**GUI implementation**

1. **Import Dependency’s**

Before we start to code the frontend we need to import some dependency’s into the Gradle file

* Tensorflow 1.2
* Camera Kit 0.13.1
* Sensey 1.8.0

Import the above dependency’s into build.gradle file as show below :-

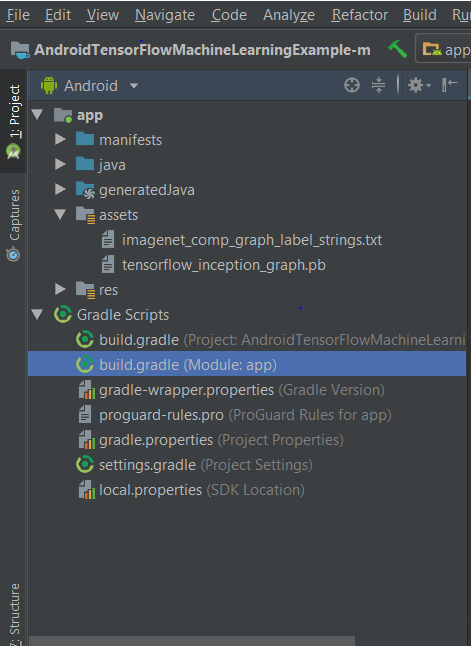
//import tensorflow

compile 'org.tensorflow:tensorflow-android:1.2.0'

//import camerakit   
compile 'com.wonderkiln:camerakit:0.13.1'  
//implement sensey  
compile 'com.github.nisrulz:sensey:1.8.0'

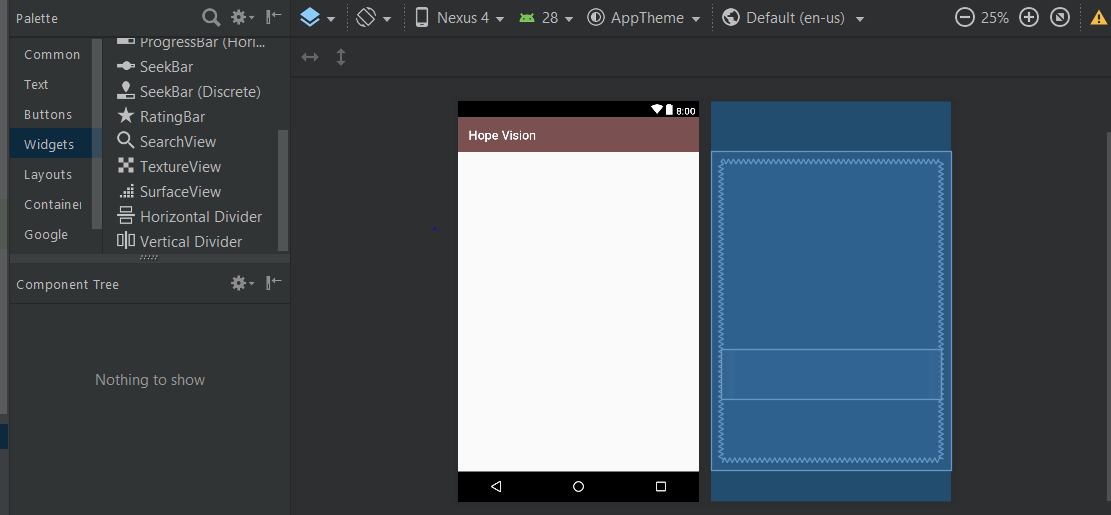
1. **Import assets**

**A**fter importing all the required dependency’s, now we import the assets into the assets folder as shown below



* The tensorflow\_inception\_graph.pb consists of inception v3 model which we have trained using tensorflow .
* imagenet\_comp\_graph\_label\_strings.txt consists of all the labels corresponding graph

1. **Implementing Camera Activity**

The first step to implement the camera activity we start by building layout for the camera activity 

The associated xml layout code for the above layout

# activity\_main.xml

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.swarathesh.hopevision.MainActivity">  
  
 <com.wonderkiln.camerakit.CameraView  
 android:id="@+id/cameraView"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:layout\_gravity="center|top" />  
  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="80dp"  
 android:layout\_gravity="center|top"  
 android:layout\_marginTop="300dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:id="@+id/imageViewResult"  
 android:layout\_width="75dp"  
 android:visibility="gone"  
 android:layout\_height="75dp"  
 android:padding="2dp" />

<TextView  
 android:id="@+id/textViewResult"  
 android:layout\_width="match\_parent"  
 android:layout\_height="80dp"  
 android:fadeScrollbars="false"  
 android:maxLines="15"  
 android:scrollbars="vertical"  
 android:gravity="center"  
 android:textColor="@android:color/black" />  
  
 </LinearLayout>  
  
  
  
</FrameLayout>

Now to implement the business logic for the above view we implement the following java code

# MainActivity.java

In this activity we use TensorFlow native interface to classify and label the images

package com.swarathesh.hopevision;  
  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.os.Bundle;  
import android.support.v7.app.AppCompatActivity;  
import android.text.method.ScrollingMovementMethod;  
import android.view.View;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import android.speech.tts.TextToSpeech;  
import android.widget.Toast;

import all the library’s necessary

import com.github.nisrulz.sensey.Sensey;  
import com.wonderkiln.camerakit.CameraKitError;  
import com.wonderkiln.camerakit.CameraKitEvent;  
import com.wonderkiln.camerakit.CameraKitEventListener;  
import com.wonderkiln.camerakit.CameraKitImage;  
import com.wonderkiln.camerakit.CameraKitVideo;  
import com.wonderkiln.camerakit.CameraView;  
  
import java.text.Normalizer;  
import java.util.List;  
import java.util.Locale;  
import java.util.concurrent.Executor;  
import java.util.concurrent.Executors;  
  
//import senesy  
import com.github.nisrulz.sensey.ProximityDetector;

*/\*\*  
 \* Created by swarathesh on 06/10/18.  
 \*/*  
  
  
public class MainActivity extends AppCompatActivity implements TextToSpeech.OnInitListener{  
 //size of sampling  
 private static final int *Size* = 224;  
 //size of mean for sampling   
 private static final int *Mean* = 117;  
 private static final float *std* = 1;  
 private static final String *INPUT* = "input";  
 private static final String *OUTPUT* = "output";  
   
 //implement text to speech for reading out the results to users  
 private TextToSpeech tts;  
  
 //import assets  
 private static final String *MODEL\_FILE* = "file:///android\_asset/tensorflow\_inception\_graph.pb";  
 private static final String *LABEL\_FILE* = "file:///android\_asset/imagenet\_comp\_graph\_label\_strings.txt";  
  
 private Classifier classifier;  
 private Executor Theadexecutor = Executors.*newSingleThreadExecutor*();  
 private TextView Result;  
 private ImageView imageViewResult;  
 private CameraView cameraView;  
 private Sensey sensey;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);

//initiate the camera view and text view to display the results on the screen

cameraView = (CameraView) findViewById(R.id.*cameraView*);  
 imageViewResult = (ImageView) findViewById(R.id.*imageViewResult*);  
 Result = (TextView) findViewById(R.id.*textViewResult*);  
  
 Result.setMovementMethod(new ScrollingMovementMethod());  
  
 //initiate sensey

Sensey.*getInstance*().init(getApplicationContext());

//initiate text to speech

We use ttp to speak out the results to the visually impaired

tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 if (i != TextToSpeech.*ERROR*){  
 tts.setLanguage(Locale.*CANADA*);  
 }  
 }  
 });  
  
  
 cameraView.addCameraKitListener(new CameraKitEventListener() {  
 @Override  
 public void onEvent(CameraKitEvent cameraKitEvent) {  
  
 }  
  
 @Override  
 public void onError(CameraKitError cameraKitError) {  
  
 }  
  
 @Override  
 public void onImage(CameraKitImage cameraKitImage) {  
  
 Bitmap bitmap = cameraKitImage.getBitmap();  
  
 bitmap = Bitmap.*createScaledBitmap*(bitmap, *Size*, *Size*, false);  
  
 imageViewResult.setImageBitmap(bitmap);

//the captured image is sent classifier instance to classify and label the image

//store all the results in a list   
 final List<Classifier.Recognition> results = classifier.recognizeImage(bitmap);  
  
 Result.setText(results.toString());  
  
 String speak = results.toString();  
  
 String input = speak;  
 String withoutAccent = Normalizer.*normalize*(input, Normalizer.Form.*NFD*);  
 String output = withoutAccent.replaceAll("[^a-zA-Z ]", "");

//speaks out the results to the visually imapired  
  
 tts.speak("i see "+output,TextToSpeech.*QUEUE\_FLUSH*,null);  
  
  
  
 }  
  
 @Override  
 public void onVideo(CameraKitVideo cameraKitVideo) {  
  
 }  
 });  
  
 //initaite the proximity sensor so when the user triggers the proximity sensor the classifier is triggered.

ProximityDetector.ProximityListener proximityListener = new ProximityDetector.ProximityListener() {  
 @Override  
 public void onFar() {  
 Toast.*makeText*(getApplicationContext(),"object to far make the user come closer !",Toast.*LENGTH\_LONG*).show();  
 }  
  
 @Override  
 public void onNear() {

//we capture the image and send it as a parameter to tensorflow classifier to classify and label the images

cameraView.captureImage();  
 }  
 };  
  
  
  
  
 //initiate tensflow and load the model using native interface  
  
 private void initTensorFlowAndLoadModel() {

//we create a new thread and execute the classifier instance because the UI thread can not manage heavy load  
 Theadexecutor.execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 classifier = TensorFlowImageClassifier.*create*(  
 getAssets(),  
 *MODEL\_FILE*,  
 *LABEL\_FILE*,  
 *Size*,  
 *Mean*,  
 *std*,  
 *INPUT*,  
 *OUTPUT*);  
   
 } catch (final Exception e) {  
 throw new RuntimeException("Error initializing TensorFlow!", e);  
 }  
 }  
 });  
 }  
  
  
}

1. **Tensorflow classifier**

In this java class we create an instance of Tensorflow classifier and classify the image that the MainActivity has sent as a parameter.

package com.swarathesh.hopevision;  
  
import android.content.res.AssetManager;  
import android.graphics.Bitmap;  
import android.support.v4.os.TraceCompat;  
import android.util.Log;  
  
import org.tensorflow.contrib.android.TensorFlowInferenceInterface;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
import java.util.PriorityQueue;  
import java.util.Vector;  
  
*/\*\*  
 \* Created by swarathesh on 06/10/18.  
 \*/*public class TFmageClassifier implements Classifier {  
  
 private static final String *TAG* = "ImageClassifier";  
  
 // Only return this many results with at least this confidence.  
 private static final int *RESULTS* = 3;  
 private static final float *TH* = 0.1f;  
  
 // Config values.  
 private String input;  
 private String output;  
 private int inSize;  
 private int Mean;  
 private float Std;  
  
 // Pre-allocated buffers.  
 private Vector<String> labels = new Vector<String>();  
 private int[] Val;  
 private float[] floatVal;  
 private float[] outputs;  
 private String[] outputNames;  
  
 private TensorFlowInferenceInterface inferenceInterface;  
  
 private boolean runStats = false;  
  
 private TFmageClassifier() {  
 }  
  
  
 public static Classifier create(  
 AssetManager assetManager,  
 String modelFilename,  
 String labelFilename,  
 int inputSize,  
 int imageMean,  
 float imageStd,  
 String inputName,  
 String outputName)  
 throws IOException {  
 TFmageClassifier c = new TFmageClassifier();  
 c.input = inputName;  
 c.outputName = outputName;  
  
  
 String actualFilename = labelFilename.split("file:///android\_asset/")[1];  
 Log.*i*(*TAG*, "Reading labels from: " + actualFilename);  
 BufferedReader br = null;  
 br = new BufferedReader(new InputStreamReader(assetManager.open(actualFilename)));  
 String line;  
 while ((line = br.readLine()) != null) {  
 c.labels.add(line);  
 }  
 br.close();  
  
 c.inferenceInterface = new TensorFlowInferenceInterface(assetManager, modelFilename);  
  
 int numClasses =  
 (int) c.inferenceInterface.graph().operation(outputName).output(0).shape().size(1);  
 Log.*i*(*TAG*, "Read " + c.labels.size() + " labels, output layer size is " + numClasses);  
  
  
 c.inSize = inputSize;  
 c.Mean = imageMean;  
 c.Std = imageStd;  
  
 // Pre-allocate buffers.  
 c.outputNames = new String[]{outputName};  
 c.Val = new int[inputSize \* inputSize];  
 c.floatVal = new float[inputSize \* inputSize \* 3];  
 c.outputs = new float[numClasses];  
  
 return c;  
 }  
  
 @Override  
 public List<Recognition> recTFimage(final Bitmap bitmap) {  
  
 TraceCompat.*beginSection*("recTFimage");  
  
 TraceCompat.*beginSection*("preprocessBitmap");  
  
 bitmap.getPixels(Val, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(), bitmap.getHeight());

//convert image to float array so that we can use it in our classifier  
 for (int i = 0; i < Val.length; ++i) {  
 final int val = Val[i];  
 floatVal[i \* 3 + 0] = (((val >> 16) & 0xFF) - Mean) / Std;  
 floatVal[i \* 3 + 1] = (((val >> 8) & 0xFF) - Mean) / Std;  
 floatVal[i \* 3 + 2] = ((val & 0xFF) - Mean) / Std;  
 }  
 TraceCompat.*endSection*();  
  
  
 inferenceInterface.feed(  
 input, floatVal, new long[]{1, inSize, inSize, 3});  
 TraceCompat.*endSection*();  
  
  
   
 inferenceInterface.run(outputNames, runStats);  
  
  
 inferenceInterface.fetch(outputName, outputs);

Compare the captured with the graph and return the best possible results  
 PriorityQueue<Recognition> pq =  
 new PriorityQueue<Recognition>(  
 3,  
 new Comparator<Recognition>() {  
 @Override  
 public int compare(Recognition lhs, Recognition rhs) {  
  
 return Float.*compare*(rhs.getConfidence(), lhs.getConfidence());  
 }  
 });  
 for (int i = 0; i < outputs.length; ++i) {  
 if (outputs[i] > *TH*) {  
 pq.add(  
 new Recognition(  
 "" + i, labels.size() > i ? labels.get(i) : "unknown", outputs[i], null));  
 }  
 }  
 final ArrayList<Recognition> recognitions = new ArrayList<Recognition>();  
 int recognitionsSize = Math.*min*(pq.size(), *RESULTS*);  
 for (int i = 0; i < recognitionsSize; ++i) {  
 recognitions.add(pq.poll());  
 }  
 TraceCompat.*endSection*();   
 return recognitions;  
 }  
  
   
}

1. **Classifier model classs**

The model class through which the tensorflow instance objects are modeled after.

*/\*\*  
 \* Created by swarathesh on 06/10/18.  
 \*/*  
package com.swarathesh.hopevision;  
  
import android.graphics.Bitmap;  
import android.graphics.RectF;  
  
import java.util.List;  
  
  
public interface Classifier {  
   
 public class Recognition {  
   
 private final String id;  
 private final String ImageLable;  
 private final Float ConfidencePercentage;  
 private RectF loc;  
  
 public Recognition(  
 final String id, final String title, final Float confidence, final RectF location) {  
 this.id = id;  
 this.ImageLable = title;  
 this.ConfidencePercentage = confidence;  
 this.loc = location;  
 }  
  
 public String getId() {  
 return id;  
 }  
  
 public String getImageLable() {  
 return ImageLable;  
 }  
  
 public Float getConfidencePercentage() {  
 return ConfidencePercentage;  
 }  
  
 public RectF getLoc() {  
 return new RectF(loc);  
 }  
  
 public void setLoc(RectF loc) {  
 this.loc = loc;  
 }  
  
 @Override  
 public String toString() {  
 String resultString = "";  
 if (id != null) {  
 resultString += "[" + id + "] ";  
 }  
  
 if (ImageLable != null) {  
 resultString += ImageLable + " ";  
 }  
  
 if (ConfidencePercentage != null) {  
 resultString += String.*format*("(%.1f%%) ", ConfidencePercentage \* 100.0f);  
 }  
  
 if (loc != null) {  
 resultString += loc + " ";  
 }  
  
 return resultString.trim();  
 }  
 }  
  
 List<Recognition> recognizeImage(Bitmap bitmap);  
  
 void enableStatLogging(final boolean debug);  
  
 String getStatString();  
  
 void close();  
}