HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE
-------------------- o0o ---------------------

Facial Recognition System with 3D Approximation

Project III

Supervisor: Prof. Nguyen Linh Giang

Student: Nguyen Ba Duong
20214886
duong.nb214886@sis.hust.edu.vn

Hanoi 2024

1. Introduction

Facial recognition systems have become integral in various applications, ranging from device authentication to surveillance systems. Advanced systems like Apple's Face ID use 3D facial recognition with specialized hardware like the TrueDepth camera system to accurately map a user’s face. However, the high cost of such systems makes them inaccessible for many developers and researchers. But only using 2D images for facial recognition raises many concerns about security problems.

Therefore, this project aims to strike a balance between cost and security by developing a facial recognition system that can achieve 3D-like accuracy using only 2D images. The system will focus on creating an accurate and secure face recognition solution by approximating 3D facial structures from 2D images.

2. Problem Description

A Facial Recognition System is a comprehensive technology designed to identify or verify a person's identity by analyzing their facial features from an image or video. This system is built upon several interconnected modules, each responsible for a specific stage in the recognition process. The primary modules are: Face Detection, Face Feature Extraction and Face Authentication.

2.1. Face Detection

The Face Detection module is the first and crucial step in the facial recognition pipeline. Its primary goal is to automatically identify and locate human faces within an image or a frame from a video stream. This involves distinguishing faces from other objects and the background in potentially complex scenes.

Input: A digital image (e.g., JPEG, PNG) or a video frame (a single image within a video sequence).

Output: A list of bounding boxes, each representing the coordinates (e.g., x, y, width, height) of a detected face within the input image.

2.2. Face Feature Extraction

Once a face has been detected, the Face Feature Extraction module comes into play. Its purpose is to capture the unique and distinguishing characteristics of the detected face in a way that can be used for comparison and identification. This involves transforming the raw pixel data of the face region into a compact and informative representation called a feature vector

Input: An image region (cropped from the original image) containing a detected face, typically aligned and normalized.

Output: A feature vector: A numerical representation of the face's unique characteristics. This vector is typically a fixed-length array of floating-point numbers.

2.3. Face Authentication

The Face Authentication module is the final stage of the facial recognition system. Its goal is to determine whether a detected face matches a known identity or to verify a claimed identity. This is achieved by comparing the feature vector of the detected face extracted in the previous stage with feature vectors stored in a database.

Input: A feature vector representing the detected face. A database of feature vectors representing known individuals (the "gallery" feature vectors).

Output:
- Identification (1:N matching): A ranked list of potential matches from the database, along with similarity scores, indicating the likelihood of each match. If no match is found above a certain threshold, the output may indicate an "unknown" individual.
- Verification (1:1 matching): A binary decision (yes/no) indicating whether the feature vector matches the feature vector associated with the claimed identity, along with a confidence score.

3. System Architecture

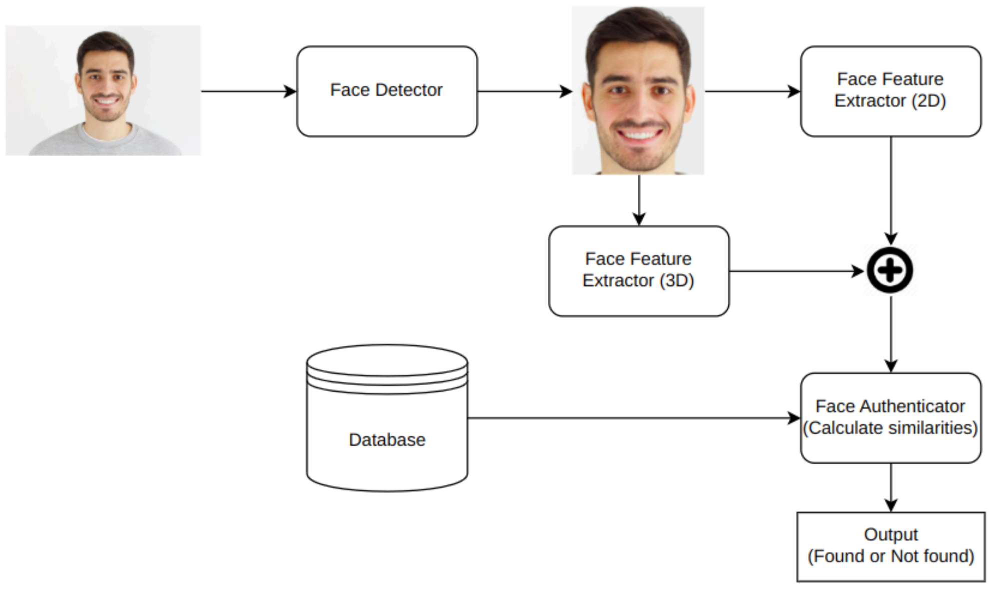

Face Detector
Face Feature Extractor (2D)
Face Feature Extractor (3D)
Database
Face Authenticator (Calculate similarities)
Output (Found or Not found)

Figure 1. Overall Architecture

The facial recognition system, as illustrated in Figure 1, comprises several interconnected components that sequentially process an input image to achieve face recognition. The system pipeline can be broken down into the following stages.

3.1. Input

The system begins with a digital image as input. This image can be captured from various sources, such as a live camera feed, a stored image file (JPEG, PNG, etc.), or a frame extracted from a video.

3.2. Face Detector

The input image is first fed into the Face Detector module. This module is responsible for identifying and localizing human faces within the input image.

The specific algorithm used for face detection can be chosen from various options:

- **Viola-Jones**: Fast but less accurate with variations in pose and lighting.
- **HOG + SVM**: More robust than Viola-Jones but can be slower.
- **Deep Learning** (MTCNN, SSD, etc.): Most accurate, especially with challenging conditions, but computationally more expensive.

To achieve 3D-like accuracy with 2D images, a deep learning-based method would likely be preferred for its higher accuracy.

The Face Detector outputs:

- **Bounding Boxes**: A list of rectangular regions, each defined by coordinates (e.g., x, y, width, height), that enclose the detected faces.
- **Confidence Scores** (Optional): A probability associated with each bounding box, reflecting the detector's confidence that the region contains a face.
- **Cropping**: The detected face regions are cropped from the original image based on the bounding box coordinates. These cropped images are then passed to the next stage.

If no human face is detected in the image by the Face Detector, the system will not move to the next stage.

3.3. Face Feature Extractor (2D)

This module extracts a set of 2D features from the cropped face image. These features are designed to capture the unique visual characteristics of the face in a two-dimensional representation.

We use deep learning techniques, such as FaceNet, to extract 2D facial features. Deep convolutional neural networks are typically employed to derive highly discriminative 2D embeddings directly from pixel data.

Output of this stage is a 2D-features vector (embedding), which is a numerical representation of the face's 2D appearance. The length of this vector is fixed and determined by the chosen feature extraction method.

3.4. Face Feature Extractor (3D)

This is a key module in this system's goal of achieving 3D-like accuracy from 2D images. It attempts to infer or approximate 3D facial structure information from the 2D cropped face image.

Since no actual 3D data is available, the system needs to use methods that can estimate 3D information from 2D. Some potential techniques include:

- **3D Morphable Models** (3DMM): Statistical models that represent 3D face shapes and textures. These models can be fitted to 2D images to estimate the underlying 3D shape.
- **Deep Learning for 3D Reconstruction**: CNNs can be trained to predict 3D face models or depth maps from single 2D images.

The output of this stage is a 3D feature vector (embedding) that represents an approximate 3D facial structure derived from the 2D input face image.

3.5. Feature Fusion

To use both feature vectors 2D and 3D, I implemented the most straightforward approach, indicated in the diagram by the **+** symbol which is concatenation. The 2D and 3D feature vectors are simply joined together, end-to-end, to create a longer feature vector that captures both 2D appearance and (approximated) 3D structural information.

3.6. Face Authenticator

This module performs the actual face recognition (identification) by comparing the combined feature vector with a database of known faces. The system maintains a database of feature vectors (embeddings) representing enrolled individuals. These feature vectors are pre-computed using the same pipeline (2D and 3D feature extraction).

The Face Authenticator calculates the similarity between the combined feature vector of the input face and the feature vectors in the database. Common similarity metrics include:

- **Cosine Similarity**: Measures the angle between vectors (preferred for high-dimensional embeddings).
- **Euclidean Distance**: Measures the straight-line distance between vectors.

The system evaluates the input feature vector against all entries in the database, generating a ranked list of potential matches with their corresponding similarity scores. If the highest score surpasses the predefined threshold, the user is considered a match.

4. Experiment

4.1. Dataset

I conduct experiments to select models for the system using the Labeled Faces in the Wild (LFW) Dataset which includes 13,233 images of 5,749 people.

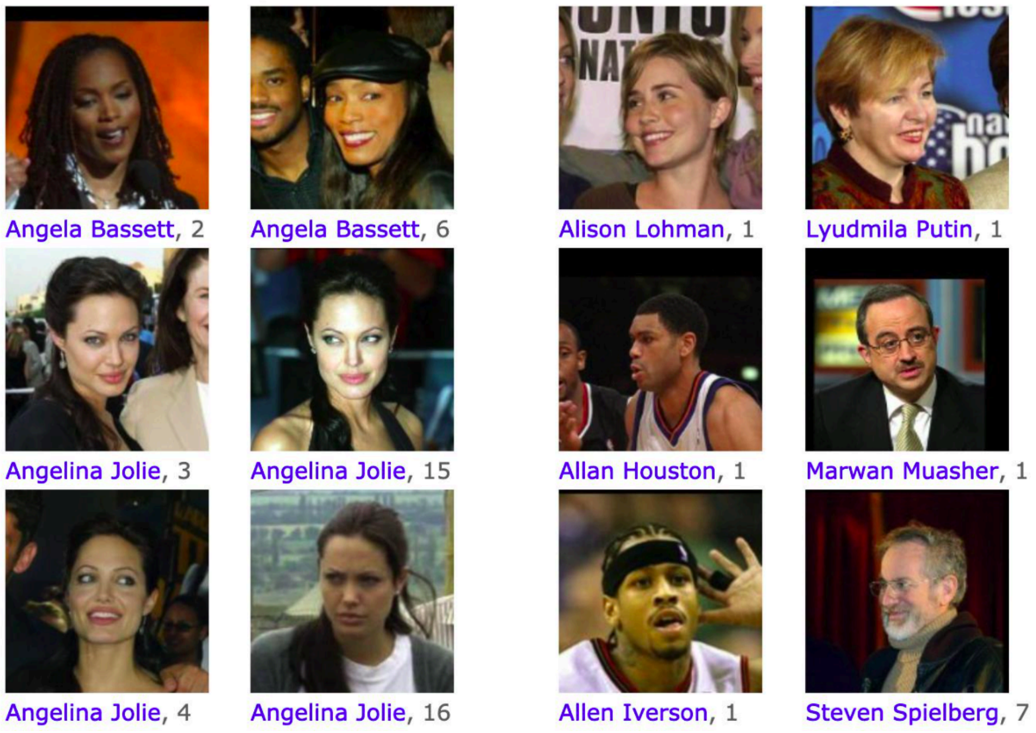


Figure 2. LFW Dataset

4.2. Experimental Setup

4.2.1. Face Detection

There are several accurate and efficient deep learning-based face detection models.

Model	AP@0.5
SSD	0.931
MTCNN	0.915
MediaPipe	0.743
RetinaFace Resnet50	0.994
RetinaFace MobilenetV1	0.994
Dual Shot Face Detector	0.989
YuNet	0.994

Table 1. Comparison of the models, based on AP

For the system, I utilize the MTCNN model, a robust and lightweight deep learning framework known for its accuracy and efficiency in face detection.

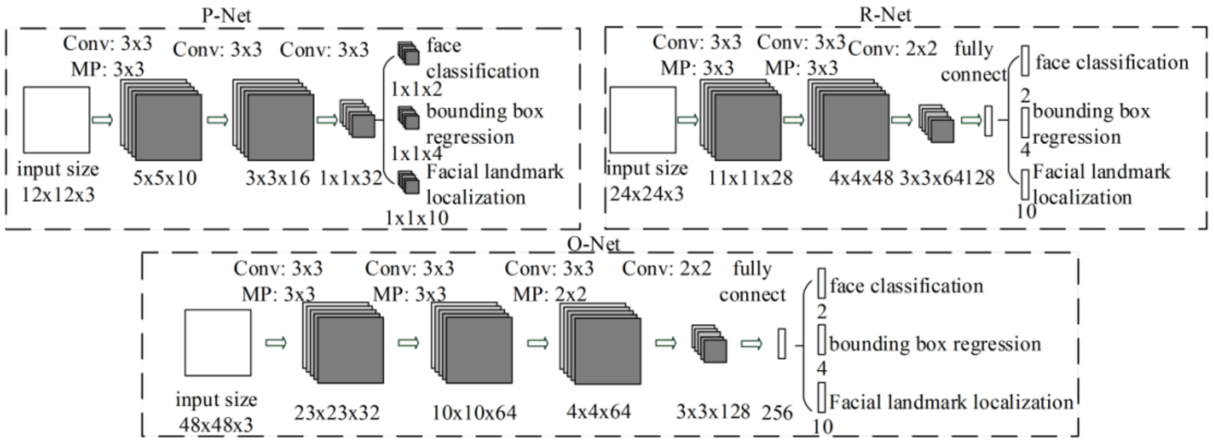
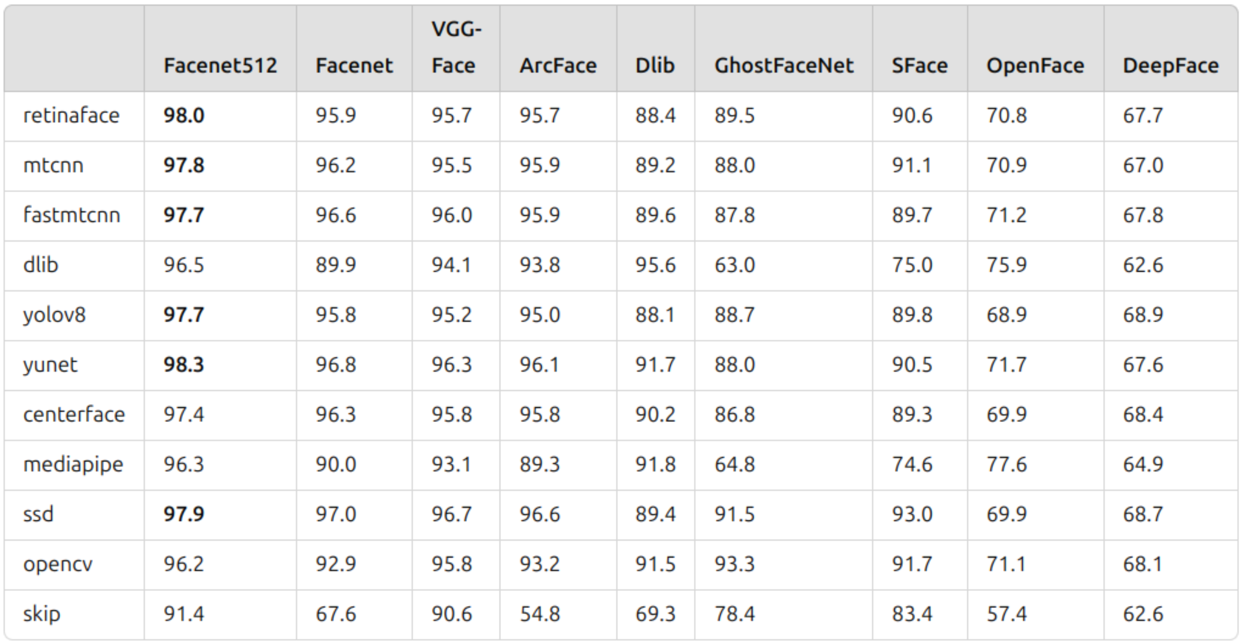


Figure 3. MTCNN Overall Architecture

MTCNN operates through three stages, where the input of each stage is a resized version of the input image. This multi-stage pipeline allows the model to efficiently identify faces of varying sizes and scales by focusing on different levels of detail at each stage.

4.2.2. Face Feature Extraction (2D)

Once the cropped face image is obtained from the face detector model, it is passed to a 2D face feature extractor to generate the face's embedding vector. There are several options to choose:



	Facenet512	Facenet	VGG-Face	ArcFace	Dlib	GhostFaceNet	SFace	OpenFace	DeepFace
retinaface	**98.0**	95.9	95.7	95.7	88.4	89.5	90.6	70.8	67.7
mtcnn	**97.8**	96.2	95.5	95.9	89.2	88.0	91.1	70.9	67.0
fastmtcnn	**97.7**	96.6	96.0	95.9	89.6	87.8	89.7	71.2	67.8
dlib	96.5	89.9	94.1	93.8	95.6	63.0	75.0	75.9	62.6
yolov8	**97.7**	95.8	95.2	95.0	88.1	88.7	89.8	68.9	68.9
yunet	**98.3**	96.8	96.3	96.1	91.7	88.0	90.5	71.7	67.6
centerface	97.4	96.3	95.8	95.8	90.2	86.8	89.3	69.9	68.4
mediapipe	96.3	90.0	93.1	89.3	91.8	64.8	74.6	77.6	64.9
ssd	**97.9**	97.0	96.7	96.6	89.4	91.5	93.0	69.9	68.7
opencv	96.2	92.9	95.8	93.2	91.5	93.3	91.7	71.1	68.1
skip	91.4	67.6	90.6	54.8	69.3	78.4	83.4	57.4	62.6

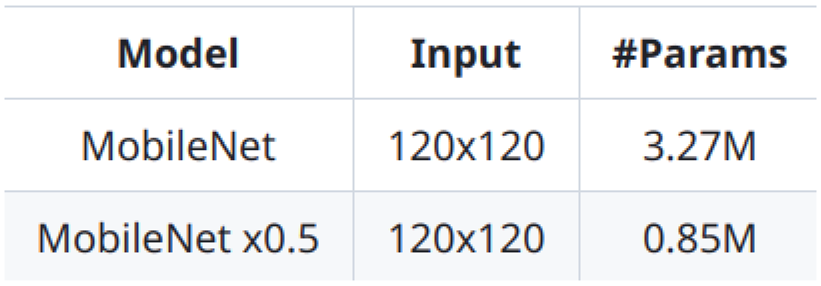
Table 2. Performance Matrix in accuracy

In Table 2, the models are compared using cosine similarity when alignment is False on LFW dataset. The first row lists the 2D face feature extractor models, and the first column lists the face detector models.

In the final system, I use the FaceNet model for the 2d face feature extraction task as it offers a good balance between being lightweight and providing moderate accuracy when used with MTCNN.

4.2.3. Face Feature Extraction (3D)

The 3DDFA_V2 model is used for 3D face feature extraction. It can generate 3D face embeddings from a 2D image. It's a lightweight model that comes with two different backbone options.



Model	Input	#Params
MobileNet	120x120	3.27M
MobileNet x0.5	120x120	0.85M

Table 3. Backbone options for 3DDFA_V2

In the system, I use the MobileNetx0.5 as the backbone for the 3DDFA_V2 model.

4.2.4. Face Authentication

I created a SQLite database to store users' information along with their face embeddings. During inference, the system combines the 2D and 3D embeddings into a single embedding using concatenation. It then calculates the cosine similarity score between this combined embedding and each embedding stored in the database to calculate a score.

$$score = \frac{embedding \cdot embedding\_db}{|embedding| \times |embedding\_db|}$$

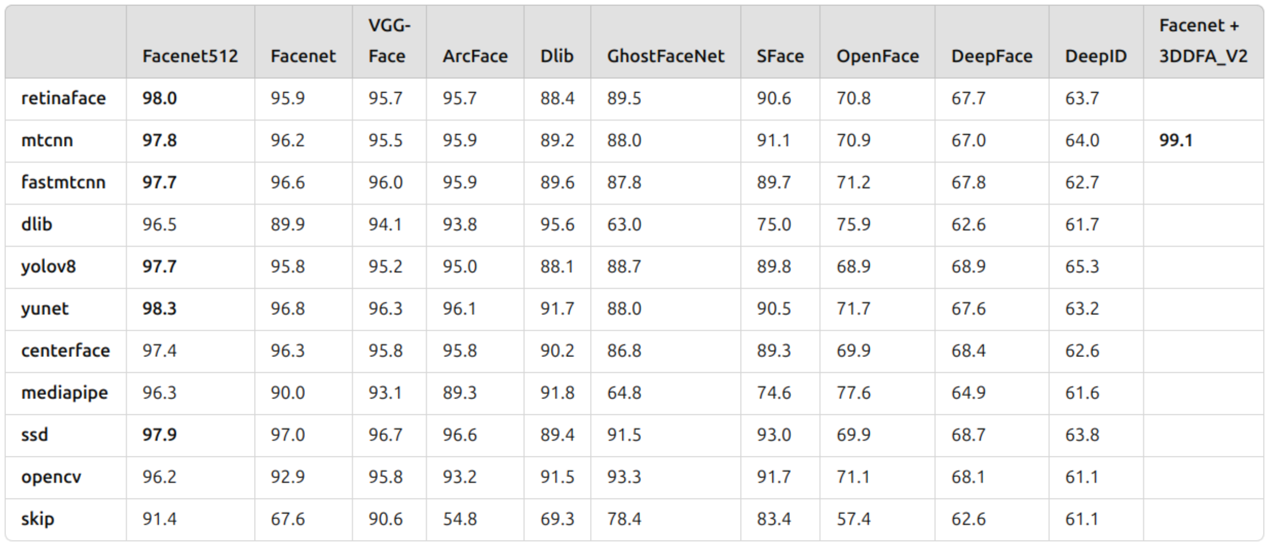
where: *embedding* is the concatenated embedding during inference.
embedding_db is one of the embeddings retrieved from the database.
score is the similarity score calculated using embedding and embedding_db.

Once the scores are calculated, if the highest score surpasses the specified threshold, the user is identified. If the score does not exceed the threshold, no match is found.

4.2.5. Hardware Setup

The system is evaluated on a computer with Ubuntu 22.04, equipped with an Intel Core i7-11800H CPU, an RTX 3050 Ti Laptop GPU, and 16GB of RAM.

4.3. Experimental Result



	Facenet512	Facenet	VGG-Face	ArcFace	Dlib	GhostFaceNet	SFace	OpenFace	DeepFace	DeepID	Facenet + 3DDFA_V2
retinaface	**98.0**	95.9	95.7	95.7	88.4	89.5	90.6	70.8	67.7	63.7	
mtcnn	**97.8**	96.2	95.5	95.9	89.2	88.0	91.1	70.9	67.0	64.0	**99.1**
fastmtcnn	**97.7**	96.6	96.0	95.9	89.6	87.8	89.7	71.2	67.8	62.7	
dlib	96.5	89.9	94.1	93.8	95.6	63.0	75.0	75.9	62.6	61.7	
yolov8	**97.7**	95.8	95.2	95.0	88.1	88.7	89.8	68.9	68.9	65.3	
yunet	**98.3**	96.8	96.3	96.1	91.7	88.0	90.5	71.7	67.6	63.2	
centerface	97.4	96.3	95.8	95.8	90.2	86.8	89.3	69.9	68.4	62.6	
mediapipe	96.3	90.0	93.1	89.3	91.8	64.8	74.6	77.6	64.9	61.6	
ssd	**97.9**	97.0	96.7	96.6	89.4	91.5	93.0	69.9	68.7	63.8	
opencv	96.2	92.9	95.8	93.2	91.5	93.3	91.7	71.1	68.1	61.1	
skip	91.4	67.6	90.6	54.8	69.3	78.4	83.4	57.4	62.6	61.1	

Table 4. Models results in accuracy

The models are compared using Cosine Similarity on the LFW dataset. The first row lists the 2D face feature extractor models, and the first column lists the face detector models.

As shown in the table, our system's result is higher than the combination of more complex models. This confirms that using 3D approximation, even with a combination of lightweight models, can improve performance of the whole system.

	CPU	GPU
Inference Time (ms)	119	83

Table 5. System Inference speed on different devices

For two given images, the system's inference time is approximately 120 milliseconds using CPU and 83 milliseconds using GPU. This demonstrates the system's efficiency, highlighting its ability to process data quickly while maintaining performance, even when utilizing lightweight models.

5. Conclusion

In this project, I developed a facial recognition system that incorporates a 3D approximation module to enhance performance. Although there are still limitations, such as the current SQLite database being less than optimal, one solution is to utilize databases capable of storing vectors, like MongoDB. Additionally, MTCNN and FaceNet could be replaced with more modern models.

References

[1] Ying et al., 2022, 3D face recognition: A comprehensive survey in 2022 (https://link.springer.com/article/10.1007/s41095-022-0317-1)

[2] Labeled Faces in the Wild (https://vis-www.cs.umass.edu/lfw/)

[3] What is Face Detection? (https://learnopencv.com/what-is-face-detection-the-ultimate-guide/)

[4] Serengil et al., 2024, A Benchmark of Facial Recognition Pipelines and Co-Usability Performances of Modules (https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077)