Face recognition Model For WMU Data Science Grad students using deep learning

Satish Gollu
Satish.gollu@wmich.edu
win:966588920

Madhurima Biswas
madhurima.biswas@wmich.edu
win:458122293

Advisor
Dr. Ajay Gupta

ABSTRACT :
There are lot of advancements in the field of face recognition over the time span and to train model to work requires a longer hours but in recent times there are few models have been introduced with the concept of transfer learning.in this paper we present a model called Face recognition library developed by Adam Geitgey. This library learns with a single image and then creates multiple images with the single image and then it learns a mapping from the images we trained and compares with the current image for the face similarity.
Our method uses wmu data science students faces to train the model collaborating with cloud big data application to deploy and maintain the model.

INTRODUCTION:

To start off with the research we used convolutional neural networks trained directly optimizing the trained face encodings and ml optimization techniques and using cloud to deploy. Down We have Provided a brief of the entire method and technology.

Face recognition problems commonly fall into one of two categories:

Face Verification : "Is this the claimed person?"
Examples:
you can pass through customs by letting a system scan your passport and then verifying that the person carrying the passport are the correct person.
A mobile that unlocks using your face is also using face verification
Face verification is a 1:1 matching problem.

Face Recognition: Instead of 1-to-1 matching, where the task is to verify if you are who you say you are, this requires the harder 1-to-N matching where we have to check if you're any one of N persons.
Example: face recognition of employees entering the office without needing to otherwise identify themselves.
Face Recognition is a 1:N matching problem.

Our application can recognize face and keep a count recognised faces on a basis of name time and date for future reference .Deep learning method is used as a basic foundation of the app . python is used a programming language , docker is to containerize the app and then deploying the model in GCP.

Fig showing the most important face encoding points that plays a vital role to recognize faces

Prior to beginning of the work on the app big data characteristics and the stages of the Big Data Characteristics are analysed .

Scope:

The initial scope of the Face Recognition Model is to identify the Faces of WMU Grad Students and then keep a track of the name date and time for future references. Although due to limitations of the GCP the Face recognition model is only trained with couple of faces . Thus the app is limited to very few faces right now but in future we can add a good number of variables for an effective output

Big Data Characteristics :
Our project applies the 5 v's in following way.

Volume : The application provides an web interface live streaming facial recognition platform for the admin (not for the users) because of security constraints. The live will be recorded and stored through GCP storage separately to accommodate the higher data volume demand. Face Recognition model is well designed to deal with high level and different kinds of image formats and image sizes

Velocity: This application detects the user and generated an real time data consists of user name, date and time for future reference. the app servers output the user verified status depending on the user trained data .the app is well positioned to scale and serve multiple number of faces at a time (in a single frame)

Variety: The application uses variety of data, initial pictures to train the model and then resized and transforming the input data into multiple images. This data comes under different sizes and different format like jpg/png/ heic etc

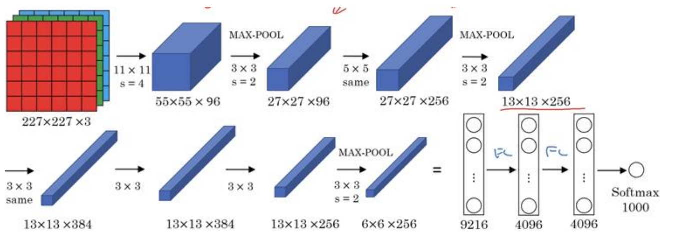
Veracity: The data used in training are the real photos of the persons so they are trusted high. However the photos can be tampered which will lead to identity mismatch .This model consider all the angels and different postures of your face and recognize it. the model does not provide any information about the admin to the users .

Value: The Face Recognition Model provides value to university department by recognizing the correct faces and can give the detailed count of how many students present or walk by through the classrooms or desired places .Further value can be added by optimizing the model which helps to prevent the academic misconducts.

DATA SET , METHOD AND TECHNOLOGIES:

Data Set:
To train the model WMU graduate student pictures are used

Method :
The model is a deep convolutional neural network trained via face recognition library. a triplet loss function that encourages vectors for the same identity to become more similar (smaller distance), whereas vectors for different identities are expected to become less similar (larger distance). The focus on training a model to create embeddings directly, These face embeddings were then used as the basis for training classifier system on standard face recognition benchmark dataset, achieving then-state-of-the-art results, here we want to ensure that an image of a specific student/user is closer to all other images of the same person to any other images of other person Which draws all possible triplets in the training set, smaller triplets results in slower convergence, even though they still pass through the network, hard triplets contributes to improve the model



Training the deep learning model will take weeks and months so, we used transfer learning method to train and develop the core face recognition algorithm using convolutional neural networks and after successful training and testing of model, now it will be deployed on GCP

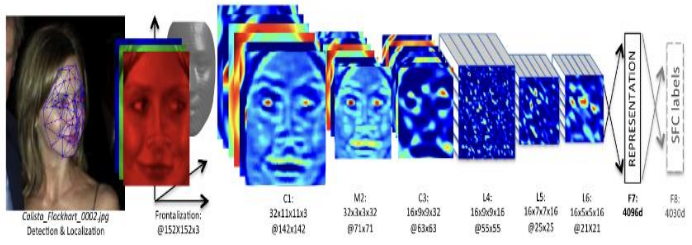
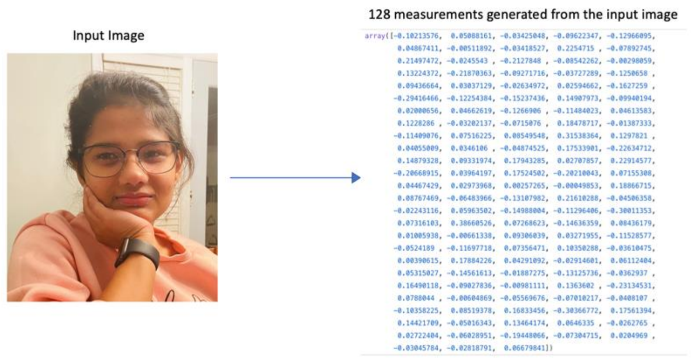


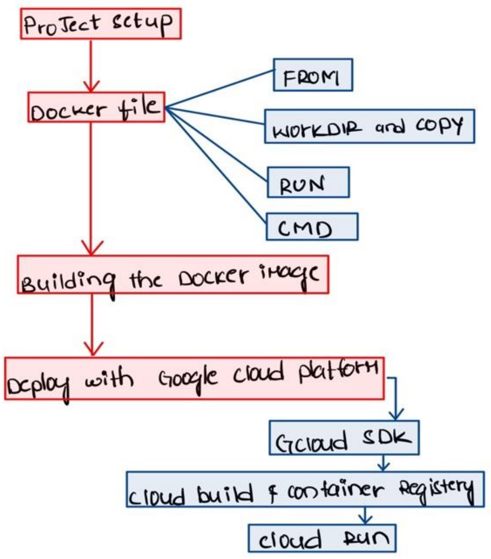
Figure: outline of the deep face architecture A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by 3 locally-connected layers and 2 fully-connected layers. Colours illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layer

After running the model we will get face encoding which will store an array of 128 values which are Euclidean distances for each comparison face. The below image will represents the face encodings of an trained image.

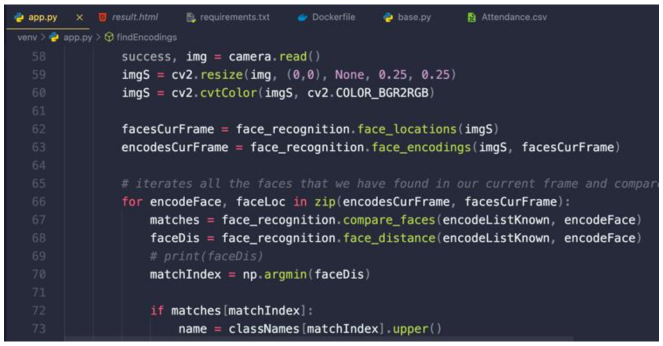


```
array([-0.10213576,  0.05088161, -0.03425048, -0.09622347, -0.12966095,
        0.04867411, -0.00511892, -0.03418527,  0.2254715 , -0.07892745,
        0.21497472, -0.0245543 , -0.2127848 , -0.08542262, -0.00298059,
        0.13224372, -0.21870363, -0.09271716, -0.03727289, -0.1250658 ,
        0.09436664,  0.03037129, -0.02634972,  0.02594662, -0.1627259 ,
       -0.29416466, -0.12254384, -0.15237436,  0.14907973, -0.09940194,
        0.02000656,  0.04662619, -0.1266906 , -0.11484023,  0.04613583,
        0.1228286 , -0.03202137, -0.0715076 ,  0.18478717, -0.01387333,
       -0.11409076,  0.07516225,  0.08549548,  0.31538364,  0.1297821 ,
        0.04055009,  0.0346106 , -0.04874525,  0.17533901, -0.22634712,
        0.14879328,  0.09331974,  0.17943285,  0.02707857,  0.22914577,
       -0.20668915,  0.03964197,  0.17524502, -0.20210043,  0.07155308,
        0.04467429,  0.02973968,  0.00257265, -0.00049853,  0.18866715,
        0.08767469, -0.06483966, -0.13107982,  0.21610288, -0.04506358,
       -0.02243116,  0.05963502, -0.14988004, -0.11296406, -0.30011353,
        0.07316103,  0.38660526,  0.07268623, -0.14636359,  0.08436179,
        0.01005938, -0.00661338,  0.09306039,  0.03271955, -0.11528577,
       -0.0524189 , -0.11697718,  0.07356471,  0.10350288, -0.03610475,
        0.00390615,  0.17884226,  0.04291092, -0.02914601,  0.06112404,
        0.05315027, -0.14561613, -0.01887275, -0.13125736, -0.0362937 ,
        0.16490118, -0.09027836, -0.00981111,  0.1363602 , -0.23134531,
        0.0788044 , -0.00604869, -0.05569676, -0.07010217, -0.0408107 ,
       -0.10358225,  0.08519378,  0.16833456, -0.30366772,  0.17561394,
        0.14421709, -0.05016343,  0.13464174,  0.0646335 , -0.0262765 ,
        0.02722404, -0.06028951, -0.19448066, -0.07304715,  0.0204969 ,
       -0.03045784, -0.02818791,  0.06679841])
```

After successful training and testing of the model the below are the steps that involved for the deployment these are the key components in order to collaborate and deploy in the google cloud.

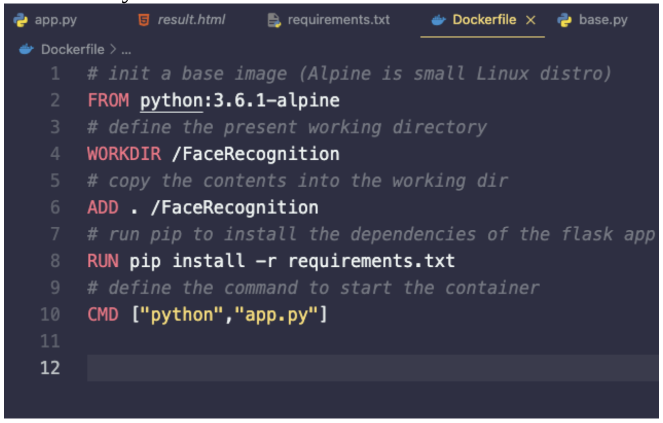

ProJect setup
Docker file
FROM
WORKDIR and COPY
RUN
CMD
Building the Docker image
Deploy with Google cloud platform
Gcloud SDK
cloud build & container Registery
cloud Run

The great benefit of Docker is the level of support for containers on Cloud platforms, such as Google Cloud (GCP). which we used in this project and built an application package it into a container with Docker, and deploy it across the globe with GCP. We will take our developed model and a required Python API, package it with Docker, and deploy it with GCP. The trained deep learning model python code file and the data set which contains the training data.



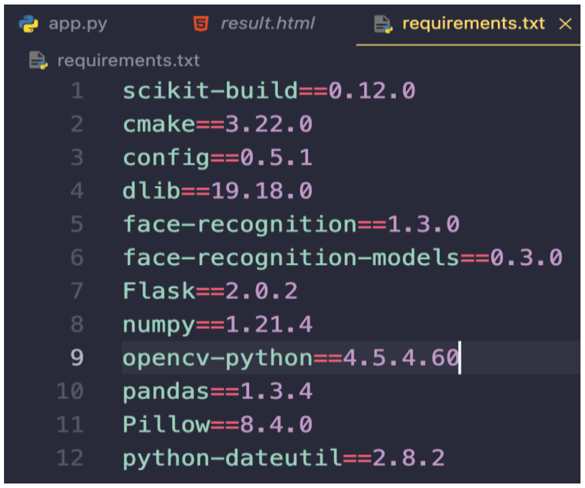
```python
        success, img = camera.read()
        imgS = cv2.resize(img, (0,0), None, 0.25, 0.25)
        imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

        facesCurFrame = face_recognition.face_locations(imgS)
        encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

        # iterates all the faces that we have found in our current frame and compar
        for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            # print(faceDis)
            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:
                name = classNames[matchIndex].upper()
```

We store our code in a directory called gcp-api under the name app.py. Alongside our script,
A **Dockerfile** — the instruction manual for Docker has been created, which helps to deploy in any running os to successfully run the model



```dockerfile
# init a base image (Alpine is small Linux distro)
FROM python:3.6.1-alpine
# define the present working directory
WORKDIR /FaceRecognition
# copy the contents into the working dir
ADD . /FaceRecognition
# run pip to install the dependencies of the flask app
RUN pip install -r requirements.txt
# define the command to start the container
CMD ["python","app.py"]


```

requirements.txt — a set of Python modules and libraries for our Docker file to install, docker will install while dockerizing the model and it will create an image file with load containers containing all of these packages.



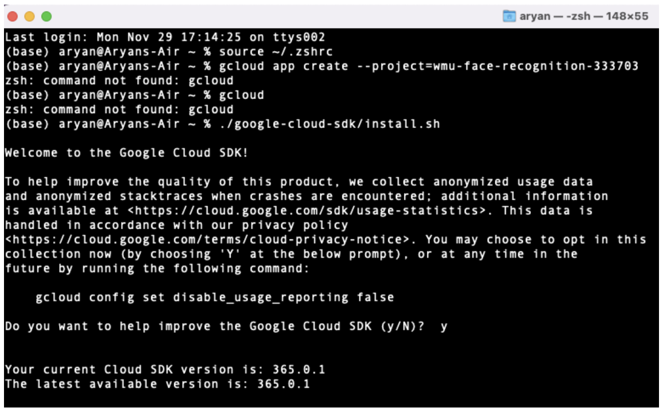
```
scikit-build==0.12.0
cmake==3.22.0
config==0.5.1
dlib==19.18.0
face-recognition==1.3.0
face-recognition-models==0.3.0
Flask==2.0.2
numpy==1.21.4
opencv-python==4.5.4.60
pandas==1.3.4
Pillow==8.4.0
python-dateutil==2.8.2
```

The Docker file is our container building blueprint. It tells Docker exactly how to and where to rearrange our model and files in a way that produces a self-contained application. It's like building a ship.
We will specify the number of CPUs, memory & hard-disk size. Etc. that requires for an app.

Later, we will also give the blueprint to our builder (Google Build), who will construct container for us.

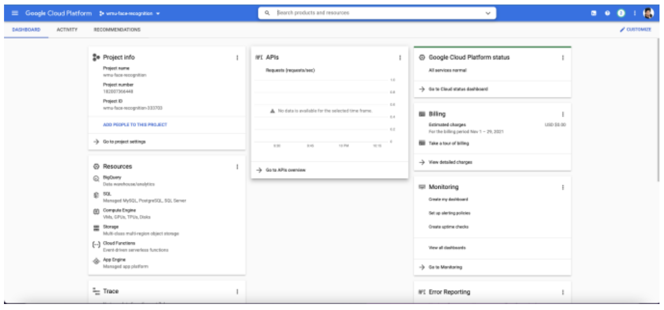
Deploy with Google Cloud Platform

The below are the steps we need to take to deploy our container to the Cloud. We
Download the Google Cloud SDK, which we will use to Build our container with Cloud Build.



```
Last login: Mon Nov 29 17:14:25 on ttys002
(base) aryan@Aryans-Air ~ % source ~/.zshrc
(base) aryan@Aryans-Air ~ % gcloud app create --project=wmu-face-recognition-333703
zsh: command not found: gcloud
(base) aryan@Aryans-Air ~ % gcloud
zsh: command not found: gcloud
(base) aryan@Aryans-Air ~ % ./google-cloud-sdk/install.sh

Welcome to the Google Cloud SDK!

To help improve the quality of this product, we collect anonymized usage data
and anonymized stacktraces when crashes are encountered; additional information
is available at <https://cloud.google.com/sdk/usage-statistics>. This data is
handled in accordance with our privacy policy
<https://cloud.google.com/terms/cloud-privacy-notice>. You may choose to opt in this
collection now (by choosing 'Y' at the below prompt), or at any time in the
future by running the following command:

    gcloud config set disable_usage_reporting false

Do you want to help improve the Google Cloud SDK (y/N)?  y


Your current Cloud SDK version is: 365.0.1
The latest available version is: 365.0.1
```

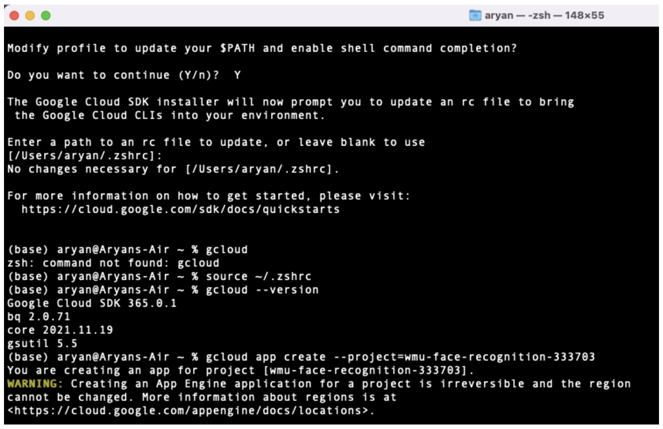
Upload the container to GCP's Container Registry and Deployed it with Cloud Run. created a new project and save the project id
The project id is: wmu-face-recognition-333703



Project info
Project name
wmu-face-recognition
Project number
182007366448
Project ID
wmu-face-recognition-333703

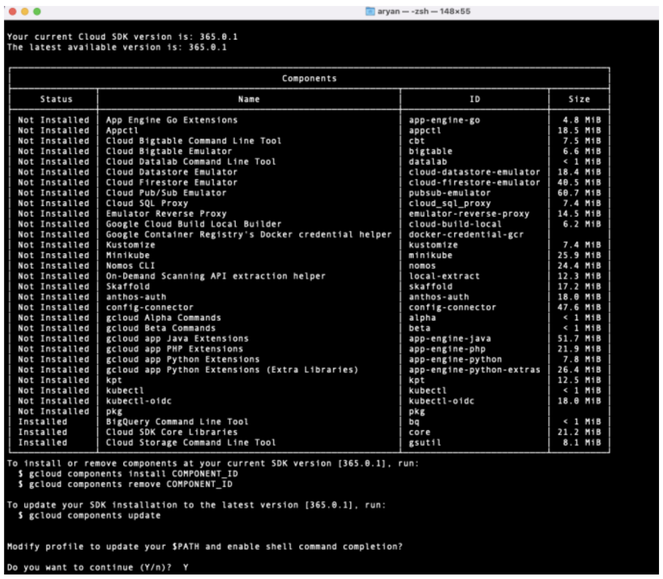
Estimated charges USD $0.00
For the billing period Nov 1 – 29, 2021

created the application by specifying the region (region is permanent) and deploying the language in which the app runs Mapping to a custom domain names if we want For security we will add proxy servers/firewall



```
Modify profile to update your $PATH and enable shell command completion?

Do you want to continue (Y/n)?  Y

The Google Cloud SDK installer will now prompt you to update an rc file to bring
 the Google Cloud CLIs into your environment.

Enter a path to an rc file to update, or leave blank to use
[/Users/aryan/.zshrc]:
No changes necessary for [/Users/aryan/.zshrc].

For more information on how to get started, please visit:
  https://cloud.google.com/sdk/docs/quickstarts


(base) aryan@Aryans-Air ~ % gcloud
zsh: command not found: gcloud
(base) aryan@Aryans-Air ~ % source ~/.zshrc
(base) aryan@Aryans-Air ~ % gcloud --version
Google Cloud SDK 365.0.1
bq 2.0.71
core 2021.11.19
gsutil 5.5
(base) aryan@Aryans-Air ~ % gcloud app create --project=wmu-face-recognition-333703
You are creating an app for project [wmu-face-recognition-333703].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.
```

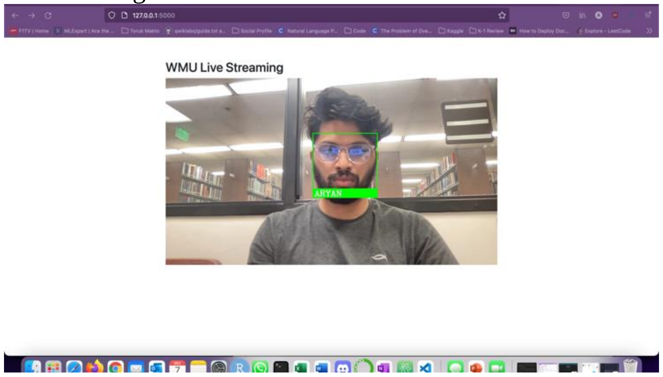
Installing all required components in gcp to run successfully and to keep checking the health.



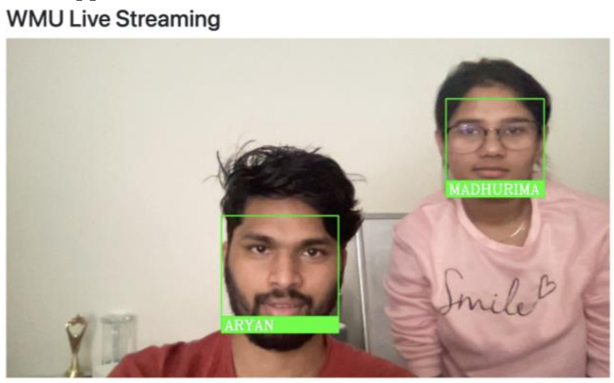
```
Your current Cloud SDK version is: 365.0.1
The latest available version is: 365.0.1

┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────────┐
│                                                    Components                                                    │
├───────────────┬──────────────────────────────────────────────────────┬──────────────────────────────┬──────────┤
│     Status    │                         Name                         │              ID              │   Size   │
├───────────────┼──────────────────────────────────────────────────────┼──────────────────────────────┼──────────┤
│ Not Installed │ App Engine Go Extensions                             │ app-engine-go                │  4.8 MiB │
│ Not Installed │ Appctl                                               │ appctl                       │ 18.5 MiB │
│ Not Installed │ Cloud Bigtable Command Line Tool                     │ cbt                          │  7.5 MiB │
│ Not Installed │ Cloud Bigtable Emulator                              │ bigtable                     │  6.6 MiB │
│ Not Installed │ Cloud Datalab Command Line Tool                      │ datalab                      │  < 1 MiB │
│ Not Installed │ Cloud Datastore Emulator                             │ cloud-datastore-emulator     │ 18.4 MiB │
│ Not Installed │ Cloud Firestore Emulator                             │ cloud-firestore-emulator     │ 40.5 MiB │
│ Not Installed │ Cloud Pub/Sub Emulator                               │ pubsub-emulator              │ 60.7 MiB │
│ Not Installed │ Cloud SQL Proxy                                      │ cloud_sql_proxy              │  7.4 MiB │
│ Not Installed │ Emulator Reverse Proxy                               │ emulator-reverse-proxy       │ 14.5 MiB │
│ Not Installed │ Google Cloud Build Local Builder                     │ cloud-build-local            │  6.2 MiB │
│ Not Installed │ Google Container Registry's Docker credential helper │ docker-credential-gcr        │          │
│ Not Installed │ Kustomize                                            │ kustomize                    │  7.4 MiB │
│ Not Installed │ Minikube                                             │ minikube                     │ 25.9 MiB │
│ Not Installed │ Nomos CLI                                            │ nomos                        │ 24.4 MiB │
│ Not Installed │ On-Demand Scanning API extraction helper             │ local-extract                │ 12.3 MiB │
│ Not Installed │ Skaffold                                             │ skaffold                     │ 17.2 MiB │
│ Not Installed │ anthos-auth                                          │ anthos-auth                  │ 18.0 MiB │
│ Not Installed │ config-connector                                     │ config-connector             │ 47.6 MiB │
│ Not Installed │ gcloud Alpha Commands                                │ alpha                        │  < 1 MiB │
│ Not Installed │ gcloud Beta Commands                                 │ beta                         │  < 1 MiB │
│ Not Installed │ gcloud app Java Extensions                           │ app-engine-java              │ 51.7 MiB │
│ Not Installed │ gcloud app PHP Extensions                            │ app-engine-php               │ 21.9 MiB │
│ Not Installed │ gcloud app Python Extensions                         │ app-engine-python            │  7.8 MiB │
│ Not Installed │ gcloud app Python Extensions (Extra Libraries)       │ app-engine-python-extras     │ 26.4 MiB │
│ Not Installed │ kpt                                                  │ kpt                          │ 12.5 MiB │
│ Not Installed │ kubectl                                              │ kubectl                      │  < 1 MiB │
│ Not Installed │ kubectl-oidc                                         │ kubectl-oidc                 │ 18.0 MiB │
│ Not Installed │ pkg                                                  │ pkg                          │          │
│ Installed     │ BigQuery Command Line Tool                           │ bq                           │  < 1 MiB │
│ Installed     │ Cloud SDK Core Libraries                             │ core                         │ 21.2 MiB │
│ Installed     │ Cloud Storage Command Line Tool                      │ gsutil                       │  8.1 MiB │
└───────────────┴──────────────────────────────────────────────────────┴──────────────────────────────┴──────────┘
To install or remove components at your current SDK version [365.0.1], run:
  $ gcloud components install COMPONENT_ID
  $ gcloud components remove COMPONENT_ID

To update your SDK installation to the latest version [365.0.1], run:
  $ gcloud components update


Modify profile to update your $PATH and enable shell command completion?

Do you want to continue (Y/n)?  Y
```

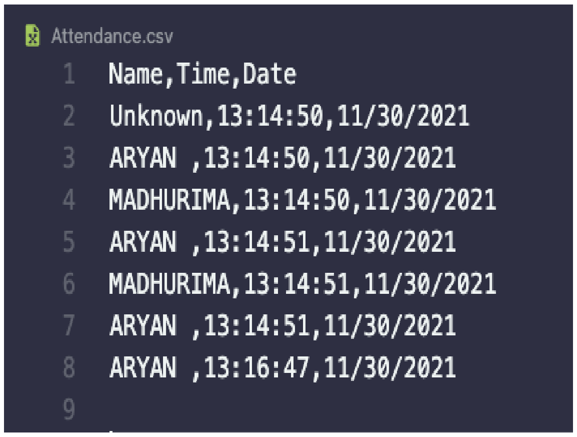
TECHNOLOGIES:
Convolutional neural networks and face recognition library to train the model and language used in the face recognition model is python and flask API frame work is used to create a responsive webpage layout. Docker tool for dockerizing model and google cloud to integrate the model and to deploy and maintain the model.

DEMONSTRATION:
By running the model, it will open a web page and will start live scanning of faces



while scanning it will also detect multiple faces and simultaneously it will record the data, like data and time of user that appeard in as excel sheet. Which is connect to GCP





```
Attendance.csv
1  Name,Time,Date
2  Unknown,13:14:50,11/30/2021
3  ARYAN ,13:14:50,11/30/2021
4  MADHURIMA,13:14:50,11/30/2021
5  ARYAN ,13:14:51,11/30/2021
6  MADHURIMA,13:14:51,11/30/2021
7  ARYAN ,13:14:51,11/30/2021
8  ARYAN ,13:16:47,11/30/2021
9
```

FUTURE SCOPE AND EXPAND USE CASES:

Expanding the face recognition model is very much possible if we can globalize the app. Face recognition app plays a big role in security .In universities it can be used as a security of labs to keep a count of the people entering the lab also can be used as a attendance system .In financial Institution in face recognition model can save both fraud and time . Also this model comes handy when then corporation have to deal with lost password. It is also useful in ATM terminals . With the further expansion of the model it can save a lot of time in Video monitoring and also in big data perspective For scaling and maintain the app we will also use GCP Kubernetes. Kubernetes allows us to use our existing Docker containers and workloads.

REFERENCES:

1. https://cs.wmich.edu/gupta/teaching/cs6100/lectureNotesBigData/Tweather-TermProjectPDF_cs6100F20_GoodProjectSample.pdf
2. image reference : https://miro.medium.com/max/276/1*LnxfkkD-HcjOUANomPTFkA.png
3. https://towardsdatascience.com/how-to-deploy-docker-containers-to-the-cloud-b4d89b2c6c31
4. 2. Abhishek Thakur: Deep Learning Model on GCP Platform (APP Engine)
5. 3. Andrew NG : Convolutional neural networks
6. FaceNet: A Unified Embedding for Face Recognition and Clustering (Schroff, Kalenichenko & Philbin, 2015)
7. 5. DeepFace: Closing the Gap to Human-Level Performance in Face Verification (Taigman, Yang, Ranzato & Wolf)
8. 6. How to Develop a Face Recognition System Using FaceNet in Keras (Jason Brownlee, 2019)
9. 7. https://face-recognition.readthedocs.io/en/latest/readme.html (official face-recognition library)

Group member Effort:
Complete Project work load has been shared equally between the group members.
From data acquisition to model deployment

Satish Gollu- 50%
Madhurima Biswas – 50%