UNIVERSITÀ DEGLI STUDI DE L'AQUILA

Erasmus+

Politechnika Śląska

BLENDED INTENSIVE PROGRAMME (B.I.P) IN MATHEMATICAL MODELLING

L'Aquila, June - July 2024

PROJECT REPORT :

FACE RECOGNITION APPLICATION

BASED ON MTCNN & FACENET

Prepared by :

- Mohamed Amine Harbaoui
- Oussama Bersellou

Introduction:

Face detection and recognition are key areas of research in computer vision that have many practical applications in the real world.

These technologies allow computers to detect and recognize human faces in images or videos, which can be used in many contexts, such as security, surveillance, speech recognition, content recommendation, and social media.

There are many algorithms and models that are used for face detection and recognition, such as **MTCNN (Multi-Task Convolutional Neural Network)** and **FaceNet**. The performance of these algorithms has increased significantly in recent years thanks to advances in machine learning and computing power.

However, there are still challenges to overcome, particularly in terms of data reliability and confidentiality.

1.Face Detection

1.1Introduction to Face Detection:

Face detection is the process of identifying all regions of an image that contain a face, regardless of its position, orientation, and three-dimensional lighting states. This step is crucial for facial recognition, which involves identifying a person from their face.

Face detection is a rapidly evolving field, with new techniques and approaches that are constantly being developed.

1.2 Face Detection Techniques:

There are several techniques for face detection, including **feature-based algorithms**, **model-based algorithms**, and **machine learning-based algorithms**.

Feature-based algorithms use specific features of the face, such as face proportions, eye shape, and mouth shape, to detect faces in images. These algorithms are often fast and efficient, but they can be sensitive to variations in the image, such as lighting and exposure.

Model-based algorithms use predefined face patterns to detect faces in images. These algorithms are generally more robust to image variations than feature-based algorithms, but they can be slower and less accurate.

Machine learning-based algorithms use training data to learn how to detect faces in images. These algorithms can be very efficient, but they often require large amounts of training data and can be expensive to train.

1.3 Advantages and disadvantages of different techniques:

Each of the face detection techniques has its own advantages and disadvantages.

Feature-based algorithms are generally fast and efficient, but they can be sensitive to image variations.

Model-based algorithms are generally more robust to image variations, but they can be slower and less accurate.

Machine learning-based algorithms can be very good, but they often require large amounts of training data and can be expensive to train.

It is important to choose the most appropriate technique according to the specific needs of each project.

1.4 Latest Developments in Face Detection:

There have been many recent developments in the field of face detection, including the use of deep learning models to improve detection performance.

Convolutional neural networks (CNN) have been widely used for face detection, especially for the detection of faces in large-scale images. CNNs have shown very good performance for face detection due to their ability to extract features from images in a robust manner.

There have also been efforts to improve the robustness of face detection algorithms to variations in the image, such as lighting and pose. Techniques such as face alignment and estimation of face points of interest have been used to assist in the detection of faces in difficult conditions.

1.5 Conclusion:

In summary, face detection is a key area of computer vision that involves identifying all regions of an image that contain a face.

There are several techniques for face detection, each with its own advantages and disadvantages. There have been many recent developments in this field, including the use of deep learning models and improving the robustness of algorithms to image variations.

Face detection is crucial for facial recognition and remains a rapidly evolving field with new research and development underway.

2.Convolutional Neural Networks "CNN"

2.1Introduction to Convolutional Neural Networks:

Convolutional neural networks (CNN) are a type of artificial neural network that have been widely used for face detection and recognition.

Their functioning is inspired by biological processes.

2.2 Why CNNs:

CNNs have been very successful at face detection and recognition due to their ability to take advantage of the spatial structure of the input data. They have also been used successfully in other areas of computer vision, such as object recognition and image segmentation.

However, CNNs are specifically designed to process input images. Their architecture is then more specific. In addition to its filtering function, the interest of the convolutional part of a CNN is that it allows the extraction of specific characteristics to each image by compressing them in order to reduce their initial size, via subsampling methods such as Max-Pooling.

2.3 How CNN Works:

Their mode of operation is as follows:

- CNNs use convolution, pooling, and full-connection layers to extract features from images and perform the final classification.
- Convolution layers apply filters to the input data to look for specific patterns, while pooling layers reduce the size of the image by selecting the maximum values in each area of the image.
- Fully-connected layers are layers of classical neurons that perform the final classification using probabilities.

The user provides an image as input in the form of a pixel matrix. It has 3 dimensions:

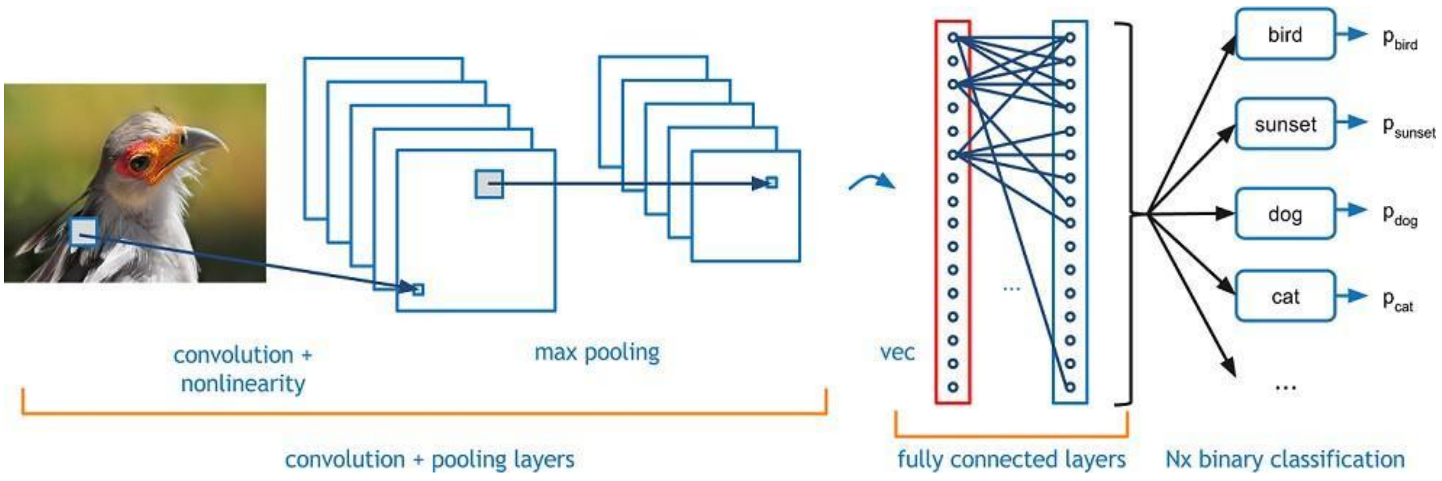
- Two dimensions for a grayscale image.
- A third dimension, with a depth of 3 to represent the essential RGB colors (red, green, blue).

2.4 CNN Network Layers:

Neurons in a fully or partially meshed multilayer structure make up a conventional neural network.
These structures reach their limits when it comes to images, since the number of inputs corresponding to the number of pixels is required.
The number of layers and the connections between them will be enormous and can only be managed by very powerful computers.
The different layers form a convolutional neural network.

The basic principle is a partial grid reaction neural network. A convolutional neural network architecture is formed by a stack of processing layers. There are five types of layers for a convolutional neural network, (see Figure 5).

FIGURE 5 – Types of layers in a convolution network



- Convolution layer (CONV).
- Pooling layer
- Correction layers (Relu)
- Fully Connected Layer (FC)
- Softmax layer.

In this section, we will explain how these different layers work and what are their characteristics

2.4.1 Convolutional Layer:

The convolution layer is the key component of convolutional neural networks, and is always at least their first layer. Its main purpose is to extract features from the input image.

The convolution layer therefore receives several images as input, which can be represented in the form of pixel value matrices.

A digital image with three RGB planes (channels) is represented by three superimposed 2D matrices (each matrix represents a color), and for each, its pixel values range from 0 to 255. Applying a convolution to this image is the same as calculating the convolution of each of them with each filter.

The filters correspond exactly to the characteristics that we want to find in the images. This operation is translated into an element-by-element multiplication between the two matrices and finally into an addition of the multiplication outputs. The final integer of this calculation forms a single element of the output matrix.

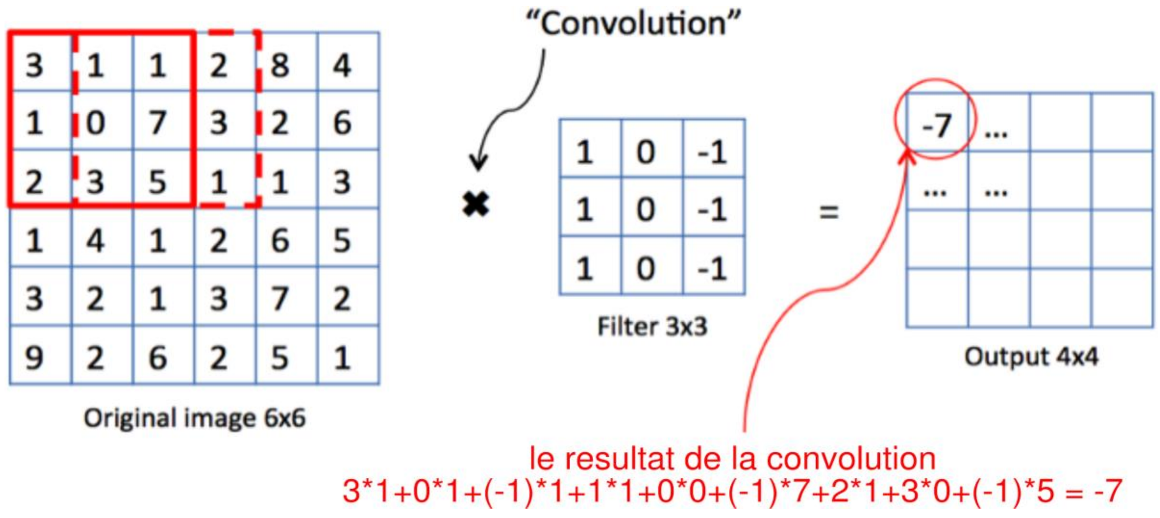


FIGURE 6 – The Convolutional Layer

Following the example below (FIGURE 6), the filter (core) is applied to detect the vertical edges of a 2D image.

A value of 1 on the core filters out the brightness, while -1 highlights darkness and 0 the gray of the original image slips in.

In this example, the filter values have already been decided in the convolution.

The output size is calculated based on the dimensions of the filter and how it slides through the image.

In the previous example, the filter moves in with 1 step and covers the entire image of another. However, there is a problem:

By moving the filter in this way, the pixels at the edges are less affected by the filter than the pixels in the image. In addition, the output image decreases with each convolution, but the problem is quickly solved with "filling".

The image is augmented by a number of zero-value lines all around its edge to allow the filter to slide up and maintain an output size equal to the input.

See the figure:

0	0	0	0	0	0	0	0
0	3	1	1	2	8	4	0
0	1	0	7	3	2	6	0
0	2	3	5	1	1	3	0
0	1	4	1	2	6	5	0
0	3	2	1	3	7	2	0
0	9	2	6	2	5	1	0
0	0	0	0	0	0	0	0

FIGURE 7 – The padding

The formula for calculating the size of the output, knowing the size of the filter (f), the pitch (s), the pad (p) and the size of the input (n):

Input: $(n \; x \; n \; x \; n_c)$ Filter: $(f \; x \; f \; x \; n_c)$ Output: $\left(\left[\frac{n+2p-f}{s}+1\right] x \left[\frac{n+2p-f}{s}+1\right] x \; n_c'\right)$

So far, we've only seen one filter at a time, but we can apply more than one.

Each filter brings us its own way out. We can stack them all together and create an output volume, such as the following figure:

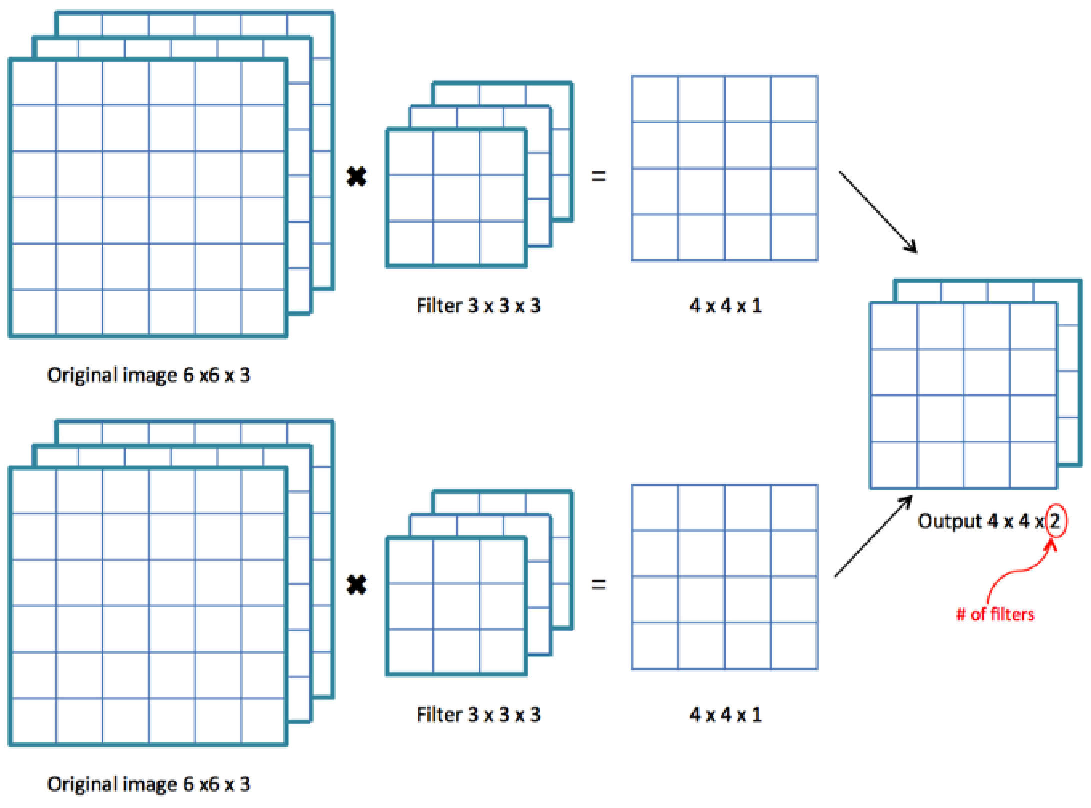


FIGURE 8 – the application of several types of filters at the same time

The final step that leads us to a convolutional neural layer is to add bias and a non-linear function.

2.4.2 Pooling layer:

There are two types of pooling layers; the maximum pooling and the average pooling.

Max pooling: A filter will be set, which will then be dragged into the inlet, the largest element in the region covered by the filter will be taken.

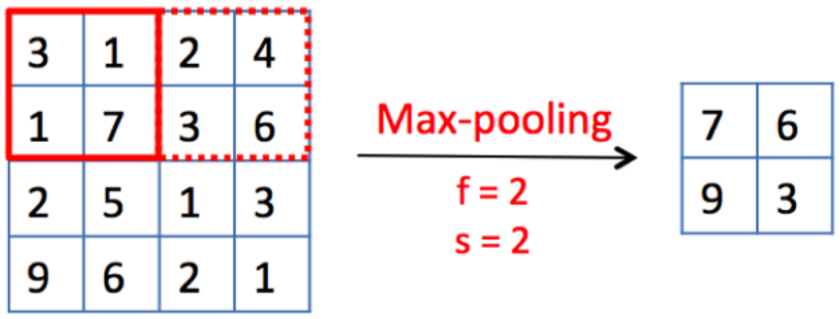


FIGURE 9 – max pooling

Average pooling: As the name suggests, it keeps the average of the values encountered in the filter.

2.4.3 Correction Layers (Reread):

The ReLU (Rectified Linear Units) correction layer therefore replaces all negative values received as inputs with zeros. It acts as an activation function.

ReLU designates the real nonlinear function defined by:

$$\mathrm{ReLU}(x)=\max(0,x).$$

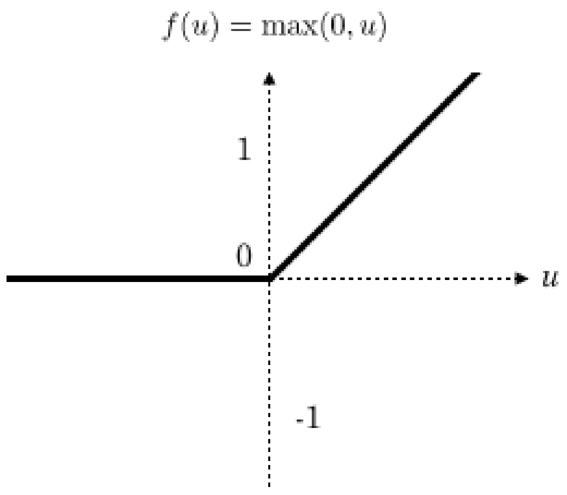


FIGURE 10 – Appearance of the ReLU function

2.4.4 Fully Connected Layer (FC)) :

The fully connected layer is simply a reactive neural network.

The fully connected layers form the last layers of the network. The output of the pooling layer represents the input of the fully connected layer, after it has been flattened.

This flattened vector is then connected to a few fully connected layers that are identical to artificial neural networks and perform the same mathematical operations. For each layer of the artificial neural network, the following calculation is performed:
g(W.x + b)

where

x: is the input vector of dimension [p, 1]

W: is the weighting matrix with dimension [p, n] where, p is the number of neurons in the previous layer and n is the number of neurons in the stream layer.

b : is the bias vector of dimension [p, 1]

g: is the activation function, which is usually ReLU.

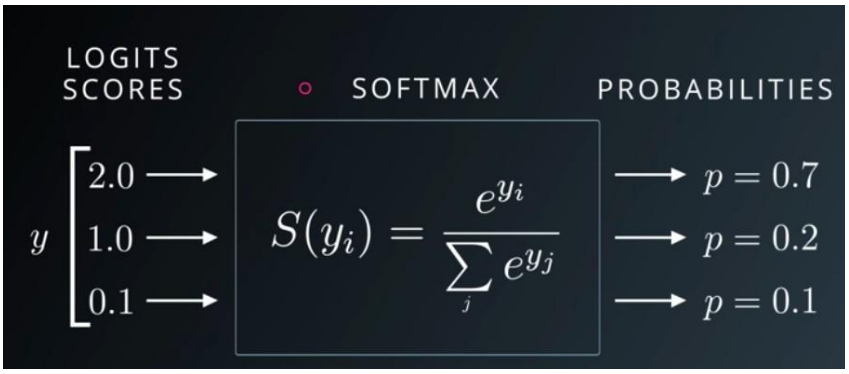
This calculation is repeated for each layer.

2.4.5 Classification output layer (the softmax layer):

The final layer uses the softmax activation function (instead of ReLU) which is used to get probabilities that the input is in a particular class (classification).

In Figure 11 the Softmax function transforms the logits [2.0, 1.0, 0.1] into probabilities [0.7, 0.2, 0.1] and that the probabilities add up to 1. Which brings us to having the probabilities of the object in the image belonging to the different classes.

FIGURE 11 – The softmax function



The input images are classified as labels, which summarizes how the convolutional neural network works.

3. The MTCNN

3.1 Introduction to MTCNN:

MTCNN (Multi-Task Convolutional Neural Network) is a convolutional neural network that has been designed to perform multiple image processing tasks simultaneously, including face detection, face alignment, and facial point of interest estimation. MTCNN has been widely used in face detection and recognition projects due to its ability to perform these tasks efficiently and accurately.

The MTCNN framework relies on a three levels, three-stage cascading architecture of deep convolutional networks carefully designed to predict the location of faces and landmarks such as eyes, nose, and mouth in coarsely to fine manner.

A new strategy for extracting hard samples online further improves performance in practice.

3.2 How MTCNN Works:

For the implementation of the face detection system, MTCNN was used as a method to integrate two tasks (recognition and alignment) using multi-tasking learning.

- First, it uses a shallow CNN to quickly generate candidate windows.

- In a second step, it refines the proposed candidate window using a more complex CNN.

- Finally, in the third step, it uses a third, more complex CNN than the others to further refine the results and generate facial cues.

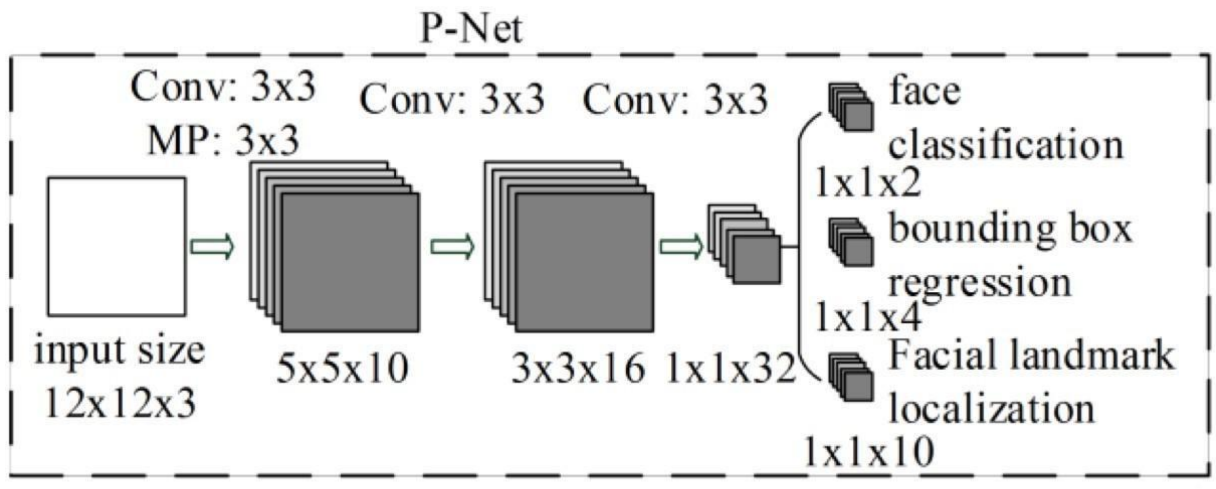
3.3 MTCNN Model Architecture:

It involves taking the image and scaling it at different scales to build the image pyramid, which is the input of the next three-levels waterfall network.

Step 1: The Proposal Network (P-Net)

This first stage is a fully convolutional network (FCN). The difference between a CNN and an FCN is that a fully convolutional network does not use a convolution layer to extract features from the input data. FCNs, on the other hand, have fully connected layers where every neuron in the next layer is connected to every neuron in the previous layer.

This proposal network is used to obtain candidate windows and their bounding box regression vectors.



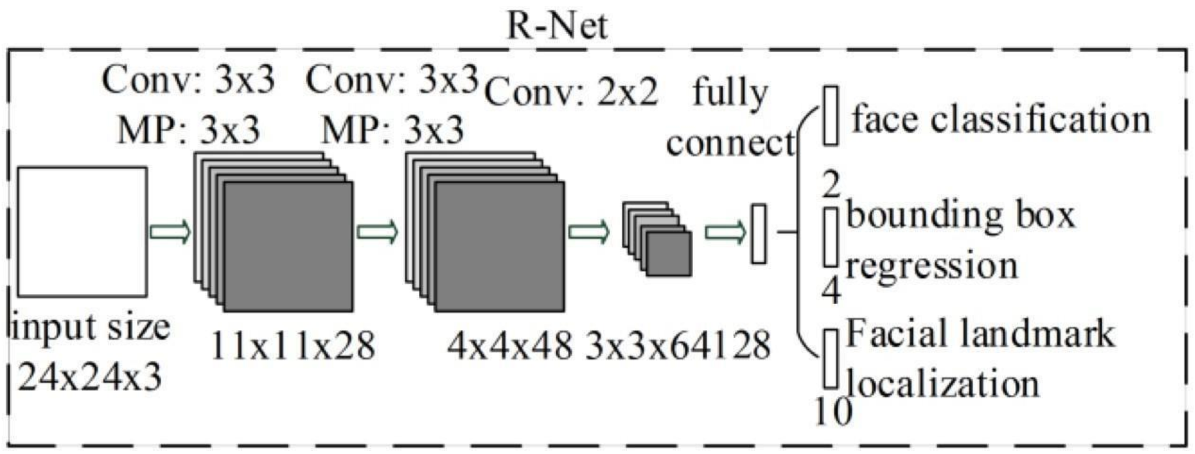
Bounding box regression is a popular technique for predicting the location of boxes when the goal is to detect an object of a certain predefined class, in this case faces.

After obtaining the vectors of the bounding boxes, some refinement is performed to combine the overlapping regions.

The end result of this step is the set of candidate windows after refinement to reduce the volume of candidates.

Step 2: The refining network (R-Net)

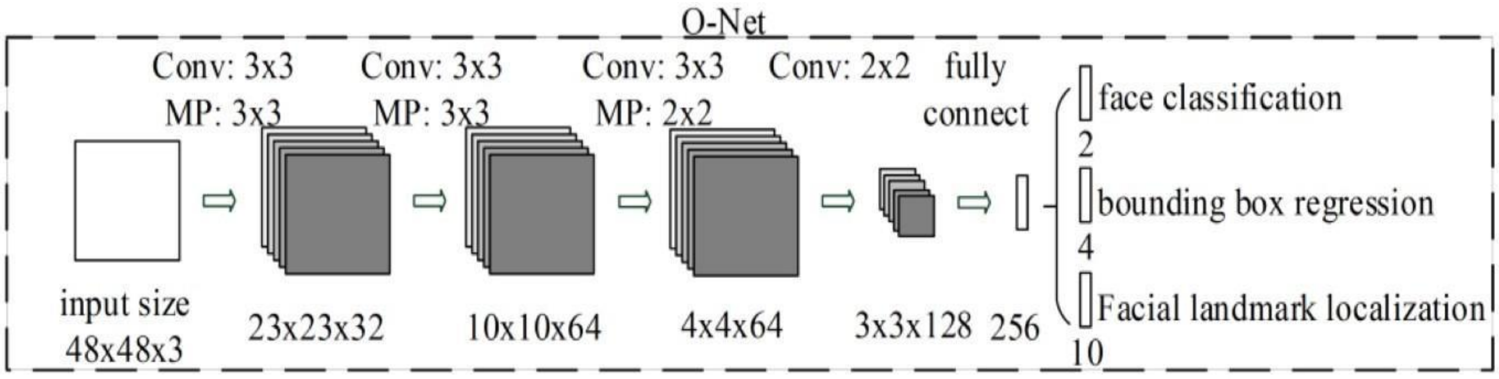
All P-Net candidates are introduced into the refinement network. Note that this network is a CNN, not an FCN like the previous one, because there is a dense layer at the last stage of the network architecture. The R-Net further reduces the number of candidates, performs calibration with bounding box regression, and uses non-maximal suppression (NMS) to merge overlapping candidates.



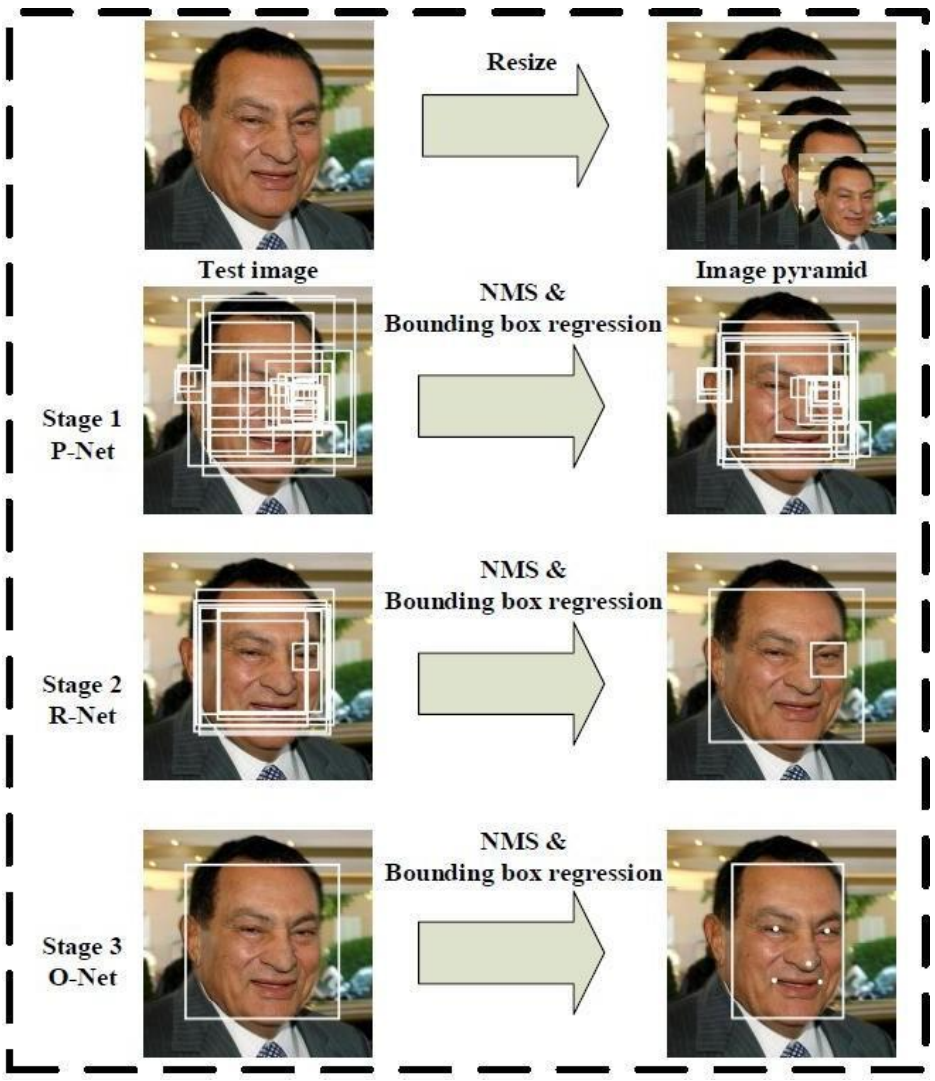
The R-Net network indicates whether the input is a face or not, a four-element vector which is the bounding box of the face, and a ten-element vector for the location of the face's landmarks.

Step 3: The Output Network (O-Net)

This step is similar to the R-Net see the following figure, but this output network aims to describe the face in more details and provide the positions of the five facial landmarks for the eyes, nose, and mouth.



The processing steps of the MTCNN model are summarized as follows:



- **Steps of the first processing block:**

1. Input image
2. Create multiple copies of different sizes of the image
3. Introducing the different images into the P-Net network
4. Gather P-Net Output
5. Remove selection frames with low confidence
6. Convert 12 x 12 core coordinates to "unresized image" coordinates
7. apply NMS method on images at different sizes
8. Group bounding boxes into squares

- **Steps of the second processing block:**

1. Introducing images into the R-Net network
2. Gather the R-Net output
3. Remove selection frames with low confidence
4. apply the NMS method for all boxes

5. Convert bounding box coordinates to "unresized image" coordinates
6. Reshape bounding boxes into squares

• **Steps of the third processing block:**

1. Feed images into O-Net
2. Gather O-Net Output
3. Remove selection boxes with low confidence
4. Convert bounding box and facial cue coordinates to "unresized image" coordinates
5. apply the NMS method

• **Final results:**

1. Consolidate all coordinates and confidence levels into a dictionary

2. Return the dictionary

3.4 MTCNN Performance:

MTCNN was compared to other face detection and recognition algorithms and showed very good performance. In particular, MTCNN was able to detect faces in high-resolution images with high accuracy and low latency. MTCNN was also able to effectively handle image variations, such as lighting and exposure, making it particularly useful for real-time applications.

3.4 Conclusion:

In summary, MTCNN is a convolutional neural network that has been designed to perform multiple image processing tasks simultaneously, including face detection, face alignment, and facial point of interest estimation.

MTCNN has shown very good performance compared to other face detection and recognition algorithms and has been used in many real-world applications.

MTCNN is a powerful tool for face detection and recognition and remains a rapidly evolving field with new research and development underway.

4. Facial recognition

Facial recognition is the use of machine learning and computer vision to identify people's faces in images or videos.

Facial recognition has many practical real-world applications, such as identity verification for unlocking phones, security monitoring, and detecting criminals.

Facial recognition includes several steps of image processing, including face detection, face alignment, and facial recognition itself.

Face detection is the process of locating faces in the image, while face alignment is the process of normalizing faces in a common orientation to facilitate subsequent facial recognition. Facial recognition is the process of comparing the characteristics of faces to those of faces previously recorded in the database to identify the face.

Facial recognition is a rapidly evolving field with new research and development underway. However, there are still challenges to overcome, especially with regard to image variations and data biases. Image variations, such as lighting and pose, can make facial recognition difficult, while data bias can lead to uneven performance across groups of people. Despite these challenges, facial recognition remains a powerful tool for identifying people and its use continues to expand in many practical applications.

5. Facenet

5.1 Introduction to Facenet:

Facenet is a convolutional neural network that was designed for facial recognition. Facenet was developed by Google and uses CNNs to extract features from faces. Facenet has been widely used in facial recognition projects due to its ability to perform recognition efficiently and accurately.

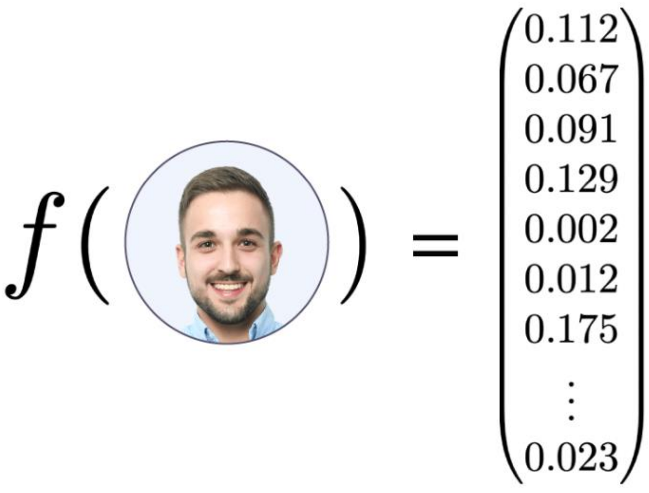
5.2 How does Facenet work:

Facenet works by using convolution and pooling layers to extract features from face images. These features are then used to perform facial recognition by comparing the features of a given face to those of faces previously recorded in the database.

FaceNet takes as input an image of a person's face see the following figure, and produces a vector of 128 numbers that represent the most important features of a face.

In machine learning, this vector is called embedding. Why this name? Because all the important information in an image is embedded in this vector.

Basically, FaceNet takes a person's face and compresses it into a 128-digit vector.



+

Embeddings are vectors and we can interpret vectors as points in the Cartesian coordinate system. This means that we can plot the image of a face in the coordinate system using its embeddings.

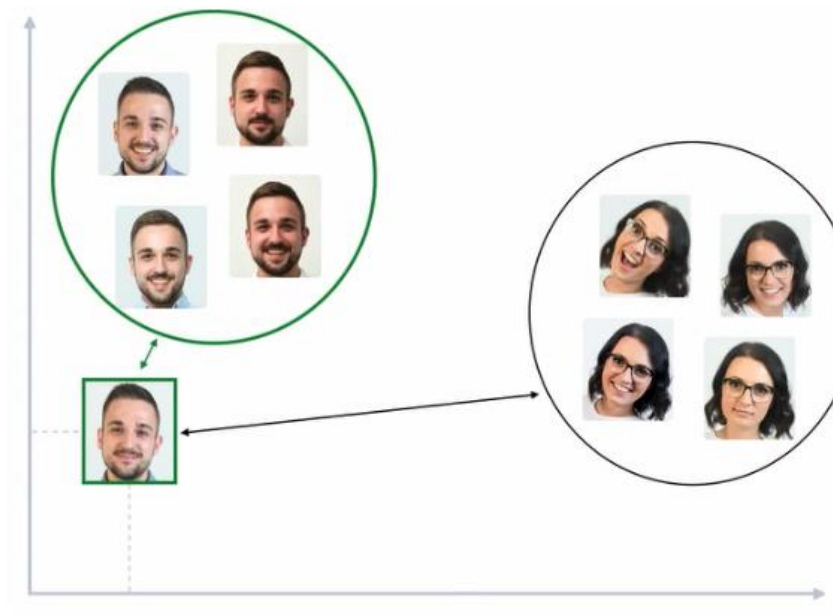
One possible way to recognize a person in an unknown image would be to calculate their embedding, calculate the distances from images of known people and if the embedding of the face is close enough to the embeddings of person A, we say that this image contains the face of person A.

Facenet also uses a technique called "triplet loss" to ensure that the features of similar faces are close to each other in the feature representation space, while the features of unsimilar faces are far from each other.

Triplet loss: In order for FaceNet to learn how to generate face embeddings, it uses this function:

1. It randomly selects an anchor image.
2. It randomly selects an image of the same person as the anchor image (positive example).
3. It randomly selects an image of a person that is different from the anchor image (negative example).
4. It adjusts the FaceNet network settings so that the positive example is closer to the anchor than the negative example.

These steps are repeated until all the faces of the same person are close to each other and far from each other and there are no more changes.



FaceNet Training Process

5.3 Facenet Performance:

Facenet has been compared to other face recognition algorithms and has shown very good performance. In particular, Facenet was able to efficiently recognize faces in large databases and with low latency. Facenet was also able to effectively handle image variations, such as lighting and exposure, making it particularly useful for real-time applications.

5.4 Conclusion:

In summary, Facenet is a convolutional neural network that was designed for facial recognition. Facenet uses CNNs to extract features from faces and has shown very good performance compared to others

6. Implementation

6.1 Introduction:

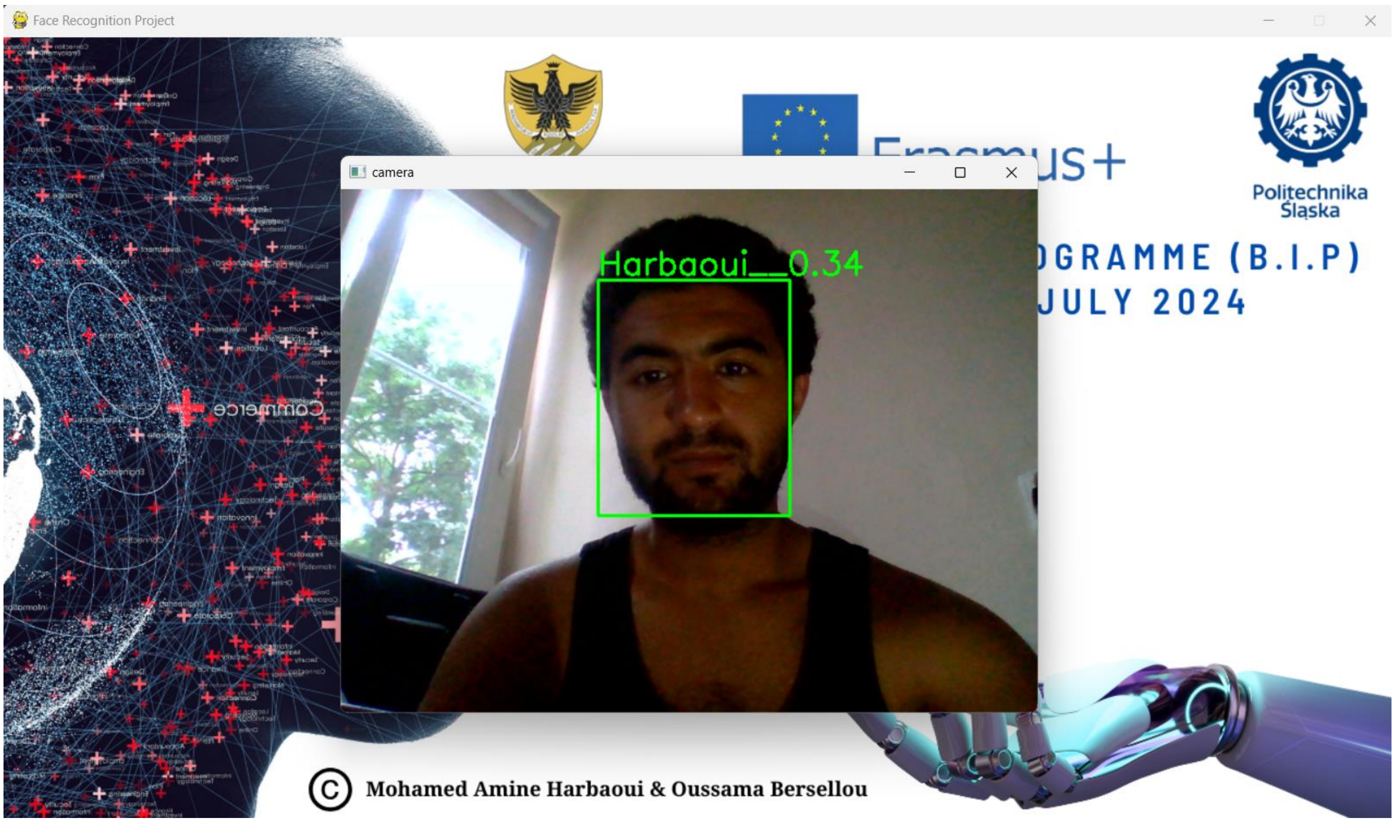
The purpose of the implementation is to develop a facial recognition application using the MTCNN and Facenet libraries. MTCNN is used to detect faces in images, while Facenet is used to encode detected faces into a single vector that can be compared to other vectors to perform facial recognition.

6.2 Input:

The app's input is a real-time image taken from the user's camera.

6.3 Output:

The output is the name of the person recognized in the image or a message indicating that the person was not found in the database.



Real-time input-output

6.4 Code Description

This code implements a real-time facial recognition application.

It imports libraries such as cv2 (OpenCV), numpy, mtcnn (a face detector), custom normalization and distance comparison functions, as well as pickle features to load recorded data.

The main detect function uses a face detector to find faces in an image, and then uses a face encoder to get a coding of each detected face, then, it compares this encoding to a collection of previously recorded reference encodings to determine whether a face has already been identified.

If a face is recognized, its name is displayed on the image, otherwise it is marked as "unknown". The result is displayed in a real-time window via a webcam.

6.5 Conclusion:

The results obtained show that the application can perform facial recognition in real time with high accuracy.

BIBLIOGRAPHY

GitHub - R4j4n/Face-recognition-Using-Facenet-On-Tensorflow-2.X

https://towardsdatascience.com/build-your-first-deep-learning-classifier-using-tensorflow-dog-breed-example-964ed0689430

https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923

https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

https://github.com/timesler/facenet-pytorch