ACKNOWLEDGEMENT

We take this opportunity to present our votes of thanks to all that guidepost who really acted as lightening pillars to enlighten our way throughout this project that has led to successful and satisfactory completion of this study.

We are really grateful to our Dr. (Prof.) Parth Shah, HOD - Department Of Information Technology, Dean - Faculty of Technology & Engineering for providing us with an opportunity to undertake this project in this university and providing us with all the facilities. We are highly thankful to Prof. Jalpesh Vasa for their active support, valuable time and advice during the study and in completing the assignment of preparing the said project within the time stipulated.

Lastly, we are thankful to all those, particularly the various friends, who have been instrumental in creating proper, healthy and conductive environment and including new and fresh innovative ideas for us during the project, their help, it would have been extremely difficult for us to prepare the project in a time bound framework.

ABSTRACT

The task of finding objects belonging to classes of interest in images has long been a focus of Computer Vision research. The ability to localize objects is useful in many applications: from self-driving cars, where it allows the car to detect pedestrians, bicyclists, road signs, and other vehicles, to security, where intruding persons can be detected. Though a lot of progress has been made since the conception of the field of Computer Vision more than five decades ago, as always, there is scope for further improvement. This is especially true in the case of object detection where a myriad of factors including variation in object instances through pose and appearance, along with other environmental factors such as the degree of occlusion, and lighting tend to cause failures. In this work we focus on improving object detection through the use of more representative features and better models. We propose new features that are not only more powerful, but also more robust and capture more information than the currently popular features. Further, we propose scalable models which can leverage large amounts of training data to improve performance.

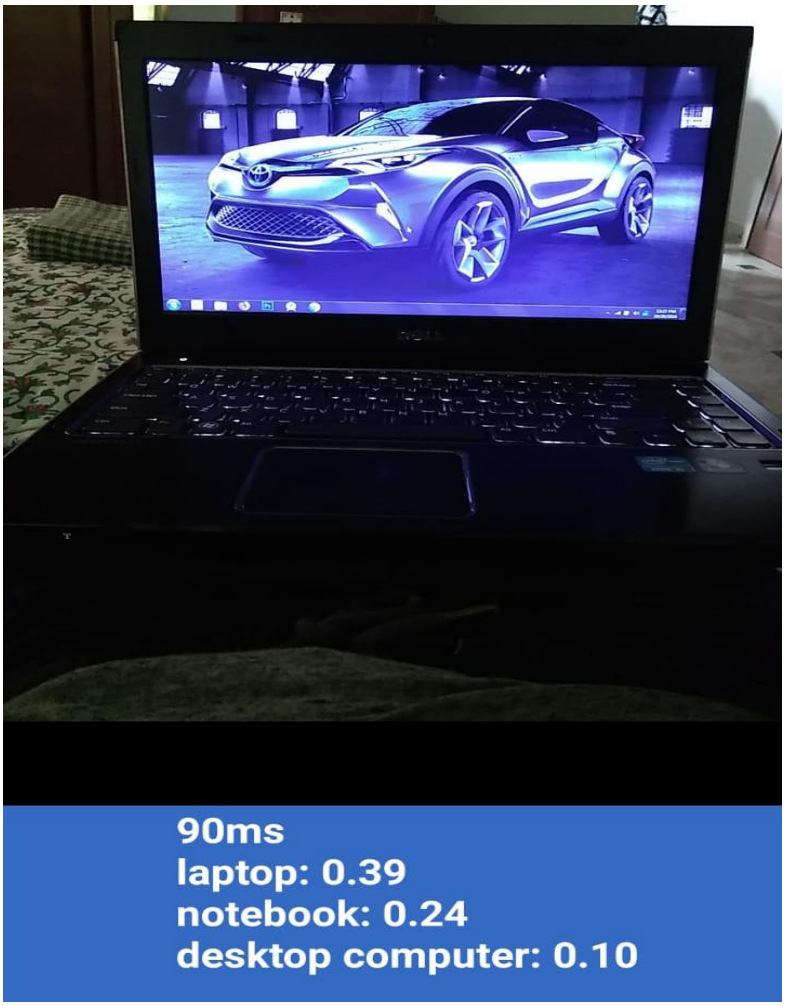

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Object Identification using Android Studio and AI (Artificial Intelligence). In this application basically, there is a camera which learns about the object near by or at some distance. It is machine learning concept. We humans can easily understand what that object is just by looking at it once, but in this application, machine is made to learn about the object nearby, not so precisely but with some accuracy.

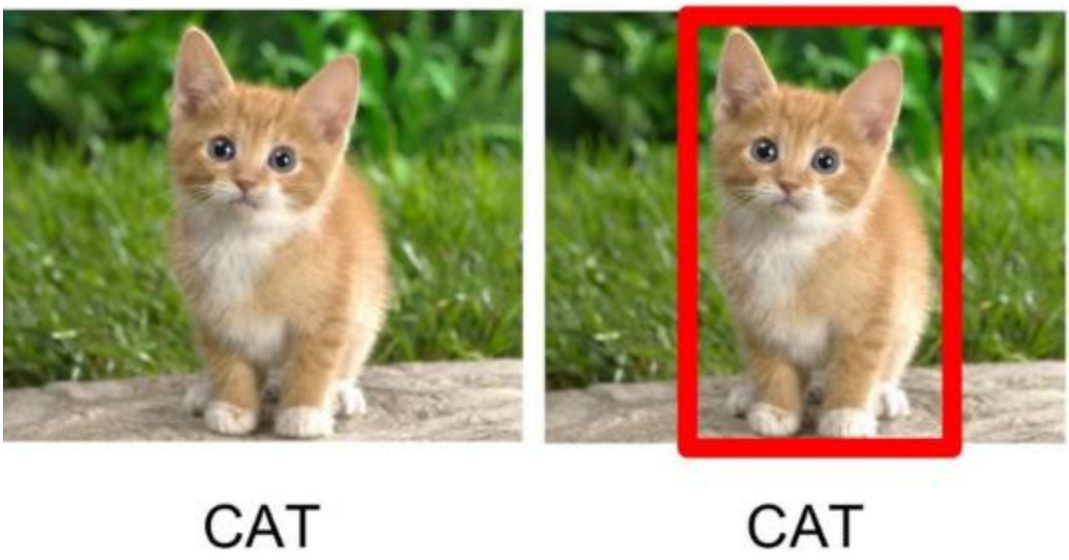

Figure 1.1 Basic idea about the application

1.1 Project overview:

- ✓ Object Identification is a computer vision technique for identifying objects in images or videos.
- ✓ When humans look at a photograph or watch a video, we can readily spot people, objects, scenes, and visual details.
- ✓ The goal is to teach a computer to do what comes naturally to humans: to gain a level of understanding of what an image contains.
- ✓ The project uses fundamentals of Android Studio and google library TensorFlow.
- ✓ 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.

1.2 Scope:

- ✓ The Object Identification has been done through the use of ordinary camera.
- ✓ The concept is well extendable in applications like Intelligent Robots, Automatic Guided Vehicles, Enhancement of Security Systems to detect the suspicious behavior along with detection of weapons, identify the suspicious movements of enemies on boarders with the help of night vision cameras and many such applications.

1.3 Objective:

- ✓ The goal of object Identification is to find the location of an object in a given picture accurately and mark the object with the appropriate category. To be precise, the problem that object identification seeks to solve involves determining where the object is, and what it is.
- ✓ However, solving this problem is not easy. Unlike the human eye, a computer processes images in two dimensions. Furthermore, the size of the object, its orientation in the space, its attitude, and its location in the image can all vary greatly.
- ✓ Technically, the above task involves image recognition and positioning.

CHAPTER-2: SYSTEM ANALYSIS

2.1 Tools &Technology

2.2.1 Tools used:

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

2.2.2 Minimum Requirement:

Software Requirement:

Update version of Android Studio. Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

Hardware Requirement:

- ✓ API level 21 or above i.e., android version 5.0 or above.
- ✓ RAM 1GB or more.

CHAPTER-3: SYSTEM DESIGN

3.1 Flow of System

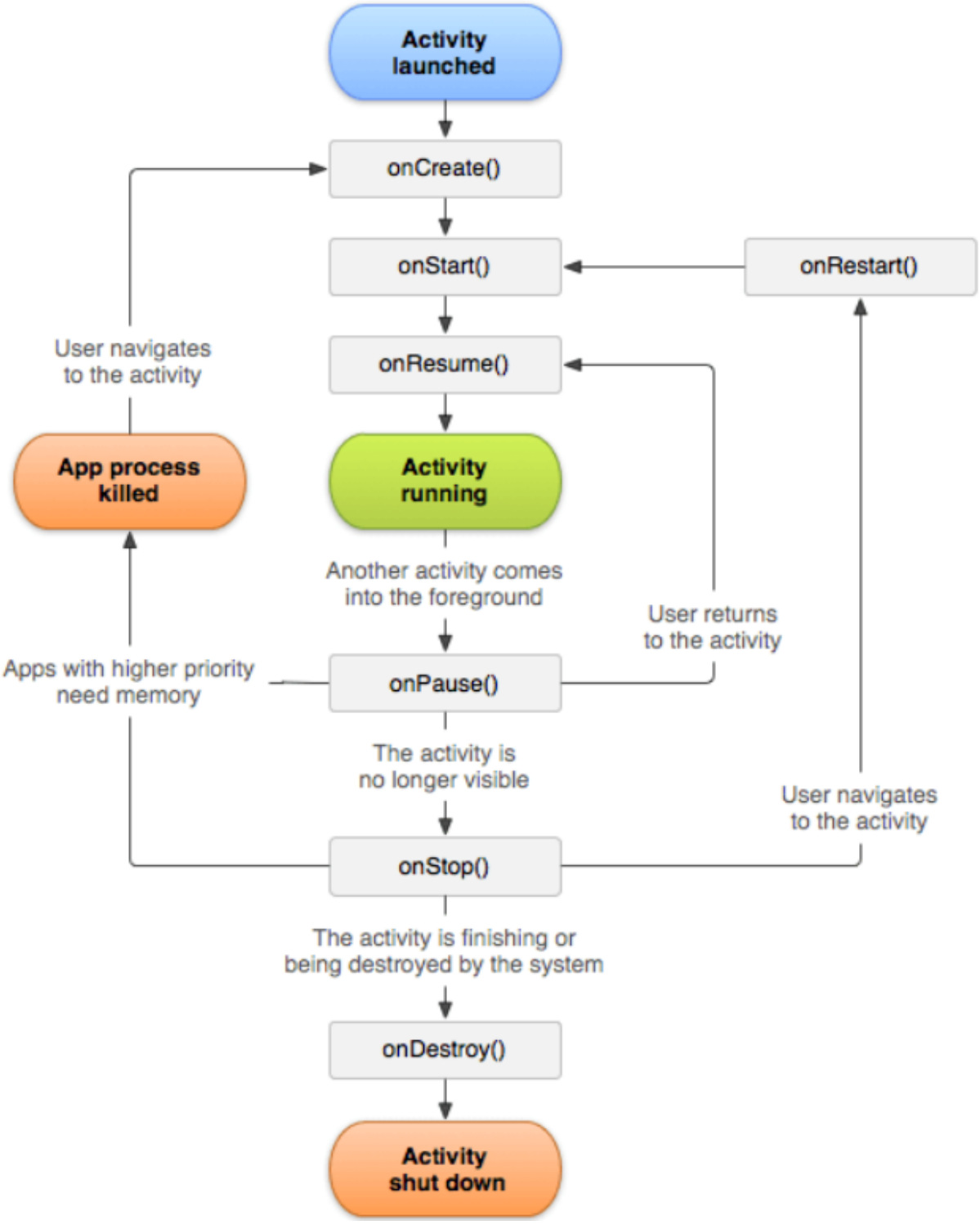

Figure 3.1 Android activity lifecycle

Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class. An activity is the single screen in android. It is like window or frame of Java. By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.
onCreate- called when activity is first created.
onStart- called when activity is becoming visible to the user.
onResume- called when activity will start interacting with the user.
onPause- called when activity is not visible to the user.
onStop- called when activity is no longer visible to the user.
onRestart- called after your activity is stopped, prior to start.
onDestroy- called before the activity is destroyed.

3.2 Major functionality

Major functionality of this application is to detect object. We humans have capability to identify the object just looking at it, this application helps machine to identify the object. It is basically an Object Identifier application using AI and tensor flow.

TensorFlow Lite is the official solution for running machine learning models on mobile and embedded devices. It enables on-device machine learning inference with low latency and a small binary size on Android, iOS, and other operating systems. TensorFlow Lite helps us introduce machine learning and AI into our app in an easy and streamlined way.

Benefits of TensorFlow:

On-device ML inference is difficult because of the many constraints—TensorFlow Lite can solve these:

- ✓ Performance: TF Lite is fast with no noticeable accuracy loss—see the metrics.
- ✓ Portability: Android, iOS, and more specialized IoT devices.
- ✓ Low latency: Optimized float- and fixed-point CPU kernels, op-fusing, and more.
- ✓ Acceleration: Integration with GPU and internal/external accelerators.
- ✓ Small model size: Controlled dependencies, quantization, and op registration.
- ✓ Tooling: Conversion, compression, benchmarking, power-consumption, and more.

CHAPTER-4: IMPLEMENTATION

4.1 Implementation Environment

The basic implementation of the application is done in android background. This application is combination of Artificial Intelligence and machine learning. This application is basically implemented on Android Studio (Updated version). Now this application is basically single user application. The basics of application is implemented in Java and C++. We humans have capability to identify the object just looking at it, this application helps machine to identify the object using AI and tensor flow. TensorFlow Lite is the official solution for running machine learning models on mobile and embedded devices. It enables on-device machine learning inference with low latency and a small binary size on Android, iOS, and other operating systems. TensorFlow Lite helped us introduce machine learning and AI into our app in an easy and streamlined way.

4.2 Module Specification

Major modules needed for this implementation of application are Android Studio and TensorFlow which is an open-source software library for dataflow programming across a range of tasks. The Camera2 API provides an interface to individual camera devices connected to an Android device. It replaces the deprecated Camera class. Use getCameraIdList to get a list of all the available cameras. You can then use getCameraCharacteristics and find the best camera that suits your need (front/rear facing, resolution etc). Create an instance of CameraDevice.StateCallback and open a camera. It is ready to start camera preview when the camera is opened. This sample uses TextureView to show the camera preview. Create a CameraCaptureSession and set a repeating CaptureRequest to it.

4.3 Coding Standards

4.3.1 CameraActivity

```
package com.example.android.tflitecamerademo;
import android.app.Activity;
import android.os.Bundle;
public class CameraActivity extends Activity {
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_camera);
   if (null == savedInstanceState) {
    getFragmentManager()
       .beginTransaction()
```

```
        .replace(R.id.container, Camera2BasicFragment.newInstance())
        .commit();
    }
  }
}
```

4.3.2 Preview width by Camera2 API

```
private static final int MAX_PREVIEW_WIDTH = 1920;
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

4.3.3 onOpened (Method called when camera is open)

```
public void onOpened(@NonNull CameraDevice currentCameraDevice) {
        // This method is called when the camera is opened.  We start camera preview here.
        cameraOpenCloseLock.release();
        cameraDevice = currentCameraDevice;
        createCameraPreviewSession();
       }
```

4.3.4 onDisconnected

```
public void onDisconnected(@NonNull CameraDevice currentCameraDevice) {
        cameraOpenCloseLock.release();
        currentCameraDevice.close();
        cameraDevice = null;
       }
```

4.3.5 onError

```
public void onError(@NonNull CameraDevice currentCameraDevice, int error) {
        cameraOpenCloseLock.release();
        currentCameraDevice.close();
        cameraDevice = null;
        Activity activity = getActivity();
        if (null != activity) {
          activity.finish();
        }
       }
```

4.3.6 showToast

```
private void showToast(final String text) {
   final Activity activity = getActivity();
   if (activity != null) {
    activity.runOnUiThread(
        new Runnable() {
          @Override
          public void run() {
            textView.setText(text);
          }
        });
   }
  }
```

4.3.7 onActivityCreated

```
public void onActivityCreated(Bundle savedInstanceState) {
   super.onActivityCreated(savedInstanceState);
   try {
    classifier = new ImageClassifier(getActivity());
   } catch (IOException e) {
    Log.e(TAG, "Failed to initialize an image classifier.");
   }
   startBackgroundThread();
  }
```

4.3.8 onResume

```
public void onResume() {
   super.onResume();
   startBackgroundThread();
   // When the screen is turned off and turned back on, the SurfaceTexture is already
   // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can open
   // a camera and start preview from here (otherwise, we wait until the surface is ready in
   // the SurfaceTextureListener).
   if (textureView.isAvailable()) {
    openCamera(textureView.getWidth(), textureView.getHeight());
   } else {
    textureView.setSurfaceTextureListener(surfaceTextureListener);}}
```

4.3.9 onPause

```
public void onPause() {
    closeCamera();
    stopBackgroundThread();
    super.onPause();
  }
```

4.3.10 onDestroy

```
public void onDestroy() {
    classifier.close();
    super.onDestroy();
  }
```

4.4 Snapshot

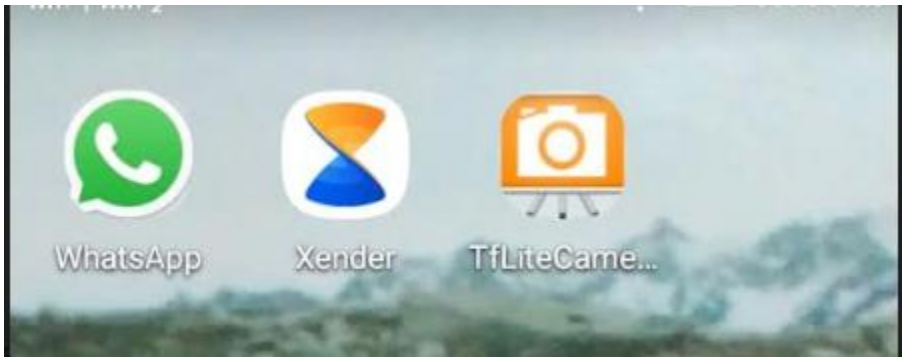

Figure 4.4 (a) Application logo

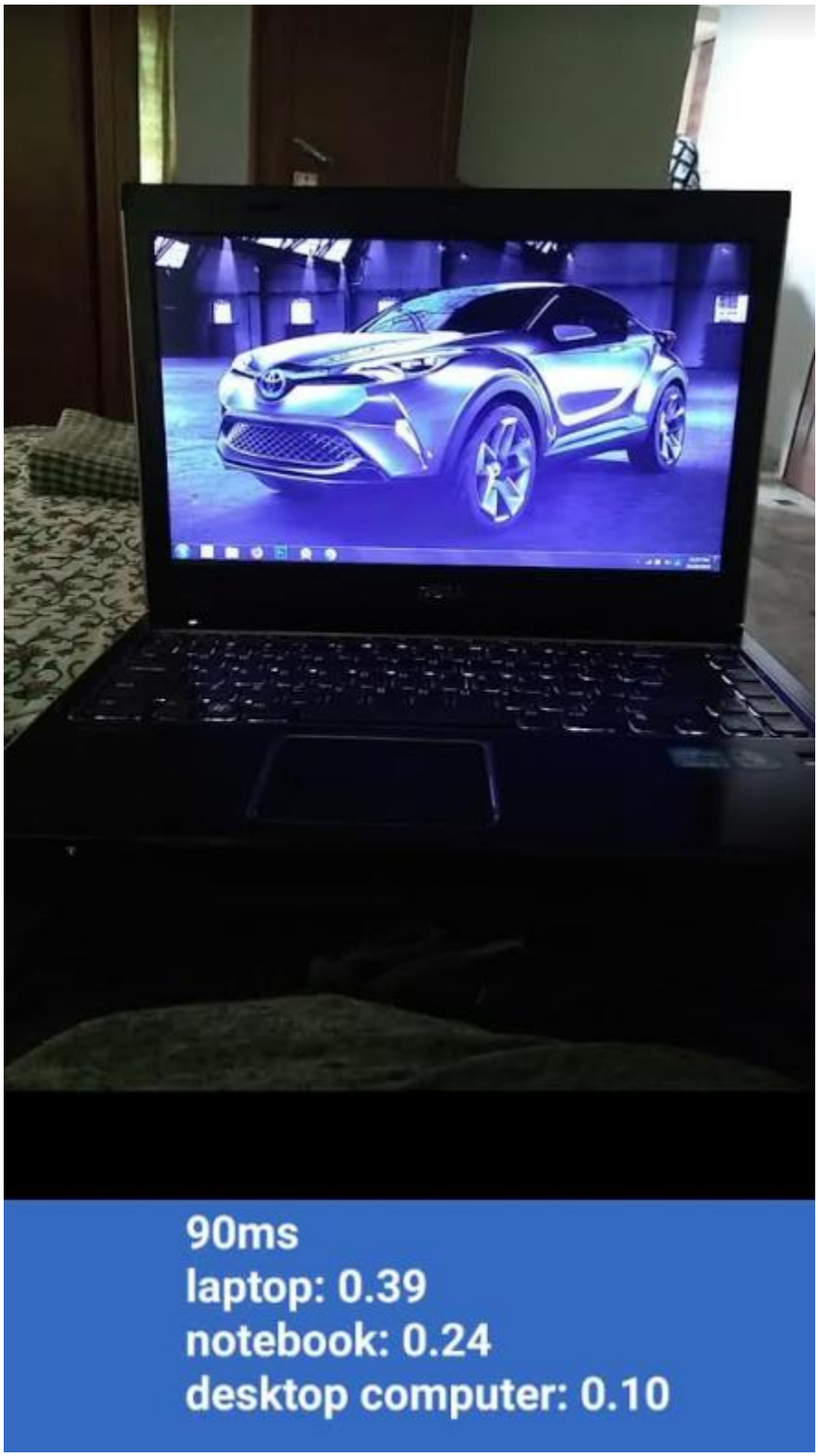

Figure 4.4 (b) Recognizing the object

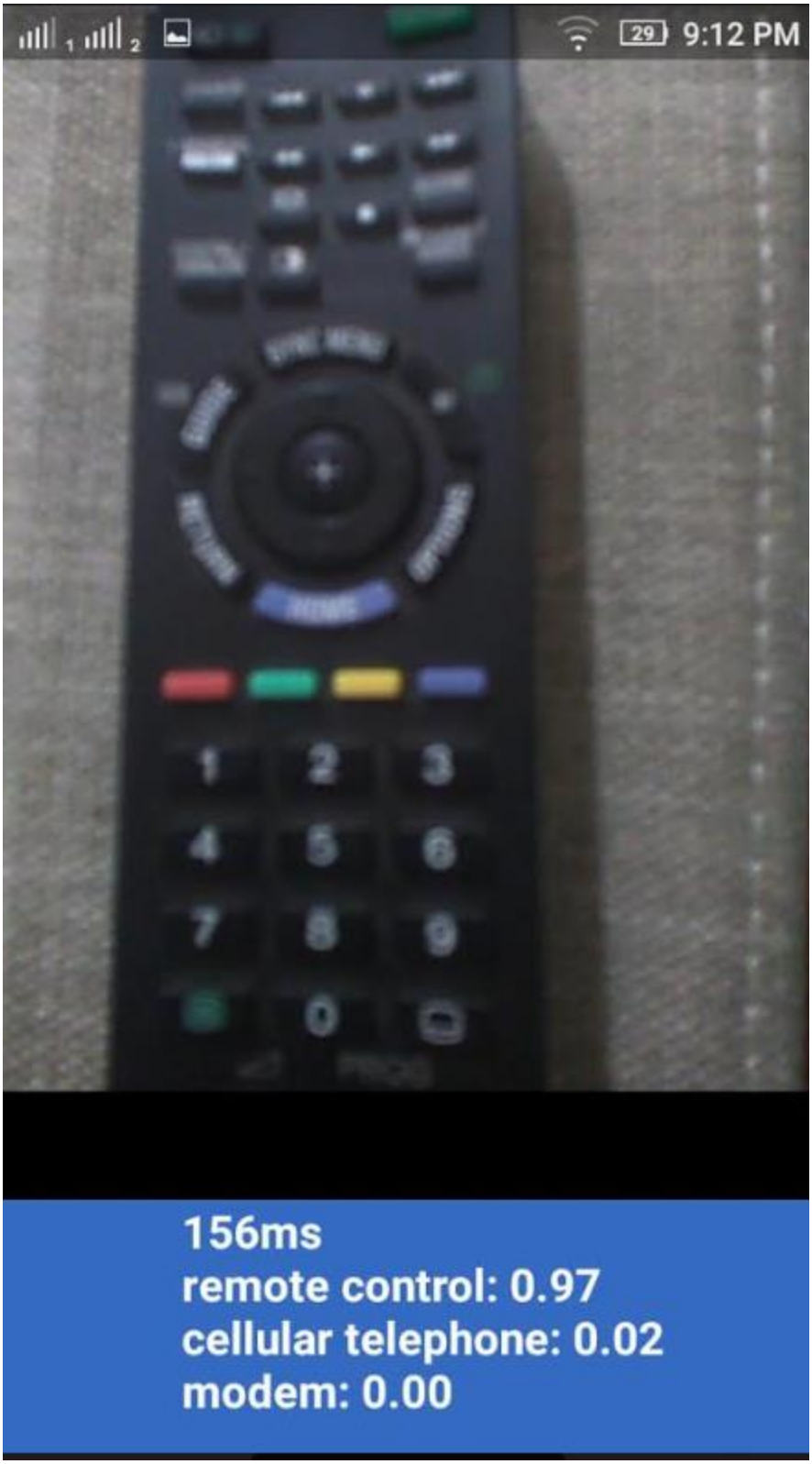

Figure 4.4 (c) Precisely identifying the object

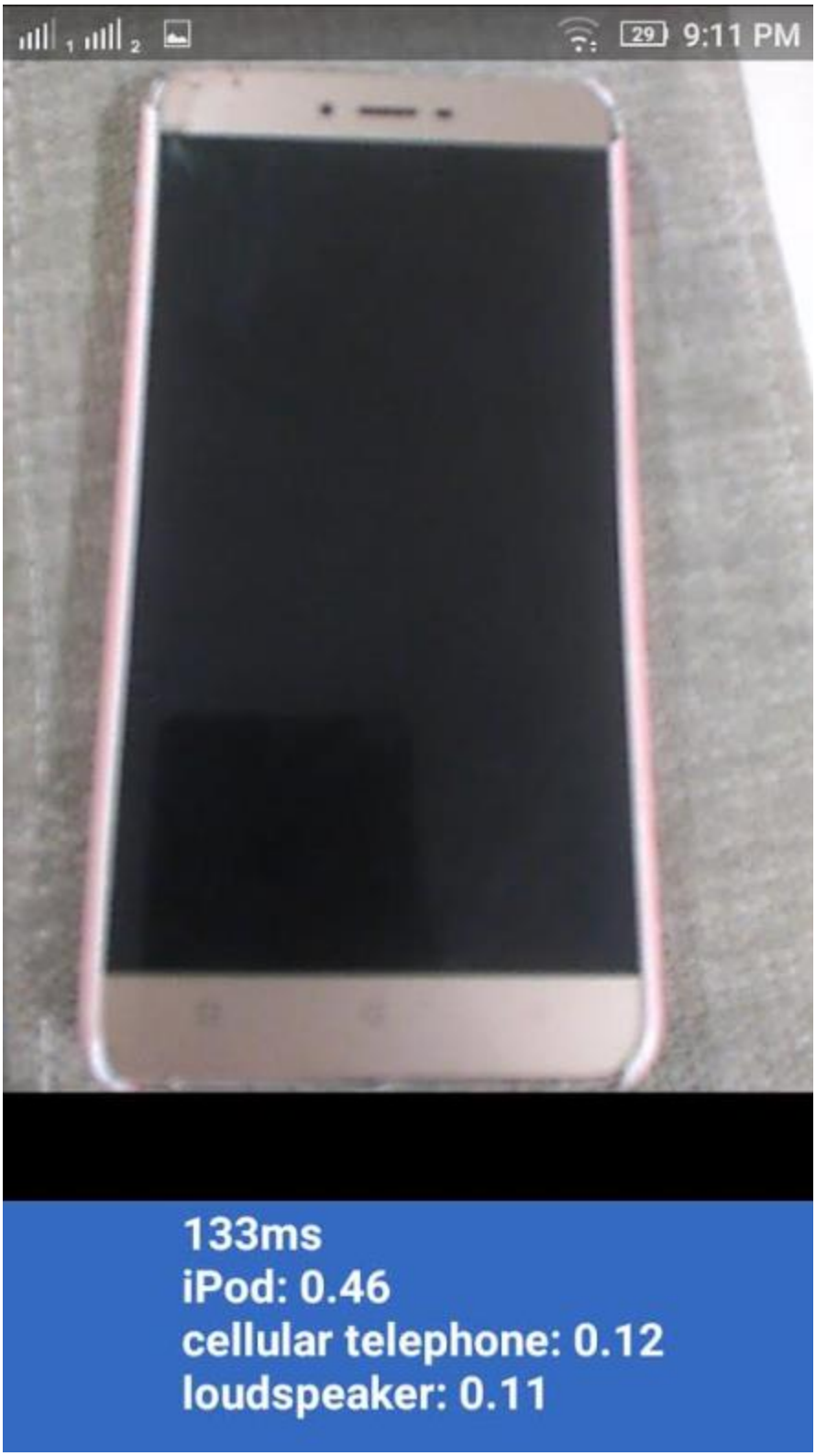

Figure 4.4 (d) Recognizing the Mobile Phone

CHAPTER-5: CONSTRAINT & FUTURE ENHANCEMENT

In this project we basically made application wherein which there is a camera which learns about the object nearby or at some distance. It is machine learning concept. We humans can easily understand what that object is just by looking at it once, but in this application, machine is made to learn about the object nearby, not so precisely but with some accuracy.

On further basis we can modify this application with more accuracy and precision, like if there is flower then we can make the camera identify the type flower. We can also measure the size of the object via some modification in the algorithm.

Face identification the most trending technology at present can also be included in this application, recognizing a person is one of the most useful thing in the case of driverless cars.

Features either the local or global used for identification can be increased, to increase the efficiency of the object identification system. Geometric properties of the image can be included in the feature vector for recognition. It can also be used in the way of video surveillance.

CHAPTER-6: CONCLUSION

Here we basically made our camera learn or we can say that we made it learn to identify the object. This application is based on Machine learning. Major functionality of this application is to detect object. We humans have capability to identify the object just looking at it, this application helps machine to identify the object. It is basically an Object Identifier application using AI and tensor flow. This application is developed in Java. It meets the full requirement of the system which it has to be developed. This project was completed within the time period given. No major problems or risks were involved during the development of the project. All bugs and errors have been removed successfully.

REFERENCES

https://www.tensorflow.org/lite/

https://www.tensorflow.org/lite/overview

https://www.datacamp.com/community/tutorials/tensorflow-tutorial?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=1t2&utm_creative=278443377089&utm_targetid=dsa-498578051924&utm_loc_interest_ms=&utm_loc_physical_ms=1007761&gclid=EAIaIQobChMIqtDik76c3gIVlzUrCh1ePwC8EAAYAiAAEgJWXvD_BwE

https://www.youtube.com/watch?v=tjsHSIG8I08&feature=youtu.be

https://www.tensorflow.org/tutorials/

https://www.truiton.com/2016/06/android-image-recognition-google-cloud-vision-api/

https://www.tensorflow.org/tutorials/images/image_recognition