MACHINE LEARNING ANDROID APPLICATION FOR IMAGE & FACE RECOGNITION

A PROJECT REPORT

Submitted by

M. PATHMA PRIYA (952317205020)

S. SUDHA (952317205033)

J. UMASUDHA (952317205036)

A. RAMALAKSHMI (952317205026)

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PSN ENGINEERING COLLEGE, TIRUNELVELI-627 152

ANNA UNIVERSITY: CHENNAI – 600 025
MARCH - 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"MACHINE LEARNING ANDROID APPLICATION FOR IMAGE AND FACE RECOGNITION"** is the bonafide Work of **M.PATHMAPRIYA (952317205020), S.SUDHA (952317205033), J.UMASUDHA (952317205036), A. RAMALAKSHMI (952317205026)** who Carried out the Project Work under my supervision.

SIGNATURE

Mr.R.ROBIN JESUBALAN M.E.,

HEAD OF THE DEPARTMENT

Information Technology

PSN Engineering College

Tirunelveli-627152

SIGNATURE

Mr.R.ROBINJESUBALAN,M.E

SUPERVISOR

Information Technology

PSN Engineering College

Tirunelveli-627152

Submitted For Viva-voce Examination held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

At the very outset, I express my thanks to **GOD ALMIGHTY** who blessed me with a healthy constitution and bestowed upon us the required skills to complete this project.

First and foremost, I would like to thank our chairman **Dr.P.SUYAMBU Ph.D.,** for providing full facilities and technical environment to start this project work.

I express my heartfelt gratitude to the Principal **Dr.M.S.RAVIKUMAR, M.E., Ph.D.,** for his Constant Support.

I profound my thanks to the HOD & Supervisor **Mr.R.ROBIN JESUBALAN ,M.E.,** Professor and Head of the Department , Information Technology, for Rendering his full support both mentally and technically by encouraging me at all times I needed it.

I extend my sincere thanks to all the Teaching and Non- Teaching Staff Members in the Department of Information Technology for their valuable help and support.

M.PATHMAPRIYA
S.SUDHA
J.UMASUDHA
A.RAMALAKSHMI

ABSTRACT

In this Project, We Create Two Android applications using the domain of Machine Learning. They are, Image Recognition Application and Face Recognition Application. In the Face Recognition application, we will learn how to compare Two Faces in an Image and Detect the face in the image is same or Different.
In the Image Recognition App, if you show the image of dog it was going to tell you the breed of dog or if you show any fruit is it going to tell you which fruit it is. Similarly it can predict 1000 categories which is using Tensor Flow Lite & Android Studio.

TABLE OF CONTENTS

S.NO	LIST OF FIGURES	PAGE NO
1.	APPLICTION OF MACHINE LEARNING	39
2.	ARCHITECTURAL DESIGN OF FACE RECOGNITION	41
3.	DATA FLOW DIAGRAM	42
4.	UML DIAGRAM	43
5.	DESIGNING PAGE 1	74
6.	TENSOR FLOW WITH KOTLIN PAGE	75
7.	MAIN PROGRAM PAGE 1	76
8.	MAIN DESIGN PAGE 2	77
9.	MAIN PROGRAM PAGE 2	78

LIST OF ABBREVIATION

- **CPU** –CENTRAL PROCESSING UNIT
- **IC** – INTEGRATED CIRCUIT
- **DSP** - DIGITAL SIGNAL PROCESSOR
- **ASIP-** APPLICATION SPECIFIC SET INSTRUCTOR PROCESSOR
- **GPU**- GRAPHICS PROCESSING UNIT
- **PPU**- PHYSICS PROCESSING UNIT
- **RAM**- RANDOM ACCESS MEMORY
- **XML**- EXTENSIBLE MARKUP LANGUAGE
- **PSTN**-PUBLIC SWITCHED TELEPHONE NETWORK
- **IOT** - INTERNET OF THINGS

5. INTRODUCTION

Nowadays People travel across different places within and outside their country for Their official purpose or it may be an excursion, most of them are not familiar with The language used within that place, recognizing the real-life object which Mentioned in the other languages, routes they need to go. In order to solve this Problem, the app is created based on a Camera application. At first, the user Captures an image using Camera in Mobile. The Captured Image which is the Object contains the face of a human then it discover the image in the face is same Or Not. In this Project, We Create Two Android applications using the domain of Machine Learning. They are, Image Recognition Application and Face Recognition Application. In the Face Recognition application, we will learn how to Compare Two Faces in an Image and Detect the face in the image is same or Different. In the Image Recognition App, if you show the image of dog it was Going to tell you the breed of dog or if you show any fruit is it going to tell you Which fruit it Is. Similarly it can predict 1000 categories which is using Tensor

Flow Lite & Android Studio.

5.1 MACHINE LEARNING (ML)

ML is the study of computer algorithms that improve automatically through Experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training Data", in order to make predictions or decisions without being explicitly Programmed to do so.Machine learning algorithms are used in a wide variety of Applications, such as email filtering and computer vision, where it is difficult or Unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which Focuses on making predictions using computers; but not all machine learning is Statistical learning. The study of mathematical optimization delivers methods, Theory and application domains to the field of machine learning. Data mining is a Related field of study, focusing on exploratory data analysis through unsupervised Learning. In its application across business problems, machine learning is also Referred toast predictive analytics.

Machine learning involves computers discovering how they can perform tasks Without being explicitly programmed to do so. It involves computers learning from Data provided so that they carry out certain tasks. For simple tasks assigned to Computers, it is possible to program algorithms telling the machine how to execute All steps required to solve the problem at hand; on the computer's part, no learning Is needed. For more advanced tasks, it can be challenging for a human to manually Create the needed algorithms. In practice, it can turn out to be more effective to help The machine develop its own algorithm, rather than having human programmers Specify every needed step. The discipline of machine learning employs various Approaches to teach computers to accomplish tasks where no fully satisfactory Algorithm is available. In cases where vast numbers of potential answers exist, one Approach is to label some of the correct answers as valid. This can then be used as Training data for the computer to improve the algorithm(s) it uses to determine Correct answers. For example, to train a system for the task of digital character Recognition, the MNIST dataset of handwritten digits has often been used.

5.2 MACHINE LEARNING APPROCHES

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

6. SYSTEM REQUIREMENT

6.1 HARDWARE REQUIREMENT

- PROCESSOR: INTEL[R] CORE [PM] 2 DUO CPU E7400@2.80 GHZ
- RAM : 2.80GHZ, 1.99 GB
- HARD DISC : 20GB
- INPUT DEVICE : STANDARD KEYBOARD AND MOUSE
- OUTPUT DEVICE : MONITOR AND MOBILE PHONE
- CAMERA : VERSION 5.3.8 , SIZE 242KIB

6.1.1 PROCESSOR

- Central processing unit(CPU), the hardware within a computer that executes a program
- Microprocessor, a central processing unit contained on a single integrated circuit (IC)

    - Application-specific instruction set processor (ASIP), a component used in system-on-a-chip design
    - Graphics processing unit (GPU), a processor designed for doing dedicated graphics-rendering computations
    - Physics processing unit (PPU), a dedicated microprocessor designed to handle the calculations of physics
    - Digital signal processor (DSP), a specialized microprocessor designed specifically for digital signal processing
        - Image processor, a specialized DSP used for image processing in digital cameras, mobile phones or other devices
    - Coprocessor
    - Floating-point unit
    - Network processor, a microprocessor specifically targeted at the networking application domain
- Multi-core processor, single component with two or more independent CPUs (called "cores") on the same chip carrier or on the same die
- Front-end processor, a helper processor for communication between a host computer and other devices

6.1.2 RAM

Random-access memory (**RAM**) is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code. A random-access memory device allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory. In contrast, with other direct-access data storage media such as hard disks, CD-RWs, DVD-RWs and the older magnetic tapes and drum memory, the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement.

RAM contains multiplexing and DE multiplexing circuitry, to connect the data lines to the addressed storage for reading or writing the entry. Usually more than one bit of storage is accessed by the same address, and RAM devices often have multiple data lines and are said to be "8-bit" or "16-bit", etc. devices.

In today's technology, random-access memory takes the form of integrated circuit(IC) chips with MOS (metal-oxide-semiconductor) memory cells. RAM is normally associated with volatile types of memory (such as dynamic random-access memory (DRAM) modules), where stored information is lost if power is removed, although non-volatile RAM has also been developed. Other types of non-

volatile memories exist that allow random access for read operations, but either do not allow write operations or have other kinds of limitations on them. These include most types of ROM and a type of flash memory called *NOR-Flash*.

The two main types of volatile random-access semiconductor memory are static random-access memory (SRAM) and dynamic random-access memory (DRAM). Commercial uses of semiconductor RAM date back to 1965, when IBM introduced the SP95 SRAM chip for their System/360 Model 95 computer, and Toshiba used DRAM memory cells for its Tosca BC-1411 electronic calculator, both based on bipolar transistors. Commercial MOS memory, based on MOS transistors, was developed in the late 1960s, and has since been the basis for all commercial semiconductor memory. The first commercial DRAM IC chip, the Intel 1103, was introduced in October 1970. Synchronous dynamic random-access memory (SDRAM) later debuted with the Samsung KM48SL2000 chip in 1992.

6.1.3 HARD DISC

A **hard disk drive** (**HDD**), **hard disk**, **hard drive**, or **fixed disk**[b] is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage and one or more rigid rapidly rotating platters coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a

moving actuator arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored and retrieved in any order. HDDs are a type of non-volatile storage, retaining stored data even when powered off.

Introduced by IBM in 1956, HDDs were the dominant secondary storage device for general-purpose computers beginning in the early 1960s. HDDs maintained this position into the modern era of servers and personal computers, though personal computing devices produced in large volume, like cell phones and tablets, rely on flash memory storage devices. More than 224 companies have produced HDDs historically, though after extensive industry consolidation most units are manufactured by Seagate, Toshiba, and Western Digital. HDDs dominate the volume of storage produced (Exabyte’s per year) for servers. Though production is growing slowly (by Exabyte’s shipped), sales revenues and unit shipments are declining because solid-state drives (SSDs) have higher data-transfer rates, higher areal storage density, better reliability, and much lower latency and access times.

6.1.4 MONITOR

A **computer monitor** is an output device that displays information in pictorial form. A monitor usually comprises the visual display, circuitry, casing, and power

supply. The display device in modern monitors is typically a thin film transistor liquid crystal display (TFT-LCD) with LED backlighting having replaced cold-cathode fluorescent lamp (CCFL) backlighting. Previous monitors used a cathode ray tube (CRT). Monitors are connected to the computer via VGA, Digital Visual Interface (DVI), HDMI, Display Port, USB-C, low-voltage differential signaling (LVDS) or other proprietary connectors and signals.

Originally, computer monitors were used for data processing while television sets were used for entertainment. From the 1980s onwards, computers (and their monitors) have been used for both data processing and entertainment, while televisions have implemented some computer functionality. The common aspect ratio of televisions, and computer monitors, has changed from 4:3 to 16:10, to 16:9.

Modern computer monitors are easily interchangeable with conventional television sets and vice versa. However, as computer monitors do not necessarily include integrated speakers nor TV tuners (such as Digital television adapters), it may not be possible to use a computer monitor as a TV set without external components.

6.1.5 MOUSE

A **computer mouse** (plural **mice**, rarely **mousse**) is a hand-held pointing device that detects two-dimensional motion relative to a surface. This motion is typically translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface of a computer.

The first public demonstration of a mouse controlling a computer system was in 1968. Mice originally used a ball rolling on a surface to detect motion, but modern mice mostly have optical sensors that have no moving parts. Originally wired to a computer, many modern mice are cordless, relying on short-range radio communication with the connected system.

In addition to moving a cursor, computer mice have one or more buttons to allow operations such as selection of a menu item on a display. Mice often also feature other elements, such as touch surfaces and scroll wheels, which enable additional control and dimensional input.

6.1.6 KEYBOARD

A **computer keyboard** is a typewriter-style device which uses an arrangement of buttons or keys to act as mechanical levers or electronic switches. Replacing early punched cards and paper tape technology, interaction via teleprompter-style

keyboards have been the main input method for computers since the 1970s, supplemented by the computer mouse since the 1980s.

Keyboard keys (buttons) typically have a set of characters engraved or printed on them, and each press of a key typically corresponds to a single written symbol. However, producing some symbols may require pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or symbols (characters), other keys or simultaneous key presses can prompt the computer to execute system commands, such as such as the Control-Alt-Delete combination used with Microsoft Windows. In a modern computer, the interpretation of key presses is generally left to the software: the information sent to the computer, the scan code, tells it only which key (or keys) on which row and column, was pressed or released.

6.1.7 MOBILE PHONE

A **mobile phone**, **cellular phone, cell phone**, **cellphone**, **hand phone**, or **hand phone**, sometimes shortened to simply **mobile**, **cell** or just **phone**, is a portable telephone that can make and receive calls over a radio frequency link while the user is moving within a telephone service area. The radio frequency link

establishes a connection to the switching systems of a mobile phone operator, which provides access to the public switched telephone network (PSTN). Modern mobile telephone services use a cellular network architecture and, therefore, mobile telephones are called *cellular telephones* or *cell phones* in North America. In addition to telephony, digital mobile phones (2G) support a variety of other services, such as text messaging, MMS, email, Internet access, short-range wireless communications (infrared, Bluetooth), business applications, video games and digital photography. Mobile phones offering only those capabilities are known as feature phones; mobile phones which offer greatly advanced computing capabilities are referred to as smartphones.

The development of metal-oxide-semiconductor (MOS) large-scale integration (LSI) technology, information theory and cellular networking led to the development of affordable mobile communications .The first handheld mobile phone was demonstrated by John F. Mitchell and Martin Cooper of Motorola in 1973, using a handset weighing. In 1979, Nippon Telegraph and Telephone (NTT) launched the world's first cellular network in Japan. In 1983, the Dynastic 8000x was the first commercially available handheld mobile phone. From 1983 to 2014, worldwide mobile phone subscriptions grew to over seven billion; enough to provide one for every person on Earth.[5] In the first quarter of 2016, the top smartphone developers worldwide were Samsung, Apple and Huawei;

smartphone sales represented 78 percent of total mobile phone sales. For feature phones (slang: *"dumb phones"*) as of 2016, the largest were Samsung, Nokia and Alcatel.

6.1.8 CAMERA

A camera is an optical instrument used to capture an image. At their most basic, cameras are sealed boxes (the camera body) with a small whole (the aperture) that allows light in to capture an image on a light-sensitive surface (usually photographic film or a digital sensor). Cameras have various mechanisms to control how the light falls onto the light-sensitive surface. Lenses focus the light entering the camera, the size of the aperture can be widened or narrowed to let more or less light into the camera, and a shutter mechanism determines the amount of time the photo-sensitive surface is exposed to the light.

The still image camera is the main instrument in the art of photography and captured images may be reproduced later as a part of the process of photography, digital imaging, and photographic printing. The similar artistic fields in the moving image camera domain are film, videography, and cinematography.

6.2 SOFTWARE REQUIREMENT

- OPERATING SYSTEM : WINDOWS 11
- BACK END : TENSOR FLOW LITE
- LANGUAGE : KOTLIN , JAVA, XML
- BROWSER : GOOGLE CHROME
- FRONT END : ANDROID STUDIO

6.2.1 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on Jet Brains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, machos and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

FEATURES

The specific feature of the Android Studio is an absence of the possibility to switch auto save feature off.

The following features are provided in the current stable version:

- Grade -based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects back port some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

6.2.2 TENSOR FLOW

Tensor Flow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Tensor flow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

Tensor Flow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

Tensor Flow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, Tensor Flow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

Tensor Flow computations are expressed as stateful dataflow graphs. The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned Tensor Flow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of Tensor Flow on Kubernetes. In March 2018, Google announced

TensorFlow.js version 1.0 for machine learning in JavaScript. In Jan 2019, Google announced Tensor Flow 2.0. It became officially available in Sep 2019.In May 2019, Google announced Tensor Flow Graphics for deep learning in computer graphics.

In May 2017, Google announced a software stack specifically for mobile development, Tensor Flow Lite In January 2019, Tensor Flow team released a developer preview of the mobile GPU inference engine with OpenGL ES 3.1 Compute Shaders on Android devices and Metal Compute Shaders on iOS devices. In May 2019, Google announced that their Tensor Flow Lite Micro (also known as Tensor Flow Lite for Microcontrollers) and ARM's Tensor would be merging. Tensor Flow Lite uses Flat Buffers as the data serialization format for network models, eschewing the Protocol Buffers format used by standard Tensor Flow models.

6.2.3 JAVA

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms

that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine(JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Oracle offers its own Hotspot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open source software and used by most developers and is the default JVM for almost all Linux distributions.

As of March 2021, the latest version is Java 16, with Java 11, a currently supported long-term support (LTS) version, released on September 25, 2018. Oracle released the last zero-cost public update for the legacy version Java 8 LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors have begun to offer zero-cost builds of OpenJDK 8 and 11 that are still receiving security and other upgrades.

Oracle (and others) highly recommend uninstalling outdated versions of Java because of serious risks due to unresolved security issues. Since Java 9, 10, 12, 13, 14, and 15 are no longer supported, Oracle advises its users to immediately transition to the latest version (currently Java 16) or an LTS release.

6.2.4 KOTLIN

Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Kotlin's standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript (e.g., for frontend web applications using React)

or native code (via LLVM); e.g., for native iOS apps sharing business logic with Android apps. Language development costs are borne by Jet Brains, while the Kotlin Foundation protects the Kotlin trademark.

On 7 May 2019, Google announced that the Kotlin programming language is now its preferred language for Android app developers. As a result many developers have switched to Kotlin. Since the release of Android Studio 3.0 in October 2017, Kotlin has been included as an alternative to the standard Java compiler. The Android Kotlin compiler produces Java 6 byte code by default (which runs in any later JVM), but lets the programmer choose to target Java 8 up to 15, for optimization, or allows for more features, e.g. Java 8 related with Kotlin 1.4, and has experimental record class support for Java 16 compatibility.

Kotlin support for JavaScript (i.e. classic back-end) is considered stable in Kotlin 1.3 by its developers, while Kotlin/JS (IR-based) in version 1.4, and is considered alpha. Kotlin/Native Runtime (for e.g. Apple support) is considered beta.

When Kotlin was announced as an official Android development language at Google I/O in May 2017, it became the third language fully supported for Android, in addition to Java and C++. As of 2020, Kotlin is still most widely used

on Android, with Google estimating that 70% of the top 1000 apps on the Play Store are written in Kotlin. Google itself has 60 apps written in Kotlin, including Maps and Drive. Many Android apps, such as Google's Home, are in the process of being migrated to Kotlin, and so use both Kotlin and Java. Kotlin on Android is seen as beneficial for its null-pointer safety as well as for its features that make for shorter, more readable code.

6.2.5 XML

Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications all of them free open standards define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

7. SYSTEM ANALYSIS

7.1 EXISTING SYSTEM

The impact of this project is made by the existing system where Google Lens process the image into text, translate, navigate and search about the text. The image is processed based on two major modules TEXT and IMAGE. TEXT module contains translation, maps, related Images. IMAGE module contains face emotions and image recognition. Where face emotion detects a face and reveals one's reactions such as happiness or sadness, etc. This project is being developed using Android Studio with the Cognitive Services from Microsoft and Google API services.

7.1.1 DE-MERITS

- The background of the object also plays a significant role in Face detection.
- The facial recognition system is highly sensitive to pose variations.

- The movement of head or different camera positions can cause changes of facial texture and it will generate the wrong result.
- Change in facial expressions may produce a different result for the same individual.
- Occlusion means the face as beard, mustache, accessories (goggles, caps, mask, etc.**)**

7.2 PROPOSED SYSTEM

In this Project, We Create Two Android applications using the domain of Machine Learning. They are, Image Recognition Application and Face Recognition Application.

In this Face Recognition application, we will learn how to compare

Two Faces in an Image and Detect the face in the image is same or Different Faces.

In the Image Recognition App, if you show the image of dog it was going to tell You the breed of dog or if you show any fruit is it going to tell you which fruit it Is.Similarly it can predict 1000 categories which is using Tensor Flow Lite & Android Studio.

7.2.1 MERITS

- Face detection API is used for the detection of faces in images, key facial features, and build contours of detected faces.
- This feature is used only offline.
- Using this API, users can extract faces from pictures and edit those using different filters. Also, it is possible to generate avatars from users' photos.
- Since ML Kit provides users with a possibility to apply face detection feature in real-time mode, developers may integrate it into video chats or games.
- So if you need to develop face recognition app, you may this API from Tensor Flow Lite.
- Using this tensor-flow API, image clearly predicted

8. LITERATURE SURVEY

8.1 FOR FACE RECOGNITION APPLICATION

Human recognition modules are available in two separate lens options for long distance and wide angle detection. It uses an Omron developed image recognition

algorithm to determine not only human face and body detection but also estimates gender, age, expression, and five other facial traits.

Customers can also choose from two camera heads, a long-distance detection type and a wide-angle detection type, depending on their specific application purposes. A piece of equipment embedded with the HVC-P2 can detect and presume attributes and conditions of a user coming in its vicinity, without the user knowing the presence of a camera, making it possible to provide services deemed most suitable in view of the user's attributes.

We will be building our facial recognition model using Keras (A Python library) and MobileNetV2 (a model built by Google). Training a model of your own requires a good amount of diverse data for training.

Just to make you aware of, in one of the Google Colabs example where they are classifying flowers, they are using at least 600 images for each flower to train the model. The model will work even for 50 photographs, but won't be very accurate. To improve accuracy, you need more and more 'diverse' photographs.

8.2 FOR IMAGE RECOGNITION APPLICATION

Just like the phrase “What-you-see-is-what-you-get” says, human brains make vision easy. It doesn’t take any effort for humans to tell apart a dog, a cat or a flying saucer. But this process is quite hard for a computer to imitate: they only seem easy because God designs our brains incredibly good in recognizing images. A common example of image recognition is optical character recognition (OCR). A scanner can identify the characters in the image to convert the texts in an image to a text file. With the same process, OCR can be applied to recognize the text of a license plate in an image.

First, a great number of characteristics, called features are extracted from the image. An image is actually made of “pixels”, as shown in Figure (A). Each pixel is represented by a number or a set of numbers — and the range of these numbers is called the color depth (or bit depth). In other words, the color depth indicates the maximum number of potential colors that can be used in an image. In an (8-bit) greyscale image (black and white) each pixel has one value that ranges from 0 to 255. Most images today use 24-bit color or higher. An RGB color image means the color in a pixel is the combination of red, green and blue. Each of the colors ranges from 0 to 255. This RGB color generator shows how any color can be generated by RGB. So a pixel contains a set of three values RGB (102, 255, and 102) refers to color **#66ff66. A**n image 800 pixel wide, 600 pixels high has 800 x 600 = 480,000 pixels = 0.48 megapixels (“megapixel” is 1 million pixels). An image with a

resolution of 1024×768 is a grid with 1,024 columns and 768 rows, which therefore contains 1,024 × 768 = 0.78 megapixels.

8.3 APPLICATION OF MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it to learn for themselves. When the computer/machine-learning-model learns, It forms '**Inference Rules'** by finding out common patterns in the input to reach out to the desired output.

You can assume a machine learning model as a black-box, you give it an input and the desired output. The black-box itself will form its own understanding/rules so that when you give it a similar input in the future, it infers out a similar desired output. This is how 'intelligence' is built into the computer.

A machine learning model is made of up of nodes which are similar to Neurons in our human brains. These neurons are structured as layers. There is an Input Layer, Hidden Layer, and Output Layer.

The Input layer takes the input, pre-processes it for the next layers and sends it to the hidden layer.

The hidden layer itself can have multiple layers within itself which do the inferencing/processing of the input to get to output. There is some '***weight***' associated with each node of the model (just like Neurons in our brain). These weights are tuned while the model is being trained until we get the desired accuracy in the output.

The output layer gets the inferred output from the Hidden layer and gives the output in the desired format.

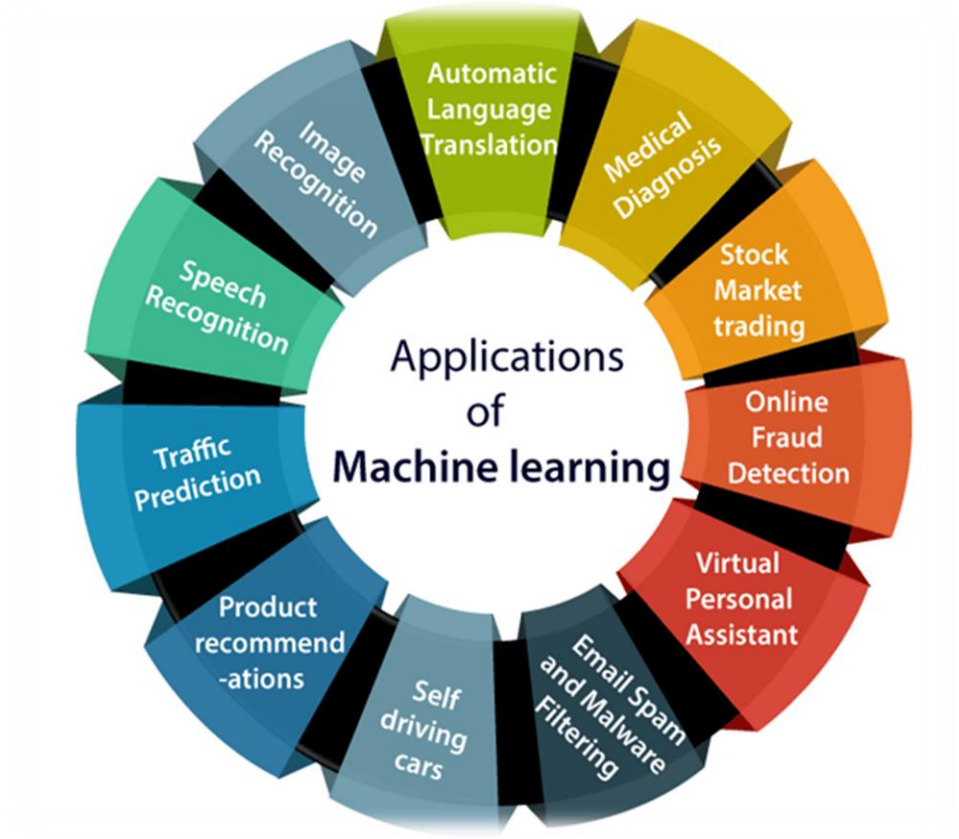
Tensor Flow is a multipurpose machine learning framework. Tensor Flow can be used anywhere from training huge models across clusters in the cloud to running models locally on an embedded system like your phone/IoT devices.

Machine Learning has been here for a while, there are a lot of open-source libraries like Tensor Flow where you can find a lot of pre-trained models and build cool stuff on top of them, without starting from Scratch. What we are trying to achieve here falls under Image Classification, where our Machine learning model has to classify the faces in the images amongst the recognized people.

There is a Github repo called Tensor Flow Zoo, where you can find the models. There are some factors involved which you should consider while choosing your model, most importantly, the speed is in milliseconds and the accuracy. If you are trying to build something which works in real-time, like in a live Camera Stream then we need the speed otherwise it would be a bad user experience as each frame will be processed. There is an obvious tradeoff between speed and accuracy, so this is one of the things you should look out for while choosing your model.

Image Recognition: Image recognition is one of the most common **applications of machine learning**. ...

- Speech Recognition
- Traffic prediction
- Product recommendations
- Self-driving cars
- Email Spam and Malware Filtering
- Virtual Personal Assistant
- Online Fraud Detection

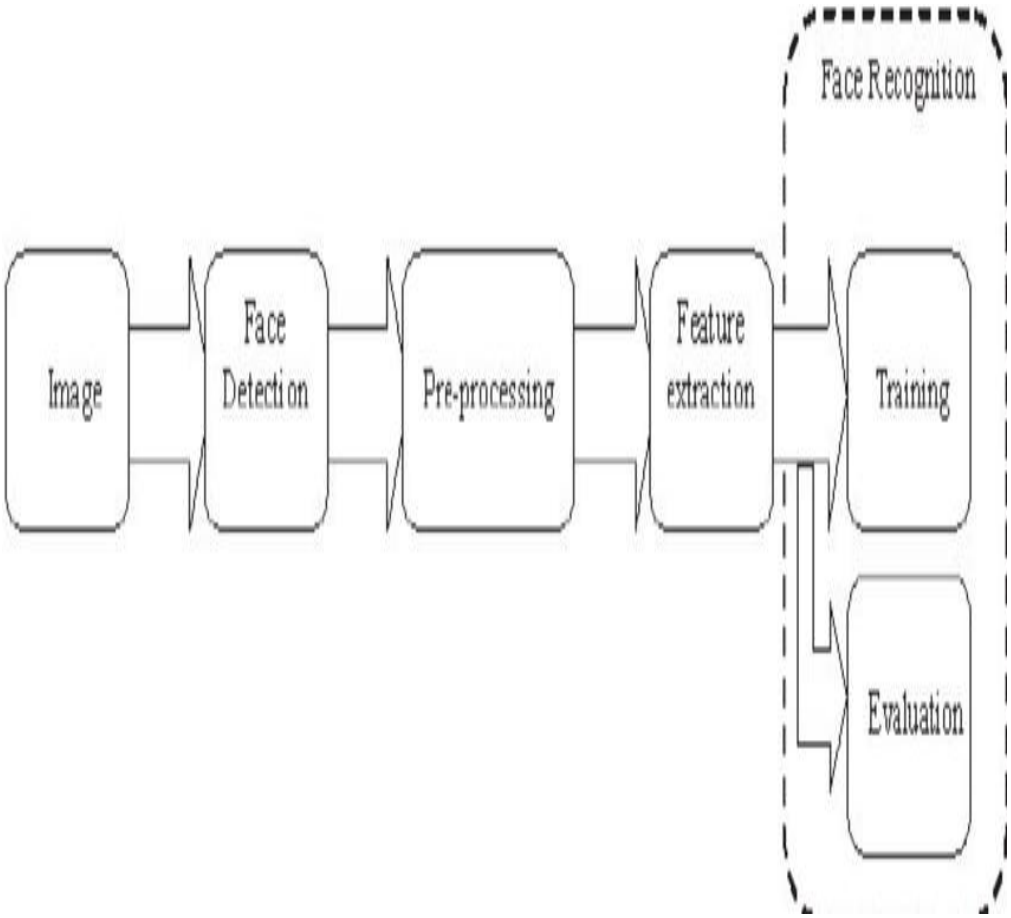


9.SYSTEM DESIGN

9.1 ARCHITECTURAL DESIGN

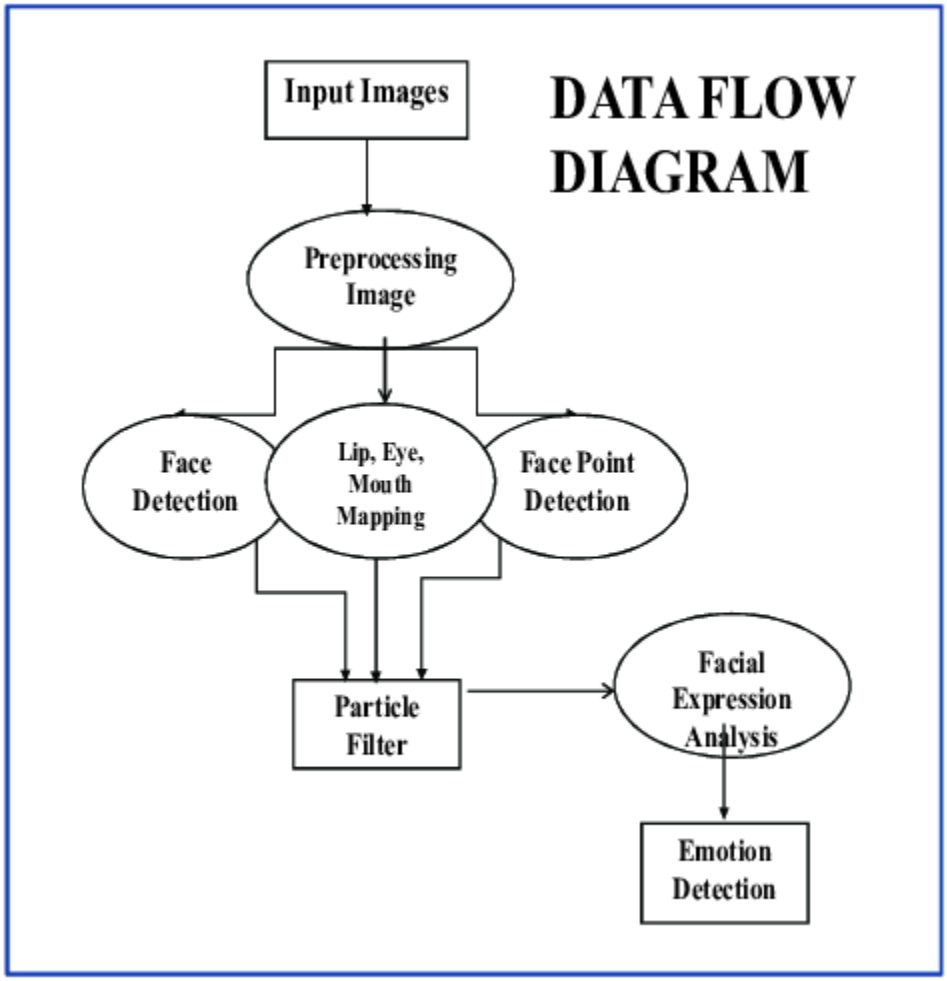
Design and implementation of a fast parallel architecture based on an improved principal component analysis (PCA) method called Composite PCA suitable for real-time face recognition is presented in this paper. The proposed architecture performs the tasks of both feature extraction and classification. Composite PCA takes in to consideration the local features of face images, which do not vary

widely between face images of the same person taken under varying expression, illumination and pose. Hence it leads to a better recognition rate than PCA. Composite PCA has more parallelism than conventional PCA and this parallelism is utilized to design an efficient architecture capable of performing real-time face recognition. The face recognition system is implemented in an FPGA environment and tested using standard databases. The system is able to recognize a person from a database of 110 images of 10 individuals in approximately 4 Ms.

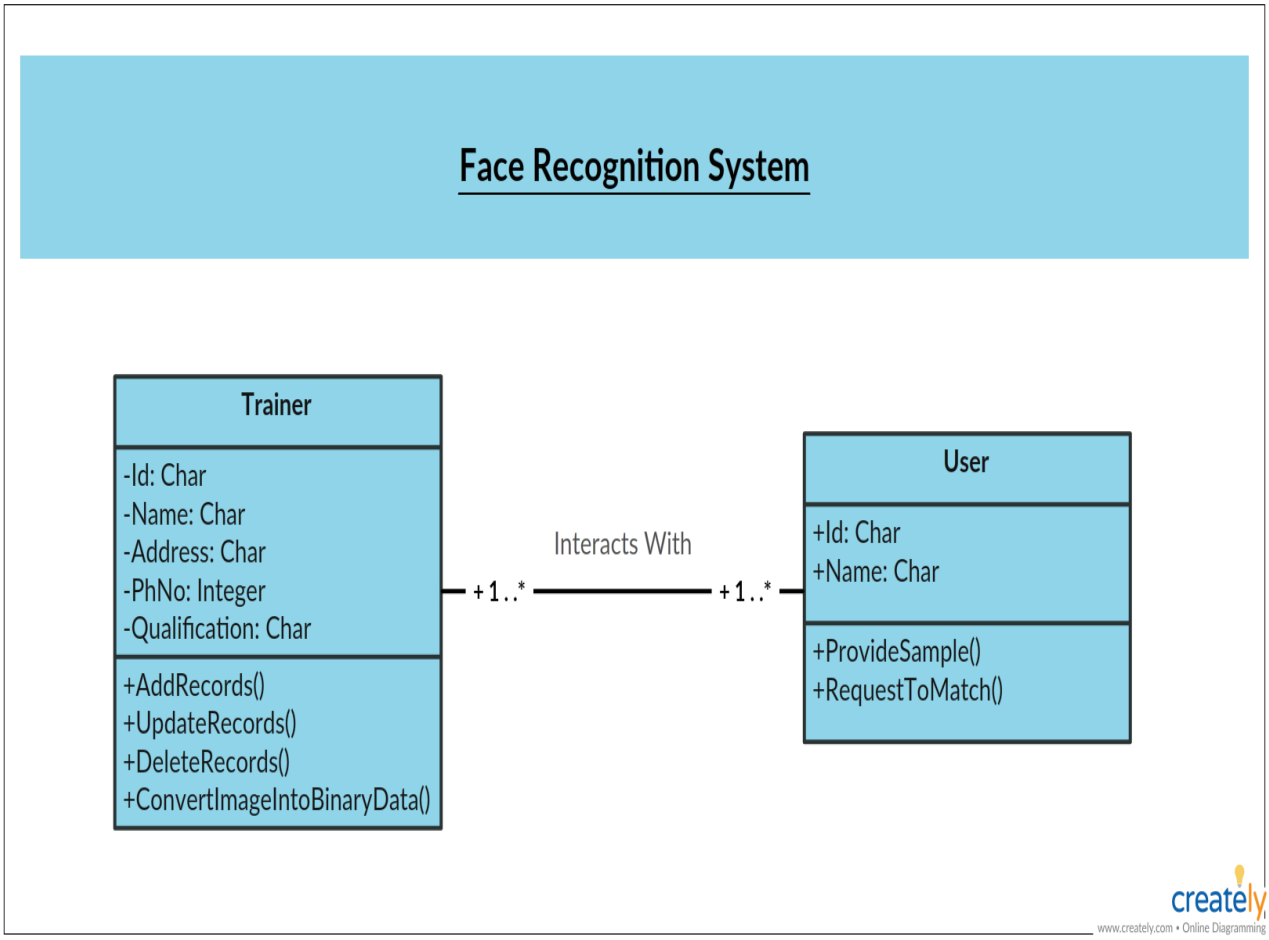


ARCHITECTURAL DESIGN OF FACE RECOGNITION

9.2 DATA FLOW DIAGRAM



9.3 UML DIAGRAM



10. CONCLUSION

In this Project, Android applications of image and face recognition using the domain Of Machine Learning is developed. In the Face Recognition application, we was Learn how to compare Two Faces in an Image and detect the face in the image is Same or Different. In the Image Recognition App, if you show the image of dog it Is going to tell you the breed of dog or if you show any fruit is it going to tell you Which fruit it Is. Similarly it can predict 1000 categories which is using Tensor Flow Lite & Android Studio.

11. FUTURE ENHANCEMENT

11.1 FACE RECOGNITION APPLICATION

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 defence 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 identification of criminals the system can be used by police force also.

11.2 IMAGE RECOGNITION APPLICATION

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image

processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence6 will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated. The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image processing app1ications. The future image processing applications of satellite based imaging ranges from planetary exploration to surveillance applications.

Using large scale homogeneous cellular arrays of simple circuits to perform image processing tasks and to demonstrate pattern-forming phenomena is an emerging topic. The cellular neural network is an implementable alternative to fully connected neural networks and has evolved into a paradigm for future imaging techniques. The usefulness of this technique has applications in the areas of silicon retina, pattern formation, etc.

APPENDIX

12.1 FACE RECOGNITION APPLICATION CODING

MainActivity.java

```
Package com.example.facerecognition;


import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;


import android.app.Activity;

import android.content.Intent;

import android.content.res.AssetFileDescriptor;

import android.graphics.Bitmap;

import android.graphics.Rect;

import android.net.Uri;

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 com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.mlkit.vision.common.InputImage;

import com.google.mlkit.vision.face.Face;

import com.google.mlkit.vision.face.FaceDetection;

import com.google.mlkit.vision.face.FaceDetector;


import org.tensorflow.lite.DataType;

import org.tensorflow.lite.Interpreter;

import org.tensorflow.lite.support.common.TensorOperator;

import org.tensorflow.lite.support.common.ops.NormalizeOp;

import org.tensorflow.lite.support.image.ImageProcessor;

import org.tensorflow.lite.support.image.TensorImage;

import org.tensorflow.lite.support.image.ops.ResizeOp;

import org.tensorflow.lite.support.image.ops.ResizeWithCropOrPadOp;
```

```
import java.io.FileInputStream;

import java.io.IOException;

import java.nio.MappedByteBuffer;

import java.nio.channels.FileChannel;

import java.util.List;


public class MainActivity extends AppCompatActivity {

    protected Interpreter tflite;

    private int imageSizeX;

    private int imageSizeY;


    private static final float IMAGE_MEAN=0.0f;

    private static final float IMAGE_STD=1.0f;


    public Bitmap oribitmap,testbitmap;

    public static Bitmap cropped;

    Uri imageuri;


    ImageView oriImage,testImage;

    Button buverify;
```

```
TextView result_text;


float [][] ori_embedding=new float[1][128];
float [][] test_embedding=new float[1][128];



@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initComponents();
}

private void initComponents() {
    oriImage=(ImageView)findViewById(R.id.image1);
    testImage=(ImageView)findViewById(R.id.image2);
    buverify=(Button)findViewById(R.id.verify);
    result_text=(TextView)findViewById(R.id.result);

    try{
```

```
            tflite=new Interpreter(loadmodelfile(this));
        }catch (Exception e){
            e.printStackTrace();
        }
        oriImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent=new Intent();
                intent.setType("image/*");
                intent.setAction(Intent.ACTION_GET_CONTENT);
                startActivityForResult(Intent.createChooser(intent,"Select
Picture"),12);
            }
        });
        buverify.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                double distance =calculate_distance(ori_embedding,test_embedding);
```

```
            if (distance < 6.0){
                result_text.setText("Result: Same Faces");
            }
            else{
                result_text.setText("Result: Different Faces");
            }
        }
    });


}
private double calculate_distance(float[][] ori_embedding,float[][]
test_embedding){
    double sum=0.0;
    for(int i=0;i<128; i++){
        sum=sum+Math.pow(ori_embedding[0][1]-test_embedding[0][1],2.0);
    }
    return Math.sqrt(sum);
}
private TensorImage loadImage(final Bitmap bitmap,TensorImage
inputImageBuffer){
```

```
        inputImageBuffer.load(bitmap);


        int cropSize=Math.min(bitmap.getWidth(),bitmap.getHeight());



        ImageProcessor imageProcessor=new ImageProcessor.Builder()
                .add(new ResizeWithCropOrPadOp(cropSize,cropSize))
                .add(new
ResizeOp(imageSizeX,imageSizeY,ResizeOp.ResizeMethod.NEAREST_NEIGH
BOR))
                .add(getPreprocessNormalizeOp())
                .build();
        return imageProcessor.process((inputImageBuffer));
    }
    private MappedByteBuffer loadmodelfile(Activity activity)throws
IOException{


        AssetFileDescriptor
fileDescriptor=activity.getAssets().openFd("Qfacenet.tflite");
        FileInputStream inputStream=new
```

```
FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel=inputStream.getChannel();
        long startoffset=fileDescriptor.getStartOffset();
        long declaredLength=fileDescriptor.getDeclaredLength();
        return
fileChannel.map(FileChannel.MapMode.READ_ONLY,startoffset,declaredLength
);


    }
    private TensorOperator getPreprocessNormalizeOp(){
        return new NormalizeOp(IMAGE_MEAN,IMAGE_STD);
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == 12 && requestCode == RESULT_OK && data!=null){
            imageuri=data.getData();
            try{
```

```
            oribitmap=
MediaStore.Images.Media.getBitmap(getContentResolver(),imageuri);
            oriImage.setImageBitmap(oribitmap);
            face_detector(oribitmap,"original");
        }catch (IOException e){
            e.printStackTrace();
        }
    }
    if(requestCode == 13 && requestCode == RESULT_OK && data!=null){
        imageuri=data.getData();
        try{

testbitmap=MediaStore.Images.Media.getBitmap(getContentResolver(),imageuri);
            testImage.setImageBitmap(testbitmap);
            face_detector(testbitmap,"test");
        }catch (IOException e){
            e.printStackTrace();
        }

    }
```

```
}
public void face_detector(final Bitmap bitmap,final String imageType){
   final InputImage image=InputImage.fromBitmap(bitmap,0);
   FaceDetector detector= FaceDetection.getClient();
   detector.process(image)
         .addOnSuccessListener(new OnSuccessListener<List<Face>>() {
            @Override
            public void onSuccess(List<Face> faces) {

               for(Face face:faces)
               {
                  Rect bounds=face.getBoundingBox();

                  cropped=Bitmap.createBitmap(bitmap,bounds.left,bounds.top,
                        bounds.width(),bounds.height());
                  get_embaddings(cropped,imageType);
               }
            }
         })
         .addOnFailureListener(new OnFailureListener() {
```

```
                @Override
                public void onFailure(@NonNull Exception e) {

Toast.makeText(getApplicationContext(),e.getMessage(),Toast.LENGTH_SHORT
).show();
                }
            });

    }
    public void get_embaddings(Bitmap bitmap,String imageType){
        TensorImage inputImageBuffer;
        float [][] embedding=new float[1][128];
        int imageTensorIndex=0;
        int [] imageShape=tflite.getInputTensor(imageTensorIndex).shape();
        imageSizeX=imageShape[1];
        imageSizeY=imageShape[2];

        DataType
imageDataType=tflite.getInputTensor(imageTensorIndex).dataType();
        inputImageBuffer=new TensorImage(imageDataType);
```

```
        inputImageBuffer=loadImage(bitmap,inputImageBuffer);


        tflite.run(inputImageBuffer.getBuffer(),embedding);


        if(imageType.equals("original")){
            ori_embedding=embedding;
        }else if(imageType.equals("test")){
            test_embedding=embedding;
        }


    }
}
```

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"
```

```
android:padding="20dp"

tools:context=".MainActivity">



<TextView

    android:id="@+id/textView"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Face Verification"

    android:textColor="#EC0C58"

    android:textSize="30sp"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHorizontal_bias="0.497"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintVertical_bias="0.06" />


<TextView

    android:id="@+id/result"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Result :"
    android:textSize="24sp"
    android:textColor="#EC0C58"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.045" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="159dp"
    android:layout_height="73dp"
    android:text="Original Image"
    android:textColor="#EC0C58"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.062"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/result"
    app:layout_constraintVertical_bias="0.067" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="165dp"
    android:layout_height="78dp"
    android:text="   Test Image"
    android:textColor="#EC0C58"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.407"
    app:layout_constraintStart_toEndOf="@+id/textView3"
    app:layout_constraintTop_toBottomOf="@+id/result"
    app:layout_constraintVertical_bias="0.072" />
```

```
<ImageView
    android:id="@+id/image1"
    android:layout_width="151dp"
    android:layout_height="130dp"
    android:layout_marginTop="48dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/image2"
    app:layout_constraintHorizontal_bias="0.404"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3"
    app:layout_constraintVertical_bias="0.068"
    app:srcCompat="@drawable/gr" />

<ImageView
    android:id="@+id/image2"
    android:layout_width="156dp"
    android:layout_height="132dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView4"
```

```
        app:layout_constraintVertical_bias="0.301"
        app:srcCompat="@drawable/gr" />

    <Button
        android:id="@+id/verify"
        android:layout_width="144dp"
        android:layout_height="48dp"
        android:layout_marginTop="356dp"
        android:text="Verify"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/result"
        app:layout_constraintVertical_bias="0.0" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

IMAGE RECOGNITION APPLICATION CODING

MainActivity.kt

```
package com.example.imagerecognition
```

```
import android.content.Intent
import android.graphics.Bitmap
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.MediaStore
import android.provider.MediaStore.Images
import android.view.ActionMode
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import com.example.imagerecognition.ml.MobilenetV110224Quant
import org.tensorflow.lite.DataType
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer

class MainActivity : AppCompatActivity() {

    lateinit var bitmap:Bitmap
```

```
    lateinit var imgview:ImageView

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

        imgview=findViewById(R.id.imageView)
        val fileName="label.txt"
        val inputString=application.assets.open(fileName).bufferedReader().use {
it.readText() }
        var townList=inputString.split("\n")
        var tv:TextView=findViewById(R.id.textView)

        var select: Button =findViewById(R.id.button)

        select.setOnClickListener(View.OnClickListener {

            var intent:Intent = Intent(Intent.ACTION_GET_CONTENT)
            intent.type="image/*"
```

```
            startActivityForResult(intent, 100)


        })

        var predict:Button=findViewById(R.id.button2)
        predict.setOnClickListener(View.OnClickListener {

            var resized:Bitmap=Bitmap.createScaledBitmap(bitmap,224,224,true)
            val model = MobilenetV110224Quant.newInstance(this)

            val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1, 224, 224,
3), DataType.UINT8)

            var tbuffer=TensorImage.fromBitmap(resized)
            var byteBuffer=tbuffer.buffer

            inputFeature0.loadBuffer(byteBuffer)

            val outputs = model.process(inputFeature0)
            val outputFeature0 = outputs.outputFeature0AsTensorBuffer
```

```
            var max=getMax(outputFeature0.floatArray)

            tv.setText(townList[max])

            model.close()


        }

    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        imgview.setImageURI(data?.data)

        var uri: Uri?= data ?.data
        bitmap= MediaStore.Images.Media.getBitmap(this.contentResolver,uri)
```

```
}

fun getMax(arr:FloatArray):Int{

    var ind=0
    var min=0.0f


    for(i in 0..100)
    {
        if (arr[i]>min)
        {
            ind=i
            min=arr[i]
        }


    }
    return ind
```

```
    }


}
```

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">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="370dp"
        android:layout_height="231dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

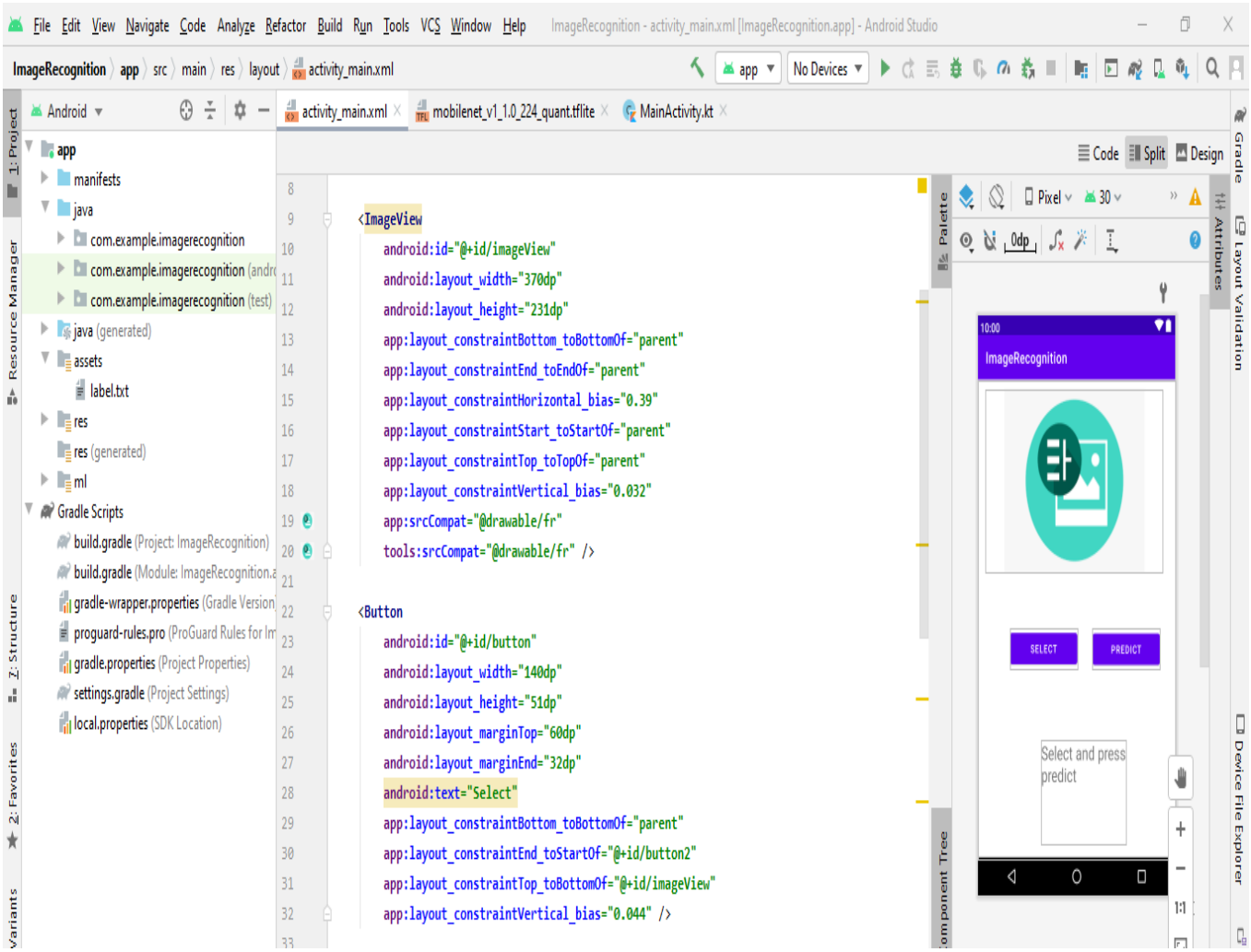
```
    app:layout_constraintHorizontal_bias="0.39"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.032"
    app:srcCompat="@drawable/fr"
    tools:srcCompat="@drawable/fr" />

<Button
    android:id="@+id/button"
    android:layout_width="140dp"
    android:layout_height="51dp"
    android:layout_marginTop="60dp"
    android:layout_marginEnd="32dp"
    android:text="Select"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button2"
    app:layout_constraintTop_toBottomOf="@+id/imageView"
    app:layout_constraintVertical_bias="0.044" />

<Button
```

```
    android:id="@+id/button2"
    android:layout_width="140dp"
    android:layout_height="51dp"
    android:layout_marginTop="72dp"
    android:layout_marginEnd="32dp"
    android:text="Predict"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageView" />

<TextView
    android:id="@+id/textView"
    android:layout_width="177dp"
    android:layout_height="132dp"
    android:text="Select and press predict"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.564"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageView"
```

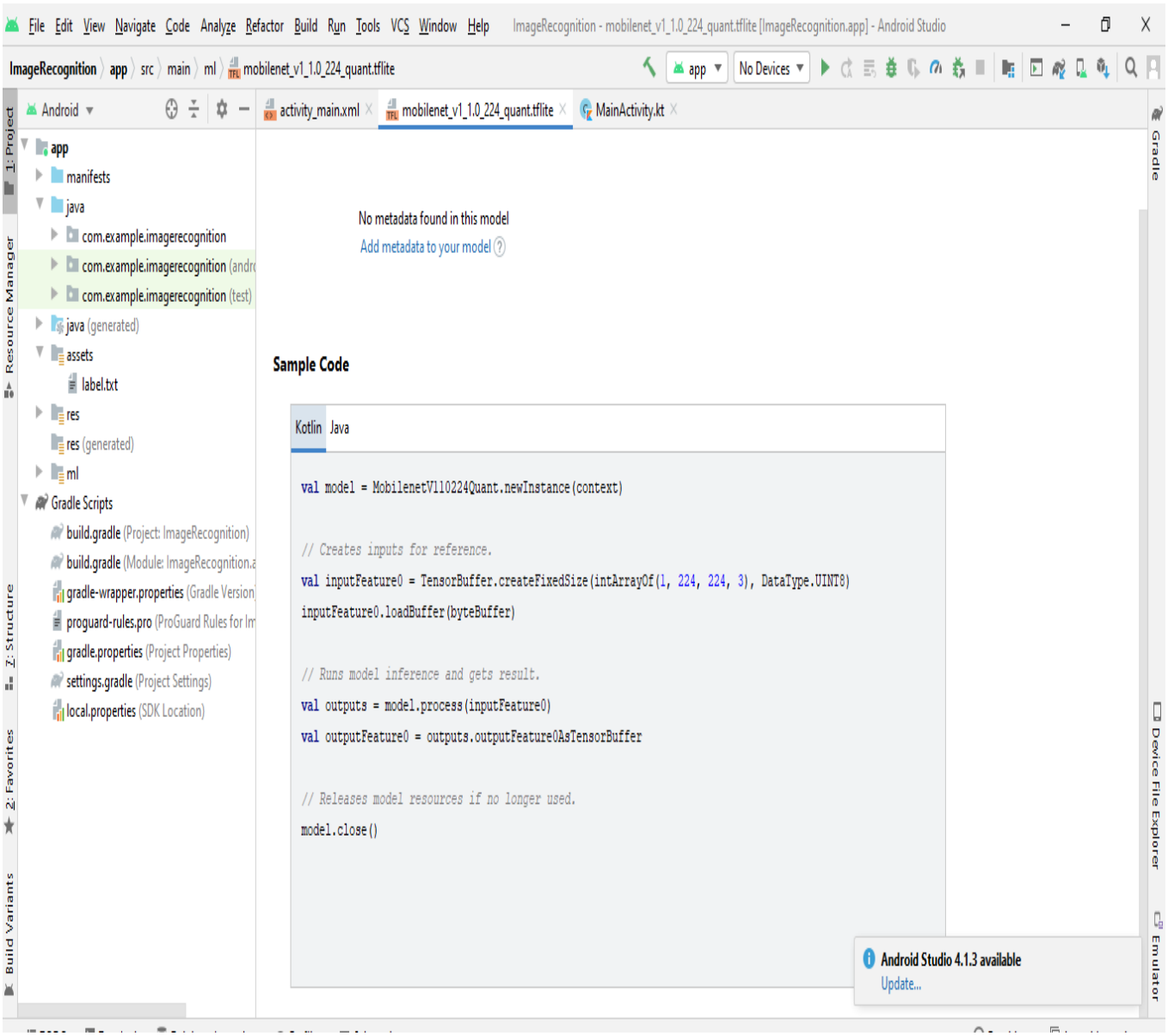
```
        app:layout_constraintVertical_bias="0.929" />

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

12.2 SCREENSHOTS

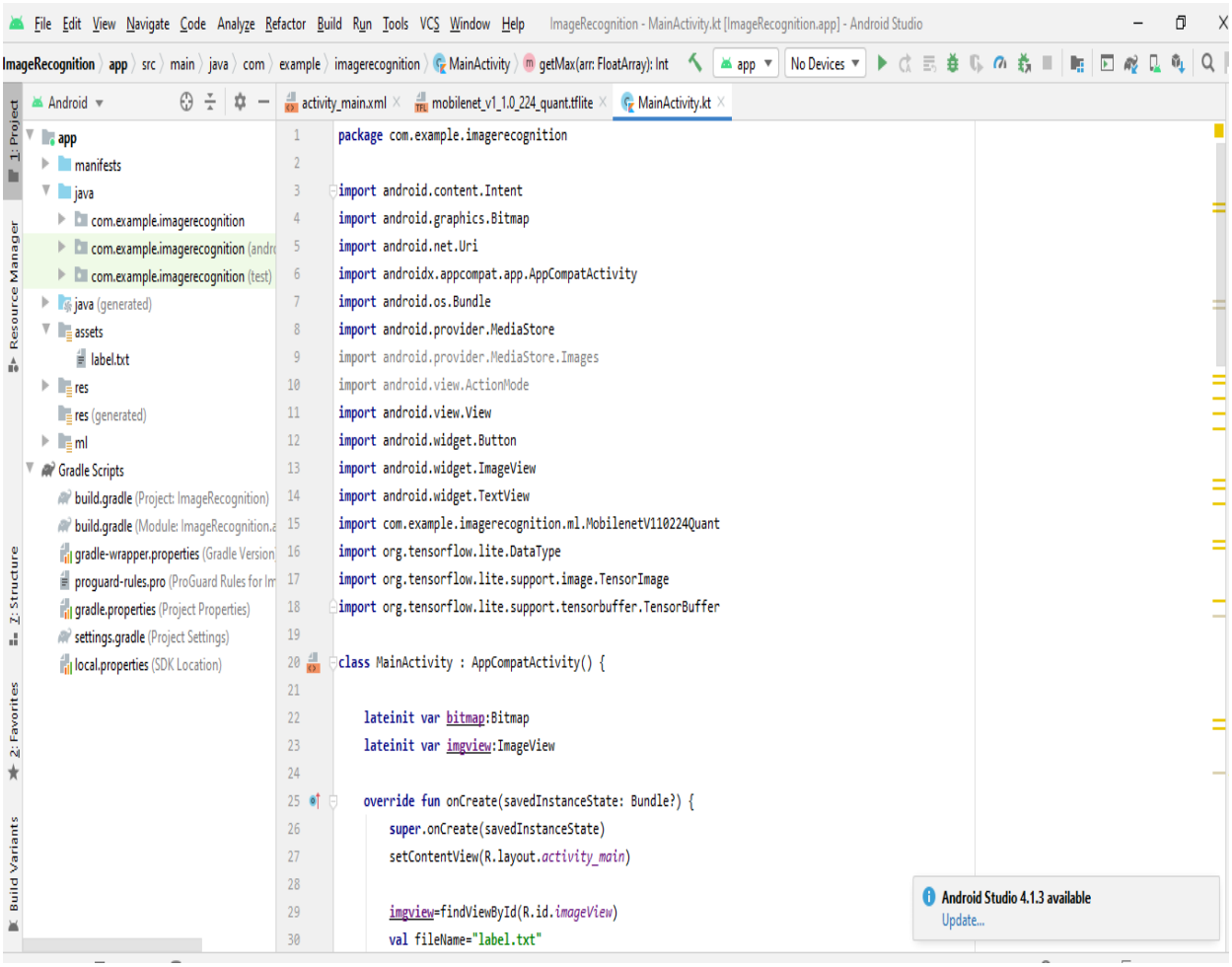
IMAGE RECOGNITION APPLICATION



DESIGNING PAGE 1

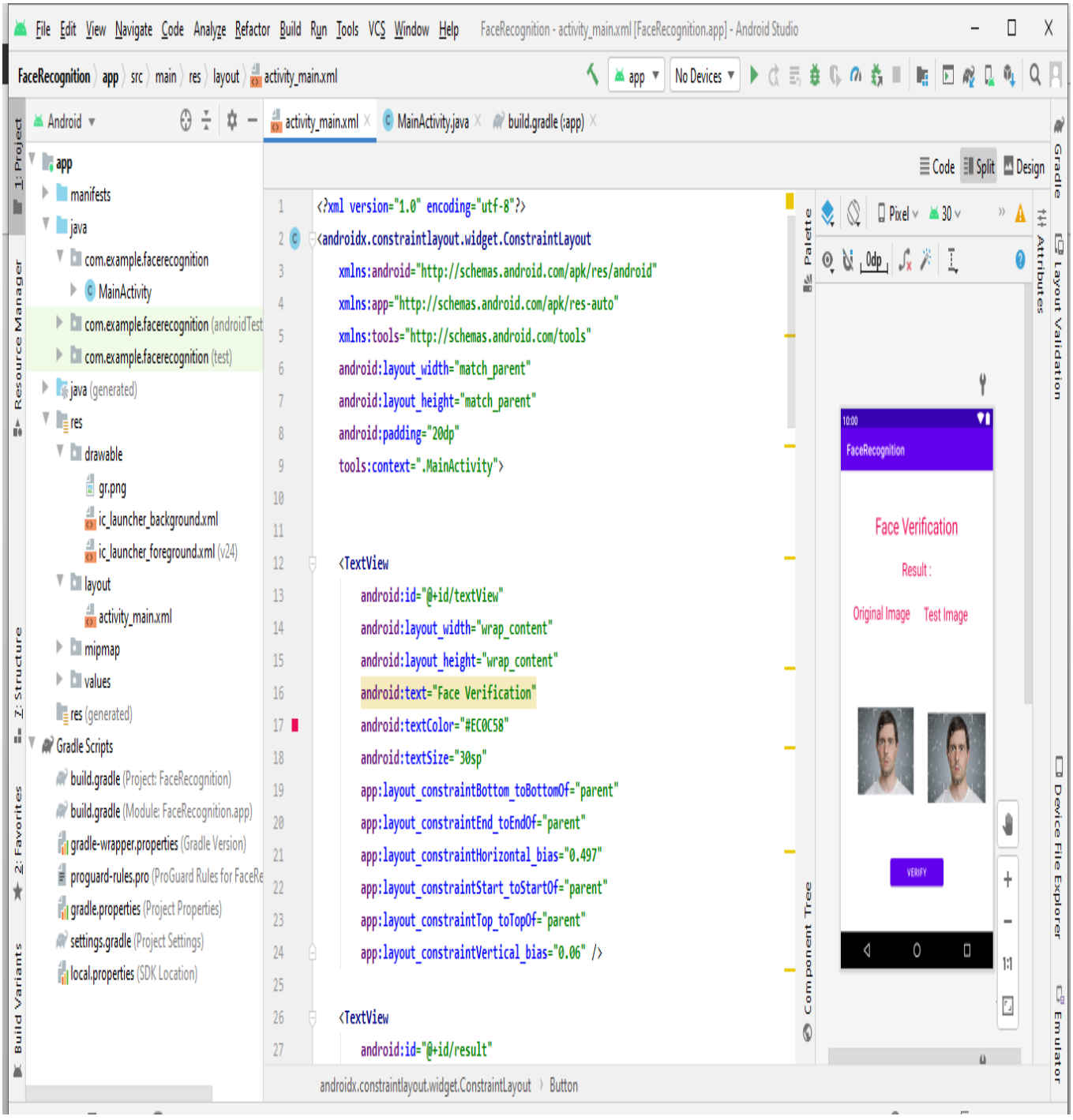


TENSOR FLOW WITH KOTLIN PAGE

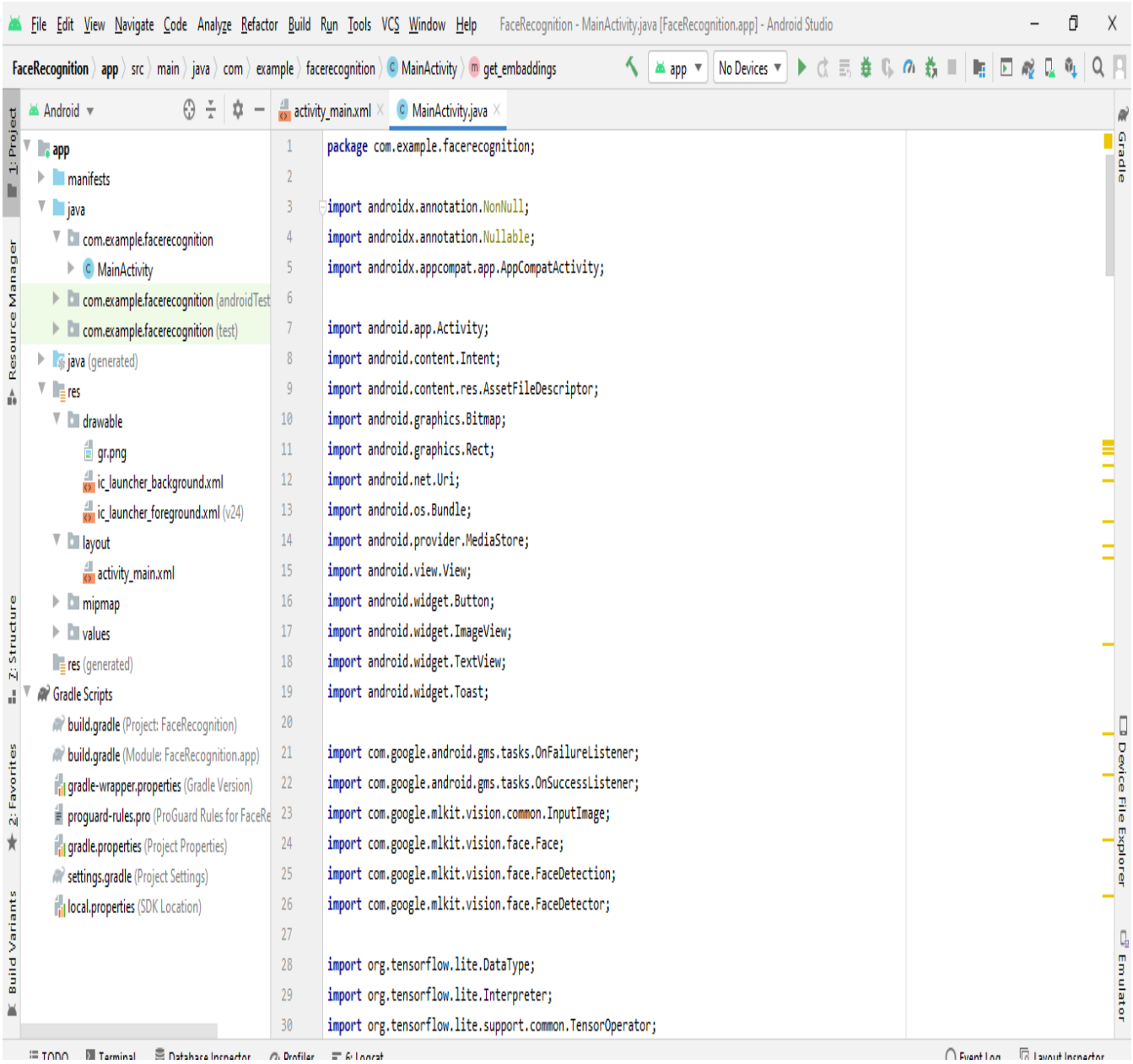


MAIN PROGRAM PAGE 1

FACE RECOGNITION APPLICATION



MAIN DESIGN PAGE 2



MAIN PROGRAM PAGE 2

13. REFERENCES

- Marian Schluter, Carsten Niebuhr, Jan Lehr, Jorg Kruger, “Vision Based Identification Service For Remanufacturing Sorting”, Elsevier, Procedia Manufacturing Volume 21, pp. 384 – 391, 2018.
- Barret Zoph, Vijay Vasudevan, Jonathan Shlens, Quoc V.Le, ”Learning Transferable Architectures For Scalable Image Recognition”, The IEEE Conference on Computer Vision and Pattern Recognition, pp. 8697-8710, 2018.
- C. V. Arulkumar, P. Vivekanandan, "Multi-feature based automatic face identification on kernel eigen spaces (KES) underunstable lighting conditions", Advanced Computing and Communication Systems 2015 International Conference on. IEEE, 2015.
- V. Arulkumar. "An Intelligent Technique for Uniquely Recognising Face and Finger Image Using Learning Vector Quantisation (LVQ)- based Template Key Generation," International Journal of Biomedical Engineering and Technology 26, no. 3/4 (February 2, 2018): 237-49.
- Yueqi Duan, Jiwen Lu, Jianjiang Feng, Jie Zhou, “Context-Aware Local Binary Feature Learning For Face Recognition”, IEEE Transaction on Pattern Analysis and Machine Intelligence, Volume 40, Issue 5, pp. 1139-1153, 2017