RAMAIAH
Institute of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

V Semester – NON-CIE 20 marks component
IMAGE PROCESSING WITH PYTHON (ECOE02)

TERM: Sep - Dec 2022

PROJECT REPORT 2

PROJECT TEAM MEMBERS

Sl. No	USN	Name
1	1MS20IS049	HAMMAD FEROZ
2	1MS20CS107	SANSKAR R GONDKAR
3	1MS20CS129	TEJAS HEGDE

SUPERVISED BY FACULTY

Dr. S Sethu Selvi

RAMAIAH
Institute of Technology

Contents

RAMAIAH
Institute of Technology

Implementation Of The Facial Recognition Attendance System

Description:

The facial-recognition based attendance system can be divided into three parts:

i. Classroom image extractor.
ii. Face recognition module
iii. Attendance Marker

Each of the following component will be discussed in detail.

i. Classroom image extractor:

The classroom image extractor is an android app that uploads the file selected by the user to a specific Google Drive folder for the next part of the program to work on. The folder to which it uploads to has been made available to everyone to upload to, thus it has no access problems from any android device.

It is divided into different components:

1. The app itself (apk)
2. The deployable app script running in the server (Google app script)

The app was made using MIT App Inventor. It is a very streamlined and intuitive way of creating GUI apps for Android. This particular app uses Google drive API to upload the images.
The way the app functions is shown in the diagrams below:

Fig. Initializing the URL of the app script

The above block initializes the app and sets the text WebAppUrl to the address the script app that is running in the server, since it'll be used later on.

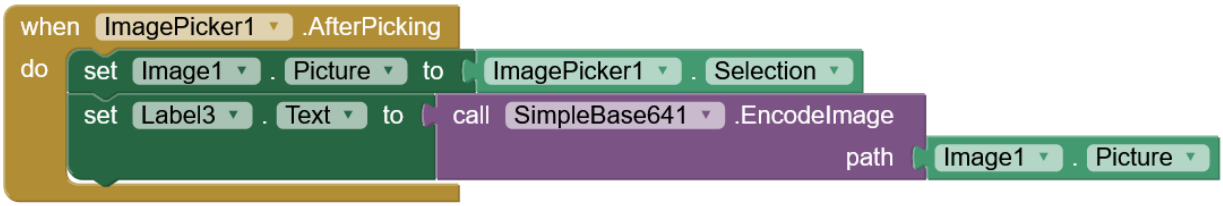

Fig. Selecting an image to encode and then upload

The first button with the text: 'Select an image' functions as an image picker, which when pressed will take the user to the gallery of the smartphone they're using. The first dark green block selects the image and sets the displayed image to that image. The second dark green block encodes the photo to base64 so that it can be uploaded later. It is stored in a hidden label called Label 3.

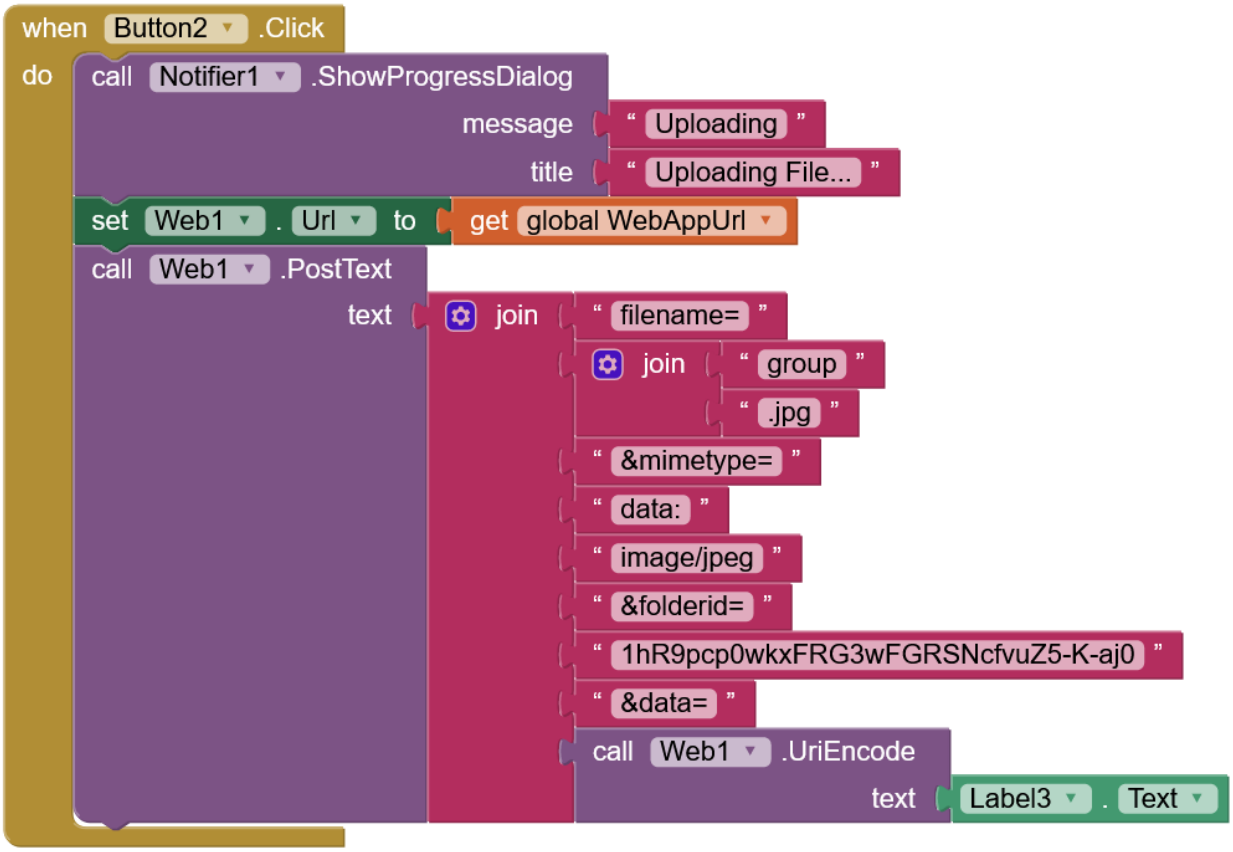

Fig. Uploading the encoded data to the specific folder in Google Drive

Button 2's task is to upload the selected and encoded image. For that it does the following in order as shown by the block:

- It shows a screen in which it says the file is being uploaded.
- It sets the upload URL to the one we had specified before
- It sets the text to be uploaded to be a join of the filename to be set, datatype, folder to which it has to upload to and the data which we had stored which has to be encoded to Uri to be sent. It does this via Post method

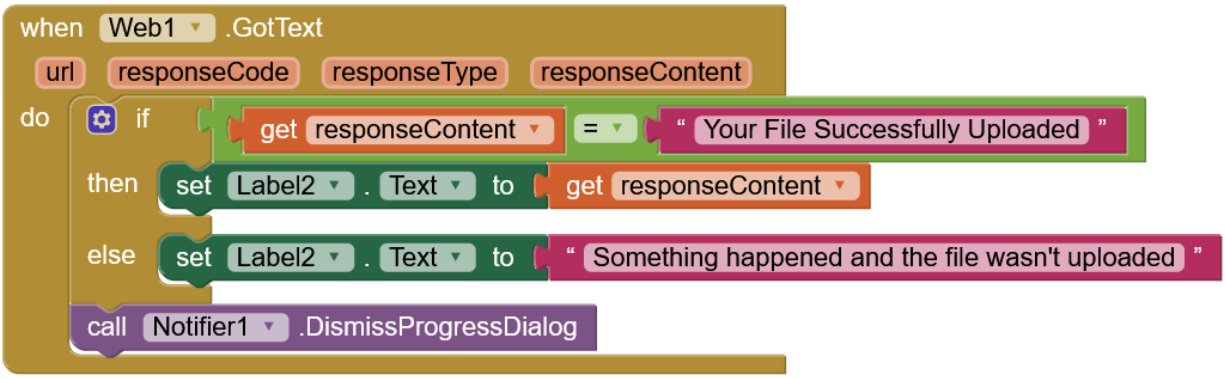

Fig. Setting response messages on the screen depending on whether or not the upload was successful

Now after the upload has been requested, the app receives a response which it indicates whether or not it was successful. Regardless of whether or not it was successful, the progress circle for uploading has to be dismissed, since its part is over and this is done in the last box.

ii. Face recognition module:

This module is the most important part of the attendance system. The core of the module is the **DLib** face recognition library, one of the most popular packages for face recognition applications. The Dlib library is a C++ library. This automatically ensures compatibility for all systems i.e.; it is a cross platform library. Moreover, it is open-source. Hence, it has been extensively used for various face recognition applications including embedded ones. An article in Analytics Vidhya has used OpenCV along with dlib for face recognition. Dlib also offers a wide range of functionality across a number of machine learning sectors, including classification and regression,

numerical algorithms such as quadratic program solvers, an array of image processing tools, and diverse networking functionality, among many other facets.

Dlib uses two different face recognition algorithms for facial capture:

- HoG + Linear SVM:

The Histogram of Oriented Gradients (HoG) + Linear Support Vector Machine (SVM) algorithm in Dlib offers very fast recognition of front-on faces, but has limited capabilities in terms of recognizing face poses at acute angles (such as CCTV footage, or casual surveillance environments where the subject is not actively participating in the ID process).

It also supports passport-style profile faces, though with very little margin for error (faces pointing up or down, etc.). HoG + SVM is suitable for constrained situations where the sensor can expect a direct and unobstructed view of the participant's face, such as ATM and mobile framework ID systems, as well as mobile traffic surveillance recognition systems, where cameras are able to obtain a straight profile shot of drivers.

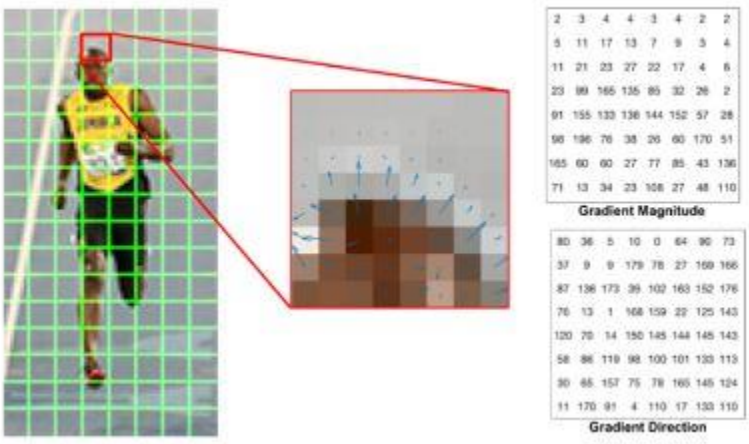

Fig. The HoG algorithm

- Max-Margin (MMOD) CNN face detector:

MMOD is a robust and reliable, GPU-accelerated face detector that leverages a convolutional neural network (CNN), and is far more capable of capturing faces at obscure angles and in challenging conditions, suiting it for casual surveillance and urban analysis.

MMOD is not a distinct alternative to HoG + Linear SVM, but rather can be applied to HoG itself, or to any bag-of-visual-word model, which treats discovered pixel groupings as explorable entities for potential labeling — including the identification of faces.

For implementation in Python, we make use of Geitgey's CUDA-capable python library, **face_recognition**, which has demonstrated 99.83% accuracy on the University of Massachusetts' Labeled Faces in the Wild benchmark dataset. The project encodes Dlib's face captures into 128 data points per face, resulting in unique parameters for the hash of each face across a variety of different photos. Subsequently a Support Vector Machine (SVM) is trained on the derived faces via scikit-learn, resulting in an agile FR model that can run with minimal latency in the right conditions.

Working:

Before discussing the working of each component of the attendance system, we need to talk about the prerequisites which are as follows:

1. The teacher must possess a **smartphone** for acquiring image of the classroom. The higher the image quality, better the results.
2. The teacher must use a **google form** of a standard template to collect student name, USN and a clear and recent photograph of his/her face, and then create a **google spreadsheet**.
3. Since the system is run on **google colab**, which uses dlib with enabled **CUDA** configuration, it is necessary to **change the runtime type to GPU.**

The step for creating a spreadsheet is required because extracting responses from the form is not only cumbersome, but also requires enabling of the forms api in the Google Cloud Console, which will be difficult for the teacher. Instead, we use the google spreadsheet for which the access method, i.e., the **gspread api** is very simple and requires no other setup. The biggest advantage of the spreadsheet is that information is present in the form of a Relational Database and hence information retrieval becomes simplified.

i. Classroom image extractor:

The home page of the app interface is made to look like this:

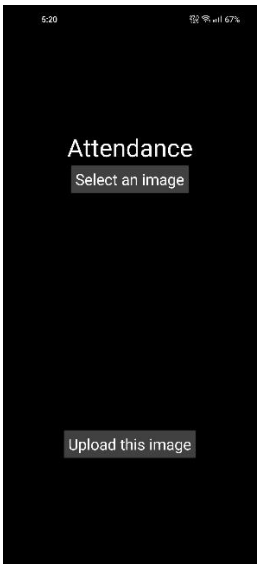

Fig. The home page of the app

It consists of the following parts:

1. Button 1 saying ‘Select an image’: This button opens up an image selector via a gallery which contains previously captured images

2. Button 2 saying ‘Upload this image’: This button uploads the photo selected by the image selector to the specified Drive folder.

After the image has been selected it looks like this (it displays selected image in the middle):

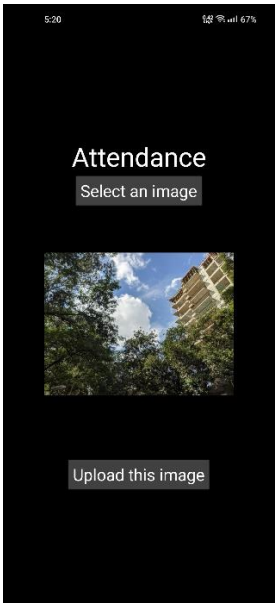

Fig. The application GUI after an image has been selected

After the button ‘Upload this image’ has been pressed the image is uploaded. While that happens this screen is shown:

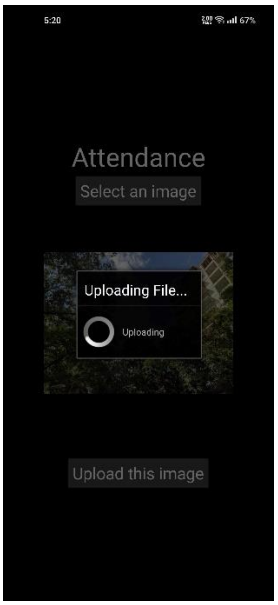

Fig. Loading screen when the image is being uploaded

After the upload has been done this screen is shown:

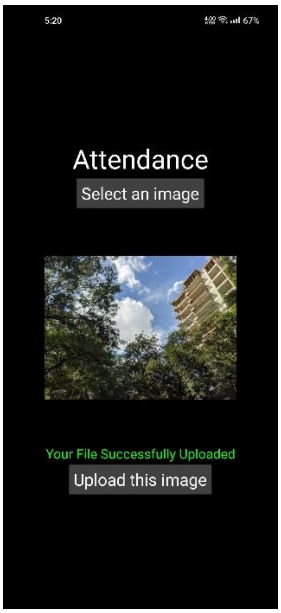

Fig. App GUI when uploading is successful

As we can see the image has been uploaded to the correct folder in Drive:

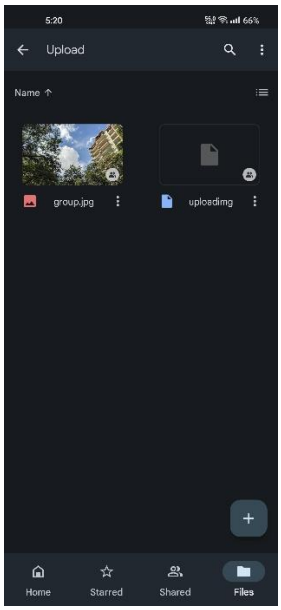

Fig. The photo is correctly uploaded to Drive

The file ‘uploadimg’ on the side of it is the deployed app script. It consists of the following code:

```
function doPost(e) {

  var data = Utilities.base64Decode(e.parameters.data);

  var blob = Utilities.newBlob(data, e.parameters.mimetype, e.parameters.filename);

  DriveApp.getFolderById('1hR9pcp0wkxFRG3wFGRSNcfvuZ5-K-aj0').createFile(blob);

  return ContentService.createTextOutput("Your File Successfully Uploaded");

}

```

Fig. App script code to handle upload to the correct folder

In this code, the function receives the data e in text form. It is then decoded (base64) to the variable data. This is done because uploading in image formats directly is not supported via the Drive API which this application uses, and thus the image had to encoded to base64 before uploading in the app. The variable blob is what contains the entire details of the file, including its extension and name. The next line stores the ID of the folder to upload to.

The next line then creates the file with the properties of blob. If this was successful then it proceeds to the next line which returns the string: 'Your File Successfully Uploaded' to the app which it displays in green at the bottom. If it doesn't upload the file, then exception handling occurs and some other string is returned to the app for which it'll say: 'Something Happened and the file wasn't uploaded'

ii. Face recognition module:

The module first extracts the form responses present in the google spreadsheet. These responses include the Name, USN and link to the photograph of the student, present in google drive. We basically obtain the spreadsheet as a **Pandas DataFrame**. Pandas library in python allows to manipulate data in the form of tables, known as Dataframes. After a bit of preprocessing, we obtain the table consisting of the responses.

Note that only for the first time the system is initialized, it has to download every photo from the respective links in the google drive and extract the face encoding of each person. Hence it takes a long time. However, we persist the face encoding in Google Drive using **pickle** library in python. Thus, the text time the system is used, we need not perform the face encoding extraction process again. So, the execution is faster.

Going a bit deeper into the download part; this process is performed automatically by the **gdown** library in python which parses the photograph links in the table and downloads them.

Now, we must understand what a face encoding means. It is basically a **numpy** array that represent the feature vector of a face. As discussed earlier, we are basically using Dlib's MMOD method which records 128 data points per face.

So basically, we extract the face encoding of each person listed in the table. This list of face encodings forms the **known face encoding**.

Now, we load the image of the classroom. The face_recognition module automatically identifies faces in the photo and extracts the encoding of each face. The list of face encodings obtained from this step is called **unknown face encoding**.

Note that the face_recognition library uses CUDA i.e., GPU acceleration for the MMOD+CNN method of Dlib to capture and extract face_encoding. The GPU acceleration helps to increase the speed as well as the accuracy of the feature extraction process.

Now, that we have the known face encoding and unknown face encoding, we just take a face in the unknown face encoding, compare it against entries in the known face encoding and find the face having the closest match. The closeness is basically a number between 0 and 1. If it falls within a certain **threshold,** the student corresponding to that entry is marked present otherwise absent.

Note that we have provided an option to update the known face encoding. We have also provided an option to verify the attendance process by comparing the person's face with the name with which he/she has been marked. This has been performed using **matplotlib.**

We then pass the marked attendance to the attendance marker for actual recording of attendance.

Given below is a diagram summarizing the face recognition module of the attendance system.

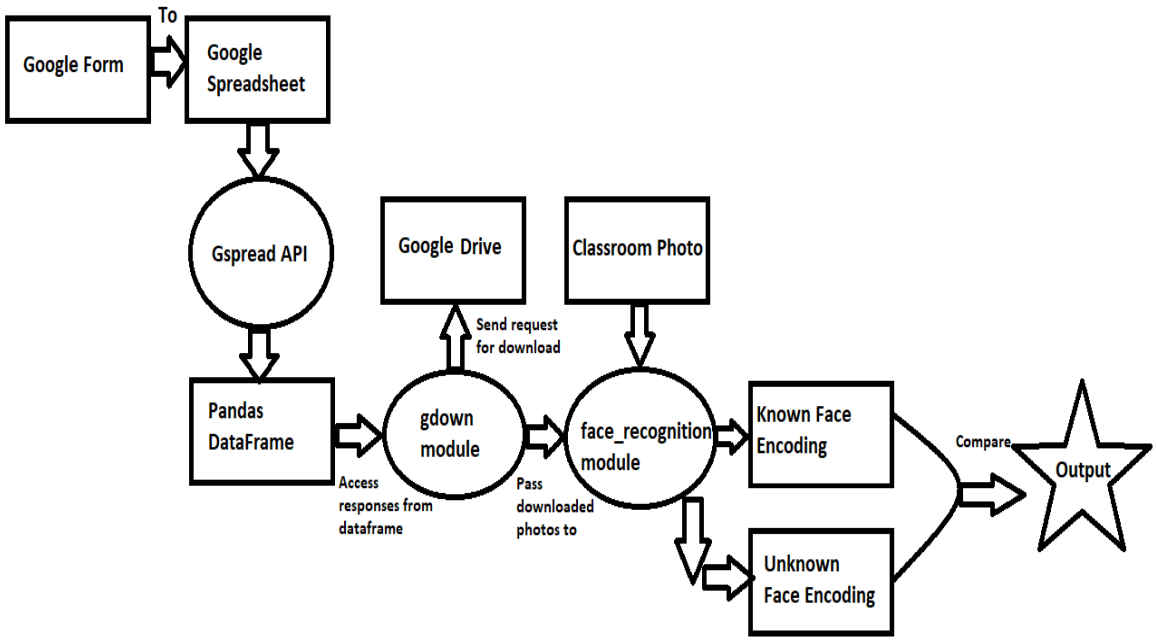

Fig. The Face Recognition Module Of The Attendance System

Results:

The following picture was given to the attendance system.

Fig. Classroom Photo given to the attendance system

The results for the system were obtained as follows:

Total No of Students=38

Total No of participants=19

Detected=17 Obscure=2

Correctly Recognized=10

Incorrectly Recognized:1

Unrecognized:6

The output of our code:

```
Not able to recognize unknown face  1
Not able to recognize unknown face  9
Not able to recognize unknown face  11
Not able to recognize unknown face  14
Not able to recognize unknown face  15
Not able to recognize unknown face  16
ARNAV A RAJESH   is Present
Vinay B Sahani   is Absent
```

```
MADHUMITA   is Present
Swastik Puri  is Present
Tejas Hegde   is Present
Sanskar R Gondkar  is Absent
ARKODEEP KOLEY  is Present
Chakradhar  is Absent
ANKESH MOHAN NAIK   is Absent
Kushal R   is Absent
Ninaad P S  is Absent
GANESH SHEKHAR NAIK  is Present
SUBHAJJI SUHAS   is Absent
Shiya Singh  is Absent
SKANDA S KUMAR   is Present
Bharath M B  is Absent
Abhinav S  is Absent
Manas Kulshrestha  is Absent
SHAWN DANIEL RODRIGUES  is Absent
R Jayanth Jadhav  is Absent
Supreeth KG  is Present
Pratham Jain   is Absent
BHARGAVA M  is Absent
Snehal Vats  is Absent
Rohan Sriharsha  is Absent
Aman bhatnagar  is Absent
Amaan Adil   is Present
Aditya Anand   is Absent
Haarish Anandan  is Absent
Pratham M  is Present
dhanashekar  is Absent
Naina Y  is Absent
Raghvendra Goundi   is Absent
Shivam Sharma   is Absent
Vigya awasthi  is Absent
Karthik Gowda K A  is Absent
Rishi Priyadarshi  is Absent
Hammad Feroz   is Present
```

Observations:

The photo chosen for the testing is an extreme one. The photo is extreme because of the following conditions:

i. Large distance between students and camera

ii. Some students have their face behind another student's face

iii. Some students have worn masks

iv. The left side of the classroom (with respect to camera) is less illuminated compared to the right side

v. Some students in corners have positioned faces in such a way that their entire face is not visible; rather only one part of face is visible.

With regard to these conditions, the system has understandably not detected two faces(one wearing mask, other behind a student).

The system has not recognized 6 faces. In that two faces are behind someone else's face. Three faces have not been detected mostly because of their large distance from the camera. One student has not turned his head towards the camera causing only one part of his face to get captured and hence he went undetected.

The only incorrect recognition is possibly because of the large variation in the faces provided in the database and the classroom photo combined with his distance from the camera. The system understandably mistook this classroom photo of that person with a very similar looking student who actually was not present.

Thus, with respect to this regard, the system has done a fantastic job of recognizing 10 faces correctly at one shot. This is something that most systems struggle to do; they just work a person or two. The accuracy can be improved if:

i. Better quality camera is used
ii. Students are brought closer to the camera
iii. Students position their face correctly towards the camera
iv. Students provide photos which are similar to their classroom avatar.
v. The room is uniformly illuminated
vi. Students don't block each other's heads or wear masks.

Overall, the system is an efficient one that asks lesser data, resources and time compared to other systems while providing a respectable accuracy, at, very importantly, free of cost.