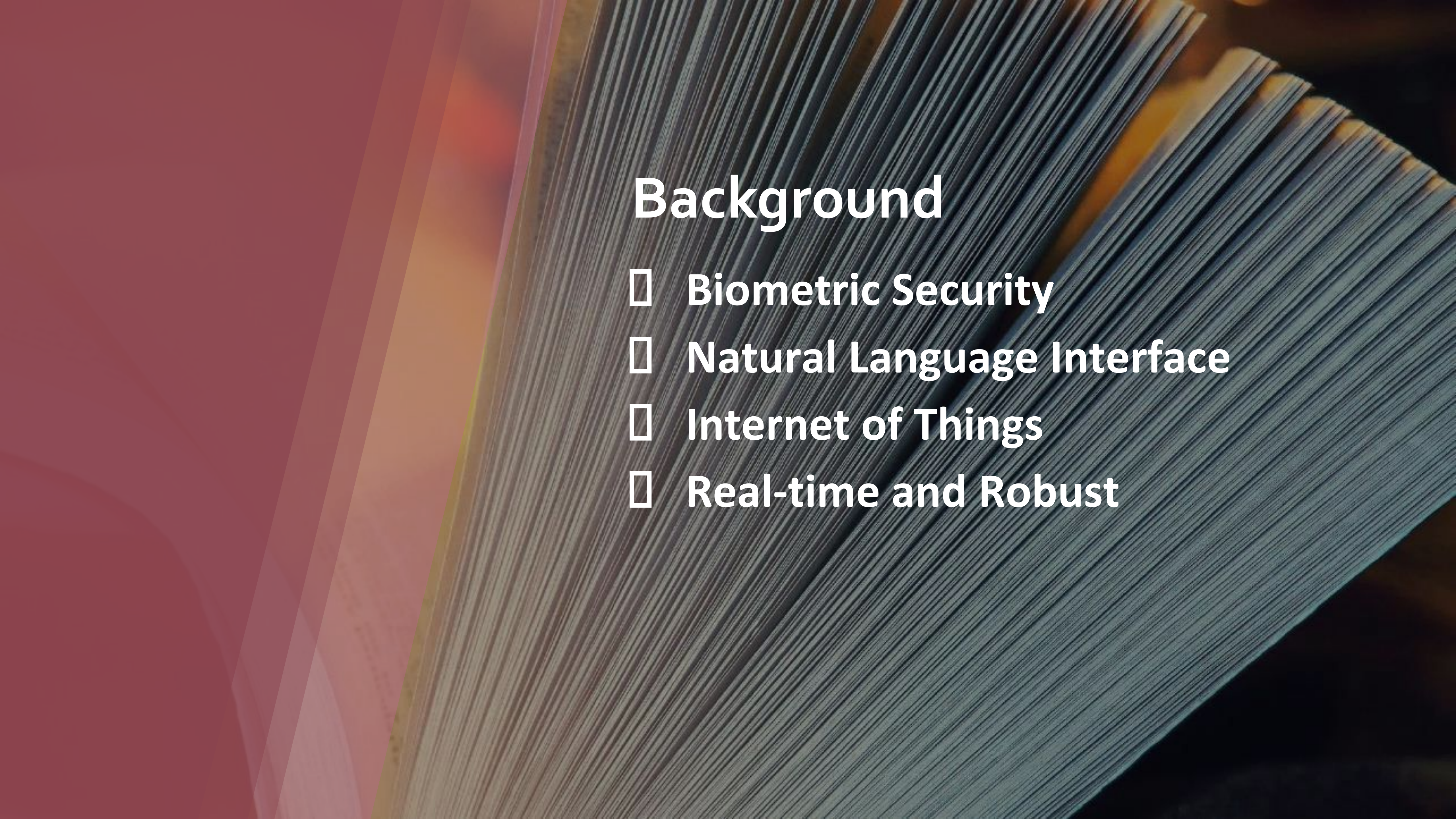
FACE RECOGNITION FOR ATTENDANCE SYSTEM

Christian NP – 0860810
Ivan Surya H – 0860812
M Farhan Tandia – 0860814

OUTLINE

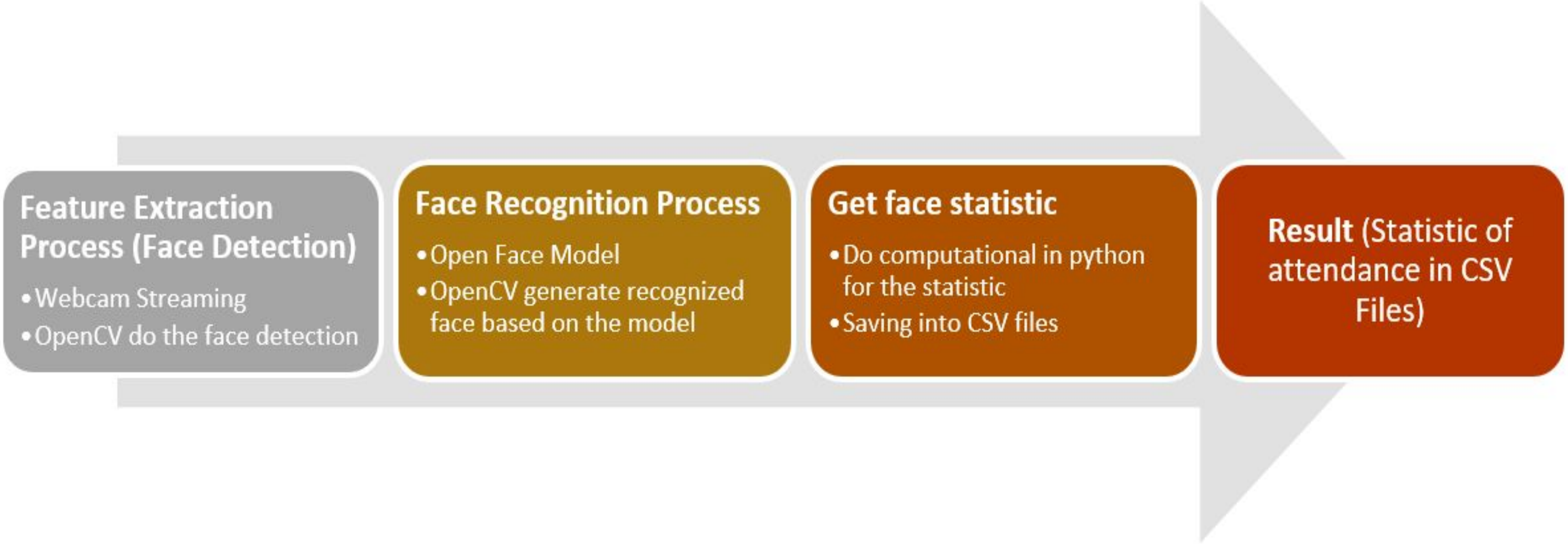
- BACKGROUND
- ARCHITECTURE
- OPEN FACE
- IMPLEMENTATION
- DEMO
- CONCLUSION



Background

- Biometric Security
- Natural Language Interface
- Internet of Things
- Real-time and Robust

ARCHITECTURE



Feature Extraction Process (Face Detection)

- Webcam Streaming
- OpenCV do the face detection

Face Recognition Process

- Open Face Model
- OpenCV generate recognized face based on the model

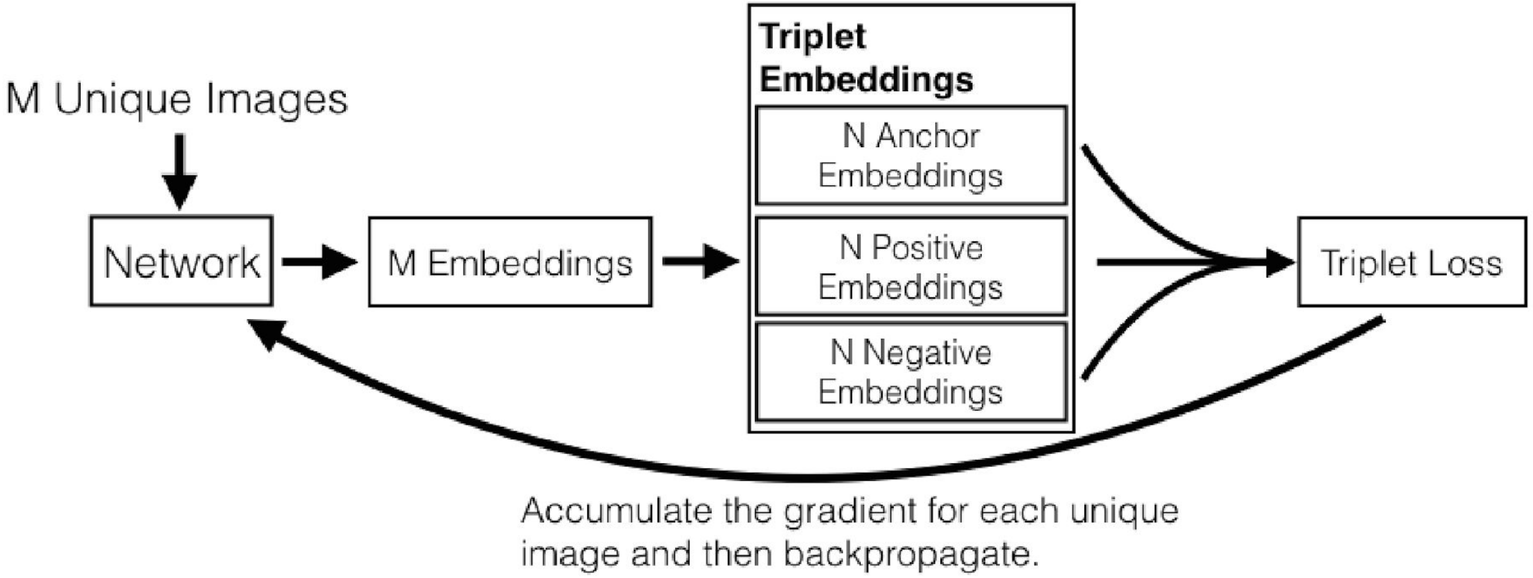
Get face statistic

- Do computational in python for the statistic
- Saving into CSV files

Result (Statistic of attendance in CSV Files)

OPEN FACE

OpenFace is a pre-trained model which use FaceNet from Google

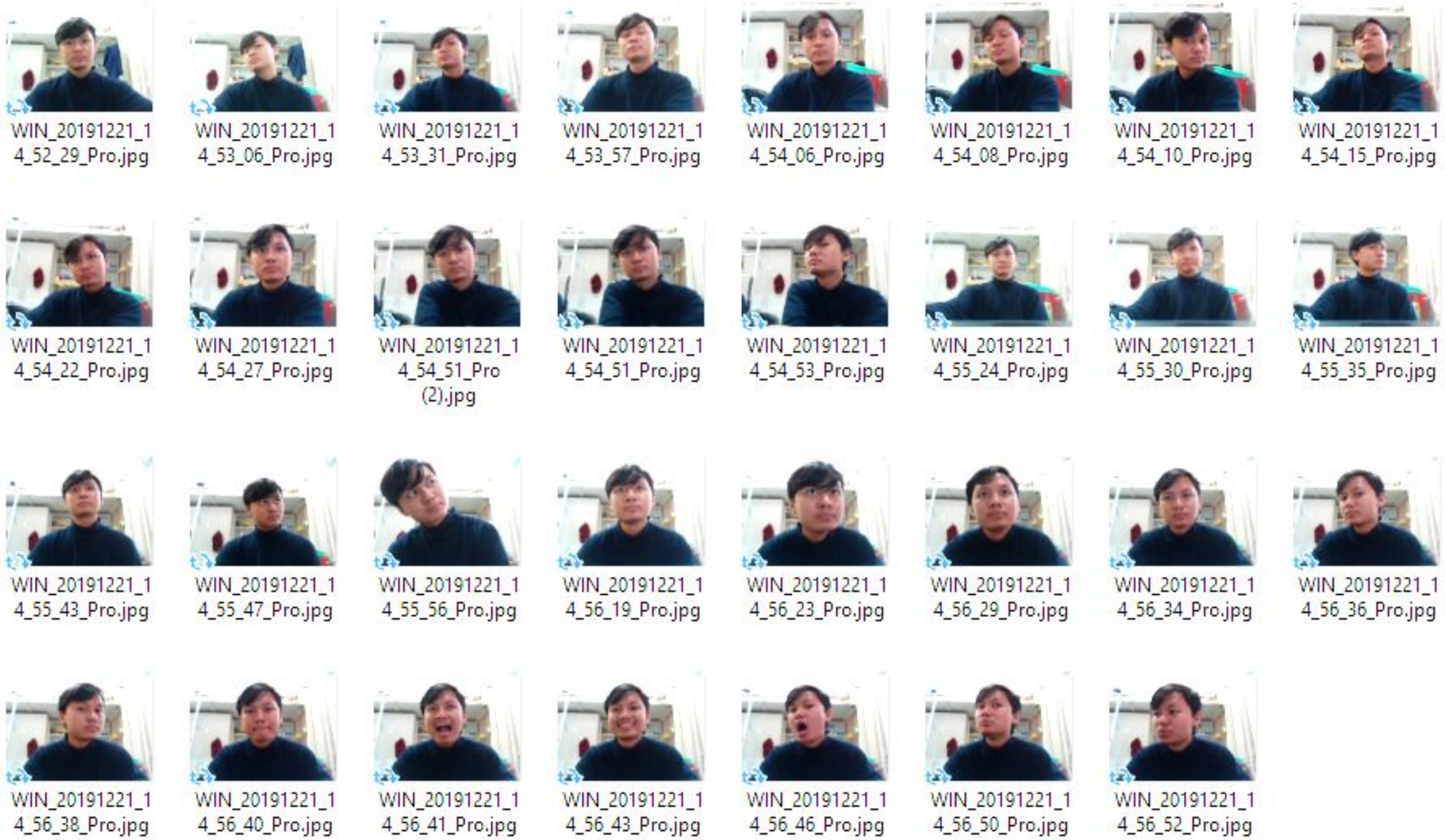


IMPLEMENTATION

ATTENDANCE SYSTEM

CAPTURE DATASET

Use webcam to capture 30 face images of each person with Python and OpenCV



WIN_20191221_14_52_29_Pro.jpg
WIN_20191221_14_53_06_Pro.jpg
WIN_20191221_14_53_31_Pro.jpg
WIN_20191221_14_53_57_Pro.jpg
WIN_20191221_14_54_06_Pro.jpg
WIN_20191221_14_54_08_Pro.jpg
WIN_20191221_14_54_10_Pro.jpg
WIN_20191221_14_54_15_Pro.jpg
WIN_20191221_14_54_22_Pro.jpg
WIN_20191221_14_54_27_Pro.jpg
WIN_20191221_14_54_51_Pro (2).jpg
WIN_20191221_14_54_51_Pro.jpg
WIN_20191221_14_54_53_Pro.jpg
WIN_20191221_14_55_24_Pro.jpg
WIN_20191221_14_55_30_Pro.jpg
WIN_20191221_14_55_35_Pro.jpg
WIN_20191221_14_55_43_Pro.jpg
WIN_20191221_14_55_47_Pro.jpg
WIN_20191221_14_55_56_Pro.jpg
WIN_20191221_14_56_19_Pro.jpg
WIN_20191221_14_56_23_Pro.jpg
WIN_20191221_14_56_29_Pro.jpg
WIN_20191221_14_56_34_Pro.jpg
WIN_20191221_14_56_36_Pro.jpg
WIN_20191221_14_56_38_Pro.jpg
WIN_20191221_14_56_40_Pro.jpg
WIN_20191221_14_56_41_Pro.jpg
WIN_20191221_14_56_43_Pro.jpg
WIN_20191221_14_56_46_Pro.jpg
WIN_20191221_14_56_50_Pro.jpg
WIN_20191221_14_56_52_Pro.jpg

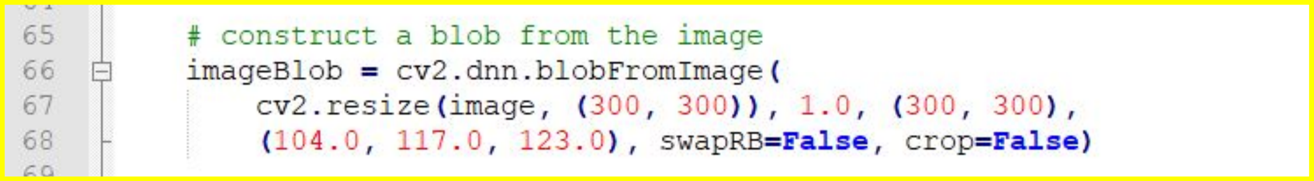
TRAIN FACE DATASET

❖ Extract Embeddings with OpenFace

(Preprocessing input image)

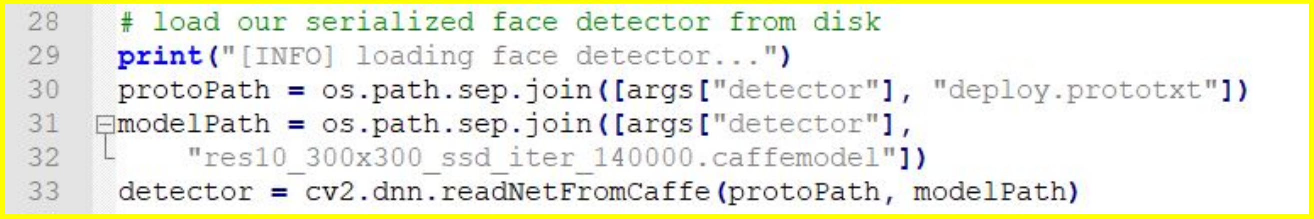
Deep learning feature (OpenFace) extractor to generate a 128-D vector describing a face. All faces **in our dataset will be passed through the neural network to generate embeddings**.

1. **(Preprocessing) Input Image using blob function**



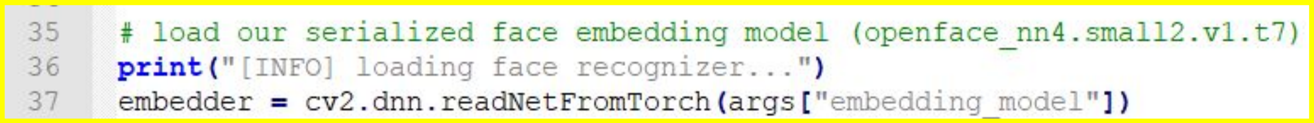
```
    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(image, (300, 300)), 1.0, (300, 300),
        (104.0, 117.0, 123.0), swapRB=False, crop=False)
```

2. **(Preprocessing) Face detector using Resnet caffe model**

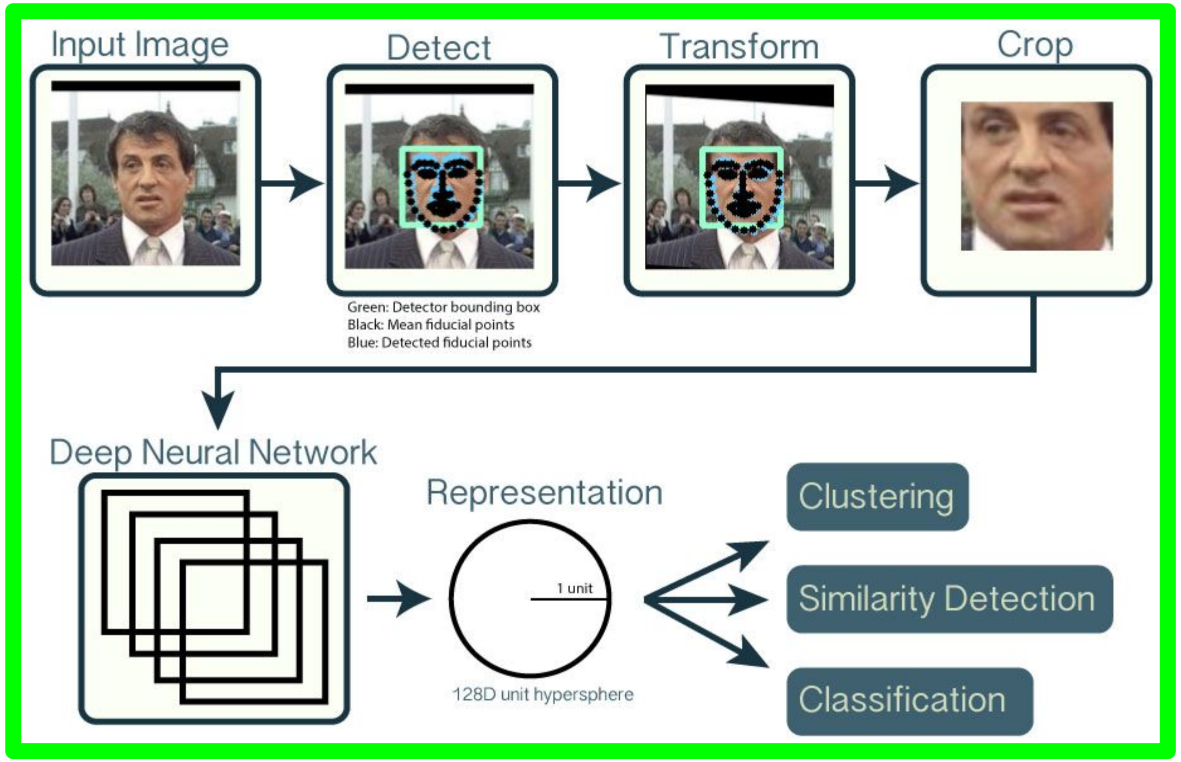


```
# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
```

3. **Pass the face image through <u>OpenFace</u> model (128-D vector)**



```
# load our serialized face embedding model (openface_nn4.small2.v1.t7)
print("[INFO] loading face recognizer...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])
```



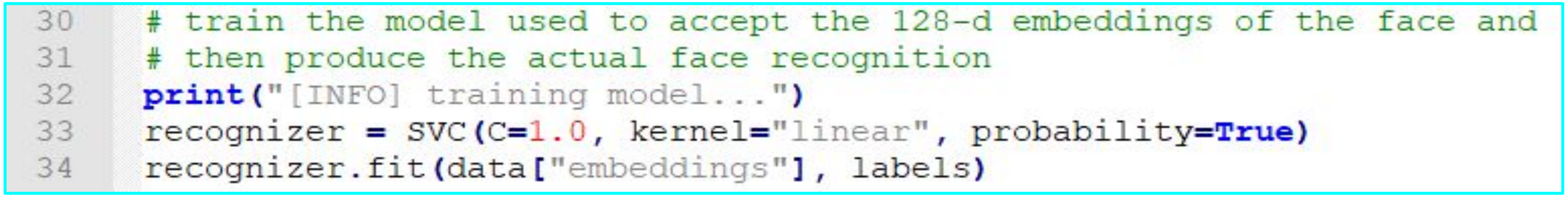
TRAIN FACE DATASET

“Farhan”

“Ivan”

Unknown

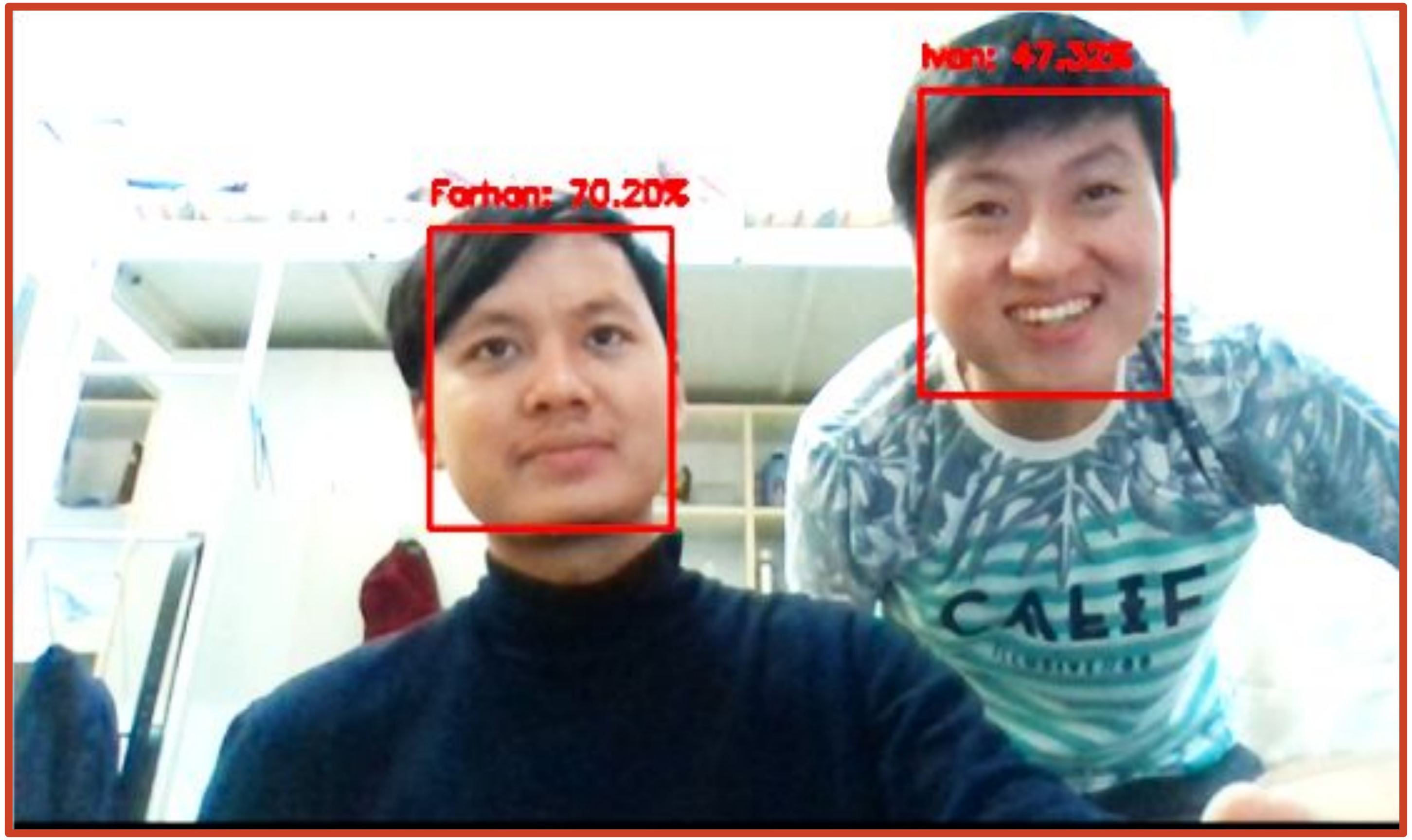
- **Linear SVC** (Support Vector Classifier) for **classifying the detected faces** to the embeddings data and accept the 128-d embeddings of the face and then **produce the actual face recognition.**



```
# train the model used to accept the 128-d embeddings of the face and
# then produce the actual face recognition
print("[INFO] training model...")
recognizer = SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)
```

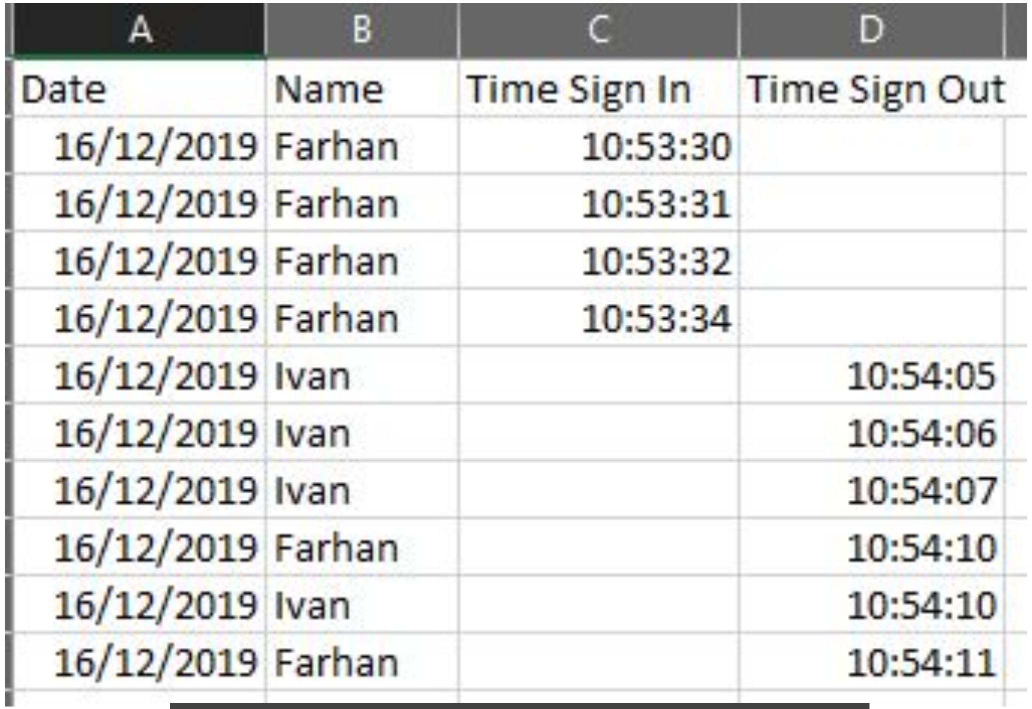
RECOGNIZE FACES

- Recognize faces in frames of a video stream using web camera
- Extract face embedding and query SVM model to determine who is in an image, then draw boxes with its name.



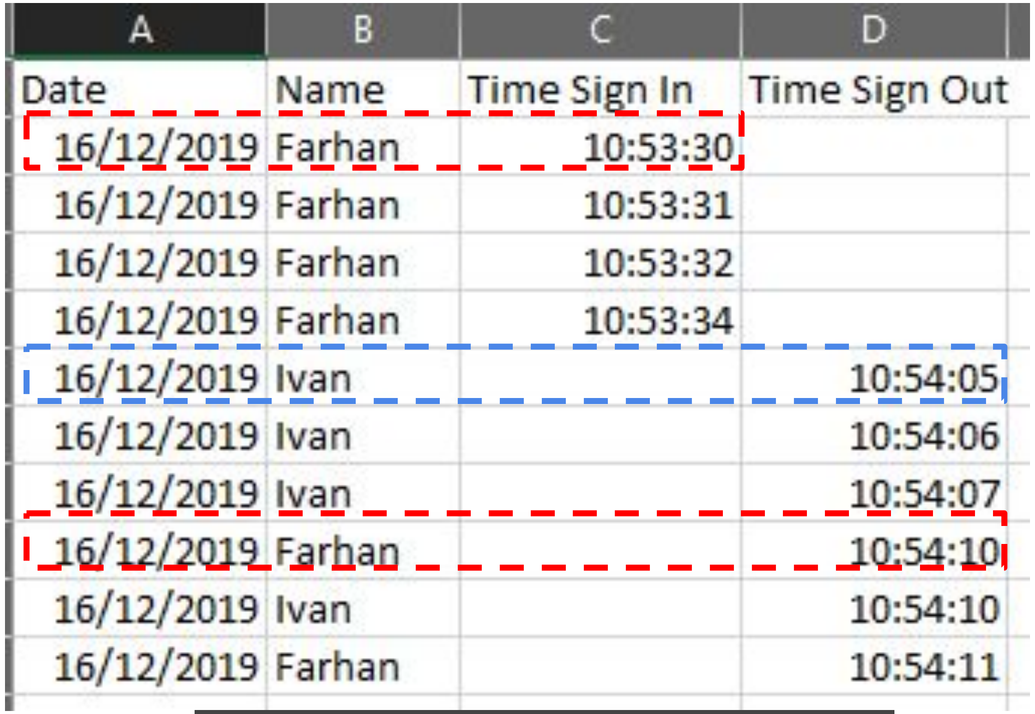
SAVE RECOGNIZED FACES

1. We make the mechanism to save recognized faces by **record the first detection which has the threshold** of above 70% accuracy for 20x frames .
2. We only record attendance within **a certain time frame for entering and leaving** the classroom.



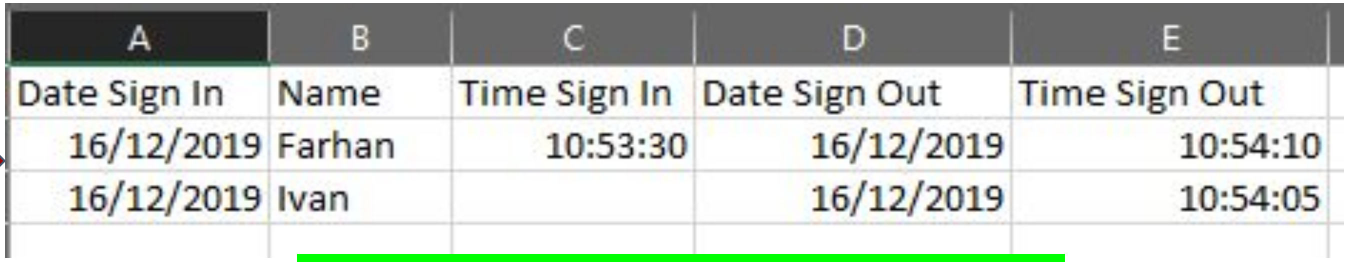
Date	Name	Time Sign In	Time Sign Out
16/12/2019	Farhan	10:53:30	
16/12/2019	Farhan	10:53:31	
16/12/2019	Farhan	10:53:32	
16/12/2019	Farhan	10:53:34	
16/12/2019	Ivan		10:54:05
16/12/2019	Ivan		10:54:06
16/12/2019	Ivan		10:54:07
16/12/2019	Farhan		10:54:10
16/12/2019	Ivan		10:54:10
16/12/2019	Farhan		10:54:11

RAW DATA OF ATTENDANCE SYSTEM



Date	Name	Time Sign In	Time Sign Out
16/12/2019	Farhan	10:53:30	
16/12/2019	Farhan	10:53:31	
16/12/2019	Farhan	10:53:32	
16/12/2019	Farhan	10:53:34	
16/12/2019	Ivan		10:54:05
16/12/2019	Ivan		10:54:06
16/12/2019	Ivan		10:54:07
16/12/2019	Farhan		10:54:10
16/12/2019	Ivan		10:54:10
16/12/2019	Farhan		10:54:11

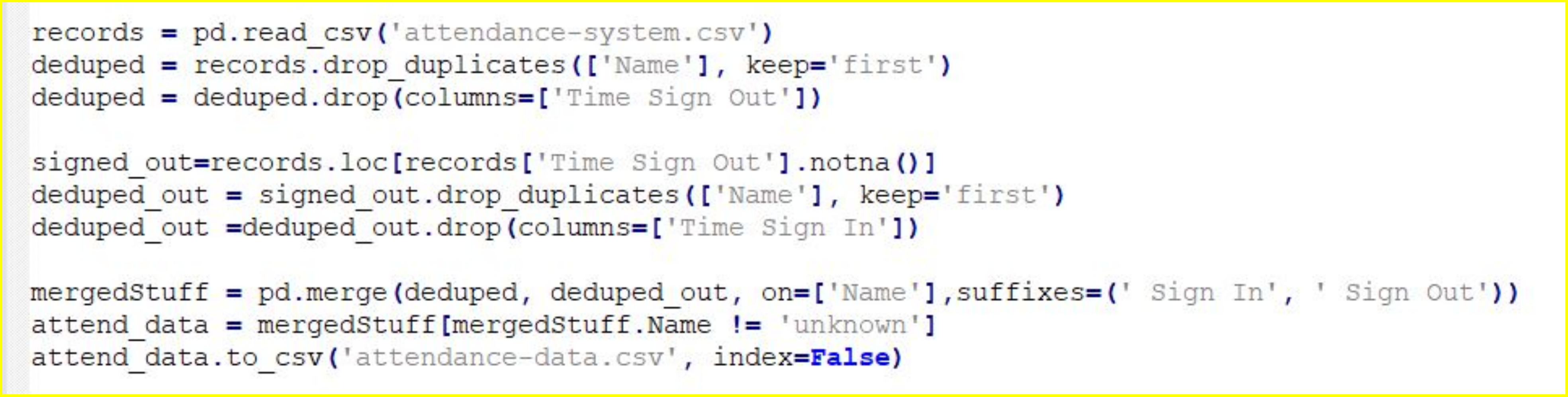
RAW DATA OF ATTENDANCE SYSTEM



Date Sign In	Name	Time Sign In	Date Sign Out	Time Sign Out
16/12/2019	Farhan	10:53:30	16/12/2019	10:54:10
16/12/2019	Ivan		16/12/2019	10:54:05

FINAL ATTENDANCE SYSTEM DATA

Using Pandas Dataframe for data processing



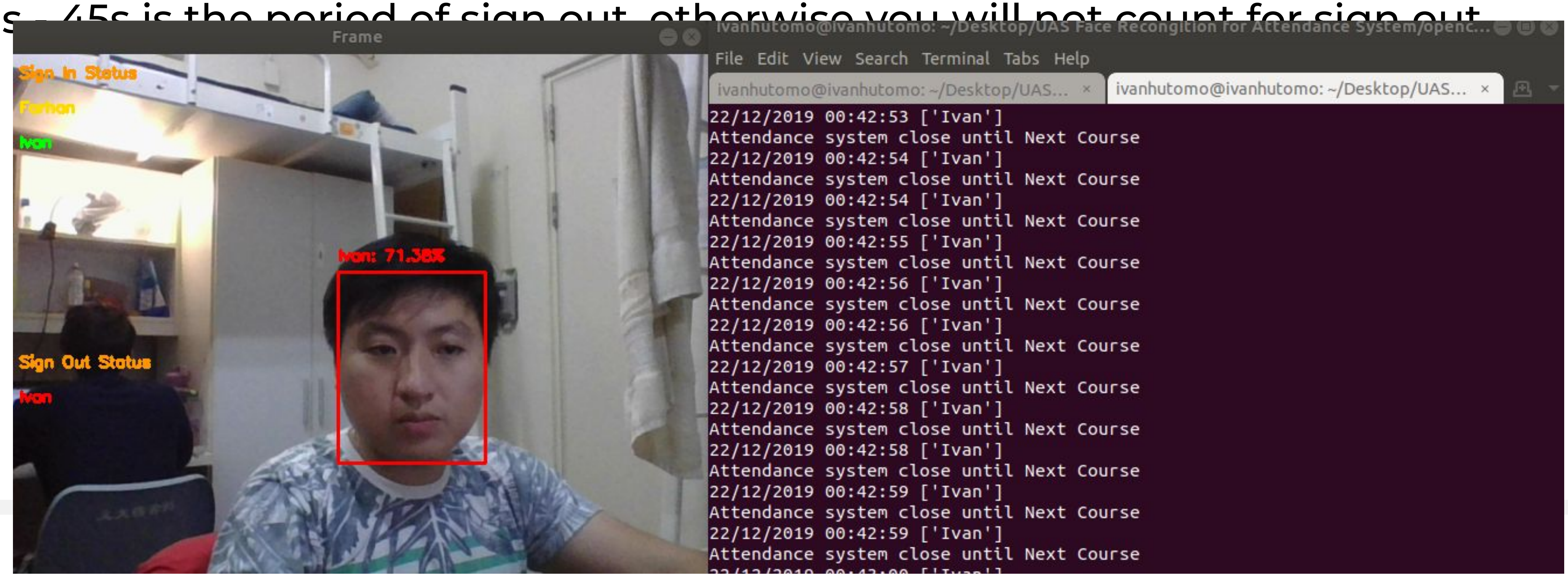
```
records = pd.read_csv('attendance-system.csv')
deduped = records.drop_duplicates(['Name'], keep='first')
deduped = deduped.drop(columns=['Time Sign Out'])

signed_out=records.loc[records['Time Sign Out'].notna()]
deduped_out = signed_out.drop_duplicates(['Name'], keep='first')
deduped_out =deduped_out.drop(columns=['Time Sign In'])

mergedStuff = pd.merge(deduped, deduped_out, on=['Name'],suffixes=(' Sign In', ' Sign Out'))
attend_data = mergedStuff[mergedStuff.Name != 'unknown']
attend_data.to_csv('attendance-data.csv', index=False)
```

DEMO

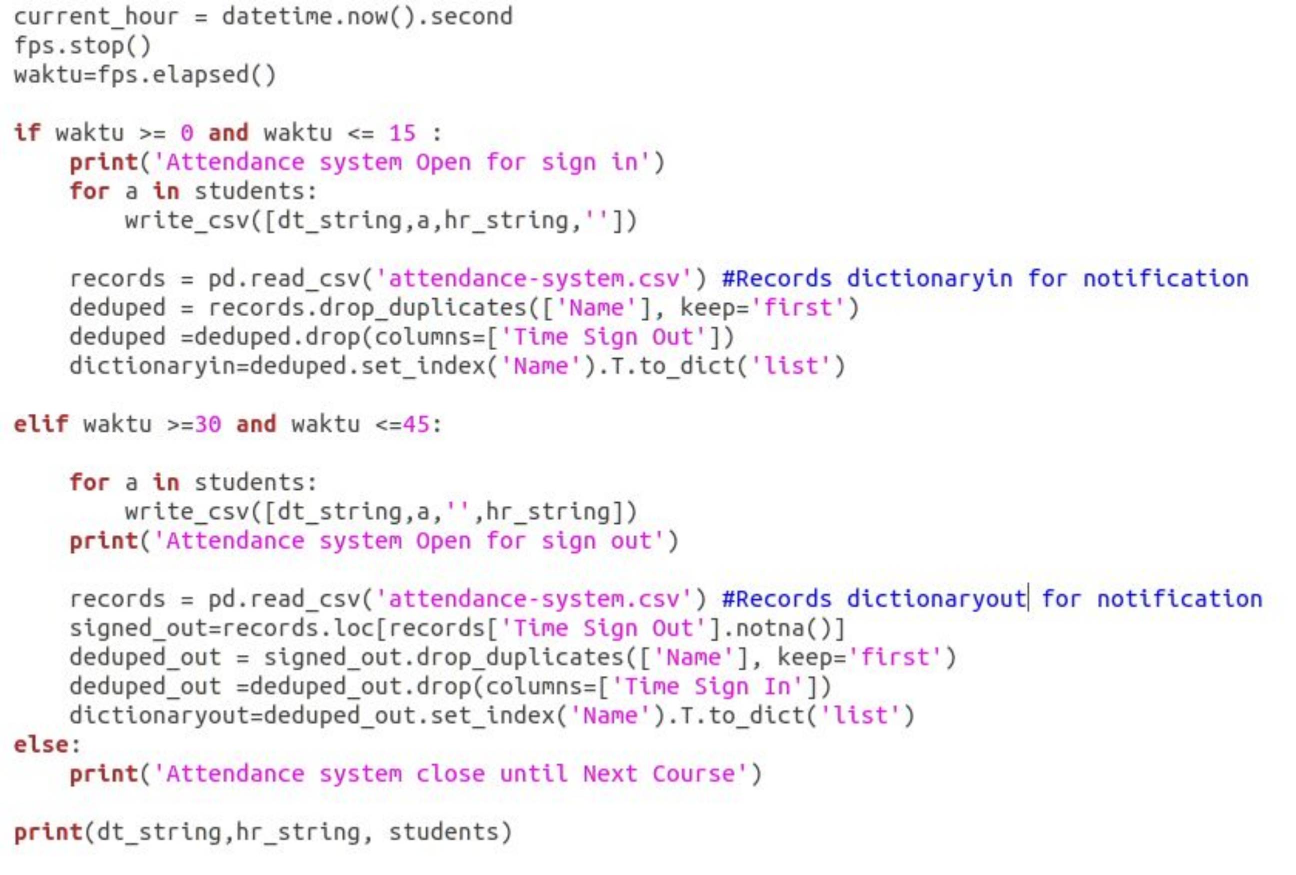
ATTENDANCE SYSTEM

HOW THE DEMOS WORK?

- We will demonstrate attendance system using our face
- The face’s confidence needed to be recorded as attendance is 70%
- 0s - 15s is period of sign in, otherwise you will not count for sign in.
- 16s - 29s is the period where attendance system is inactive, your face still recognized but you will not count either for sign in and sign out.
- 30s - 45s is the period of sign out, otherwise you will not count for sign out.


Frame
Sign In Status
Farhan
Ivan
Ivan: 71.38%
Sign Out Status
Ivan
ivanhutomo@ivanhutomo: ~/Desktop/UAS Face Recongition for Attendance System/openc...
File Edit View Search Terminal Tabs Help
ivanhutomo@ivanhutomo: ~/Desktop/UAS...
ivanhutomo@ivanhutomo: ~/Desktop/UAS...

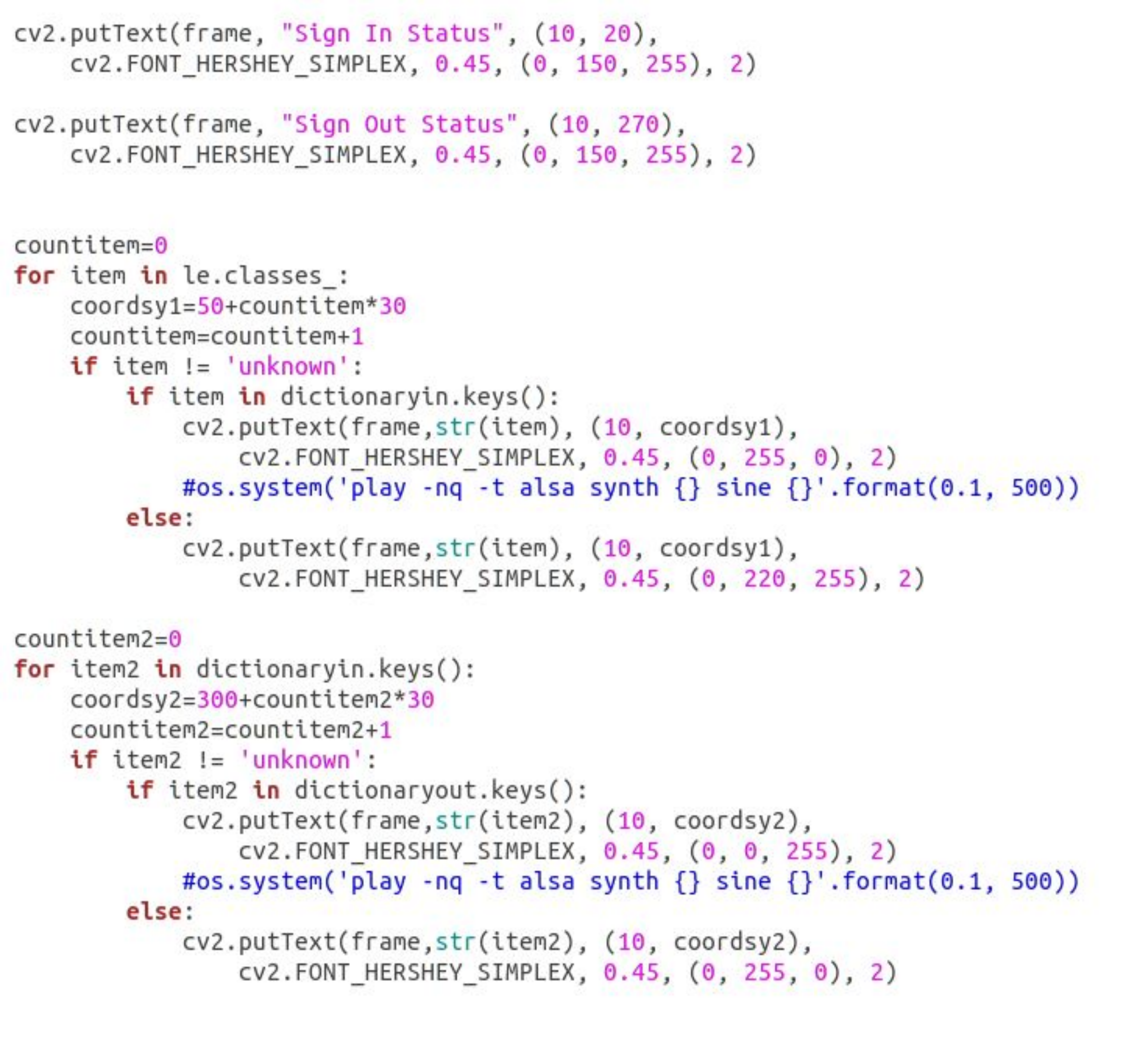
```
22/12/2019 00:42:53 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:54 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:54 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:55 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:56 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:56 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:57 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:58 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:58 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:59 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:42:59 ['Ivan']
Attendance system close until Next Course
22/12/2019 00:43:00 ['Ivan']
```



```
current_hour = datetime.now().second
fps.stop()
waktu=fps.elapsed()

if waktu >= 0 and waktu <= 15 :
    print('Attendance system Open for sign in')
    for a in students:
        write_csv([dt_string,a,hr_string,''])

    records = pd.read_csv('attendance-system.csv') #Records dictionaryin for notification
    deduped = records.drop_duplicates(['Name'], keep='first')
    deduped =deduped.drop(columns=['Time Sign Out'])
    dictionaryin=deduped.set_index('Name').T.to_dict('list')

elif waktu >=30 and waktu <=45:

    for a in students:
        write_csv([dt_string,a,'',hr_string])
    print('Attendance system Open for sign out')

    records = pd.read_csv('attendance-system.csv') #Records dictionaryout for notification
    signed_out=records.loc[records['Time Sign Out'].notna()]
    deduped_out = signed_out.drop_duplicates(['Name'], keep='first')
    deduped_out =deduped_out.drop(columns=['Time Sign In'])
    dictionaryout=deduped_out.set_index('Name').T.to_dict('list')
else:
    print('Attendance system close until Next Course')

print(dt_string,hr_string, students)
```

Code to Record Recognized Face to Dictionary

Code to Put Dictionary to Frame



```
cv2.putText(frame, "Sign In Status", (10, 20),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 150, 255), 2)

cv2.putText(frame, "Sign Out Status", (10, 270),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 150, 255), 2)


countitem=0
for item in le.classes_:
    coordsy1=50+countitem*30
    countitem=countitem+1
    if item != 'unknown':
        if item in dictionaryin.keys():
            cv2.putText(frame,str(item), (10, coordsy1),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)
            #os.system('play -nq -t alsa synth {} sine {}'.format(0.1, 500))
        else:
            cv2.putText(frame,str(item), (10, coordsy1),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 220, 255), 2)

countitem2=0
for item2 in dictionaryin.keys():
    coordsy2=300+countitem2*30
    countitem2=countitem2+1
    if item2 != 'unknown':
        if item2 in dictionaryout.keys():
            cv2.putText(frame,str(item2), (10, coordsy2),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
            #os.system('play -nq -t alsa synth {} sine {}'.format(0.1, 500))
        else:
            cv2.putText(frame,str(item2), (10, coordsy2),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)
```

CONCLUSION

- ✓ We perform face detection, face embedding, face recognition
- ✓ OpenFace can perform well in real data using deep metrics learning and SVM
- ✓ The network can learn to quantify faces and return highly robust and discriminating embeddings suitable for face recognition
- ✓ We can reuse the OpenFace model for our own applications without having to explicitly train it

THANK YOU

Any Questions?