Facial Recognition with Deep Face Library
-
Ensuring Campus Security

New Mexico

SuperComputing Challenge

Final Report

April 10, 2024

Team

Los Alamos High School

Team member:

Hyunoo (Hayden) Kim

Project Mentor:

Michela Ombelli

Executive Summary

In light of increasing concerns about campus security and rising school shooting incidents, my project aims to leverage live face recognition technology to solve this current problem. From taking inspiration from recent advancements in facial recognition systems making them more accessible to the public, my project's purpose is to enhance safety measures on campuses by implementing real-time identification and monitoring capabilities. By utilizing live video feeds from strategically placed cameras across the campus, my program can identify faces in real time within its field of view. This enables instant identification of individuals against a predefined database of authorized personnel, visitors, and potential threats.

At the current stage of development, my system can detect faces, establishing an ROI (region of interest) for every frame. It then pulls from the ROI and analyzes the face using a library called *Deepface*. Due to the lack of time caused by many different time commitments this year, the project was only able to progress to detecting multiple faces at the same time, but only being able to recognize one at a time. Despite this, there is a clear path to solving this issue in the future since I have been able to draw from a folder containing multiple images, proving that multiple-image analysis is possible given my current hardware.

The aim is to make my program work in tandem with other security measures in place and, in terms of implication, I placed a heavy emphasis on privacy by making the system optimized to work with pictures of students and faculty being limited to professional ones — such as school pictures.

Background

In just 2023, there have been 82 school shootings in the United States. This is slightly more than one school shooting every school week — Monday through Friday. Additionally, the rise of campus threats has had a general upward trend over the past decade (with an exception in 2020 when virtual school was largely used and the number of school shootings dropped), making it of utmost importance to find effective measures of preventing casualties and ensuring the safety of school students and faculty.

Methodology and Validation

My approach to creating a face recognition system was a slightly roundabout way, however, it allowed me to gain a deeper understanding of AI and how machine learning works (see Figure 1). Instead of jumping straight into recognition—which is easily possible due to advancements made by AI companies—I built a detection module using OpenCV (Computer Vision) to understand how detection works, then built up from the detection module and integrated Deepface to accomplish face recognition (see Figure 2).

To validate that my model was working properly, I tested it against some of my family's faces to see if the model could identify my face properly inside a semi-crowded environment.

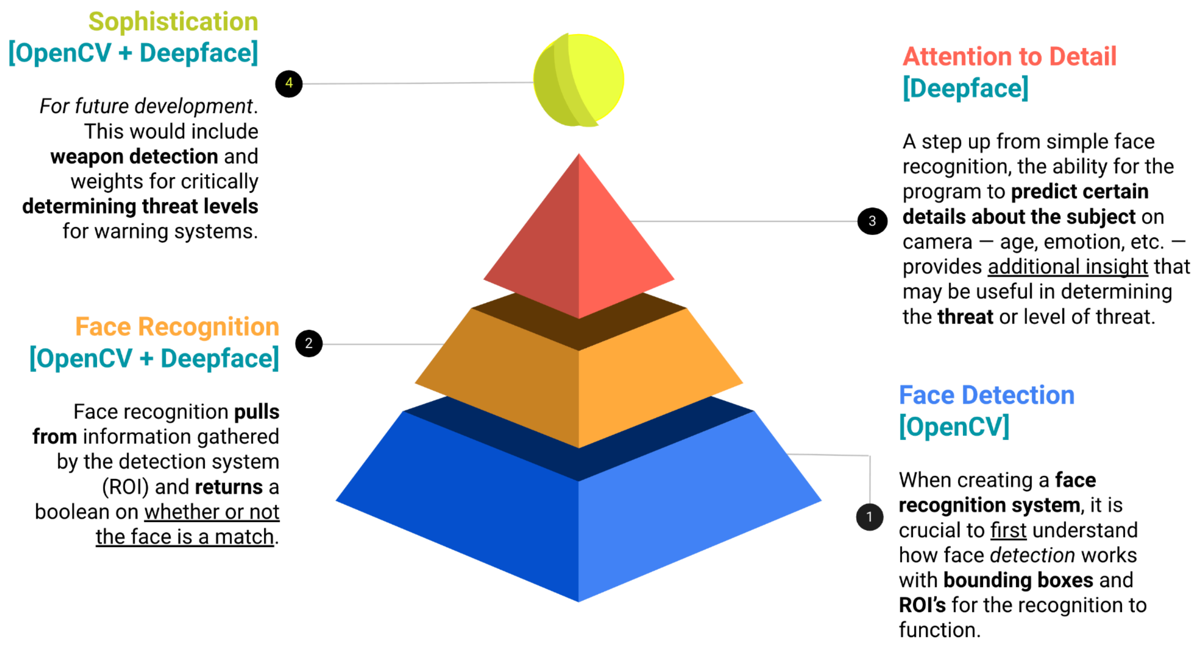
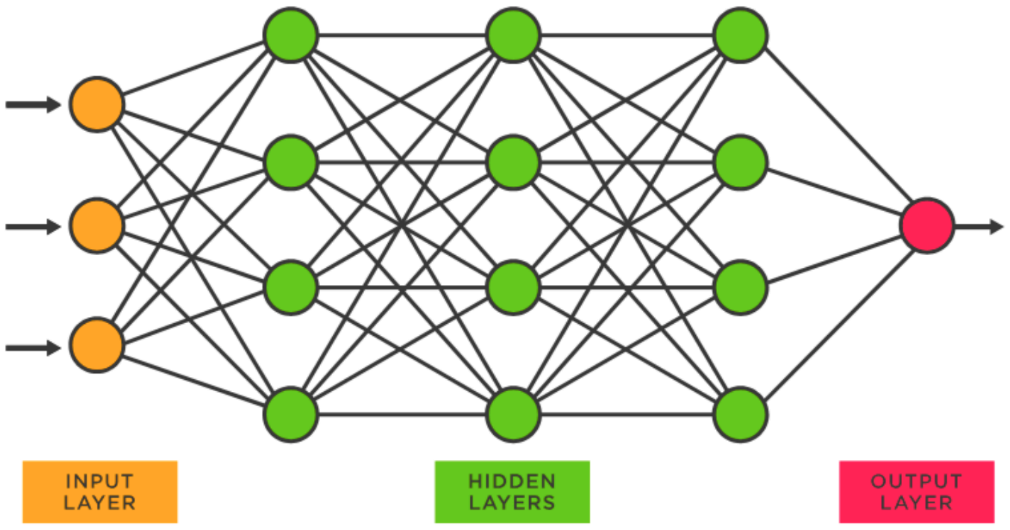

Sophistication
[OpenCV + Deepface]
4
For future development. This would include weapon detection and weights for critically determining threat levels for warning systems.
Attention to Detail
[Deepface]
3
A step up from simple face recognition, the ability for the program to predict certain details about the subject on camera — age, emotion, etc. — provides additional insight that may be useful in determining the threat or level of threat.
Face Recognition
[OpenCV + Deepface]
2
Face recognition pulls from information gathered by the detection system (ROI) and returns a boolean on whether or not the face is a match.
Face Detection
[OpenCV]
1
When creating a face recognition system, it is crucial to first understand how face detection works with bounding boxes and ROI's for the recognition to function.

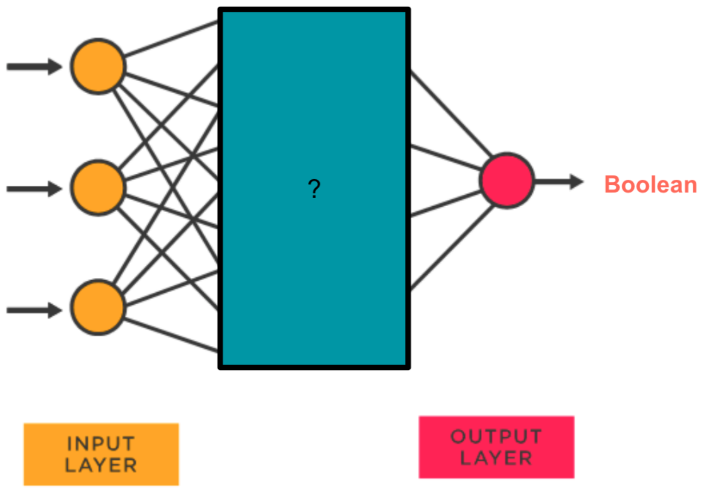
Figure 1. My approach for facial detection and recognition method using various Python-based computer vision modules (OpenCV, Deepface [1,2,3])

Since the project's purpose is to achieve campus security, I ran one additional test to determine the library's capability with my program to predict values on a gradient such as age and emotion. This was to test for further information gathering such as weapon detection or observations about an identified suspect that the system may have been able to use to understand a situation better. Unfortunately, the library didn't prove so useful.

Deepface and most other traditional neural networks:



OpenCV and other modern detection systems:

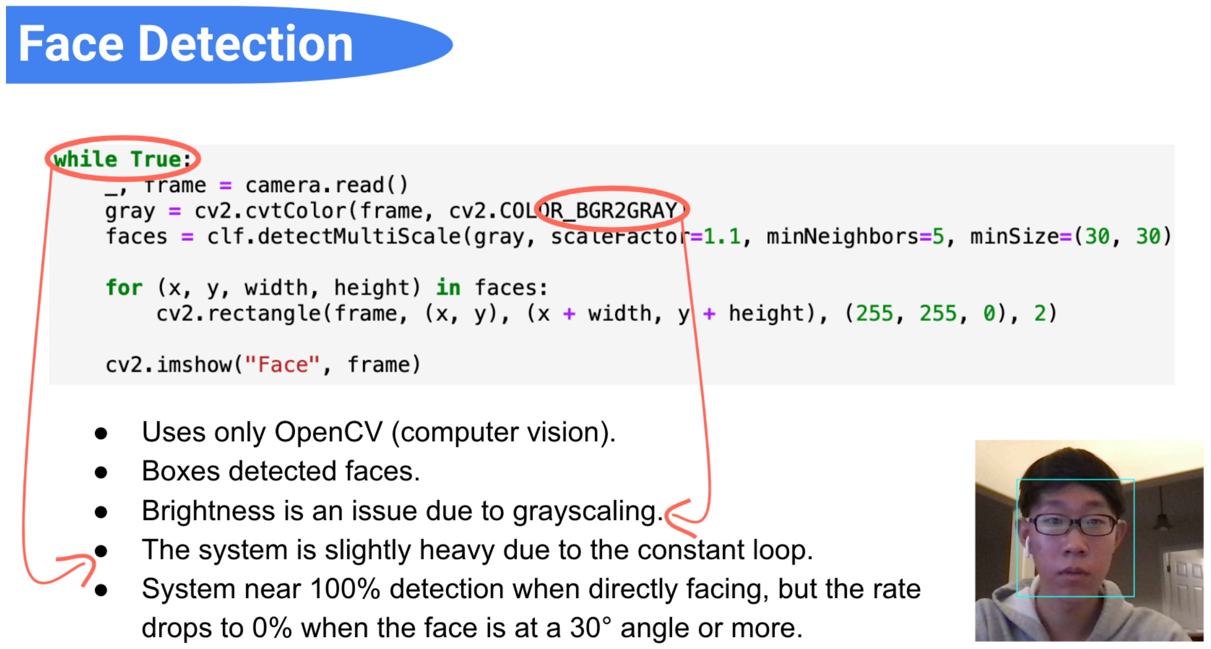


Simple detection has become more streamlined and bulky hidden node complexes are not needed. Instead, the machine learns to identify specific features to return a boolean.

Figure 2. (Top) Traditional neural networks used in Deepface, and (Bottom) streamlined modern detection using machine learning algorithm used in OpenCV [4,5,6]

Results and Conclusion

In conclusion, this project has been pieced together from free, open-source libraries and shows that enhancing campus security with algorithms is not at all implausible for school districts. The current model can pull from a folder containing multiple people (with the limit that it is only able to analyze one face per recognition loop) where each person may have multiple reference photos to help the system better recognize the face (see Figures 3,4 and 5).



Face Detection

```
while True:
    _, frame = camera.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = clf.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, width, height) in faces:
        cv2.rectangle(frame, (x, y), (x + width, y + height), (255, 255, 0), 2)

    cv2.imshow("Face", frame)
```

- Uses only OpenCV (computer vision).
- Boxes detected faces.
- Brightness is an issue due to grayscaling.
- The system is slightly heavy due to the constant loop.
- System near 100% detection when directly facing, but the rate drops to 0% when the face is at a 30° angle or more.

Figure 3. Face detection coding for boxes detected faces using OpenCV.

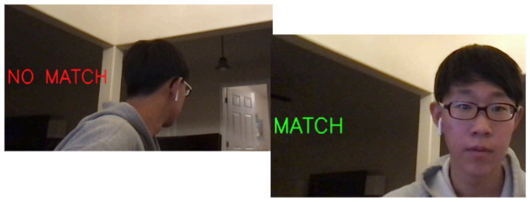
Face Recognition

```python
while True:
    ret, frame = cap.read()

    if ret:
        if counter % 30 == 0:
            try:
                threading.Thread(target = check_face(frame), args = (frame.copy(),)).start()
            except ValueError:
                pass
        counter += 1

        if face_match:
            cv2.putText(frame, "MATCH", (20, 450), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 50), 3)
        else:
            cv2.putText(frame, "NO MATCH", (20, 450), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 3)

        cv2.imshow("video", frame)
```

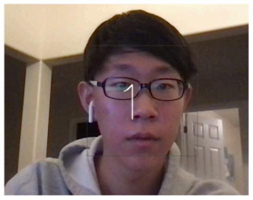


```python
def display_image(image_path):
    image = cv2.imread(image_path)
```

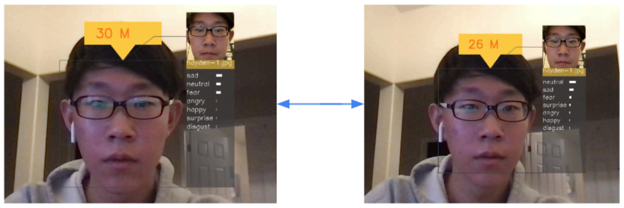
- Uses OpenCV and Deepface (tensorflow).
- Indicates if the faces match or not from a reference image.
- The FPS at which the camera is able to display lags constantly due to the refresh rate of each try.
- System near 100% detection when directly facing, but the rate drops to 0% when the face is looking away 30°+ in any direction.

Figure 4. Face recognition using TensorFlow in Deepface python module.

Attention to Detail



Queue - only guesses every ~5 seconds



1. Age estimation fluctuates too much to be relied upon
2. The time between guesses is consistent but slow. In a large crowd (when everyone is crowding through a door) people can easily be missed.

- Uses Deepface (tensorflow).
- Indicates if the face matches and with which image from a library of reference images.
- The FPS at which the camera is able to display lags even more constantly due to the refresh rate of each try. Uses a queue system to prevent the system from overloading.
- The age estimate is not accurate, mainly for younger ages.
- The system will attempt to force a match between images which makes this impractical for large-scale implementation.
- **Main issue: adjustments cannot be made on my end as this is a function made by Deepface.**

Figure 5. Challenges of facial recognition in Deepface (Examples: inaccurate age estimation, incorrect emotion sensing)

Throughout the project, I have obtained a broader understanding of how machine learning works to establish complex neural networks. With a focus on my future career path, I see this as a massive milestone in progressing in computer science—one of the majors that I am considering. This newfound knowledge of neural networks is something that I value as the most significant achievement that I have made in this project.

Acknowledgments

I would like to acknowledge Daniel Gray, a scientist at LANL who was able to help me understand the complexities of face recognition libraries. He has allowed me to gain a much deeper understanding and appreciation for machine learning.

I would like to acknowledge my father, Jun Kim, a scientist at LANL who encouraged me to continue with my project when things were not going right. He is also experienced in coding and I was able to seek help from him when I did not know how to do something. Both of these individuals played crucial roles in getting my project up to where it is now despite time challenges this year.

References

1. Rasmussen, S.H.R., Ludeke, S.G. & Klemmensen, R. Using deep learning to predict ideology from facial photographs: expressions, beauty, and extra-facial information. *Sci Rep* 13, 5257 (2023). https://doi.org/10.1038/s41598-023-31796-1
2. Tutorials for OpenCV: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
3. Tutorials for Deepface: https://viso.ai/computer-vision/deepface/
4. Neural network: https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
5. Hivalila Hangaragi, Tripty Singh, Neelima N, Face Detection and Recognition Using Face Mesh and Deep Neural Network, Procedia Computer Science, Volume 218, 2023, Pages 741-749, https://doi.org/10.1016/j.procs.2023.01.054.
6. Face recognition with Python using OpenCV: https://www.datacamp.com/tutorial/face-detection-python-opencv

Products (Appendix)

My first model (face detection):

```
import pathlib
import cv2

cascade_path = pathlib.Path(cv2.__file__).parent.absolute() /
"data/haarcascade_frontalface_default.xml"
clf = cv2.CascadeClassifier(str(cascade_path))

camera = cv2.VideoCapture(0)

while True:
    _, frame = camera.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = clf.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),
flags=cv2.CASCADE_SCALE_IMAGE)

    for (x, y, width, height) in faces:
        cv2.rectangle(frame, (x, y), (x + width, y + height), (255, 255, 0), 2)

    cv2.imshow("Face", frame)

    key = cv2.waitKey(1)
    if key == ord("q") or key == 27:  # 27 is the ASCII code for the ESC key
        break

camera.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
print("done")
```

My second model (face recognition):

```
#pip install deepface
import threading
import cv2
from deepface import DeepFace

cap = cv2.VideoCapture(0)
```

```
counter = 0

face_match = False
#print("check 1")

reference_img = cv2.imread("reference.jpg")
#print("check 2")

def check_face(frame):
    global face_match
    try:
        if DeepFace.verify(frame, reference_img.copy())['verified']:
            face_match = True
        else:
            face_match = False
    except ValueError:
        face_match = False

while True:
    ret, frame = cap.read()

    if ret:
        if counter % 30 == 0:
            try:
                threading.Thread(target = check_face(frame), args = (frame.copy(),)).start()
            except ValueError:
                pass
        counter += 1

        if face_match:
            cv2.putText(frame, "MATCH", (20, 450), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255,
50), 3)
        else:
            cv2.putText(frame, "NO MATCH", (20, 450), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0,
255), 3)

        cv2.imshow("video", frame)
    key = cv2.waitKey(1)
    if key == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```