REal-Time Facial recognition using DeepFace and FaceNet512

PREPARED BY  
Saville Atkins

In partial fulfillment of the requirements for  
 CPE 4903

PREPARED BY: Saville Atkins  
  
APPROVED BY: Saville Atkins

April 29, 2025  
  
KENNESAW STATE UNIVERSITY

ABSTRACT

This project involved building a facial recognition system that works in real-time using a Raspberry Pi. I used DeepFace, an open-source facial analysis library, to generate facial embeddings and recognize users based on cosine similarity. The system has two modes: one for enrolling faces and another for recognizing them live. It displays results using a Sense HAT’s LED matrix. Everything was run directly on the Raspberry Pi with a USB camera, making the project compact and efficient. Originally I tried to get the Raspberry Pi Cam working instead of using the USB Camera but after troubleshooting it for an extended period of time, it made more sense to use the USB Camera.

Table of contents

Section Title Page

1. INTRODUCTION ............................................................................ 1

1.1 Project Goals .......................................................................... 1

2. BACKGROUND THEORY ....................................................................... 2

2.1 CNNs (Convolutional Neural Networks) ................................................... 2

2.2 Embeddings and Cosine Similarity ....................................................... 2

2.3 FaceNet512 ............................................................................. 3

2.4 Closing Theory ......................................................................... 3

3. SYSTEM DESIGN ........................................................................... 4

3.1 Enrollment Mode ........................................................................ 4

3.2 Recognition Mode ....................................................................... 4

3.3 Visual Indicators ...................................................................... 5

4. SETUP AND DEVELOPMENT TOOLS ............................................................. 6

4.1 Hardware ............................................................................... 6

4.2 Software ............................................................................... 6

4.3 Setup Overview ......................................................................... 7

5. PROCEDURE AND IMPLEMENTATION ............................................................ 8

6. ANALYSIS ................................................................................ 9

7. CONCLUSION .............................................................................. 10

8. IMAGES .................................................................................. 11

9. CODE .................................................................................... 12

10. REFERENCES AND FOOTNOTES ............................................................... 13

1. introduction

1.1 Project Goals

The goal of this project was to design and implement a lightweight, local facial recognition system that would leave us with real world AI project under our belt. Facial recognition is a widely used biometric method, and exploring how it could be deployed on a small device like the Raspberry Pi is a fun and useful project. I used DeepFace with the pre-trained FaceNet512 model to handle face analysis, while OpenCV handles camera input and shows feedback on whether the users face matches one of the enrolled faces. The Sense HAT was used for recognition confirmation, current process feedback, and the joystick button to take the initial enrolment images.

2. Background theory

2.1 cnns(Convolutional Neural Networks)

CNNs are deep learning models that work especially well for handling image data. They look at images in small patches and use filters to pick up on patterns like edges, curves, or textures. These patterns get passed through several layers that help the system learn what important features make up a face. So basically, a CNN learns to recognize things like eyes, mouths, and overall face structure just by going deeper through the image. That’s what allows it to eventually tell different people apart.

2.2 Embeddings and cosine similarity

When it comes to face recognition, comparing images directly doesn’t work well, especially if lighting or angles change. Instead, we convert the face into a numerical format — a list of numbers called an embedding — that captures its key features. Think of it like a digital fingerprint for the face. Then we compare two embeddings using cosine similarity, which measures how close the two number lists are by checking the angle between them. If the angle is small, the faces are probably of the same person.

2.3 FACENET512

FaceNet512 is a version of the FaceNet model that turns a face image into a 512-dimensional vector. It's already been trained on huge datasets of faces, so it knows how to pull out the most important features. The cool part is it works even if your face is slightly turned or the lighting is a bit different. It’s the part of the system that makes the embeddings I compare during recognition.

2.4 CLOSING THEORY

Altogether, CNNs, embeddings, cosine similarity, and FaceNet — are what make modern face recognition possible. They take a regular image and turn it into something the computer can understand and compare reliably.

3. SYSTEM DESIGN

3.1 ENROLLMENT MODE

The user presses the joystick to take a picture, which is then turned into an embedding and saved.

3.2 RECOGNITION MODE

The system continuously checks faces using the camera. Each detected face is compared to the stored embeddings to check for a match.

3.3 VISUAL INDICATORS

Visual indicators are displayed on the Sense Hat:

* White = start
* Blue = capture mode
* Purple = saving face embeddings
* Green = known face
* Red = unknown face

4. SETUP AND DEVELOPMENT TOOLS

4.1 HARDWARe

Hardware used:

* Raspberry PI 4B
* USB Webcam
* Sense Hat
* 64GB microSD card

4.2 SOFTWARE

Software and Tools used:

* Raspberry PI OS(64-bit)
* Python 3.9 (using pyenv)
* DeepFace library
* OpenCV
* NumPy
* Sense HAT Python library
* VNC for remote access
* Geany (IDE)
* JupyterLab

4.3 SETUP OVERVIEW

Getting everything set up for this project took a bit of trial and error. I used a Raspberry Pi 4B, which I connected to a USB webcam and a Sense HAT for visual feedback and the joystick. The first step was installing a 64-bit version of Raspberry Pi OS.

I created a Python 3.9 virtual environment using pyenv, which was necessary because DeepFace doesn’t work well with newer Python versions. Installing DeepFace and TensorFlow took some time due to compatibility issues and long install times, especially when building packages like grpcio from source.

There were also issues with the Raspberry Pi camera — the Pi Camera v2 gave me distorted images and errors with libcamera, so I ended up switching to a USB webcam which worked fine with OpenCV. Remote access to the Pi was done using VNC Viewer, which made it easier to code and debug everything. I used Geany as the code editor on the Pi since I prefer an IDE over the pi’s command line and it got the job done.

5. PROCEDURE AND IMPLEMENTATION

1. Set up Python 3.9 environment with pyenv.
2. Installed all necessary packages using pip.
3. Tested the USB webcam using OpenCV.
4. Wrote most of the early version of the code in Jupyter Notebook using my laptop webcam to prototype everything before moving to the Pi.
5. After the logic worked on my laptop, I transferred the code to the Raspberry Pi and updated it to work with the hardware, including joystick input and Sense HAT feedback.
6. Saved captured images and converted them to embeddings using DeepFace.
7. Stored each user embedding with a unique ID.
8. In recognition mode, the camera runs in a continuous loop, capturing a frame every few seconds.
9. Each captured frame is checked for a match using cosine similarity.
10. The result is shown both on the Sense HAT (green for match, red for unknown) and on the screen with overlay text.

6. Analysis

This project involved a bit of trial and error to get to a working and reliable facial recognition system. One of the most important things I learned was how small adjustments to preprocessing like converting images from BGR to RGB can affect recognition accuracy. At first, my recognition accuracy was seemingly decent but somewhere in the process I started getting bad results, even when I used pictures of the same person. Once I added the image conversion, I saw an improvement in how well the system could recognize faces.

Performance-wise, I found that DeepFace's Facenet512 model could run on the Raspberry Pi surprisingly well, especially when checking one image at a time. Even doing full live recognition with multiple users while on VNC performed well. There was some lag in the video feed but overall it ran smoothly.

7. CONCLUSION

This project taught me a lot about what it takes to build a working AI system on limited hardware. By the end of it, I had a fully functioning facial recognition setup that ran completely on a Raspberry Pi without needing the internet. It’s something that is somewhat close to what’s used IRL and it’s a cool project I could put in my portfolio.

One of the most valuable parts of the project was going from prototyping in Jupyter Notebook to running everything on the Pi’s hardware. I feel that it gave me a better understanding of how to adapt and scale software across different platforms. I had to translate user input from keyboard commands to joystick interaction, adjust how feedback was shown using the Sense HAT, and troubleshoot differences in camera behavior. These are the kinds of skills you only really build by going through the full process yourself.

Overall, this project pushed me in a good way. It combined AI, hardware interaction, image processing, and embedded system troubleshooting all in one project. I’m proud of the results and I’m looking forward to making more cool AI projects. I think I want to try to make my own AI assistant next!

8. IMAGES



9. Code

# imports

**import** cv2

**from** deepface **import** DeepFace

**print(**DeepFace**.**\_\_version\_\_**)**

**import** pickle

**from** numpy **import** dot

**from** numpy**.**linalg **import** norm

**from** sense\_hat **import** SenseHat

**import** time

sense **=** SenseHat**()**

# Sense Hat colors

W **=** **[**255**,** 255**,** 255**]** # White

R **=** **[**255**,** 0**,** 0**]** # Red

G **=** **[**0**,** 255**,** 0**]** # Green

B **=** **[**0**,** 0**,** 255**]** # Blue

Y **=** **[**255**,** 255**,** 0**]** # Yellow

P **=** **[**191**,** 64**,** 191**]** # Purple

\_ **=** **[**0**,** 0**,** 0**]** # Off

# LED patterns

**def** make\_circle**(**hatColor**):**

\_ **=** **[**0**,** 0**,** 0**]**

C **=** hatColor

**return** **[**

\_**,** \_**,** C**,** C**,** C**,** C**,** \_**,** \_**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

\_**,** \_**,** C**,** C**,** C**,** C**,** \_**,** \_

**]**

**def** make\_smiley**(**hatColor**):**

\_ **=** **[**0**,** 0**,** 0**]**

C **=** hatColor

**return** **[**

\_**,** \_**,** C**,** C**,** C**,** C**,** \_**,** \_**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

C**,** \_**,** C**,** \_**,** \_**,** C**,** \_**,** C**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C**,**

C**,** C**,** \_**,** \_**,** \_**,** C**,** \_**,** C**,**

C**,** \_**,** C**,** C**,** C**,** \_**,** \_**,** C**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

\_**,** \_**,** C**,** C**,** C**,** C**,** \_**,** \_

**]**

**def** make\_face**(**hatColor**):**

\_ **=** **[**0**,** 0**,** 0**]**

C **=** hatColor

**return** **[**

\_**,** \_**,** \_**,** C**,** C**,** \_**,** \_**,** \_**,**

\_**,** \_**,** C**,** \_**,** \_**,** C**,** \_**,** \_**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

\_**,** C**,** \_**,** \_**,** \_**,** \_**,** C**,** \_**,**

\_**,** \_**,** C**,** \_**,** \_**,** C**,** \_**,** \_**,**

\_**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,**

\_**,** C**,** C**,** C**,** C**,** C**,** C**,** \_**,**

C**,** \_**,** \_**,** \_**,** \_**,** \_**,** \_**,** C

**]**

# ---------- Enrollment Mode ----------

**print(**"Starting Enrollment Mode"**)**

circle **=** make\_circle**(**W**)**

sense**.**set\_pixels**(**circle**)**

# Webcam Capture

max\_users **=** 2

current\_id **=** 1

known\_faces **=** **[]**

**while** current\_id **<=** max\_users**:**

cam **=** cv2**.**VideoCapture**(**0**)**

cv2**.**namedWindow**(**f"Enrollment - User {current\_id} (Press joystick or ESC)"**)**

circle **=** make\_circle**(**B**)**

sense**.**set\_pixels**(**circle**)**

exit\_requested **=** **False**

latest\_frame **=** **None**

**def** handle\_joystick\_enroll**(**event**):**

**global** exit\_requested

**if** event**.**action **==** "pressed" **and** event**.**direction **==** "middle"**:**

exit\_requested **=** **True**

sense**.**stick**.**direction\_middle **=** handle\_joystick\_enroll

**while** **not** exit\_requested**:**

ret**,** frame **=** cam**.**read**()**

**if** **not** ret**:**

**print(**"Failed to grab frame."**)**

**break**

latest\_frame **=** frame

cv2**.**imshow**(**f"Enrollment - User {current\_id}"**,** frame**)**

key **=** cv2**.**waitKey**(**1**)**

**if** key **==** 27**:**

exit\_requested **=** **True**

time**.**sleep**(**0.01**)**

cam**.**release**()**

cv2**.**destroyAllWindows**()**

**if** latest\_frame **is** **not** **None:**

filename **=** f"enroll\_face\_{current\_id}.jpg"

latest\_frame **=** cv2**.**cvtColor**(**latest\_frame**,**cv2**.**COLOR\_BGR2RGB**)**

cv2**.**imwrite**(**filename**,** latest\_frame**)**

**print(**"Captured image size:"**,**latest\_frame**.**shape**)**

**print(**f"Saved {filename}"**)**

profile\_face **=** make\_face**(**P**)**

sense**.**set\_pixels**(**profile\_face**)**

**print(**"Generating embedding..."**)**

embedding\_obj **=** DeepFace**.**represent**(**img\_path**=**filename**,** model\_name**=**"Facenet512"**,** enforce\_detection**=True)**

embedding **=** embedding\_obj**[**0**][**"embedding"**]**

known\_faces**.**append**({**"id"**:** f"user\_{current\_id}"**,** "embedding"**:** embedding**})**

**print(**f"user\_{current\_id} enrolled."**)**

current\_id **+=** 1

# Save all embeddings

**with** **open(**"known\_faces.pkl"**,** "wb"**)** **as** f**:**

pickle**.**dump**(**known\_faces**,** f**)**

**print(**"All embeddings saved to known\_faces.pkl"**)**

smiley **=** make\_smiley**(**Y**)**

sense**.**set\_pixels**(**smiley**)**

# ---------- Live Recognition Mode ----------

**print(**"\nSwitching to Live Continuous Recognition Mode"**)**

circle **=** make\_circle**(**B**)**

sense**.**set\_pixels**(**circle**)**

cam **=** cv2**.**VideoCapture**(**0**)**

cv2**.**namedWindow**(**"Live Recognition (ESC to exit)"**)**

**with** **open(**"known\_faces.pkl"**,** "rb"**)** **as** f**:**

known\_faces **=** pickle**.**load**(**f**)**

**def** cosine\_similarity**(**a**,** b**):**

**return** dot**(**a**,** b**)** **/** **(**norm**(**a**)** **\*** norm**(**b**))**

last\_recognition\_time **=** time**.**time**()**

**try:**

**while** **True:**

ret**,** frame **=** cam**.**read**()**

**if** **not** ret**:**

**print(**"Failed to grab frame."**)**

**break**

# Convert to RGB for DeepFace

frame\_rgb **=** cv2**.**cvtColor**(**frame**,** cv2**.**COLOR\_BGR2RGB**)**

# Save resized temporary frame

resized\_frame **=** cv2**.**resize**(**frame\_rgb**,** **(**320**,** 240**))**

cv2**.**imwrite**(**"live\_frame.jpg"**,** resized\_frame**)**

current\_time **=** time**.**time**()**

**if** current\_time **-** last\_recognition\_time **>=** 2**:**

**try:**

embedding\_obj **=** DeepFace**.**represent**(**

img\_path**=**"live\_frame.jpg"**,**

model\_name**=**"Facenet512"**,**

enforce\_detection**=False**

**)**

test\_embedding **=** embedding\_obj**[**0**][**"embedding"**]**

best\_match **=** **None**

highest\_similarity **=** **-**1

**for** person **in** known\_faces**:**

similarity **=** cosine\_similarity**(**test\_embedding**,** person**[**"embedding"**])**

**print(**f"Comparing with {person**[**'id'**]**}: {similarity:.4f}"**)**

**if** similarity **>** highest\_similarity**:**

highest\_similarity **=** similarity

best\_match **=** person**[**"id"**]**

threshold **=** 0.6

**if** highest\_similarity **>=** threshold**:**

**print(**f"Match: {best\_match} ({highest\_similarity:.4f})"**)**

circle **=** make\_circle**(**G**)**

sense**.**set\_pixels**(**circle**)**

cv2**.**putText**(**frame**,** f"Match: {best\_match}"**,** **(**10**,** 30**),**

cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 1**,** **(**0**,** 255**,** 0**),** 2**)**

**else:**

**print(**f"Unknown face ({highest\_similarity:.4f})"**)**

circle **=** make\_circle**(**R**)**

sense**.**set\_pixels**(**circle**)**

cv2**.**putText**(**frame**,** "Unknown Face"**,** **(**10**,** 30**),**

cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 1**,** **(**0**,** 0**,** 255**),** 2**)**

last\_recognition\_time **=** current\_time

**except** **Exception** **as** e**:**

**print(**"No face detected or recognition error:"**,** e**)**

# Show the live feed with overlay

cv2**.**imshow**(**"Live Recognition (ESC to exit)"**,** frame**)**

**if** cv2**.**waitKey**(**1**)** **&** 0xFF **==** 27**:**

**break**

**finally:**

cam**.**release**()**

cv2**.**destroyAllWindows**()**

sense**.**clear**()**

**print(**"Live Recognition ended."**)**

references and footnotes

[1] DeepFace - <https://github.com/serengil/deepface>

[2] OpenCV - <https://opencv.org/>

[3] TensorFlow - <https://www.tensorflow.org/>

[4] Raspberry Pi Docs - <https://www.raspberrypi.com/documentation/>