() {

    private var photoFilePath = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        checkPermissions()
    }

    private fun checkPermissions() {
        if (arePermissionAlreadyGranted()) {
            takePhoto()
        } else {
            requestPermissions()
        }
    }

    private fun arePermissionAlreadyGranted() =
            ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED
```

```
    private fun requestPermissions() {
        ActivityCompat.requestPermissions(this,
                arrayOf(Manifest.permission.WRITE_EXTERNAL_STORAGE),
                REQUEST_PERMISSIONS)
    }

    private fun takePhoto() {

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).absol
utePath + "/${System.currentTimeMillis()}.jpg"
        val currentPhotoUri = UriHelper.getUriFromFilePath(this, photoFilePath)

        val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, currentPhotoUri)
        takePictureIntent.flags = Intent.FLAG_GRANT_READ_URI_PERMISSION

        if (takePictureIntent.resolveActivity(packageManager) != null) {
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PICTURE)
        }
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>,
grantResults: IntArray) {
        if (requestCode == REQUEST_PERMISSIONS && arePermissionGranted(grantResults)) {
            takePhoto()
        } else {
            requestPermissions()
        }
    }

    private fun arePermissionGranted(grantResults: IntArray) =
            grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        val file = File(photoFilePath)
        if (requestCode == REQUEST_TAKE_PICTURE && file.exists()) {
            //classify photo
        }
    }
```

```
    private val handler = Handler()
    private var photoFilePath = ""

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        val file = File(photoFilePath)
        if (requestCode == REQUEST_TAKE_PICTURE && file.exists()) {
            classifyPhoto(file)
        }
    }

    private fun classifyPhoto(file: File) {
        val photoBitmap = BitmapFactory.decodeFile(file.absolutePath)
        val croppedBitmap = ImageUtils.getCroppedBitmap(photoBitmap)
        classifyAndShowResult(croppedBitmap)
        imagePhoto.setImageBitmap(photoBitmap)
    }

    private fun classifyAndShowResult(croppedBitmap: Bitmap) {
        runInBackground(
                Runnable {
                    val result = classifier.recognizeImage(croppedBitmap)
                    //show result
                })
    }

    @Synchronized
    private fun runInBackground(runnable: Runnable) {
        handler.post(runnable)
    }
}private const val ENABLE_LOG_STATS = false

class ImageClassifier (
        private val inputName: String,
        private val outputName: String,
        private val imageSize: Long,
        private val labels: List<String>,
        private val imageBitmapPixels: IntArray,
        private val imageNormalizedPixels: FloatArray,
        private val results: FloatArray,
        private val tensorFlowInference: TensorFlowInferenceInterface
) : Classifier {
```

```
    override fun recognizeImage(bitmap: Bitmap): Result {
        preprocessImageToNormalizedFloats(bitmap)
        classifyImageToOutputs()
        val outputQueue = getResults()
        return outputQueue.poll()
    }

    private fun preprocessImageToNormalizedFloats(bitmap: Bitmap) {
        // Preprocess the image data from 0-255 int to normalized float based
        // on the provided parameters.
    }

    private fun classifyImageToOutputs() {
        //feed the classifier with the data via input
        tensorFlowInference.feed(inputName, imageNormalizedPixels,
                1L, imageSize, imageSize, COLOR_CHANNELS.toLong())
        //run the classification
        tensorFlowInference.run(arrayOf(outputName), ENABLE_LOG_STATS)
        //get the results from the ouptput
        tensorFlowInference.fetch(outputName, results)
    }

    private fun getResults(): PriorityQueue<Result> {
        val outputQueue = createOutputQueue()
        results.indices.mapTo(outputQueue) { Result(labels[it], results[it]) }
        return outputQueue
    }

    private fun createOutputQueue(): PriorityQueue<Result> {
        return PriorityQueue(
                initialCapacity = labels.size,
                Comparator { (_, rConfidence), (_, lConfidence) ->
                    Float.compare(lConfidence, rConfidence) })
    }
}
private fun classifyAndShowResult(croppedBitmap: Bitmap) {
        runInBackground(
                Runnable {
                    val result = classifier.recognizeImage(croppedBitmap)
                    showResult(result)
```

```
            })
    }

    @Synchronized
    private fun runInBackground(runnable: Runnable) {
        handler.post(runnable)
    }

    private fun showResult(result: Result) {
        textResult.text = result.result.toUpperCase()
        layoutContainer.setBackgroundColor(getColorFromResult(result.result))
    }

    @Suppress("DEPRECATION")
    private fun getColorFromResult(result: String): Int {
        return if (result == getString(R.string.hot)) {
            resources.getColor(R.color.hot)
        } else {
            resources.getColor(R.color.not)
        }
    }
}
}
```

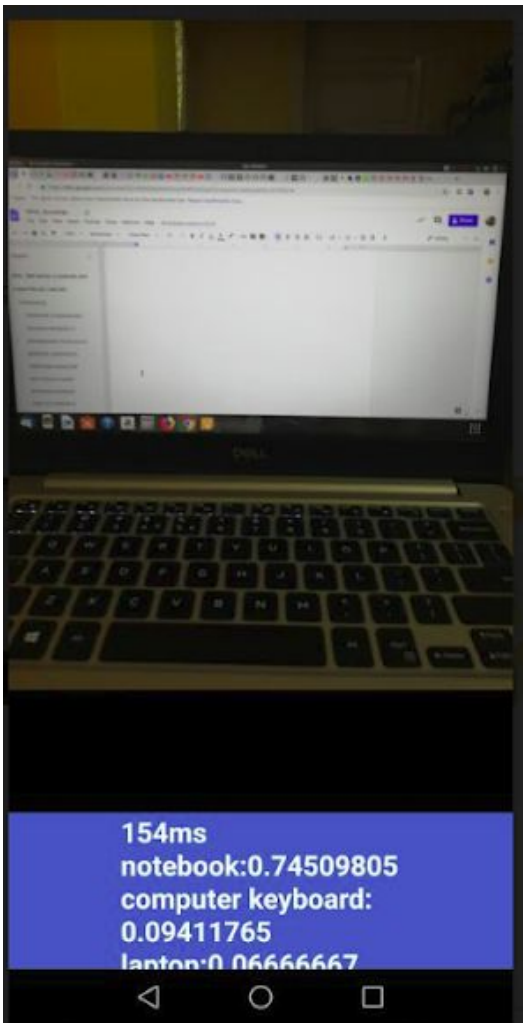
Const. Java

```
const val GRAPH_FILE_PATH = "file:///android_asset/graph.pb"
const val LABELS_FILE_PATH = "file:///android_asset/labels.txt"

const val GRAPH_INPUT_NAME = "input"
const val GRAPH_OUTPUT_NAME = "final_result"

const val IMAGE_SIZE = 224
```

8.2. SCREENSHOTS




COMPUTERS
147ms
mouse:0.8392157
CD player:0.101960786
remote control:0.011764706

CHAPTER 1

1.INTRODUCTION:

1.1 **Problem Statement:**

Recent advances in deep learning made tasks such as Image and speech recognition possible. Deep Learning is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. Deep learning excels in recognizing objects in images as it's implemented using 3 or more layers of artificial neural networks where each layer is responsible for extracting one or more feature of the image.A Neural Network is a computational model that works in a similar way to the neurons in the human brain. Each neuron takes an input, performs some operations then passes the output to the following neuron. Computers are able to perform computations on numbers and is unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand.

There are two common ways to do this in Image Processing:

1. Using Greyscale:

The image will be converted to greyscale (range of gray shades from white to black) the computer will assign each pixel a value based on how dark it is. All the numbers are put into an array and the computer does computations on that array.

2. Using RGB Values:

Colors could be represented as RGB values (a combination of red, green and blue ranging from 0 to 255). Computers could then extract the RGB value of each pixel and put the result in an array for interpretation.When the computer interprets a new image, it will convert the image to an array by using the same technique, which then compares the patterns of numbers against the already-known objects. The computer then allots confidence scores for each class. The class with the highest confidence score is usually the predicted one.

One of the most popular techniques used in improving the accuracy of image classification is Convolutional Neural Networks. Convolutional Neural Network is a special type Neural Networks that works in the same way of a regular neural network except that it has a convolution layer at the beginning.Instead of feeding the entire image as an array of numbers, the image is broken up into a number of tiles, the machine then tries to predict what each tile is. Finally, the computer tries to predict what's in the picture based on the prediction of all the tiles.

CHAPTER 2

2. REQUIREMENTS

2.1 Functional Requirements

- Smartphone with camera
- Stand-alone computer
- Android Studio
- Tensorflow classification module
- Training dataset
- Training module
- Classified data
- Image probability module
- Wifi router
- Imagenet database

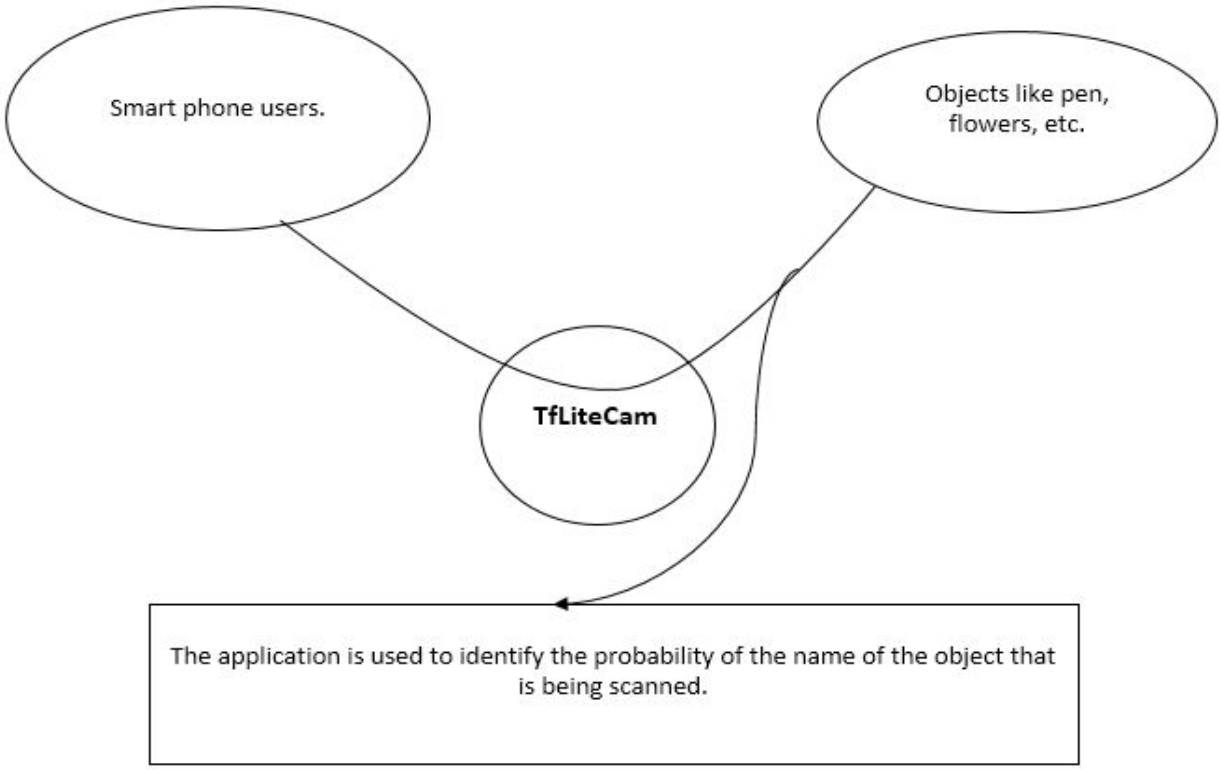
2.2 Non- Functional Requirements

- Accessibility
- Consistency
- Reliability
- Mobility
- Modifiability
- Efficiency
- Accuracy

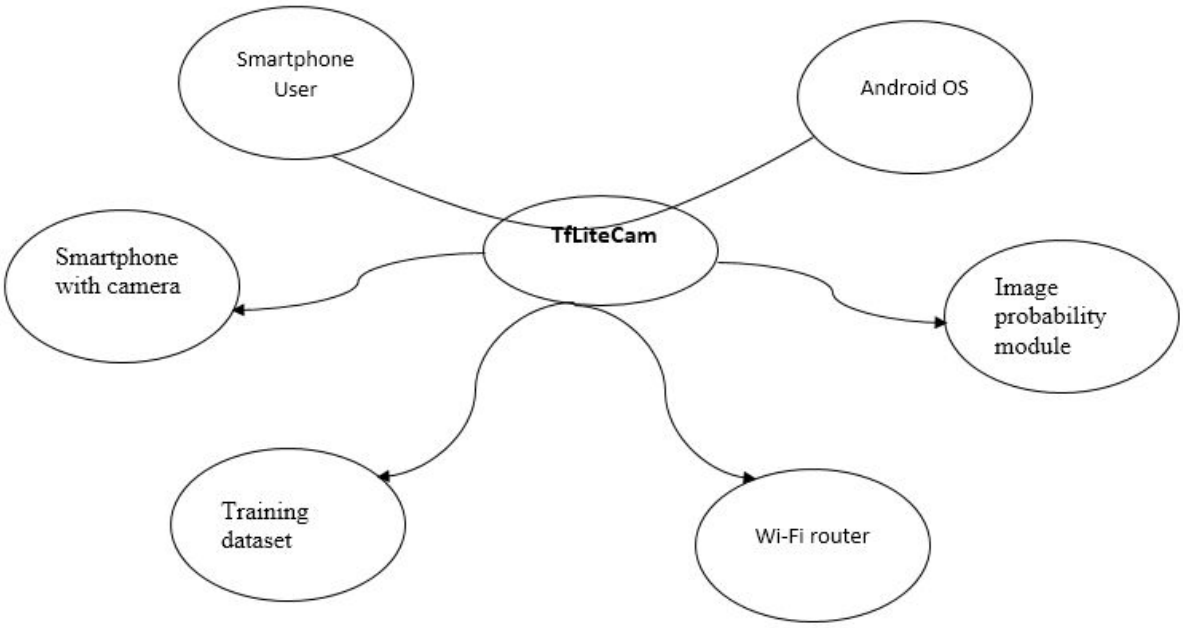
CHAPTER 3

3. Analysis

3.1 Presentation of bull diagram



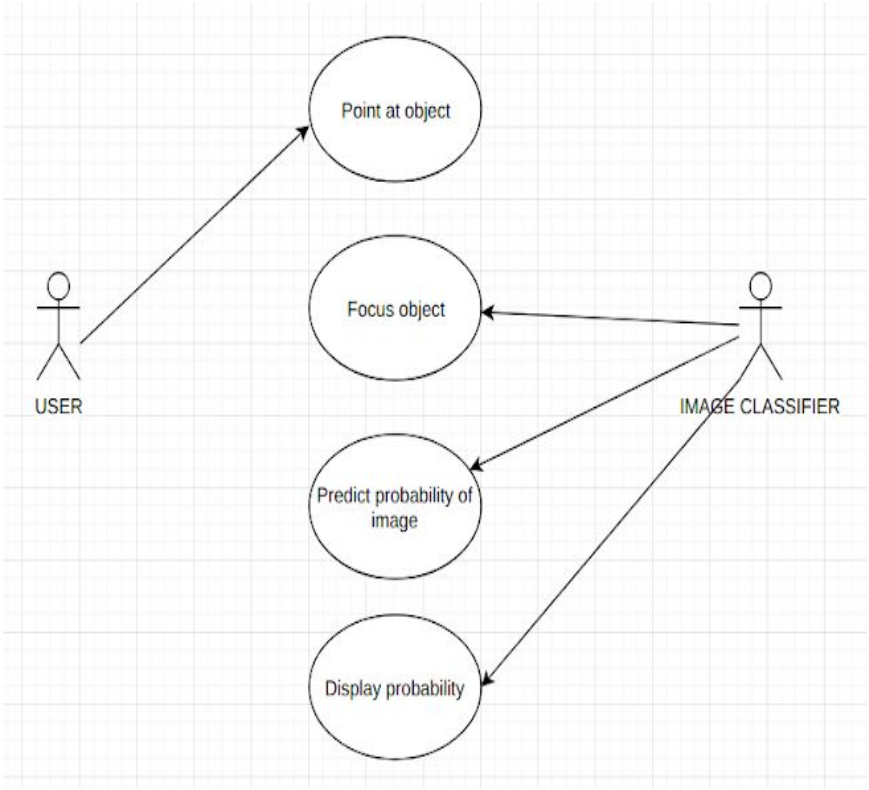
3.2 Presentation of octopus diagram



CHAPTER 4

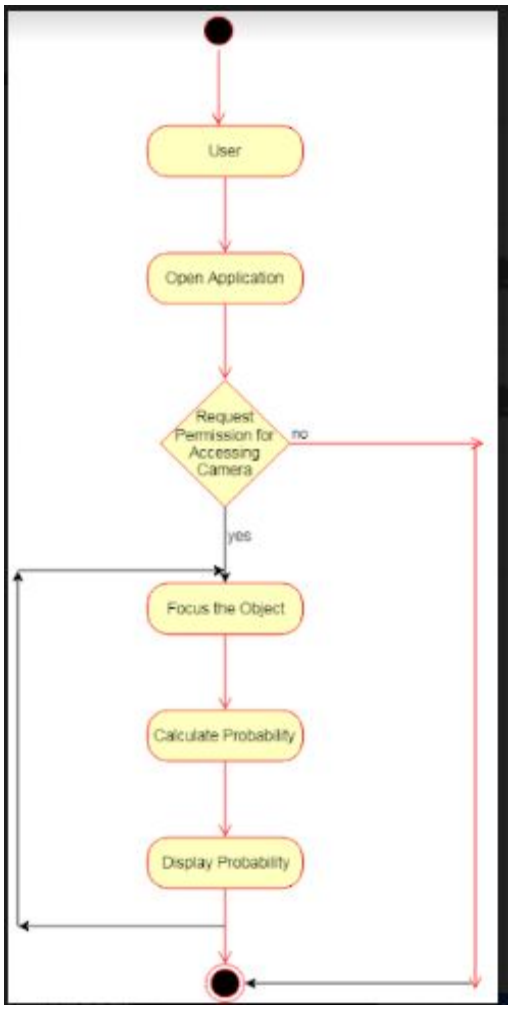
4. SYSTEM DESIGN – UML MODELING

4.1 USE CASE DIAGRAM

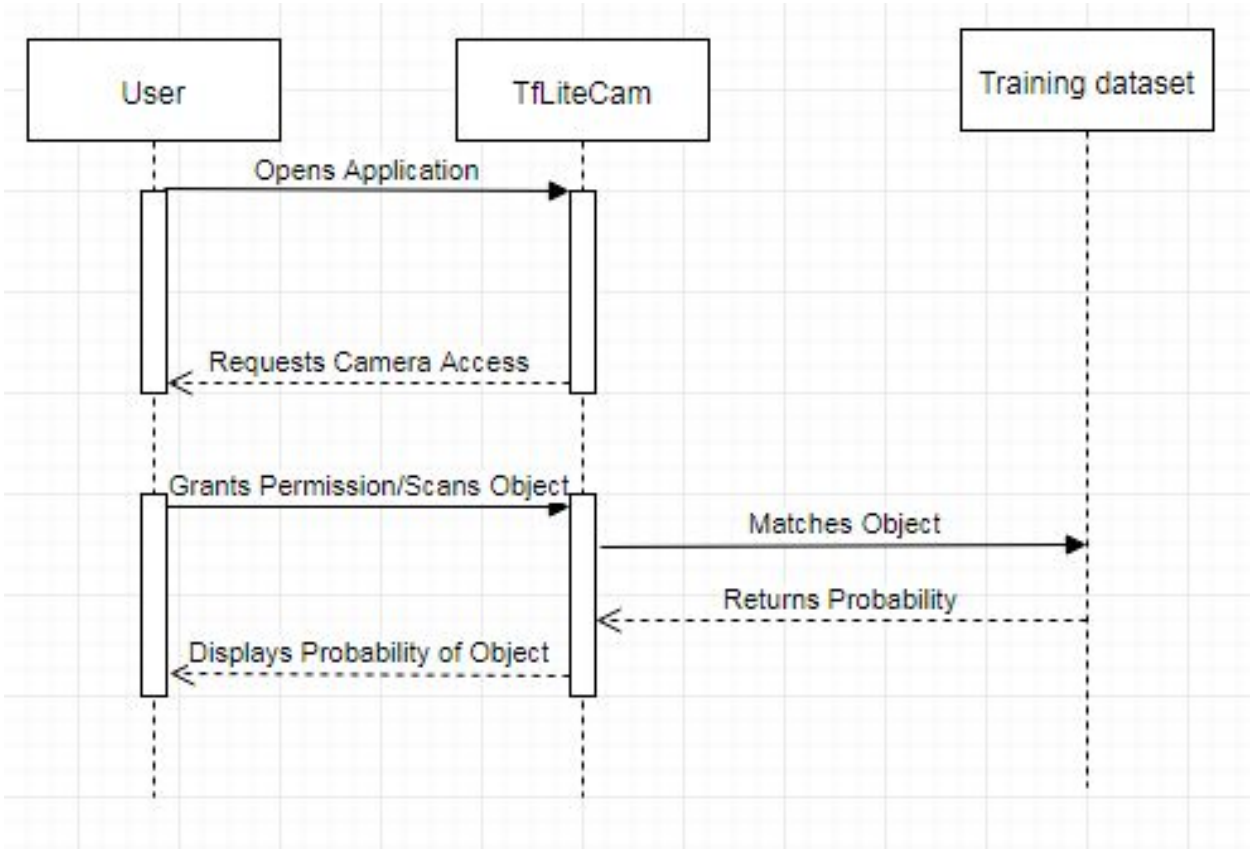


4.2 CLASS DIAGRAM

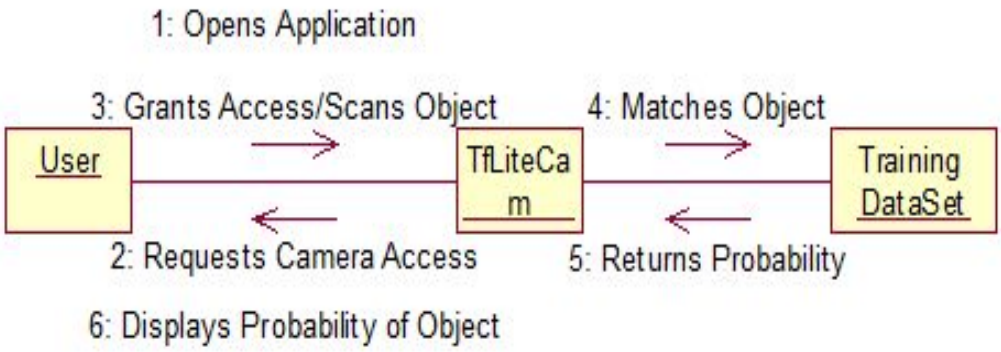
4.3 ACTIVITY DIAGRAM



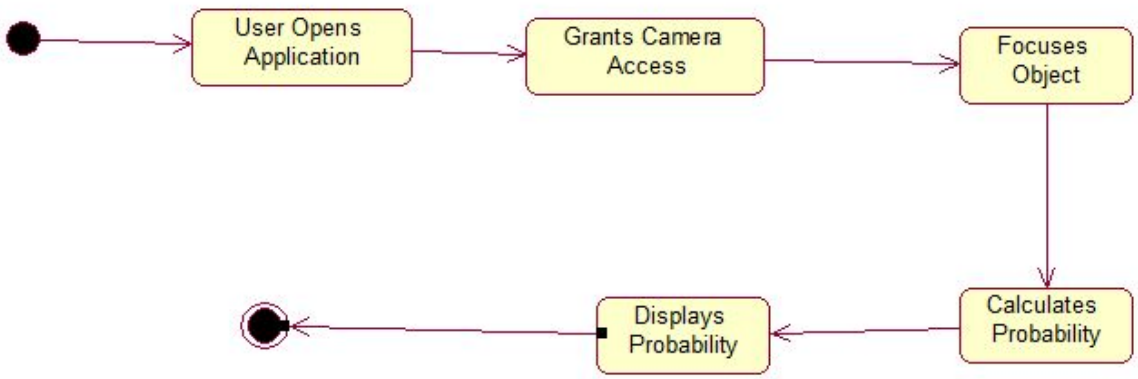
4.4 SEQUENCE DIAGRAM



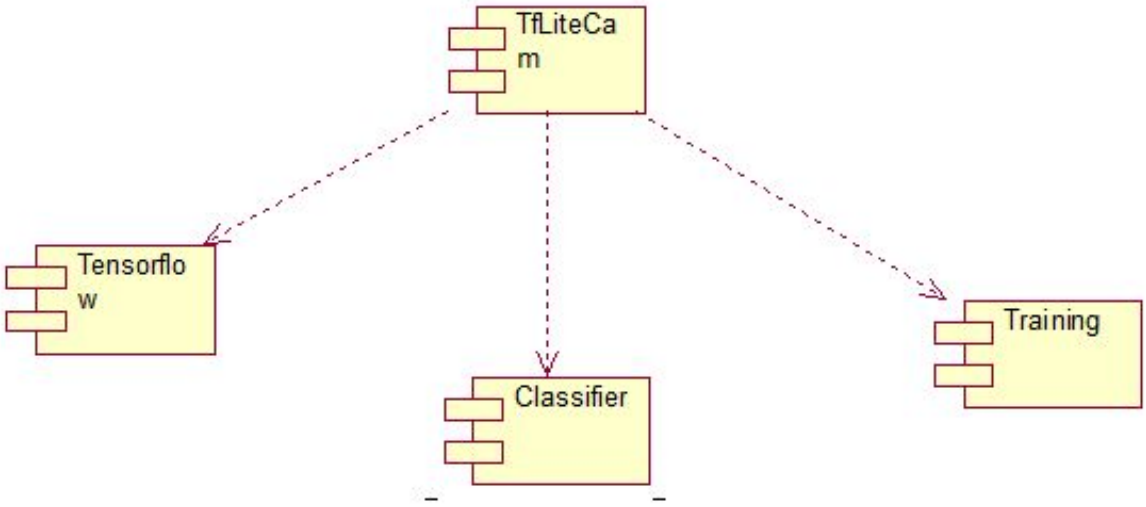
4.5 COLLABORATION DIAGRAM



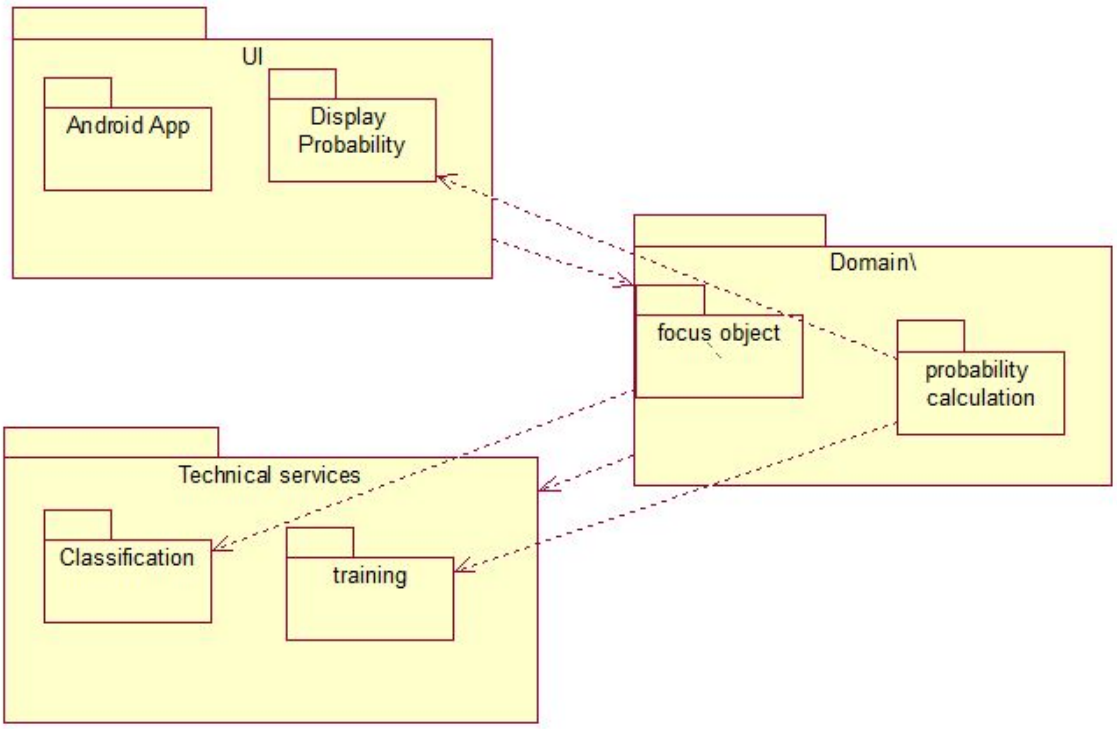
4.6 STATE CHART



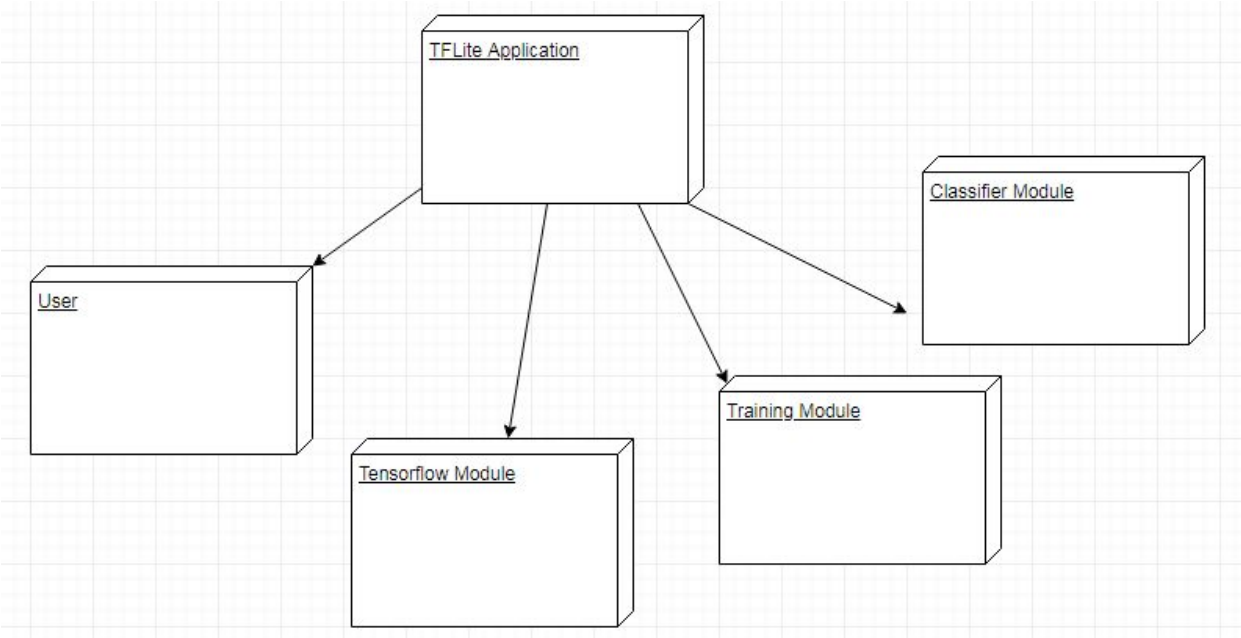
4.7 COMPONENT DIAGRAM



4.8 PACKAGE DIAGRAM



4.9. DEPLOYMENT DIAGRAM



CHAPTER 5

5. Module Description

Tensorflow module

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

Classifier module

Tensorflow comes with a pretrained module which is the classifier module. This module is capable of being trained by the user by feeding it with a dataset comprising of labelled data. The module identifies the object placed in front of the mobile camera and compared the image data with the existing image data obtained from the training dataset. Through extended training, the accuracy and efficiency of the application to classify the object is increased.

Training module

This module supports the Machine Learning paradigm of the Tensorflow framework. This module is capable of learning from a pre-trained dataset. That is , the module learns from enormous amounts of labelled data and is able to observe from the data and further learn from it.

CHAPTER 6

6. CONCLUSION:

In conclusion, we have created an android application which makes use of the Tensorflow module. The module has been trained through the mobile app to identify images and classify them through probabilities. The training dataset was derived from imagenet and google images. The dataset was used to train the module. The classification of the objects will help the user to have a clear knowledge about the focused object and also the objects related to the focused object. The probability displayed is the measure that will assist the user. More accuracy can be achieved by training the module with more number of datasets and also with different varieties.

CHAPTER 7

7. REFERENCES

- https://www.tensorflow.org/tutorials
- http://www.image-net.org/
- https://developers.google.com/
- https://developer.android.com/

CHAPTER 8

8. APPENDICES:

8.1 Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.tensorflow.demo">

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <application android:allowBackup="true"
        android:debuggable="true"
        android:label="@string/app_name"
        android:icon="@drawable/ic_launcher"
        android:theme="@style/MaterialTheme">

        <activity android:name="org.tensorflow.demo.ClassifierActivity"
                android:screenOrientation="portrait"
                android:label="@string/activity_name_classification">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
            </intent-filter>
```

```
</activity>

<activity android:name="org.tensorflow.demo.DetectorActivity"
        android:screenOrientation="portrait"
        android:label="@string/activity_name_detection">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="org.tensorflow.demo.StylizeActivity"
        android:screenOrientation="portrait"
        android:label="@string/activity_name_stylize">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="org.tensorflow.demo.SpeechActivity"
    android:screenOrientation="portrait"
    android:label="@string/activity_name_speech">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
            <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
        </intent-filter>
    </activity>
</application>


</manifest>
```

MainActivity

```
private const val REQUEST_PERMISSIONS = 1
private const val REQUEST_TAKE_PICTURE = 2

class MainActivity : AppCompatActivity() {

    private var photoFilePath = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        checkPermissions()
    }

    private fun checkPermissions() {
        if (arePermissionAlreadyGranted()) {
            takePhoto()
        } else {
            requestPermissions()
        }
    }

    private fun arePermissionAlreadyGranted() =
            ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED
```

```
    private fun requestPermissions() {
        ActivityCompat.requestPermissions(this,
                arrayOf(Manifest.permission.WRITE_EXTERNAL_STORAGE),
                REQUEST_PERMISSIONS)
    }

    private fun takePhoto() {

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).absol
utePath + "/${System.currentTimeMillis()}.jpg"
        val currentPhotoUri = UriHelper.getUriFromFilePath(this, photoFilePath)

        val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, currentPhotoUri)
        takePictureIntent.flags = Intent.FLAG_GRANT_READ_URI_PERMISSION

        if (takePictureIntent.resolveActivity(packageManager) != null) {
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PICTURE)
        }
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>,
grantResults: IntArray) {
        if (requestCode == REQUEST_PERMISSIONS && arePermissionGranted(grantResults)) {
            takePhoto()
        } else {
            requestPermissions()
        }
    }

    private fun arePermissionGranted(grantResults: IntArray) =
            grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        val file = File(photoFilePath)
        if (requestCode == REQUEST_TAKE_PICTURE && file.exists()) {
            //classify photo
        }
    }
```

```
    private val handler = Handler()
    private var photoFilePath = ""

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        val file = File(photoFilePath)
        if (requestCode == REQUEST_TAKE_PICTURE && file.exists()) {
            classifyPhoto(file)
        }
    }

    private fun classifyPhoto(file: File) {
        val photoBitmap = BitmapFactory.decodeFile(file.absolutePath)
        val croppedBitmap = ImageUtils.getCroppedBitmap(photoBitmap)
        classifyAndShowResult(croppedBitmap)
        imagePhoto.setImageBitmap(photoBitmap)
    }

    private fun classifyAndShowResult(croppedBitmap: Bitmap) {
        runInBackground(
                Runnable {
                    val result = classifier.recognizeImage(croppedBitmap)
                    //show result
                })
    }

    @Synchronized
    private fun runInBackground(runnable: Runnable) {
        handler.post(runnable)
    }
}private const val ENABLE_LOG_STATS = false

class ImageClassifier (
        private val inputName: String,
        private val outputName: String,
        private val imageSize: Long,
        private val labels: List<String>,
        private val imageBitmapPixels: IntArray,
        private val imageNormalizedPixels: FloatArray,
        private val results: FloatArray,
        private val tensorFlowInference: TensorFlowInferenceInterface
) : Classifier {
```

```
    override fun recognizeImage(bitmap: Bitmap): Result {
        preprocessImageToNormalizedFloats(bitmap)
        classifyImageToOutputs()
        val outputQueue = getResults()
        return outputQueue.poll()
    }

    private fun preprocessImageToNormalizedFloats(bitmap: Bitmap) {
        // Preprocess the image data from 0-255 int to normalized float based
        // on the provided parameters.
    }

    private fun classifyImageToOutputs() {
        //feed the classifier with the data via input
        tensorFlowInference.feed(inputName, imageNormalizedPixels,
                1L, imageSize, imageSize, COLOR_CHANNELS.toLong())
        //run the classification
        tensorFlowInference.run(arrayOf(outputName), ENABLE_LOG_STATS)
        //get the results from the ouptput
        tensorFlowInference.fetch(outputName, results)
    }

    private fun getResults(): PriorityQueue<Result> {
        val outputQueue = createOutputQueue()
        results.indices.mapTo(outputQueue) { Result(labels[it], results[it]) }
        return outputQueue
    }

    private fun createOutputQueue(): PriorityQueue<Result> {
        return PriorityQueue(
                initialCapacity = labels.size,
                Comparator { (_, rConfidence), (_, lConfidence) ->
                    Float.compare(lConfidence, rConfidence) })
    }
}
private fun classifyAndShowResult(croppedBitmap: Bitmap) {
        runInBackground(
                Runnable {
                    val result = classifier.recognizeImage(croppedBitmap)
                    showResult(result)
```

```
            })
    }

    @Synchronized
    private fun runInBackground(runnable: Runnable) {
        handler.post(runnable)
    }

    private fun showResult(result: Result) {
        textResult.text = result.result.toUpperCase()
        layoutContainer.setBackgroundColor(getColorFromResult(result.result))
    }@Suppress("DEPRECATION")
    private fun getColorFromResult(result: String): Int {
        return if (result == getString(R.string.hot)) {
            resources.getColor(R.color.hot)
        } else {
            resources.getColor(R.color.not)
        } }
}}
```

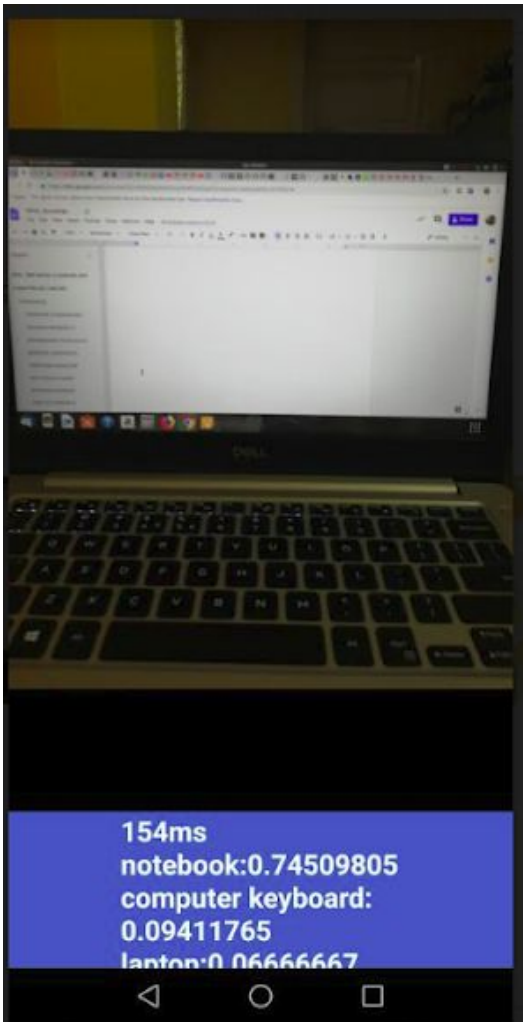
Const. Java

```
const val GRAPH_FILE_PATH = "file:///android_asset/graph.pb"
const val LABELS_FILE_PATH = "file:///android_asset/labels.txt"

const val GRAPH_INPUT_NAME = "input"
const val GRAPH_OUTPUT_NAME = "final_result"

const val IMAGE_SIZE = 224
```

SCREENSHOTS:




COMPUTERS
147ms
mouse:0.8392157
CD player:0.101960786
remote control:0.011764706