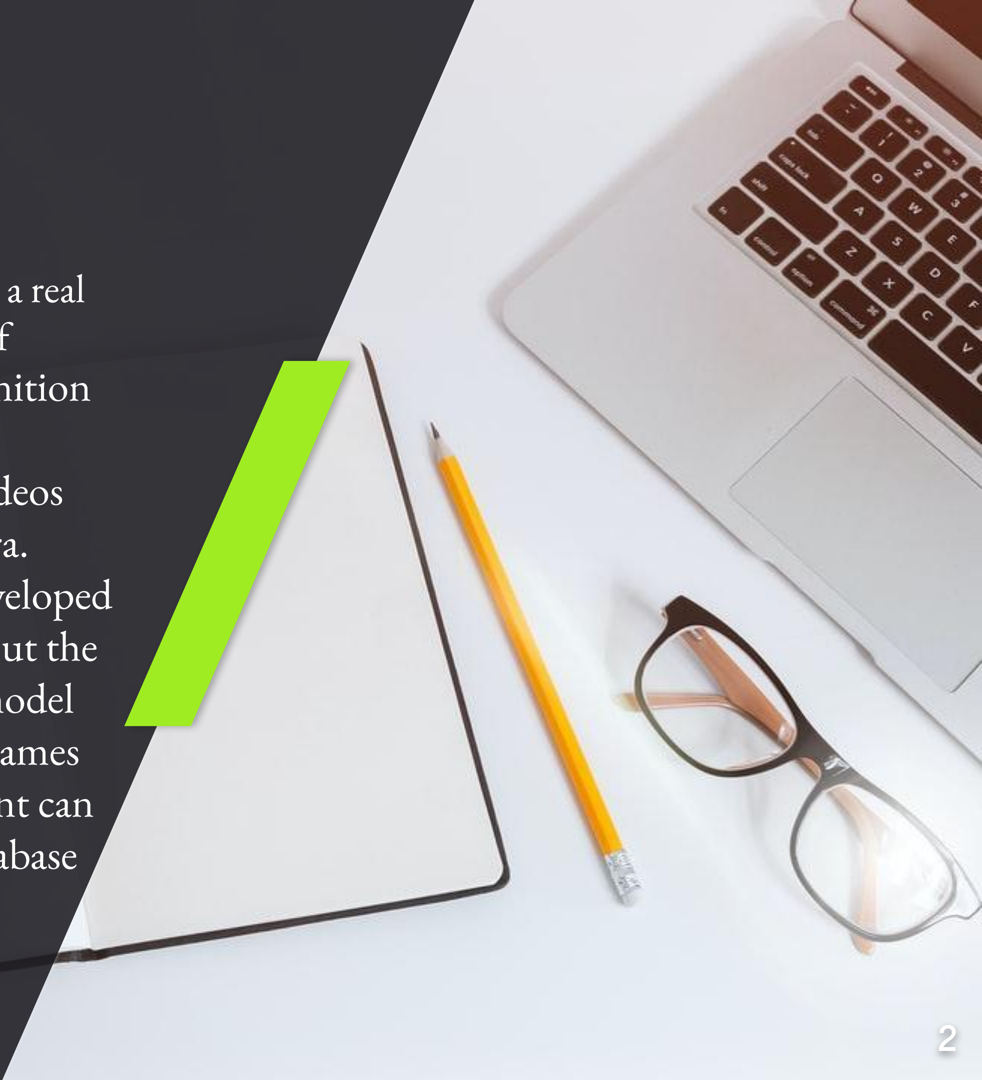
AUTOMATED ATTENDANCE BASED SYSTEM USING FACE RECOGNITION

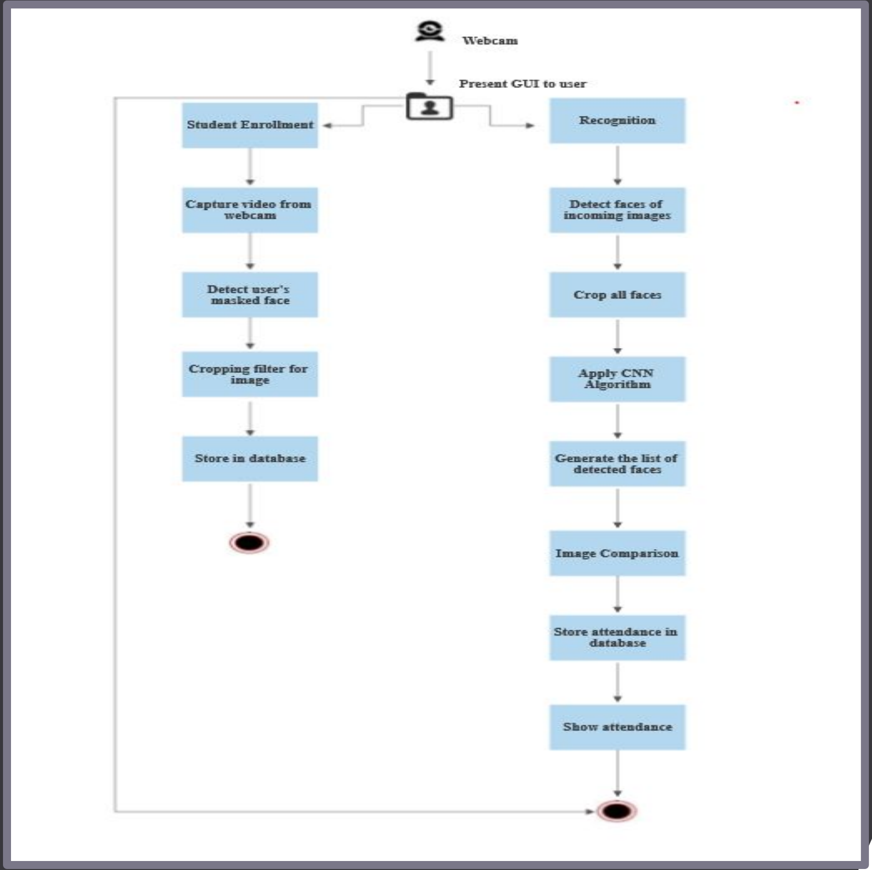
Name : Renuga Keerthi.K.R
Reg no : 20mcb1017

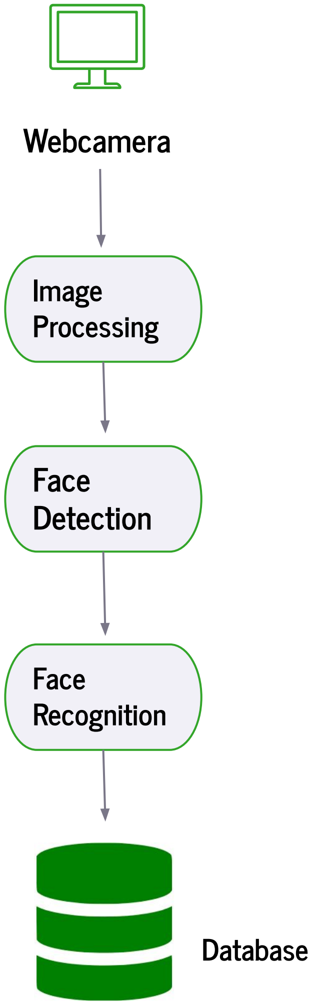
PROJECT OBJECTIVE:

Smart Attendance using Real-Time Face Recognition is a real world solution which comes with day to day activities of handling student attendance system. In this face recognition project, a computer system will be able to find and recognize users masked face precisely in images or videos that are being captured through a surveillance camera. Numerous algorithms and techniques have been developed for improving the performance of face recognition but the concept to be implemented here is Deep Learning model and CNN algorithm. It helps in conversion of the frames of the video into images so that the face of the student can be easily recognized for their attendance and the database can be easily reflected automatically.



System Architecture:





Tools and Technologies

- Hardware components
- Software components
- Technologies

- Hardware Components
  - IP Camera/Web Camera
  - Computer
- Software Components
  - Jupyter
  - Microsoft Excel
- Technologies
  - Python libraries
  - Tkinter
  - Open-CV Python
  - VGG-16 model

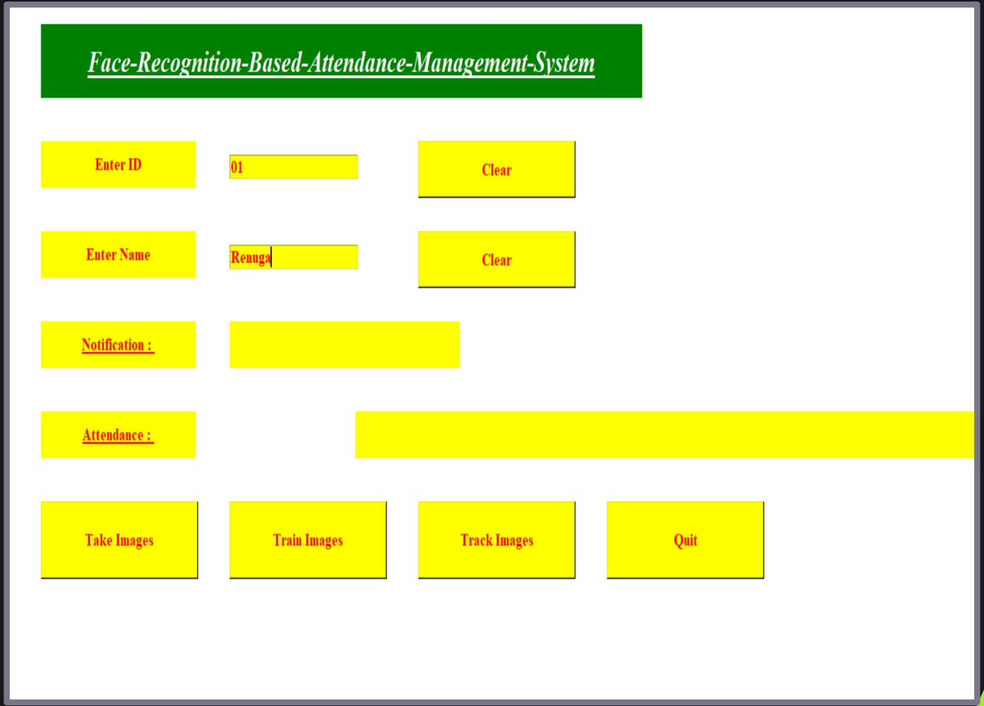
Workflow:

1. The camera will be able to get student's face images perfectly. Then it goes to further process of face detection.
2. Face detection helps to determine captured image with location and sizes of student faces.
3. Face recognition of an automatic method of identifying and verifying a person from images and videos from camera.
4. To apply a cropping filter in order to extract the non-masked region.
5. We extract deep features using VGG-16 face CNN descriptor.
6. The particular student will be marked as present in attendance when if a face from the particular date folder is matched

1. Student Registration :

GUI TO USER

Presenting GUI to the student to take, train and track images for attendance.



2. Face Detection :

- Detecting facial landmarks is a subset of the shape prediction problem. Facial landmarks such as eyes, eyebrows, nose, mouth, jaw line were used to localize and represent salient regions of the face.

- Our goal was to detect important facial structures on the face using shape prediction methods.

- Face detection has been achieved by us in two ways.

  - Using Opencv's built-in particular HaarCascades.

  - Using Cascading Classifier.

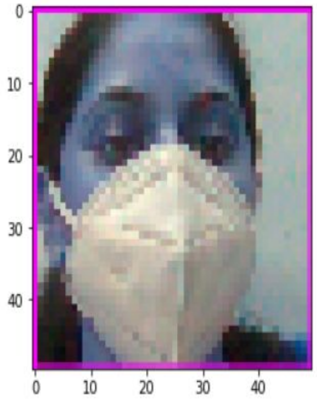
- To implement this case study, we need a lot of images of people wearing a mask and not wearing a mask.
- So first we need to collect data and we are going to collect data using our own camera.

Demo:

➢ For detecting key facial structures in the face region we have used a pre-trained facial landmark detector which estimates the location of x,y-coordinates that map to facial structures on the face.

➢ There are other facial landmark detectors, but all of them try to localize and label the following facial regions: Mouth, Right eyebrow, Left eyebrow, Right eye, Left eye, Nose, Jaw, etc

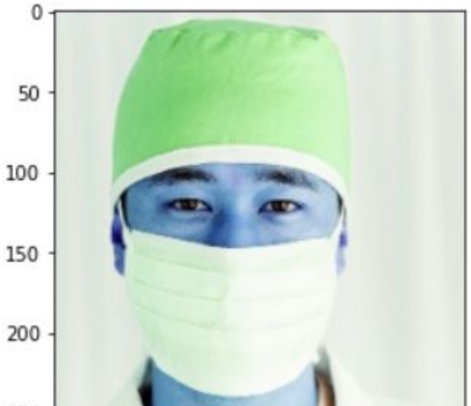
```
In [25]: plt.imshow(data[66])

Out[25]: <matplotlib.image.AxesImage at 0x19f53228430>
```



Implementation:

- Sample mask dataset:

```
img = cv2.imread('C:/Users/Renu/Desktop/3.jpg')

img.shape

(256, 256, 3)

img[0]

array([[223, 231, 224],
       [223, 231, 224],
       [224, 232, 225],
       [225, 233, 226],
       [227, 234, 227],
       [227, 234, 227],
       [227, 234, 227],
       [227, 234, 227],
       [227, 234, 227],
       [227, 234, 227],
       [227, 234, 227],
       [225, 232, 225],
       [224, 232, 225],
       [223, 231, 224],
       [223, 231, 224],
       [222, 230, 223],
       [221, 232, 224],

plt.imshow(img)

<matplotlib.image.AxesImage at 0x19f4af84f40>
```



Haarcascade Classifier:

```
haar_data = cv2.CascadeClassifier('C:/Users/Renu/Desktop/haarcascade_frontalface_default.xml')

haar_data.detectMultiScale(img)

()
```

- Object Detection using Haar feature-based cascade classifiers is an effective object detection method.

- First we need to load the required XML classifiers. Then load our input image (or video).

❑ Face Detection:

```
In [17]: capture = cv2.VideoCapture(0)
         data = []
         while True:
             flag, img = capture.read()
             if flag:
                 faces = haar_data.detectMultiScale(img)
                 for x,y,w,h in faces:
                     cv2.rectangle(img, (x,y),(x+w,y+h), (255,0,255),4)
                     face = img[y:y+h, x:x+w, :]
                     face = cv2.resize(face,(50,50))
                     print(len(data))
                 if len(data) < 100:
                     data.append(face)
                     cv2.imshow('result',img)
                 if cv2.waitKey(2) == 27:
                     break
         capture.release()
         cv2.destroyAllWindows()
```

```
49
51
54
55
55
56
57
59
62
63
64
65
71
73
74
77
82
84
85
87
```

- ➔ To implement this case study, we need a lot of images of students wearing a mask and not wearing a mask.
- ➔ So first we need to collect data and we are going to collect data using our own camera. Here is the complete code to perform face detection using camera and storing face data only:

❑ Face Detection:

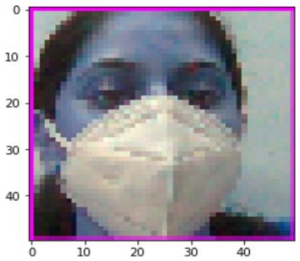
```python
import numpy as np
```

```python
np.save('StudID:_without_mask.npy',data)
```

```python
np.save('StudID:_with_mask.npy',data)
```

```python
plt.imshow(data[66])
```

```
<matplotlib.image.AxesImage at 0x19f53228430>
```




```python
Stud_with_mask = np.load('StudID:_with_mask.npy')
Stud_without_mask = np.load('StudID:_without_mask.npy')
```

```python
Stud_with_mask.shape
```

```
(100, 50, 50, 3)
```

```python
Stud_without_mask.shape
```

```
(100, 50, 50, 3)
```

➢ Save the data in a numpy file and can also plot the face data to check the data collected by OpenCV.

➢ Now we can load the data anywhere and start processing.

```python
import os

import numpy as np

import shutil

rootdir= 'C:/Users/Renu/Desktop/Images' #path of the original folder
```

```python
classes = ['Renuga']
```

```python
for i in classes:
    os.makedirs(rootdir +'/train/' + i)
    os.makedirs(rootdir +'/test/' + i)
    source = rootdir + '/' + i

    allFileNames = os.listdir(source)

    np.random.shuffle(allFileNames)

    test_ratio = 0.25

    train_FileNames, test_FileNames = np.split(np.array(allFileNames),
                                                [int(len(allFileNames)* (1 - test_ratio))])

    train_FileNames = [source+'/'+ name for name in train_FileNames.tolist()]
    test_FileNames = [source+'/' + name for name in test_FileNames.tolist()]

for name in train_FileNames:
    shutil.copy(name, rootdir +'/train/' + i)

for name in test_FileNames:
    shutil.copy(name, rootdir +'/test/' + i)
```

Dataset Preparation:

- Storing the dataset in a separate folder
- Splitting the dataset into training and testing data

3. Face Recognition:

- The important part of this system is face recognition.
- Face recognition of an automatic method of identifying and verifying a person from images and videos from camera.
- In this face recognition method , we develop a VGG-16 model to identify the masked face.

WORKFLOW:

- The trained image dataset is used as the basis for developing deep CNNs for face recognition tasks such as face identification and verification.
- The student face is recognized if it matches with the data folder along with student_name.

MODEL DEVELOPMENT:

➔ Model summary for class = 'Renuga'

block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1)	25089

MODEL DEVELOPMENT:

➔ Fitting the model

```
[ ] # fit the model
    r = model.fit_generator(
      training_set,
      validation_data=test_set,
      epochs=5,
      steps_per_epoch=len(training_set),
      validation_steps=len(test_set)
    )
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/5
3/3 [==============================] - 52s 17s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 2/5
3/3 [==============================] - 59s 21s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 3/5
3/3 [==============================] - 59s 21s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 4/5
3/3 [==============================] - 59s 21s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 5/5
3/3 [==============================] - 59s 21s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

Face Recognition:

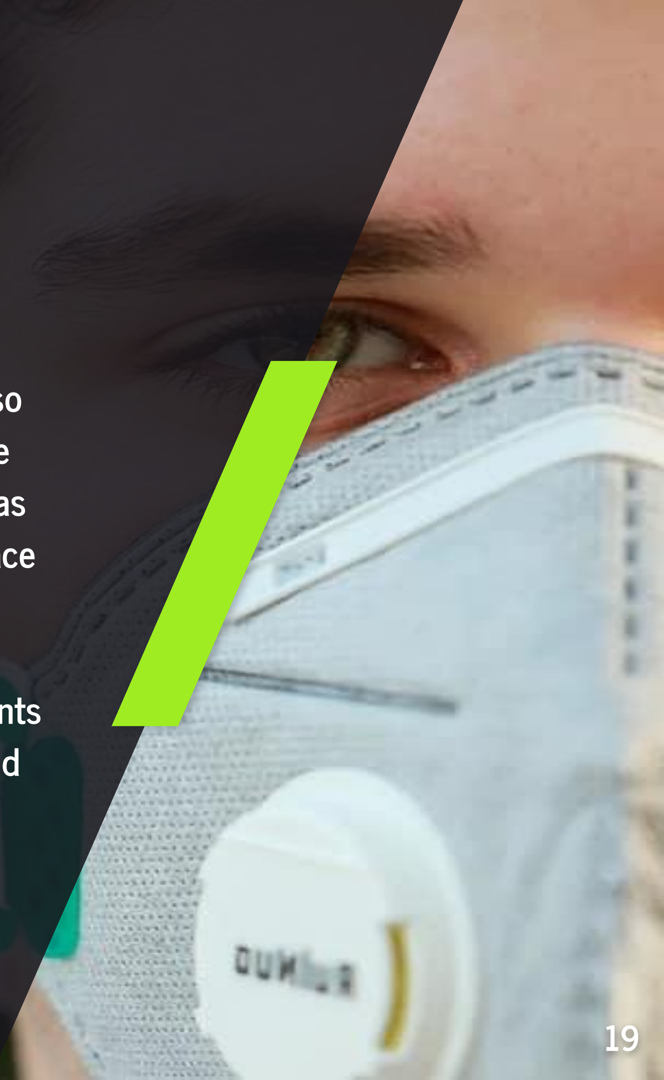
- The developed model is saved as facefeatures_new_model.h5 and further used for recognizing the face.

- The masked face is recognized as the model matches with the trained image dataset along with the student_name.

FUTURE WORK:

Attendance marker:

- ➔ Model will be further developed so that unique id of the students are identified separately and marked as present in attendance ,when if a face from particular date folder is matched.
- ➔ That is, collect the list of all students who were present in the class, and rest of the students belongs the class will be marked as absent.



CONCLUSION:

In real-world scenarios, human faces might be occluded by other objects such as facial mask. This makes the face recognition process a very challenging task. Consequently, current face recognition methods will easily fail to make an efficient recognition. The proposed method improves the generalization of face recognition process in the presence of the mask.The anticipated outcome of this project is the identification of masked face of students and mark their attendance automatically.