A Project Report

on

TEXT & FACIAL FEATURES RECOGNITION USING ML KIT

Submitted in partial fulfillment of the
requirement for the award of the degree of

B.Tech Computer Science Engineering

Under The Supervision of
Dr. Sanjeev Kumar Prasad
Associate Professor
Department of Computer Science and Engineering

Submitted By

Anju Kumari (18SCSE1010736)
Krishna Gopal (18SCSE1010489)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
December- 2021

Table of Contents

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"Text & facial features recognition using ML kit"** in partial fulfillment of the requirements for the award of the B. Tech(CSE) -submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of July,2021 to December, 2021, under the supervision of **Dr. Sanjeev Kumar Prasad**, **Professor,** Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering, Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Anju Kumari(18SCSE1010736)
Krishna Gopal(18SCSE1010489)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Sanjeev Kumar Prasad
Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Anju Kumari (18SCSE1010736) and Krishna Gopal(18SCSE1010489) has been held on and her/his work is recommended for the award of B. Tech(CSE)

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:
Place: Greater Noida

ACKNOWLEDGEMENT

In the accomplishment of completion of my project on Text & facial features recognition using ML kit. I would like to convey my special gratitude to Dr. Sanjeev Kumar Prasad of the Computer Science Department. Your valuable guidance and suggestions helped me in various phases of the completion of this project. I will always be thankful to you in this regard. I am ensuring that this project was finished by me and not copied.

Anju Kumari(18SCSE1010736)
Krishna Gopal(18SCSE1010489)

Abstract

We all face a problem when there is a need to copy text written over an image for example someone given you his contact card and you want to mail him or make a call then you will have to see the mail id and no. to do so which will take more time but with the help of text recognition feature you just need to take a snap of the card and you will get the mail id or phone no. very easily within some seconds. And we are going to solve this problem by making an android app using ML Kit.

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. It makes it easy to apply ML techniques in your apps by bringing Google's ML technologies, such as the Mobile
Vision, and TensorFlow Lite, together in a single SDK. This makes it fast and unlocks real-time use cases like the processing of camera input. It also works while offline and can be used for processing images and text that need to remain on the device. ML Kit is now Generally Available (GA), with the exception of Pose Detection, Entity Extraction, Text Recognition, and Selfie Segmentation.

Nowadays many apps are using Machine Learning inside their apps to make most of the tasks easier. We are going to build an app that will detect and display the text written in images and the facial features/condition of a person's image.

The app will :

1) Use the ML Kit Text Recognition API to detect text in images - Recognize text across Latin-based languages, Analyze the structure of the text, Identify the language of text, Small application footprint, Real-time recognition.
2) Use the ML Kit Face Contour API to identify facial features in images - Recognize and locate facial features, Get the contours of facial features, Recognize facial expressions, Track faces across video frames, Process video frames in real-time.

	TABLE OF CONTENTS	
CHAPTER	**TITLE**	**PAGE**
1	**INTRODUCTION**	**1-2**
	1.1 OVERVIEW	1
	1.2 LITERATURE SURVEY	1-2
	1.3 OBJECTIVES	2
2	**SYSTEM ANALYSIS**	**3-5**
	2.1 EXISTING SYSTEM	3
	2.2 PROBLEM DEFINITION	3
	2.3 PROPOSED SYSTEM	3-4
	2.3.1 ADVANTAGES	4-5
3	**SOFTWARE REQUIREMENTS**	**6-9**
	3.1 INTRODUCTION	6
	3.1.1 PURPOSE	6
	3.1.2 SCOPE	6-7
	3.2 GENERAL DESCRIPTION	7-8
	3.2.1 PROCESS OVERVIEW	7
	3.2.2 ASSUMPTIONS AND DEPENDENCIES	7-8
	3.3 PLATFORM DESCRIPTION	8
	3.4 SOFTWARE DESCRIPTION	8
	3.5 OPERATIONAL REQUIREMENTS	8
	3.5.1 FUNCTIONAL REQUIREMENTS	8
	3.5.2 DATA REQUIREMENTS	8
	3.5.3 PERFORMANCE REQUIREMENTS	8
	3.6 HARDWARE REQUIREMENTS	9

List of Figures

Acronyms

ML Kit	Machine Learning Kit
SDK	Software Development Kit
API	Application programming interface
NoSQL	not only SQL
XML	Xtensible Markup Language

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

Machine Learning for a beginner can be very tricky to understand the concepts and then implement them at the same time and it's also very time-consuming so Google has come with an interesting, exciting, and ready-to-go solution for Machine Learning which is ML Kit.

It is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use library. Whether you are new or experienced in machine learning, you can implement the operations you need in a clear way. No deep knowledge of neural networks or model optimization to get started. Whereas if you are an experienced ML and AI developer, ML Kit provides convenient APIs that help you use your custom TensorFlow Lite models in your mobile applications.

1.1 OVERVIEW

Nowadays many apps use Machine Learning inside their apps to make most of the tasks easier. We have seen many apps that detect text from any image. This image may include number plates, images, and many more. We are also making face detection which will display the emotions of faces by taking snaps or uploading photos. We are going to make **Text and facial features detectors in Android using Firebase ML Kit**.

1.2 LITERATURE SURVEY

Guido Albertengo **[1]** says that the Firebase Cloud messaging service, a service offered by Google Firebase. The Author says that Firebase is a backend-as-a-service (BaaS) that grew up into a next-generation app-development platform on Google Cloud Platform. It also focuses on Firebase Cloud Messaging as a service, so it explains how Google Firebase requires fewer resources in terms of time, development cost, and infrastructure. Google Firebase provides us with a Software development kit (SDK) and Application Program Interface (API) so that the developer again doesn’t need to define low-level programming logic. In this paper,
The author also tries to say that all Firebase services are cloud-based, mobile applications, so data exchange takes place directly without the use of dedicated servers.

Chunnu Khawas and Pritam Shah **[2]** explain about services offered by Google Firebase. It tells us how Firebase uses JSON (JavaScript Object Notation) format for storing data in the database. The Writer explains how cloud-based infrastructure is provided for testing apps. This includes the ML Kit Beta version used for developing Language Identification Android Application. The Author also summarizes some of the topics including adding Firebase to our Application adding required dependencies and giving appropriate permissions. This paper also tries to tell how Google Firebase helps making Android Applications faster and efficient as no PHP is required.

https://firebase.google.com/docs/mlkit/android/translate-text/ **[3]**, these docs tell about how to add firebase to the android project. This likewise incorporates the Available libraries for android. It gives a concise introduction about Add Firebase using the Firebase Assistant and Adds Firebase utilizing the Firebase console, In this doc, we likewise give the implementation of Translate text API by adding required techniques like interpreting strategy and required dependencies for the ML Kit Android libraries to your module.

1.3 OBJECTIVES

The main objective is to simplify the implementation of machine learning models in mobile applications. Image text extraction can recognize Latin-based languages which help extract the incomprehensible text out of images. This text can later be used for translation, serving various other purposes. And facial features like happy, sad, angry, etc detection.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

There are various applications with the features of text extraction and facial features detection like Google lens to extract text and mask detector which uses the facial detection features by using Machine Learning. Earlier there was the need to write code for using features of Machine Learning now it has become very easy not to need to learn ML just need to use the ML kit available in Firebase.

2.2. PROBLEM DEFINITION

As we have already discussed that in text recognition, we are given a segmented image and our task is to recognize what text is present in that segment. Thus, for the text recognition problem, the input is an image while the output is a text. So, in order to solve the text recognition problem, we need to develop a model that takes the image as an input and outputs the recognized text.
A complete face recognition system includes face detection, face preprocessing, and face recognition processes. Therefore, it is necessary to extract the face region from the face detection process and separate the face from the background pattern, which provides the basis for the subsequent extraction of the face difference features. The recent rise of the face based on the depth of learning detection methods, compared to the traditional method not only shortens the time but the accuracy is effectively improved. Face recognition of the separated faces is a process of feature extraction and contrast identification of the normalized face images in order to obtain the identity of human faces in the images.

2.3 PROPOSED SYSTEM

[1] TEXT RECOGNITION:-Text recognition can automate tedious data entry for credit cards, receipts, and business cards of various organizations. With the Google Cloud-based API, you can extract text from pictures of documents of various formats, which you can use to increase accessibility or translate documents.

[2] FACE DETECTION:- The Machine Learning Kit's face detection API, you can detect faces in an image, identify key facial features of a human in a more interactive way and get the contours of detected faces.

[3] SMART REPLY:- With the Machine Learning Kit's Smart Reply API, you can automatically generate relevant replies to messages. Smart Reply helps the broad range of users respond to messages quickly, and makes it easier to reply to messages on devices with limited input capabilities.

Some of the features of ML Kit are shown in Figure 1

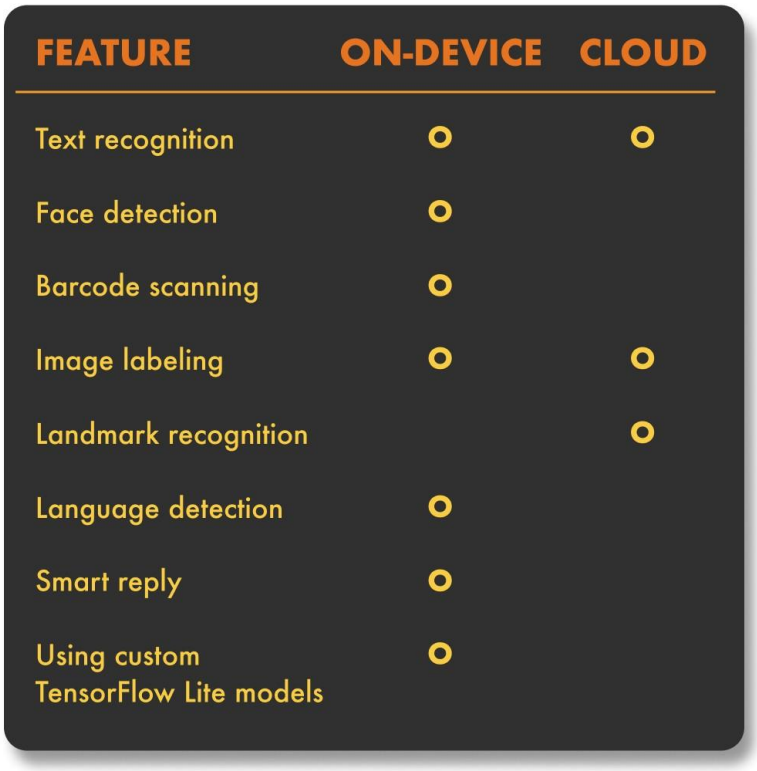


Fig-1:ML Kit Features

2.3.1. ADVANTAGES

I. *Easy-to-use framework for developers*

Typically, adding ML to a mobile app is a challenging and time-consuming process. By using this new SDK, this process is dramatically simplified. All you need is to pass data to the API, and wait until the SDK will send a response. Google's team stated, that

implementing their APIs. Therefore, you do not need deep knowledge of neural networks.

II. *Custom models*

This option is helpful for experienced developers. If base ML Kit APIs do not cover all your needs, you can have your own ML model. New SDK can work with TensorFlow and the mobile app machine learning library. Besides, it supports developers with the possibility to download their own model to the Firebase console and bundle it with their products. Another significant key factor is that models are updated dynamically. It suggests that the model will be updated even without updating the whole app.

III. Cloud and on-device APIs

In fact, developers can choose between cloud-based and on-device APIs. To make a right choice it is significant to take into consideration some differences between these two variants. Cloud APIs process data on the Google Cloud Platform; therefore, recognizes objects more accurately. However, cloud models are larger in comparison to on-device ones. Offline models need less free space, can work offline, and process data faster, but their accuracy is lower.

IV. Security Concern and privacy

As we know, most models are app-specific. Thus, probably, your customize model should be protected from other accesses from other developers. Now, you can do this task safely by using ML Kit. Furthermore, Google claims that cloud APIs do not store user's data, and it is removed when processing is done.

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 INTRODUCTION

The following describes the requirement specification for the installation and execution of the Text & Facial features detection app.

3.1.1. PURPOSE

Nowadays many apps use Machine Learning inside their apps to make most of the tasks easier. We have seen many apps that detect text from any image. This image may include number plates, images, and many more. We are also making face detection which will display the emotions of faces by taking snaps or uploading photos. We are going to make **Text and facial features detectors in Android using Firebase ML Kit**.

3.1.2. SCOPE

- **Recognize text across Latin-based languages** Supports recognizing text using Latin script.
- **Analyze structure of text** Supports detection of words/elements, lines and paragraphs.
- **Identify language of text** Identifies the language of the recognized text.
- **Small application footprint** On Android, the API is offered as an unbundled library through Google Play Services.
- **Real-time recognition** Can recognize text in real-time on a wide range of devices.
- **Recognize and locate facial features** Get the coordinates of the eyes, ears, cheeks, nose, and mouth of every face detected.

- **Get the contours of facial features** Get the contours of detected faces and their eyes, eyebrows, lips, and nose.
- **Recognize facial expressions** Determine whether a person is smiling or has their eyes closed.
- **Track faces across video frames** Get an identifier for each unique detected face. The identifier is consistent across invocations, so you can perform image manipulation on a particular person in a video stream.
- **Process video frames in real-time** Face detection is performed on the device and is fast enough to be used in real-time applications, such as video manipulation.

3.2 GENERAL DESCRIPTION

3.2.1 PROCESS OVERVIEW

This app reduces the human efforts and time in scanning the text to extract and save as well as detection of facial features or emotions.

- Installation of app
- Allow permissions

3.2.2 ASSUMPTIONS AND DEPENDENCIES

The following assumptions are made in the implementation:

- classpath 'com.google.gms:google-services:4.3.2'
- implementation'com.google.firebase:firebase-ml-vision:18.0.1'
- implementation'com.google.firebase:firebase-analytics:17.2.2'
- apply plugin: 'com.google.gms.google-services’
- implementation 'com.google.firebase:firebase-ml-vision-face-model:17.0.2'
- implementation'com.google.firebase:firebase-core:15.0.2'
- implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
- implementation 'androidx.core:core-ktx:1.6.0'

- implementation 'androidx.appcompat:appcompat:1.4.0'
- implementation 'com.google.android.material:material:1.4.0'
- implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
- testImplementation 'junit:junit:4.+'
- androidTestImplementation 'androidx.test.ext:junit:1.1.3'
- androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

3.3 PLATFORM DESCRIPTION

The app is prepared in Android studio and this project work is carried out in Ubuntu for demonstration purposes. and can be installed on Android phones version 10+

3.4 SOFTWARE DESCRIPTION

In order to simulate the project, Android Studio is used and it is connected to the Firebase to use the ML Kit.

3.5 OPERATIONAL REQUIREMENTS

3.5.1 FUNCTIONAL REQUIREMENTS

The functional requirements for the project are

1. Permission to access the camera.
2. Permission to internal storage.
3. Permissions to save the output.

3.5.2 DATA REQUIREMENTS

Snap through any camera app or upload images/photos.

3.5.3 PERFORMANCE REQUIREMENTS

1. Fast internet connection
2. Local storage
3. Controlling devices
4. Low power consumption

3.6 HARDWARE REQUIREMENTS

3.6.1 REQUIREMENTS FOR MAKING APP

1. Processor : Intel Pentium 4
2. Hard Disk Space : 40 GB
3. Monitor : 13.5”
4. Keyboard : 102 KEYS
5. Internal Memory Capacity : 256 MB
6. Mouse : Optical Mouse

3.6.2 REQUIREMENTS FOR INSTALLATION

1. Android : 10+ version
2. RAM : 4+ GB

3.7 SOFTWARE REQUIREMENTS

1. Operating System : Any Operating System
2. Coding Language : Kotlin, XML
3. Tool Used : Android Studio, Firebase

3.8 FEASIBILITY STUDY

This section provides an overview of the economic, operational, and technical feasibility of the project for real-time implementation.

3.8.1 ECONOMIC FEASIBILITY

The project uses Android Studio, Firebase, Kotlin, and XML which is Open Source. All the android libraries used in this project are also open source. There is some expenditure of money involved for using ML Kit features of Firebase after completion of the free plan.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

Flow diagram for Face detection

Facial emotion recognition is a complex task and the machine learning approach to recognizing faces requires several steps to perform it.

Feature selection: This stage refers to attribute selection for the training of the machine learning algorithm. The process includes the selection of predictors for the construction of the learning system. It helps in improving prediction rate, efficiency, and cost-effectiveness. Many tools such as Weka and sci-kit-learn have inbuilt tools for automated feature selection.

Feature classification: When it comes to supervised learning algorithms, classification consists of two stages. Training and classification, where training helps in discovering which features are helpful in classification. Classification is where one comes up with new examples and, hence, assigns them to the classes that are already made through training the features.

Feature extraction: Machine learning requires numerical data for learning and training. During feature extraction, the processing is done to transform arbitrary data, text or images, to gather the numerical data. Algorithms used in this step include principal component analysis, local binary patterns, linear discriminant analysis, independent component analysis, etc.

Classifiers: This is the final step in this process. Based on the inference from the features, the algorithm performs data classification. It comprises classifying the emotions into a set of predefined emotion categories or mapping to a continuous space where each point corresponds to an expressive trait. It uses various algorithms such as Support Vector Machine (SVM), Neural Networks, and Random Forest Search.

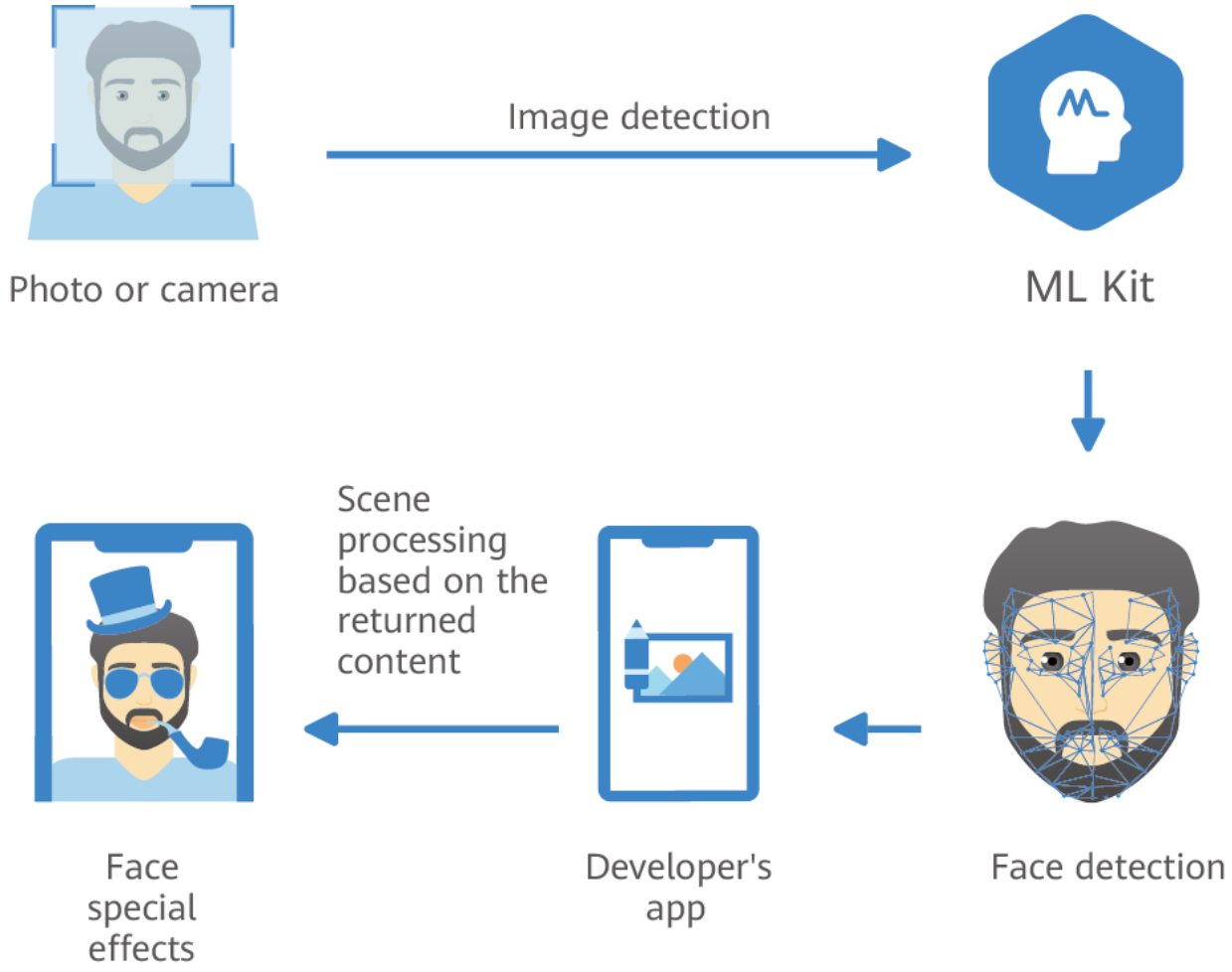


Fig-2: Facial feature detection process

Flow diagram for text detection

- Text is read from the camera stream or static image.
- Image is sent to the machine learning model.
- The model analyzes this image via ML Kit and sends a String response.

Below is the list of languages supported by ML Kit:

On-device:

Simplified Chinese, Japanese, Korean, Latin-based languages

Almost all languages can be supported on the cloud.

Text Recognition Prerequisites : Add the dependencies for the ML Kit Android libraries to your module (app level) Gradle file (usually app/build.gradle) implementation **'com.google.firebase:firebase-ml-vision:24.0.1'**

a. In Android Manifest File: If you use the on-device API, configure your app to automatically download the ML model to the device after your application is installed from the Play Store.

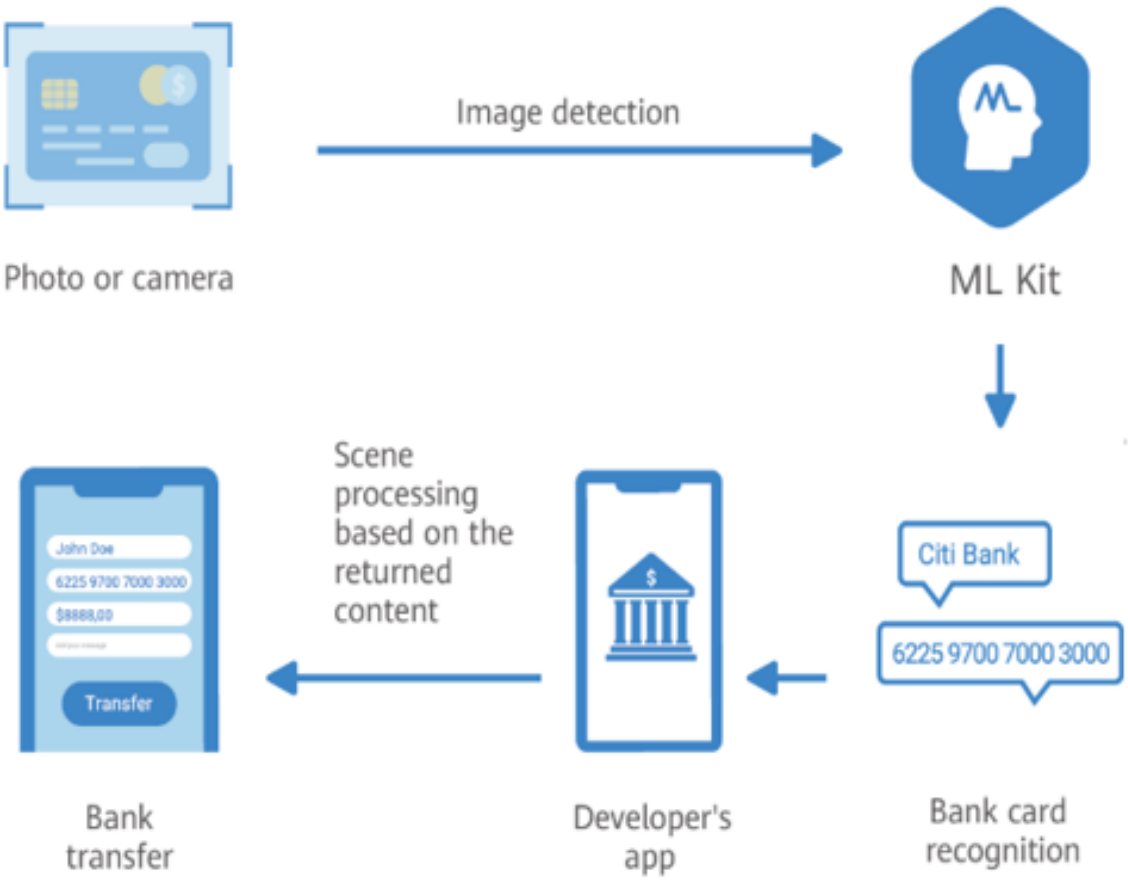


Fig-3:Text detection process

4.2 CONTEXT DIAGRAM

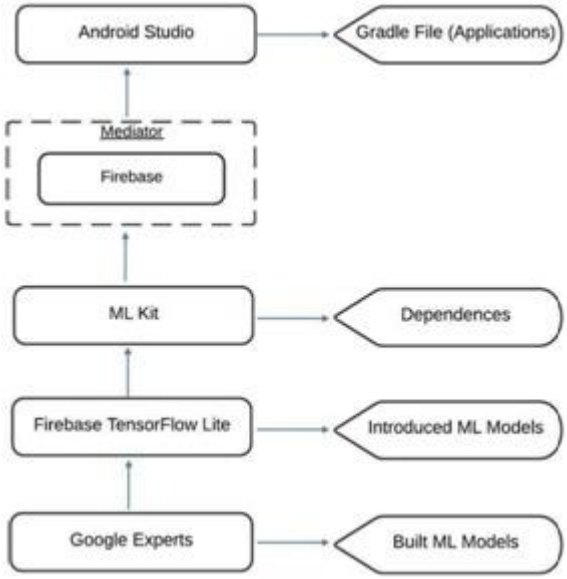


Fig. 4: Firebase ML Model Architecture

In this architecture, the Google Experts built ML models. These ML models are introduced in Firebase TensorFlow Lite then these models are converted into TensorFlow Lite models. For each model ML Kit provides dependencies, now ML Kit is integrated into Android Studio through using Firebase which acts as a mediator. By this, Gradle File (Applications) add the dependencies for the Firebase products that you want to use in your app which is available in the Android Studio.

CHAPTER 5

FUNCTION MODULE

5.1 METHODOLOGY FOR TEXT EXTRACTION

ML Kit's on-gadget Image Text Extraction (Recognize text) Dependency is utilized to separate the client input text which can later be utilized for language ID and interpretation. The Cloud Recognize text API in ML Kit as of now can separate client input text out of records, pictures, and pictures. This API likewise has the capacity to perceive numerous other Latin texts like Romanized, Hindi, Japanese, and Chinese, and so on.
Note: On Device app-only extract English which doesn't require internet.

Before we begin:
1] If we haven't already, we need to add Firebase to your Android Project.
Integrating Firebase to our Android Project:

Prerequisites:

- Android Studio is to be installed or if present, updated to the latest version.
- The target API level should be 16 (Jelly Bean) or later.
- 4.1 or later Gradle should be used.
- Set up a device or emulator (for 16 G.B RAM PC) for running the app or else deploy via USB port.
- Google account should be used to Sign in to the firebase.Our Android App can be connected to the Firebase using one of the two options:

•**Option 1:** Firebase console-setup workflow can be
used
(recommended).
•**Option 2:** Android Studio Firebase Assistant can also be used, but it requires additional configuration.

Option 1: Adding Firebase using Firebase Console:
Adding Firebase through the console requires two tasks to be performed i.e. we need to download Firebase Config Files from the console, and then add them into Gradle in the form of dependencies Registering app to a Firebase project:

Step 1: Register your app with firebase.

- Open Google Firebase & now go to the Firebase Console.
- Click on add project.
- Now enter a name for the PROJECT name and then click on continue.

- After creating a project in the Firebase, click on the android icon available on the screen.

Now, give the package name of the android application (i.e. com.example.appname) and then enter SHA-1 (recommended) which is available in the gradle file at the top-right in the Android Studio code panel. Go to app Android signingReports, DoubleClick on it. Then select SHA-1 from the Run Console. Paste it in Firebase. Now add the dependencies in Android Studio as per the firebase instructions. Also, Download the google.json file and paste it in the project-level app and complete the process as per firebase instructions. For more info visit firebase docs in the reference module.

1. We can enable Firebase Products in our app by adding the google-services Plugin to our Project level Gradle file(build.gradle). We also need to check whether we have got Google's Maven repository (google()) in Project Level Gradle file which should be inside the repositories and all projects repositories. add Google Services plugin **(classpath'com.google.gms:google-services:4.3.2')** inside the dependencies

```
repositories{
google() //Maven
jcenter()
}
dependencies {
classpath 'com.google.gms:google-services:4.3.2' //
Google Services plugin
}
allprojects {
repositories {
google()
jcenter()
}
}
```

Now go to app level build.gradle and add apply
plugin: 'com.android.application' at bottom of the file.

Step 2: Add Firebase SDKs to your app.

We need to add the dependencies for the Firebase product that we want to integrate in our app. It is to be added in our module (app-level) Gradle file (usually app/build.gradle) as per instructions in the Firebase Console.
Note: Some Firebase SDKs for Android offer an alternative Kotlin extensions library.

1. Now we need to sync our app so that all the dependencies have the necessary versions. Logs in the device will now display the Firebase

Verification that Initialization is complete. When we run our app on an emulator with network access, Firebase Console will notify us that we have successfully connected our app to Firebase.

Option 2: Adding Firebase with the Android studio Firebase Assistant:
Necessary Firebase Files are added into the Gradle and our app is registered with a Firebase Project all from within Android Studio using Firebase Assistant.

Caution: All the Firebase dependencies and settings should be configured and built correctly.

1. Open your Android project in AndroidStudio.

2. Click on Tools and then Firebase to open the FirebaseAssistant.

3. Select Connect to Firebase to register our app with an existing or a new Firebase project and automatically add the required files and code to your android project.

4. All the plugin versions should be up-to-date. These include:
- All the Project-level Gradle files and Google Services plug-inversion.
- All the App-level Gradle files and Firebase Android LibraryVersions.

5. Now, we need to sync our app in order to make sure that all dependencies are synced to the latest- version.

6. Lastly, we should run our app to check the connection to the firebase. Google's Maven repository should be included in both build scripts and all projects sections in build.gradle file at the project level. We now need to add ML Kit Android library dependencies to app-level GradleFile:
The dependencies are:
•implementation'com.google.firebase:firebase-ml-vision:18.0.1'
•implementation'com.google.firebase:firebase-analytics:17.2.2'
And the plug-ins are:
apply plugin: 'com.google.gms.google-services'

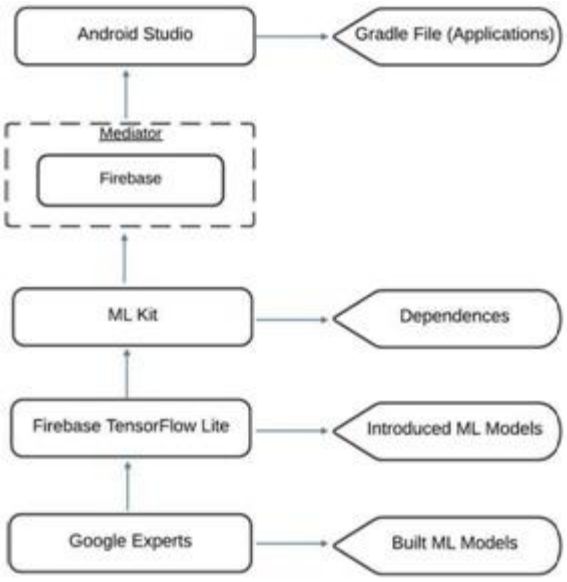


Fig. 5: Firebase ML Model Architecture

In this architecture, the Google Experts built ML models these ML models are introduced in Firebase TensorFlow Lite then these models are converted into TensorFlow Lite models. For each model ML Kit provides dependences, now ML Kit is integrated into Android Studio through using Firebase which acts as
a mediator. By this, Gradle File (Applications) add the dependencies for the Firebase products that you want to use in your app which is available in the Android Studio.

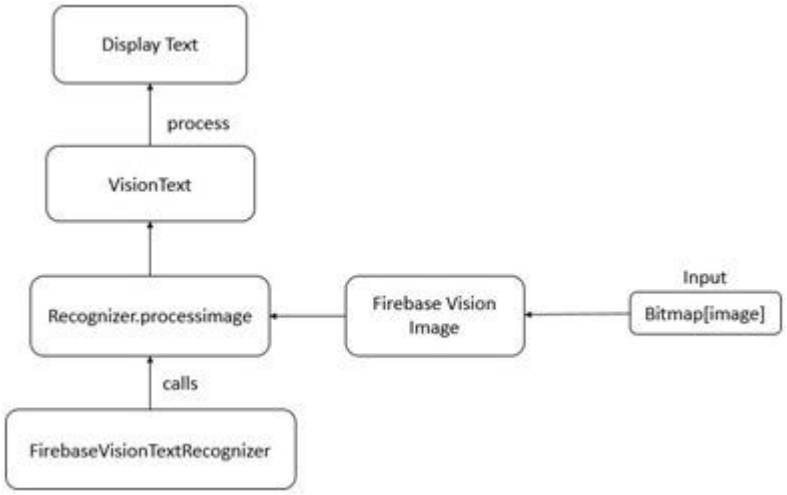


Fig. 6: Firebase Text Recognition Block Diagram

In figure 5, In the first place, we start with a bitmap of the chosen pictures, then, at that point, we need to introduce aFirebaseVisionImage object, passing the bitmap to the constructor. A FirebaseVisionImage addresses an image object that can be utilized for both on-gadget and cloud API locators. From that point, we need to process the image utilizing FirebaseVisionText Recognizer, which is an on-gadget or cloud text recognizer that recognizes text ina picture then

we need to consider Process Image on the FirebaseVisionImage on the FirebaseVisionImageas a boundary. By utilizing an OnSuccessListener to decide when identification is finished. In the event that fruitful, we can access a FirebaseVisionText object which is perceived text in a picture. At long last passing a FirebaseVisionTextfrom an OnSuccess capacity to a capacity to deal with the text and shown on the screen. For code and other info visit Firebase docs and CodeLab.

5.2. METHODOLOGY FOR FACE DETECTION

Just follow the same steps above as that of text extraction. It is important to configure the Face detection API and add the right dependencies in our project once the project is configured.

Before we begin:
1] If we haven't already, we need to add Firebase to your Android Project.
Integrating Firebase to our Android Project:

Prerequisites:

- Android Studio is to be installed or if present, updated to the latest version.
- The targets API level should be 16 (Jelly Bean) or later.
- 4.1 or later Gradle should be used.
- Set up a device or emulator (for 16 G.B RAM PC) for running the app or else deploy via USB port.
- Google account should be used to Sign in to the firebase.Our Android App can be connected to the Firebase using one of the two options:

•**Option 1:** Firebase console-setup workflow can be
used
(recommended).
•**Option 2:** Android Studio Firebase Assistant can also be used, but it requires additional configuration.

Option 1: Adding Firebase using Firebase Console:
Adding Firebase through the console requires two tasks to be performed i.e. we need to download Firebase Config Files from the console, and then add them into Gradle in the form of dependencies Registering app to a Firebase project:

Step 1: Register your app with firebase.

- Open Google Firebase & now go to the Firebase Console.
- Click on add project.

- Now enter a name for the PROJECT name and then click on continue.
- After creating a project in the Firebase, click on the android icon available on the screen.

Now, give the package name of the android application (i.e. com.example.appname) and then enter SHA-1 (recommended) which is available in the gradle file at the top-right in the Android Studio code panel. Go to app Android signingReports, DoubleClick on it. Then select SHA-1 from Run Console. Paste it in Firebase. Now add the dependencies in Android Studio as per the firebase instructions. Also, Download the google.json file and paste it in project-level app and complete the process as per firebase instructions. For more info visit firebase docs in the reference module.

1. We can enable Firebase Products in our app by adding the google-services Plugin to our Project level Gradle file(build.gradle). We also need to check whether we have got Google's Maven repository (google()) in Project Level Gradle file which should be inside the repositories and all projects repositories. add Google Services plugin **(classpath'com.google.gms:google-services:4.3.2')** inside the dependencies

```
repositories{
google() //Maven
jcenter()
}
dependencies {
classpath 'com.google.gms:google-services:4.3.2' //
Google Services plugin
}
allprojects {
repositories {
google()
jcenter()
}
}
```

Now go to app level build.gradle and add apply
plugin: 'com.android.application' at bottom of the file.

Step 2: Add Firebase SDKs to your app.

We need to add the dependencies for the Firebase product that we want to integrate in our app. It is to be added in our module (app-level) Gradle file (usually app/build.gradle) as per instructions in the Firebase Console.
Note: Some Firebase SDKs for Android offer an alternative Kotlin extensions library.

1. Now we need to sync our app so that all the dependencies have the necessary versions. Logs in the device will now display the Firebase
Verification that Initialization is complete. When we run our app on an emulator with network access, Firebase Console will notify us that we have successfully connected our app to the Firebase.

Option 2: Adding Firebase with the Android studio Firebase Assistant:
Necessary Firebase Files are added into the Gradle and our app is registered with a Firebase Project all from within Android Studio using Firebase Assistant.

Caution: All the Firebase dependencies and settings should be configured and built correctly.

1. Open your Android project in AndroidStudio.

2. Click on Tools and then Firebase to open the FirebaseAssistant.

3. Select Connect to Firebase to register our app with an existing or a new Firebase project and automatically add the required files and code to your android project.

4. All the plugin versions should be up-to-date. These include:
- All the Project-level Gradle files and Google Services plug-inversion.
- All the App-level Gradle files and Firebase Android LibraryVersions.

5. Now, we need to sync our app in order to make sure that all dependencies are synced to the latest- version.

6. Lastly, we should run our app to check the connection to the firebase. Google's Maven repository should be included in both build scripts and all projects sections in build.gradle file at the project level. We now need to add ML Kit Android library dependencies to app-level GradleFile:

The dependencies are:
•implementation'com.google.firebase:firebase-ml-vision:18.0.1'
•implementation'com.google.firebase:firebase-analytics:17.2.2'
And the plug-ins are:
apply plugin: 'com.google.gms.google-services'

7. Add some more dependencies
```
dependencies{
   implementation 'com.google.firebase:firebase-ml-vision:18.0.2'
   implementation 'com.google.firebase:firebase-ml-vision-face-model:17.0.2'
}
```

Add these lines in the Manifest.xml :

<uses-permission
android:name="android.permission.CAMERA"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<meta-data android:name="com.google.firebase.ml.vision.DEPENDENCIES"
android:value="face" />

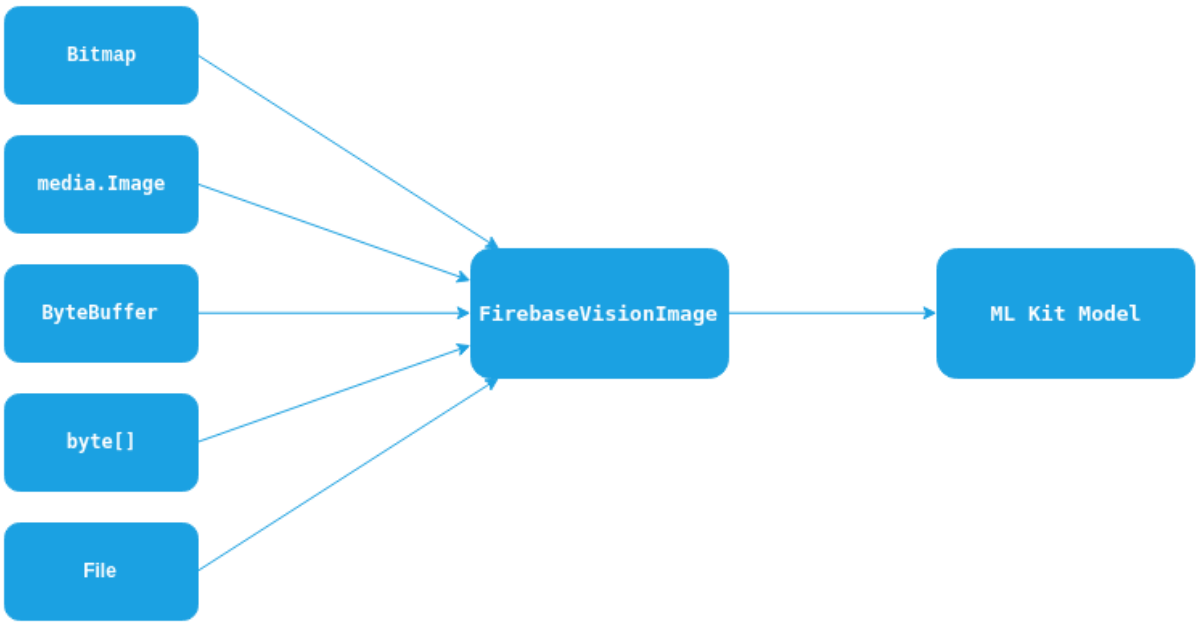


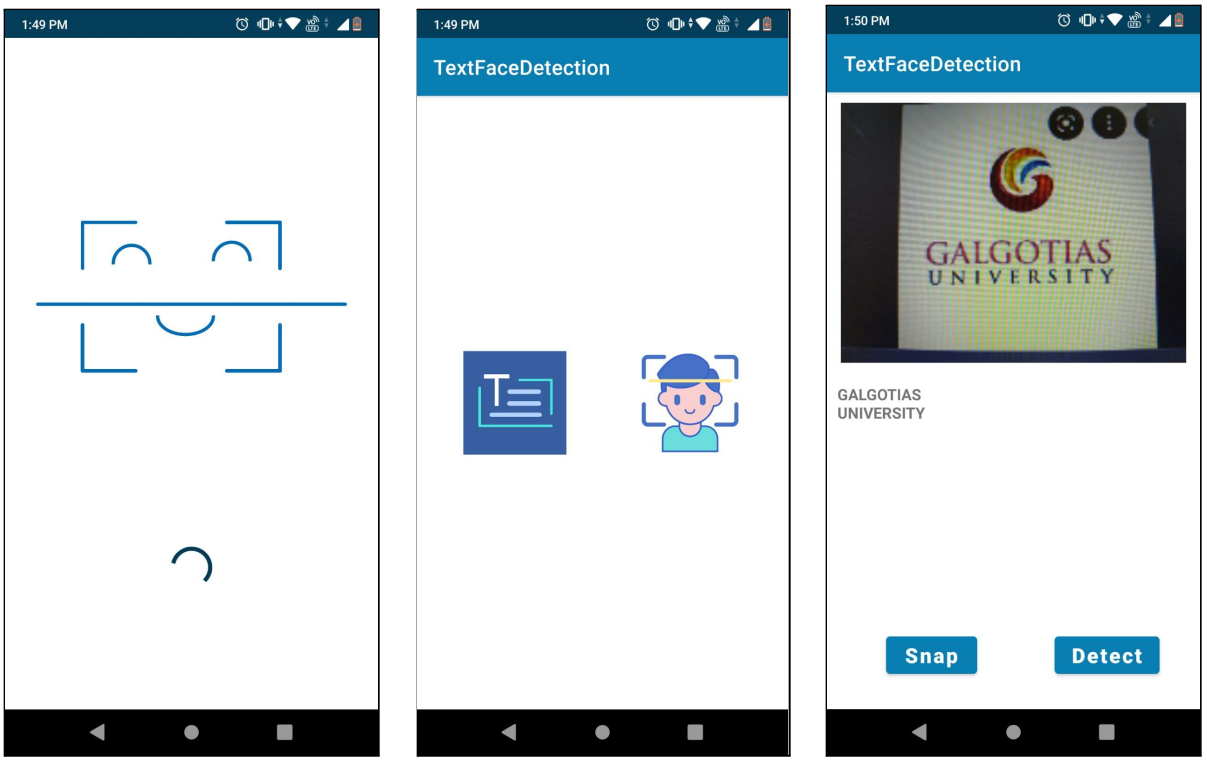
Fig 7: FirebaseVisionImage object

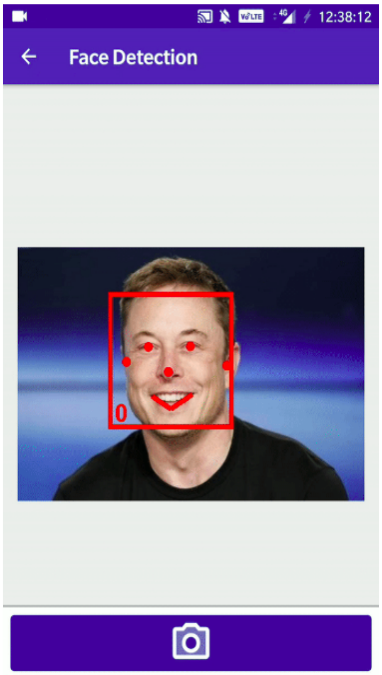
In the above figure 6, ML Kit provides an easy way to detect faces from a variety of image types like Bitmap, media. Image, ByteBuffer, byte[], or a file on the device. You just need to create a FirebaseVisionImage object from the above-
mentioned image types and pass it to the model.

CHAPTER 6

6.1 RESULT

The app thus gives the result of the snap or uploaded photo in the following way





6.1 RESULT & DISCUSSION

Machine Learning, Natural Language Processing is in itself a deep subject. As an Android Developer, it is very difficult to understand every algorithm and evaluate the result of the algorithm, and study this result for using it in the Android Application. But this can be made easy with Google Firebases' Machine Learning (ML) Kit. ML Kit provides us with many APIs which can be used by Android developers by importing the required dependencies. Thus, this is the best approach for creating faster, secure, and better applications implementing Machine Learning concepts. Recognize text API in Google Firebase accepts an image as an input and gives the available text in the image as an output user can use this for translating and other purposes. It also Recognizes the non-Latin-based languages if we use cloud API.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

As global needs and expectations from Mobile applications are rising, it is very essential to satisfy users with all the features and capabilities in a handheld device. We require " **Machine learning** " to meet global expectations, so understanding it is very necessary. But to have a good understanding of machine learning is not easy. A lot of research exercise is to be done before using them. So to overcome this problem, we can use "**Firebase**" a google product for the backend which has inbuilt **ML kit products/packages** to develop the "**Image Text Recognition & Facial Features Detection " API**. We can integrate ML Competence in our App through the ML kit and thus saving a lot of time and providing an efficient solution for our problem. Hence, an Android Developer now need not learn Machine Learning Algorithms and can focus on Android Application development. Firebase also has TestLab with capabilities of testing mobile apps in all aspects. It hence helps organizations to reach the business standards with very little effort and high efficiency.

7.2 FUTURE WORK

The world is using facial recognition technology and enjoying its benefits. Why should India be left out? There is a huge scope of this technology in India and it can help improve the country in various aspects. The technology and its applications can be applied across different segments in the country.

- Preventing the frauds at ATMs in India. A database of all customers with ATM cards in India can be created and facial recognition systems can be installed. So, whenever user will enter in ATM his photograph will be taken to permit the access after it is being matched with stored photo from the database.
- Reporting duplicate voters in India.
- Passport and visa verification can also be done using this technology.
- Also, driving license verification can be done using the same approach.
- In defense ministry, airports, and all other important places the technology can be used to ensure better surveillance and security.
- It can also be used during examinations such as Civil Services Exam, SSC, IIT, MBBS, and others to identify the candidates.

- This system can be deployed for verification and attendance tracking at various government offices and corporates.
- For access control verification and identification of authentic users, it can also be installed in bank lockers and vaults.
- For the identification of criminals, the system can be used by the police force also.

Proposed android application can be further extended to deal with any target and source language for translation. It can be further modified to deal with text having vertical or arbitrary orientation.

CHAPTER 8

SAMPLE CODE AND OUTPUT

APPENDIX 1: MAIN MODULE CODE

-> AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  package="com.ml.textfacedetection">
  <!-- below line is use to add camera feature in our app -->
  <uses-feature
     android:name="android.hardware.camera"
     android:required="true" /> <!-- permission for internet -->
  <uses-permission android:name="android.permission.INTERNET" />

  <application
     android:allowBackup="true"
     android:icon="@mipmap/ic_launcher"
     android:label="@string/app_name"
     android:roundIcon="@mipmap/ic_launcher_round"
     android:supportsRtl="true"
     android:theme="@style/Theme.TextFaceDetection">
     <activity android:name=".TextExtractionActivity"/>
     <activity android:name=".MainActivity"/>
     <activity android:name="com.ml.textfacedetection.SplashActivity"
```

```
        android:configChanges="orientation|screenSize"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.TextFaceDetection.NoActionBar"
        tools:ignore="MissingClass">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

-> build.gradle(Module)

```
plugins {
  id 'com.android.application'
  id 'kotlin-android'
  id 'com.google.gms.google-services'
}

android {
  compileSdkVersion 31
  buildToolsVersion "30.0.3"

  defaultConfig {
    applicationId "com.ml.textfacedetection"
    minSdkVersion 19
    targetSdkVersion 31
```

```
        versionCode 1
        versionName "1.0"
        multiDexEnabled true

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
                    proguardFiles  getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:4.+'
```

```
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    //dependency for firebase ml kit
    //noinspection OutdatedLibrary
    implementation 'com.google.firebase:firebase-ml-vision:15.0.0'
// dependency for firebase core.
implementation'com.google.firebase:firebase-core:15.0.2'

    implementation 'androidx.multidex:multidex:2.0.1'
}
```

-> build.gradle(Project)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    ext.kotlin_version = "1.4.32"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.3"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        classpath 'com.google.gms:google-services:4.3.8'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

```
allprojects {
  repositories {
     google()
     jcenter()
  }
}

task clean(type: Delete) {
  delete rootProject.buildDir
}
```

-> splactActivity.kt

```
package com.ml.textfacedetection

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler

class SplashActivity : AppCompatActivity() {

  // This is the loading time of the splash screen
  private val SPLASH_TIME_OUT: Long = 3000
  override fun onCreate(savedInstanceState: Bundle?) {
     super.onCreate(savedInstanceState)
     setContentView(R.layout.activity_splash)
     Handler().postDelayed({
```

```
            startActivity(Intent(this, MainActivity::class.java))
            finish()
        }, SPLASH_TIME_OUT)
    }
}
```

-> activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:backgroundTint="#0BA4EA"
    tools:context="com.ml.textfacedetection.SplashActivity">


    <ImageView
        android:id="@+id/logoImg"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_marginStart="100dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="100dp"
        android:src="@drawable/scan"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
  <ProgressBar
      android:id="@+id/progressBar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/logoImg"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="48dp"
      android:layout_marginBottom="16dp"
      android:indeterminateTint="#043C55"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.0"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/logoImg"
      app:layout_constraintVertical_bias="0.29"
      tools:targetApi="lollipop" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

-> mainActivity.kt

```
package com.ml.textfacedetection
import android.content.Intent
import android.media.Image
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ImageView
```

```
class MainActivity : AppCompatActivity() {

   lateinit var textScan : ImageView
   lateinit var faceScan : ImageView
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)

      textScan = findViewById(R.id.text)
      faceScan = findViewById(R.id.face)

      textScan.setOnClickListener{
         val intent : Intent = Intent(this,TextExtractionActivity::class.java)
         startActivity(intent)
      }

   }
}
```

-> activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">
```

```
  <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal"
     android:weightSum="2"
     android:layout_margin="10dp"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toTopOf="parent">

<ImageView
  android:id="@+id/text"
  android:layout_width="0dp"
  android:layout_weight="1"
  android:layout_height="100dp"
  android:src="@drawable/textscan"
  android:scaleType="fitCenter"
  />

     <ImageView
        android:id="@+id/face"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="100dp"
        android:src="@drawable/face"
        android:scaleType="fitCenter"
        />
  </LinearLayout>
```

</androidx.constraintlayout.widget.ConstraintLayout>

-> textExtraction.kt

```
package com.ml.textfacedetection
import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.Bitmap
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.ml.vision.FirebaseVision
import com.google.firebase.ml.vision.common.FirebaseVisionImage
import com.google.firebase.ml.vision.text.FirebaseVisionText

class TextExtractionActivity : AppCompatActivity() {
  // creating variables for our
  // image view, text view and two buttons.
  private lateinit var img: ImageView
  private lateinit var textview: TextView
  private lateinit var snapBtn: Button
  private lateinit var detectBtn: Button
```

```
// variable for our image bitmap.
private var imageBitmap: Bitmap? = null
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_text_extraction)

    img = findViewById<View>(R.id.image) as ImageView
    textview = findViewById<View>(R.id.text) as TextView
    snapBtn = findViewById<View>(R.id.snapbtn) as Button
    detectBtn = findViewById<View>(R.id.detectbtn) as Button



    // adding on click listener for detect button.

    // adding on click listener for detect button.
    detectBtn.setOnClickListener{

        // calling a method to
        // detect a text .
        detectTxt()

    }
    snapBtn.setOnClickListener {

        // calling a method to capture our image.
        dispatchTakePictureIntent()

    }

}
```

```
    private val REQUEST_IMAGE_CAPTURE = 1

    @SuppressLint("QueryPermissionsNeeded")
    private fun dispatchTakePictureIntent() {
        // in the method we are displaying an intent to capture our image.
        val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)

        // on below line we are calling a start activity
        // for result method to get the image captured.
        if (takePictureIntent.resolveActivity(packageManager) != null) {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
        }
    }


    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        // calling on activity result method.
            if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==
RESULT_OK) {
            // on below line we are getting
            // data from our bundles. .
            val extras = data!!.extras
            imageBitmap = extras!!["data"] as Bitmap?

            // below line is to set the
            // image bitmap to our image.
            img.setImageBitmap(imageBitmap)
        }
    }
```

```
  private fun detectTxt() {
      // this is a method to detect a text from image.
      // below line is to create variable for firebase
      // vision image and we are getting image bitmap.
      val image = FirebaseVisionImage.fromBitmap(imageBitmap!!)

      // below line is to create a variable for detector and we
      // are getting vision text detector from our firebase vision.
      val detector = FirebaseVision.getInstance().visionTextDetector

      // adding on success listener method to detect the text from image.
          detector.detectInImage(image).addOnSuccessListener { firebaseVisionText -> // calling a method to process
          // our text after extracting.
          processTxt(firebaseVisionText)
      }.addOnFailureListener { // handling an error listener.
              Toast.makeText(this@TextExtractionActivity, "Fail to detect the text from image..", Toast.LENGTH_SHORT).show()
      }
  }


  private fun processTxt(text: FirebaseVisionText) {
      // below line is to create a list of vision blocks which
      // we will get from our firebase vision text.
      val blocks = text.blocks
```

```
        // checking if the size of the
        // block is not equal to zero.
        if (blocks.size == 0) {
            // if the size of blocks is zero then we are displaying
            // a toast message as no text detected.
                                Toast.makeText(this@TextExtractionActivity,  "No  Text  ",
Toast.LENGTH_LONG).show()
            return
        }
        // extracting data from each block using a for loop.
        for (block in text.blocks) {
            // below line is to get text
            // from each block.
            val txt = block.text

            // below line is to set our
            // string to our text view.
            textview.text = txt
        }
    }

}
```

-> activity_text_extraction.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".TextExtractionActivity">




<!--image view to display our image-->
<ImageView
    android:id="@+id/image"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_margin="10dp"
    android:scaleType="fitCenter" />

<!--text view to display our extracted text-->
<TextView
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_margin="10dp"
    android:layout_below="@+id/image"
    android:textSize="15sp"
    android:textStyle="bold" />
```

```
<!--button to capture our image-->

<!--button to detect text from our image-->
<Button
    android:id="@+id/snapbtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="57dp"
    android:layout_marginLeft="57dp"
    android:layout_marginBottom="28dp"
    android:text="Snap"
    android:textAllCaps="false"
    android:textSize="20sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/detectbtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/snapbtn"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginEnd="39dp"
    android:layout_marginRight="39dp"
    android:text="Detect"
    android:textAllCaps="false"
```

```
    android:textSize="20sp"
    android:textStyle="bold" />

</RelativeLayout>
```

-> color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="purple_200">#FFBB86FC</color>
  <color name="purple_500">#087EB3</color>
  <color name="purple_700">#033F5A</color>
  <color name="teal_200">#FF03DAC5</color>
  <color name="teal_700">#FF018786</color>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
</resources>
```

-> themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
                                    <style          name="Theme.TextFaceDetection"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/purple_500</item>
    <item name="colorPrimaryVariant">@color/purple_700</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    <!-- Status bar color. -->
                                        <item      name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
  </style>
```

```
  <!-- This style is for the splash screen -->

  <style name="Theme.TextFaceDetection.NoActionBar">
     <item name="windowActionBar">false</item>
     <item name="windowNoTitle">true</item>
  </style>

</resources>
```

->faceDetectionActivity.kt

```
package com.ml.textfacedetection

import android.app.Activity
import android.content.Intent
import android.graphics.*
import android.os.Bundle
import android.provider.MediaStore
import android.support.design.widget.BottomSheetBehavior
import android.support.v7.app.AppCompatActivity
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.view.View
import android.widget.FrameLayout
import android.widget.ImageView
import android.widget.Toast
import com.google.firebase.ml.vision.FirebaseVision
import com.google.firebase.ml.vision.common.FirebaseVisionImage
import com.google.firebase.ml.vision.common.FirebaseVisionImageMetadata
import com.google.firebase.ml.vision.face.FirebaseVisionFace
import com.google.firebase.ml.vision.face.FirebaseVisionFaceContour
import com.google.firebase.ml.vision.face.FirebaseVisionFaceDetectorOptions
import com.google.firebase.ml.vision.face.FirebaseVisionFaceLandmark
import com.hitanshudhawan.firebasemlkitexample.R
import com.otaliastudios.cameraview.Facing
import com.otaliastudios.cameraview.Frame
import com.otaliastudios.cameraview.FrameProcessor
import com.theartofdev.edmodo.cropper.CropImage
import kotlinx.android.synthetic.main.activity_face_detection.*
```

```
import kotlinx.android.synthetic.main.content_face_detection.*

class FaceDetectionActivity : AppCompatActivity(), FrameProcessor {

  private var cameraFacing: Facing = Facing.FRONT

                    private val imageView by lazy { findViewById<ImageView>(R.id.face_detection_image_view)!! }

                    private val bottomSheetButton by lazy { findViewById<FrameLayout>(R.id.bottom_sheet_button)!! }
                    private val bottomSheetRecyclerView by lazy { findViewById<RecyclerView>(R.id.bottom_sheet_recycler_view)!! }
                    private val bottomSheetBehavior by lazy { BottomSheetBehavior.from(findViewById(R.id.bottom_sheet)!!) }

  private val faceDetectionModels = ArrayList<FaceDetectionModel>()

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_face_detection)
    setSupportActionBar(toolbar)
    supportActionBar?.setDisplayHomeAsUpEnabled(true)

    face_detection_camera_view.facing = cameraFacing
    face_detection_camera_view.setLifecycleOwner(this)
    face_detection_camera_view.addFrameProcessor(this)
    face_detection_camera_toggle_button.setOnClickListener {
          cameraFacing = if (cameraFacing == Facing.FRONT) Facing.BACK else Facing.FRONT
      face_detection_camera_view.facing = cameraFacing
    }

    bottomSheetButton.setOnClickListener {
      CropImage.activity().start(this)
    }

    bottomSheetRecyclerView.layoutManager = LinearLayoutManager(this)
```

```
bottomSheetRecyclerView.adapter = FaceDetectionAdapter(this, faceDetectionModels)
  }

  override fun process(frame: Frame) {

    val width = frame.size.width
    val height = frame.size.height

    val metadata = FirebaseVisionImageMetadata.Builder()
        .setWidth(width)
        .setHeight(height)
        .setFormat(FirebaseVisionImageMetadata.IMAGE_FORMAT_NV21)
                .setRotation(if (cameraFacing == Facing.FRONT)
FirebaseVisionImageMetadata.ROTATION_270 else
FirebaseVisionImageMetadata.ROTATION_90)
        .build()

        val firebaseVisionImage = FirebaseVisionImage.fromByteArray(frame.data,
metadata)
    val options = FirebaseVisionFaceDetectorOptions.Builder()
        .setContourMode(FirebaseVisionFaceDetectorOptions.ALL_CONTOURS)
        .build()
    val faceDetector = FirebaseVision.getInstance().getVisionFaceDetector(options)
    faceDetector.detectInImage(firebaseVisionImage)
        .addOnSuccessListener {
                face_detection_camera_image_view.setImageBitmap(null)

                val bitmap = Bitmap.createBitmap(height, width,
Bitmap.Config.ARGB_8888)
            val canvas = Canvas(bitmap)
            val dotPaint = Paint()
            dotPaint.color = Color.RED
            dotPaint.style = Paint.Style.FILL
            dotPaint.strokeWidth = 4F
            val linePaint = Paint()
            linePaint.color = Color.GREEN
            linePaint.style = Paint.Style.STROKE
            linePaint.strokeWidth = 2F

            for (face in it) {
```

```
                                                    val faceContours =
face.getContour(FirebaseVisionFaceContour.FACE).points
                for ((i, contour) in faceContours.withIndex()) {
                    if (i != faceContours.lastIndex)
                            canvas.drawLine(contour.x, contour.y, faceContours[i + 1].x,
faceContours[i + 1].y, linePaint)
                    else
                                canvas.drawLine(contour.x, contour.y, faceContours[0].x,
faceContours[0].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                        val leftEyebrowTopContours =
face.getContour(FirebaseVisionFaceContour.LEFT_EYEBROW_TOP).points
                for ((i, contour) in leftEyebrowTopContours.withIndex()) {
                    if (i != leftEyebrowTopContours.lastIndex)
                        canvas.drawLine(contour.x, contour.y, leftEyebrowTopContours[i +
1].x, leftEyebrowTopContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                    val leftEyebrowBottomContours =
face.getContour(FirebaseVisionFaceContour.LEFT_EYEBROW_BOTTOM).points
                for ((i, contour) in leftEyebrowBottomContours.withIndex()) {
                    if (i != leftEyebrowBottomContours.lastIndex)
                                                    canvas.drawLine(contour.x, contour.y,
leftEyebrowBottomContours[i + 1].x, leftEyebrowBottomContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                        val rightEyebrowTopContours =
face.getContour(FirebaseVisionFaceContour.RIGHT_EYEBROW_TOP).points
                for ((i, contour) in rightEyebrowTopContours.withIndex()) {
                    if (i != rightEyebrowTopContours.lastIndex)
                        canvas.drawLine(contour.x, contour.y, rightEyebrowTopContours[i
+ 1].x, rightEyebrowTopContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                val rightEyebrowBottomContours =
```

```
face.getContour(FirebaseVisionFaceContour.RIGHT_EYEBROW_BOTTOM).points
                for ((i, contour) in rightEyebrowBottomContours.withIndex()) {
                    if (i != rightEyebrowBottomContours.lastIndex)
                                                    canvas.drawLine(contour.x, contour.y,
rightEyebrowBottomContours[i + 1].x, rightEyebrowBottomContours[i + 1].y,
linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                            val leftEyeContours =
face.getContour(FirebaseVisionFaceContour.LEFT_EYE).points
                for ((i, contour) in leftEyeContours.withIndex()) {
                    if (i != leftEyeContours.lastIndex)
                            canvas.drawLine(contour.x, contour.y, leftEyeContours[i + 1].x,
leftEyeContours[i + 1].y, linePaint)
                    else
                                canvas.drawLine(contour.x, contour.y, leftEyeContours[0].x,
leftEyeContours[0].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                            val rightEyeContours =
face.getContour(FirebaseVisionFaceContour.RIGHT_EYE).points
                for ((i, contour) in rightEyeContours.withIndex()) {
                    if (i != rightEyeContours.lastIndex)
                            canvas.drawLine(contour.x, contour.y, rightEyeContours[i + 1].x,
rightEyeContours[i + 1].y, linePaint)
                    else
                                canvas.drawLine(contour.x, contour.y, rightEyeContours[0].x,
rightEyeContours[0].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                        val upperLipTopContours =
face.getContour(FirebaseVisionFaceContour.UPPER_LIP_TOP).points
                for ((i, contour) in upperLipTopContours.withIndex()) {
                    if (i != upperLipTopContours.lastIndex)
                            canvas.drawLine(contour.x, contour.y, upperLipTopContours[i +
1].x, upperLipTopContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }
```

```
                                                    val upperLipBottomContours =
face.getContour(FirebaseVisionFaceContour.UPPER_LIP_BOTTOM).points
                for ((i, contour) in upperLipBottomContours.withIndex()) {
                    if (i != upperLipBottomContours.lastIndex)
                        canvas.drawLine(contour.x, contour.y, upperLipBottomContours[i +
1].x, upperLipBottomContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                    val lowerLipTopContours =
face.getContour(FirebaseVisionFaceContour.LOWER_LIP_TOP).points
                for ((i, contour) in lowerLipTopContours.withIndex()) {
                    if (i != lowerLipTopContours.lastIndex)
                            canvas.drawLine(contour.x, contour.y, lowerLipTopContours[i +
1].x, lowerLipTopContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                    val lowerLipBottomContours =
face.getContour(FirebaseVisionFaceContour.LOWER_LIP_BOTTOM).points
                for ((i, contour) in lowerLipBottomContours.withIndex()) {
                    if (i != lowerLipBottomContours.lastIndex)
                        canvas.drawLine(contour.x, contour.y, lowerLipBottomContours[i +
1].x, lowerLipBottomContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                    val noseBridgeContours =
face.getContour(FirebaseVisionFaceContour.NOSE_BRIDGE).points
                for ((i, contour) in noseBridgeContours.withIndex()) {
                    if (i != noseBridgeContours.lastIndex)
                        canvas.drawLine(contour.x, contour.y, noseBridgeContours[i + 1].x,
noseBridgeContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }

                                                    val noseBottomContours =
face.getContour(FirebaseVisionFaceContour.NOSE_BOTTOM).points
                for ((i, contour) in noseBottomContours.withIndex()) {
                    if (i != noseBottomContours.lastIndex)
```

```
                        canvas.drawLine(contour.x, contour.y, noseBottomContours[i +
1].x, noseBottomContours[i + 1].y, linePaint)
                    canvas.drawCircle(contour.x, contour.y, 4F, dotPaint)
                }


                if (cameraFacing == Facing.FRONT) {
                    val matrix = Matrix()
                    matrix.preScale(-1F, 1F)
                     val flippedBitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.width,
bitmap.height, matrix, true)
                    face_detection_camera_image_view.setImageBitmap(flippedBitmap)
                } else {
                    face_detection_camera_image_view.setImageBitmap(bitmap)
                }
            }

        }
        .addOnFailureListener {
            face_detection_camera_image_view.setImageBitmap(null)
        }
  }

   public override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?)
{
     if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
        val result = CropImage.getActivityResult(data)

        if (resultCode == Activity.RESULT_OK) {
            val imageUri = result.uri
                    analyzeImage(MediaStore.Images.Media.getBitmap(contentResolver,
imageUri))
            face_detection_camera_container.visibility = View.GONE
                                                } else if (resultCode ==
CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
                Toast.makeText(this, "There was some error : ${result.error.message}",
Toast.LENGTH_SHORT).show()
        }
     }
  }
```

```
  private fun analyzeImage(image: Bitmap?) {
      if (image == null) {
          Toast.makeText(this, "There was some error", Toast.LENGTH_SHORT).show()
          return
      }

      imageView.setImageBitmap(null)
      faceDetectionModels.clear()
      bottomSheetRecyclerView.adapter?.notifyDataSetChanged()
      bottomSheetBehavior.state = BottomSheetBehavior.STATE_COLLAPSED
      showProgress()

      val firebaseVisionImage = FirebaseVisionImage.fromBitmap(image)
      val options = FirebaseVisionFaceDetectorOptions.Builder()
              .setPerformanceMode(FirebaseVisionFaceDetectorOptions.ACCURATE)

.setLandmarkMode(FirebaseVisionFaceDetectorOptions.ALL_LANDMARKS)

.setClassificationMode(FirebaseVisionFaceDetectorOptions.ALL_CLASSIFICATIONS
)
              .build()
      val faceDetector = FirebaseVision.getInstance().getVisionFaceDetector(options)
      faceDetector.detectInImage(firebaseVisionImage)
              .addOnSuccessListener {
                  val mutableImage = image.copy(Bitmap.Config.ARGB_8888, true)

                  detectFaces(it, mutableImage)

                  imageView.setImageBitmap(mutableImage)
                  hideProgress()
                  bottomSheetRecyclerView.adapter?.notifyDataSetChanged()
                  bottomSheetBehavior.state = BottomSheetBehavior.STATE_EXPANDED
              }
              .addOnFailureListener {
                                          Toast.makeText(this, "There was some error",
Toast.LENGTH_SHORT).show()
                  hideProgress()
              }
  }
```

```
   private fun detectFaces(faces: List<FirebaseVisionFace>?, image: Bitmap?) {if (faces
== null || image == null) {
        Toast.makeText(this, "There was some error", Toast.LENGTH_SHORT).show()
        return
      }


      for ((index, face) in faces.withIndex()) {

        canvas.drawRect(face.boundingBox, facePaint)
                        canvas.drawText("Face$index",  (face.boundingBox.centerX()  -
face.boundingBox.width()  /  2)  +  8F,  (face.boundingBox.centerY()  +
face.boundingBox.height() / 2) - 8F, faceTextPaint)

        if (face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EYE) != null) {
          val leftEye = face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EYE)!!
          canvas.drawCircle(leftEye.position.x, leftEye.position.y, 8F, landmarkPaint)
        }
        if (face.getLandmark(FirebaseVisionFaceLandmark.RIGHT_EYE) != null) {
                                                    val      rightEye      =
face.getLandmark(FirebaseVisionFaceLandmark.RIGHT_EYE)!!
                         canvas.drawCircle(rightEye.position.x,  rightEye.position.y,  8F,
landmarkPaint)
        }
        if (face.getLandmark(FirebaseVisionFaceLandmark.NOSE_BASE) != null) {
          val nose = face.getLandmark(FirebaseVisionFaceLandmark.NOSE_BASE)!!
          canvas.drawCircle(nose.position.x, nose.position.y, 8F, landmarkPaint)
        }
        if (face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EAR) != null) {
          val leftEar = face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EAR)!!
          canvas.drawCircle(leftEar.position.x, leftEar.position.y, 8F, landmarkPaint)
        }
        if (face.getLandmark(FirebaseVisionFaceLandmark.RIGHT_EAR) != null) {
                                                    val      rightEar      =
face.getLandmark(FirebaseVisionFaceLandmark.RIGHT_EAR)!!
          canvas.drawCircle(rightEar.position.x, rightEar.position.y, 8F, landmarkPaint)
        }
           if (face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_LEFT) != null
&& face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_BOTTOM) != null &&
face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_RIGHT) != null) {
```

```
                                                    val leftMouth =
face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_LEFT)!!
                                                    val bottomMouth =
face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_BOTTOM)!!
                                                    val rightMouth =
face.getLandmark(FirebaseVisionFaceLandmark.MOUTH_RIGHT)!!
                            canvas.drawLine(leftMouth.position.x, leftMouth.position.y,
bottomMouth.position.x, bottomMouth.position.y, landmarkPaint)
                        canvas.drawLine(bottomMouth.position.x, bottomMouth.position.y,
rightMouth.position.x, rightMouth.position.y, landmarkPaint)
        }

            faceDetectionModels.add(FaceDetectionModel(index, "Smiling Probability
${face.smilingProbability}"))
                faceDetectionModels.add(FaceDetectionModel(index, "Left Eye Open
Probability ${face.leftEyeOpenProbability}"))
                faceDetectionModels.add(FaceDetectionModel(index, "Right Eye Open
Probability ${face.rightEyeOpenProbability}"))
    }
  }

  private fun showProgress() {
    findViewById<View>(R.id.bottom_sheet_button_image).visibility = View.GONE
                findViewById<View>(R.id.bottom_sheet_button_progress).visibility =
View.VISIBLE
  }

  private fun hideProgress() {
                  findViewById<View>(R.id.bottom_sheet_button_image).visibility =
View.VISIBLE
                findViewById<View>(R.id.bottom_sheet_button_progress).visibility =
View.GONE
  }

}
```

REFERENCES

[1] Guido Albertengo, Fikru G. Debele, Waqar Hassan, Dario Stramandino, “On the performance of web services, google cloud messaging and firebase” Digital Communications and Network,2019.

[2] David Ward, Catheryn Peoples, “An IOS Application using Firebase for Gym Membership Management”, IEEE Potentials, Vol. 38, Issue:3,2019.

[3] Chunnu Khawas and Pritam Shah, “Application of Firebase in Android App Development-A study”, In International Journal of Computer Applications, Vol. 136, Issue.:146,2018.

[4] https://firebase.google.com/docs/ml-kit/identify-languages