Report

IT 360

Security project

Authors:

Jaouher ElMani

Jihene Gazzeh

Farah Jaouadi

Nour Elloumi

Contents

Introduction

In the digital age, the security of sensitive information and personal data is a constant concern. Traditional authentication methods, such as passwords and PINs, are increasingly seen as insufficient in the face of advanced cybersecurity threats. As a result, organizations are compelled to enhance their security posture by permitting only authenticated users or processes to access protected resources.

To address these challenges, more companies are turning to innovative technologies. Artificial Intelligence (AI) and Machine Learning (ML), particularly deep learning-based techniques, are being leveraged to develop more secure and effective authentication approaches.

One such promising solution is **facial recognition-based biometric authentication**. This technology has emerged as a highly effective solution for addressing security challenges.

In this comprehensive report, we will delve into main components, functional flow, mathematical demonstrations, algorithms, standard requirements ... of facial recognition-based authentication systems.

1- What is face recognition?

Facial recognition is a **biometric security technology** that identifies or verifies a person's identity using their face. Facial recognition systems can match a human face from a digital image or a video frame against a database of faces.

Face-based authentication finds **utility across sectors**:

- **Healthcare**: Enhancing patient identification and securing electronic medical records.
- **Banking**: Ensuring secure logins and transactions.
- **Law Enforcement**: Aiding in suspect identification and public safety.
- **Retail**: Facilitating personalized customer experiences and targeted marketing.

Examples of facial recognition technology

Apple uses facial recognition to help users quickly unlock their phones, log in to apps, and make purchases.

Google incorporates the technology into Google Photos and uses it to sort pictures and automatically tag them based on the people recognized.

MAC make-up, uses facial recognition technology in some of its brick-and-mortar stores, allowing customers to virtually "try on" make-up using in-store augmented reality mirrors.

McDonald's has used facial recognition in its Japanese restaurants to assess the quality of customer service provided there, including analyzing whether its employees are smiling while assisting customers.

2- Main components

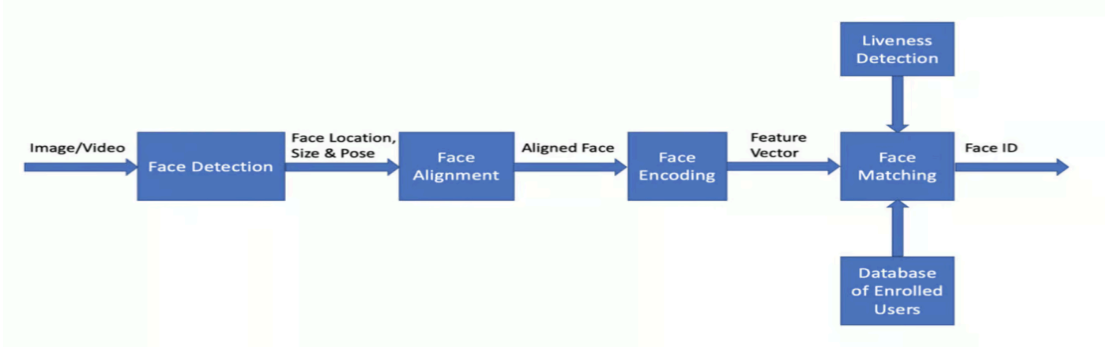
Physical components:

- **Camera/Webcam**: This is the hardware component that captures the user's face. It can be an external device or a built-in feature of the user's device.
- **pc/phone** ...

Non-Physical components:

- **Database**: This is where the system stores the facial data of registered users. When a user tries to authenticate, their facial data is compared with the data in this database.
- **Server**: This is where the main processing happens. The server receives the facial data from the user interface, processes it, and then sends back the results.

3- Functional flow



Face detection:

It involves locating a face in an image or video stream, and it doesn't concern itself with identifying the person in the image.

Face alignment:

It deals with the issue of faces not being directly oriented towards the camera when a picture is taken. The system can still recognize the person even if the face is turned in different directions.

The "face landmark estimation" algorithm is used to locate facial landmarks, which are specific points that exist on every face, such as the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc.

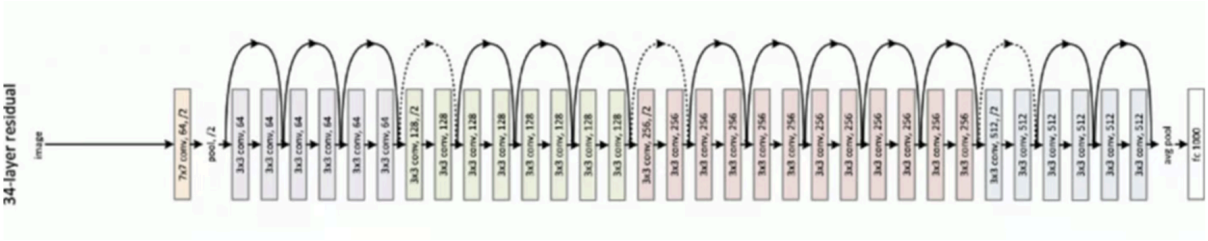
The 68-point face landmark model is a common example used in the pipeline to locate these specific points on every face. Once the locations of these key geometric face structures are identified, any rotation, translation, and scale representation of the face can be normalized.

This means that no matter how the face is turned, the eyes and mouth can be centered in roughly the same position in the image.

Face encoding:

This step is crucial as it converts the detected and aligned faces into a numerical representation that can be processed by a computer.

Deep learning models, such as **Convolutional Neural Networks (CNNs)**, are often used for this task. They are trained on large databases of face images to calculate the best numerical representation for each face.



Face matching:

After a face has been detected, aligned, and encoded, the system then performs face matching. This involves comparing the numerical representation (or encoding) of a face with the encodings of other faces in a database.

There are two main types of face matching tasks:

1. **Face Identification**: This process involves searching a database of known users to find the person who has the closest encoding (i.e., smallest distance) to the test face image. This is often used in surveillance systems or for identifying individuals in a large crowd.
2. **Face Verification**: This process compares the encoding of the test face image with a specific encoding (i.e., the encoding of an authorized user). If the two encodings are close enough (i.e., the distance between them is smaller than a certain threshold), the test person is verified. This is often used in authentication systems, such as unlocking a smartphone or accessing a secure area

Face liveness detection:

In addition to the main functionality of the face recognition pipeline, liveness detection has been incorporated into the pipeline. This is an optional feature to make sure the

authenticated face is from a real person, not from a photograph or from a video frame. The movement-based models such as "**eye-blink detection**" and "**mouth-moving detection**" are used as part of the liveness check. Once the liveness detection feature is triggered, the system detects eye-blinks followed by the mouth close-open-close pattern to verify if it is a live person.

Mathematical demonstrations

1- Mathematical model

A mathematical model represents a system in terms of mathematical ideas and terminology. To understand a system, a model may be used to examine the interactions between its many components and forecast the system's behavior.

Our system's mathematical modelling looks like this:

S= {Σ, F, δ, C} S = Face Recognition. ∑ = set of input symbols = {Video File, image, character information} F = set of output symbol = {Match Found then notification to user, Not Found} δ = 1.

1. Start w

2. Read the training set of images **N*N images**

3. Resize image dimensions to **N2*1**

4. Select training set of **N2*M** Dimensions, M: number of sample images

5. Find the average face, subtract from the faces in the training set, and create matrix A

Ψ=$(1/M)$ **Σ Γi**

Where, Ψ= average image, M= number of images, and Γi= image vector. Φi = Γi – Ψ Where, i = 1, 2, 3, ..., M. A = [Φ1, Φ2, Φ3...ΦM]

6. Calculate covariance matrix: **AA' C=AT*A**

7. Calculate the c covariance matrix's eigenvectors.

8. To find the eigenfaces, divide the entire number of eigenvectors by the total number of training pictures.

9. The chosen eigenvectors are multiplied by the A matrix to construct a reduced eigenface space.

10. Determine the eigenface of the picture.

11. Calculate the eigenfaces' Euclidean distances from the picture.

12. Calculate the shortest distance between two points using the Euclidean formula.

13. An image with the smallest Euclidean distance or an image that cannot be recognised are the output options. Eigenfaces will create the grayscale pictures, and the method will only execute on keyframes if C is true.

2- Face Recognition Methods

There are a variety of ways to recognise a person's face, including the following:

a. Geometric Based / Template Based:

Algorithms for facial recognition may be categorized as either geometry-based or template-based. Tools like SVM, PCA, LDA, LDA-like statistical approaches, kernel methods, and trace transforms may be used to build template-based methods, as can other statistical methods like SVM. The geometric feature-based approaches examine **the geometric connection between local face characteristics**. Additionally, it's called feature-based.

b. Piecemeal / Wholistic:

To identify the most important qualities, numerous academics used this method to examine **the relationship between the parts of a face and its functions**. The eyes, a combination of traits, and so on, have been used in many ways to identify someone. Hidden Markov Models and facial recognition feature processing come within this area.

c. Appearance-Based / Model-Based:

Images of a face are shown using an appearance-based technique. As a high-dimensional vector, images are often taken into account. A feature space may be derived from an image divide using this method. The training set compared to the

sample picture. The model-based method, on the other hand, **aims to model a face**. The model's parameters were utilized to recognise the picture and the fresh sample it was fed into the model.

There are two ways to categorize the appearance-based technique. In the direct method, we employ Ex-PCA, LDA, and IDA; in the nonlinear approach, we use Kernel PCA. As an alternative, the model-based technique is divided into two categories: either 2D or 3D. Matching of an Elastic Bunch Graph from the past is used.

3- Template matching

This is a digital image processing technique for finding small parts of an image that resemble a template image.

The main challenges in template matching work are: detecting irrational changes, light and background changes, background clutter and scale changes.

One basic method of matching a template is using an image patch (template), tailored to a particular feature of the search image we want to search for. This procedure can be easily achieved on grey images or edge images. The cross-linking output will be highest in places where the image structure matches the mask structure, where the values of the large image are multiplied by the values of the large mask.

This method is usually applied by selecting a portion of the search icon to use as the first template: we will call the search image **S (x, y)**, where (x, y) the search represents the coordinates of each pixel in the image. We will call the template T (X t, Y t), where (Xt, Yt) represent the points of each pixel in the template. We then basically move the center (or the origin) of the template T(x t, y t) over each (x, y) point in the exploration image and calculate the sum of products between the coefficients in S(x, y) and T(xt, yt) over the entire area spanned by the pattern. Since all possible positions of the template are considered in terms of the search image, a position with a high score is the best position. This method is sometimes called “Linear SpatialFiltering” and the template is called a filter mask.

One more way to deal with transformation problems on images using template matching is to **compare the intensity of the pixels using the SAD (combination of absolute differences) measurement**.

The intensity of a pixel in a search image with coordinates (xs, ys) has intensity Is(xs, ys) and a pixel in the template with coordinates (xt, yt) has intensity It(xt, yt). Consequently, the absolute difference in the pixel intensities is well-defined as

Diff(xs, ys, x t, y t) = | **Is(xs, ys) – It(x t, y t)** |.

SAD(x,y)= $\sum_{i=0}^{T_{rows}} \sum_{j=0}^{T_{cois}} \mathrm{Diff}(x+i, y+j, i, j)$

The mathematical representation of the idea of looping through pixels in the search image when we translate the original of the template on each pixel and measure SAD is:

$$\sum_{x=0}^{S_{rows}} \sum_{y=0}^{S_{cols}} \mathrm{SAD}(x,y)$$

Srows and Scols signify the rows and the columns of the search image and Trows and Tcols signify the rows and the columns of the template image, separately. In this method, the lowermost SAD score gives the approximation for the best position of the template within the search image. The method is simple to contrivance and recognize, but it is one of the measured methods.

4-Neural Networks

Nerve networks are nonlinear networks that are suitable for representing nonlinear faces. Neural networks are used in many applications such as pattern recognition problems, character recognition, object recognition, and so on.

The main purpose of the neural network is to capture the complex facial features. **Multi-Layer Perceptron (MPL)** with feed forward learning algorithm was applied for facial recognition. T.J Stonham made his first attempt at face recognition using a single-layer adaptive network called WISARD, which has a separate network for each stored

individual. S. Lawrence, Giles, Tsoi, and A.D. Back have proposed a hybrid neural network that combines local image sampling, self-mapping (SOM) neural networks, and a virtual neural network. SOM provides the amount of sample images, thus minimizing the dimensions and conveying the message. The SOM consists of N nodes ordered in a two-dimensional lattice structure and each node has 2 or 4 neighboring nodes, respectively. Typically, a SOM's life cycle consists of three phases: the learning phase, the training phase, and the testing phase. During the learning phase, the neuron weighing near the input data vector is declared the winner. The learning algorithm may be summarized as follows:

Initialization: Select random values for the initial vector weights Wj(0),

for j=1,2,...,l where l is the total number of neurons

$$w_i = [w_{i1}, w_{i2}, w_{i3} \ldots\ldots w_{il}]^T \in \Re^n$$

Sampling: Draw a sample x from the input space with a confident prospect.

$$x = [x_1, x_2, \ldots\ldots\ldots x_l]^T \in \Re^n$$

Similarity Matching: Find the best matching (winning) neuron i(x) at time t, $0 < t \leq n$ by using the minimum distance Euclidean criterion:

Updating: Adjust the synaptic weight vector of all neurons by using the bring up-to-date

$$i(x) = \arg min \min_j \|x(n) - w_j\|, j = 1,2, \ldots, l$$

formula:

$$w_j(n+1) = w_j(n) + \eta(n) h_{j,i(x)}(n)\,(x(n) - w_j(n))$$

Where $\eta(n)$ is the learning rate parameter, and $h_{j,i(x)}$ is the neighbourhood function cantered around the winning neuron i(x).

Repeat from step 2 until no changes in the feature map are observed.

The self-generated map recorded two values: the total number of winning times for both titles and the image is not in the database. During the training phase, the number of wins is also recorded with the input sample label for each node. During the testing phase, each input vector is compared with all the nodes in the SOM, and some of the best matches are found based on a minimum distance calculator.

Face recognition, which uses features derived from the isolated cosine transform (DCT) coefficient with SOM-based classification, was performed using an image database of 25 face images of each of the 5 subjects. After training for 850 positions, he achieved an identification rate of 81.36% for 10 consecutive trials. There are 5 different facial expressions.

Algorithms Or Protocol Explanation

1- Deep learning

Fundamental concepts

Linear regression vs neural networks

In linear regression, the relationship between the input features and the output is assumed to be linear. This means that the output can be obtained by a weighted sum of the input features, where each feature is multiplied by a weight and then summed together with a bias term.
While linear regression can capture simple relationships between variables, it struggles to capture complex patterns or interactions between features. Each feature contributes independently to the output without considering potential interactions with other features.

Neural networks, on the other hand, are highly flexible models capable of capturing complex patterns and interactions within data.

Deep learning uses powerful neural networks to predict interactions in neural networks.

input layer: predictive features

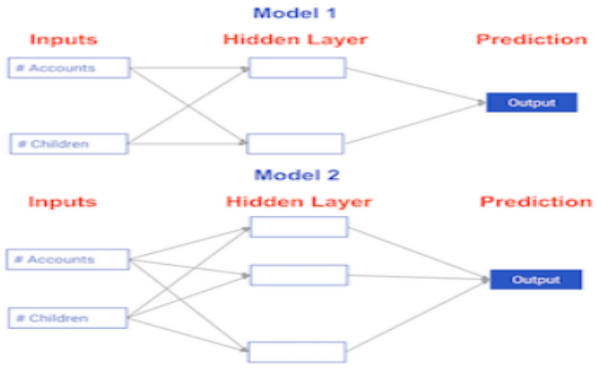
output layer: the prediction from our model

hidden layers: all layers that aren't input or output layers

Hidden layer values: Unobservable representations in neural networks. while Inputs and outputs correspond to observable events in the world, which can be recorded as data.

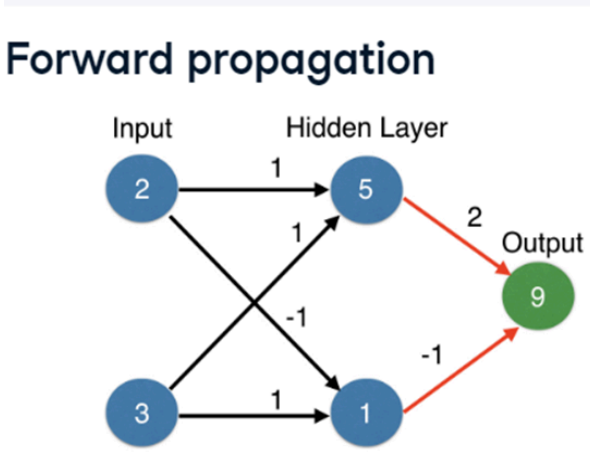
Each dot, called a **node**, in the hidden layer, represents an aggregation of information from our input data. **Each node** adds to the model's ability to capture interactions, the more nodes we have the more interactions we capture.

Model 2 has a greater ability to account for interactions.



Forward propagation:

an algorithm that shows how neural networks use data to make predictions.



Multiply – add process (example: 5 = 2*1 + 3*1)

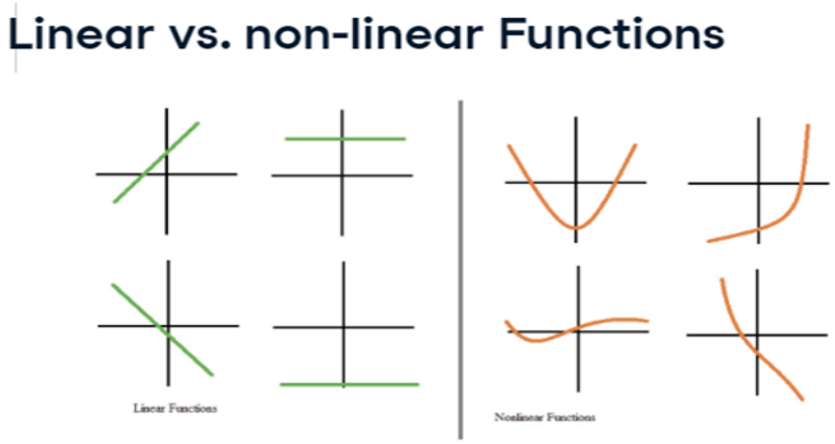
Forward propagation for one data point at a time

Output is the prediction of that data point

Activation functions

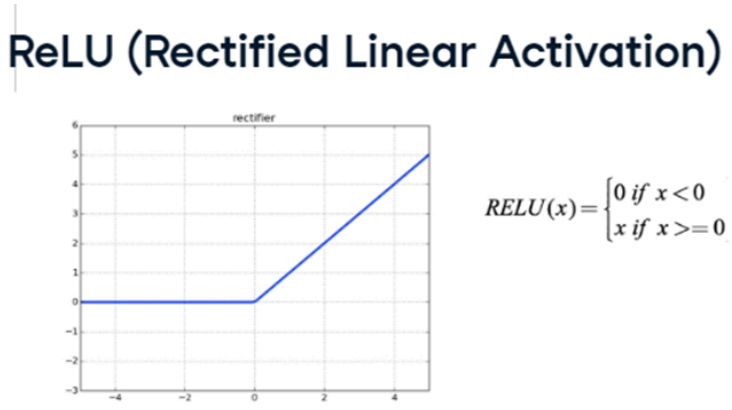
For neural networks to achieve maximum predictive power, we must apply the activation function in the hidden layers.

Activation functions capture non-linearities.

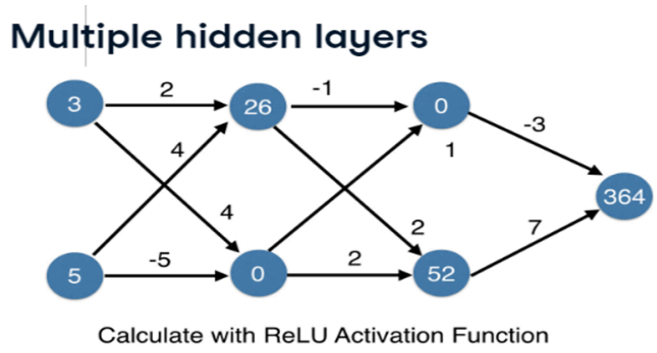


Activation function applied to node inputs to produce node output

The **rectified linear activation function** is mostly used today.



Modern deeper networks: use of models with not just one hidden layer, but with many successive hidden layers

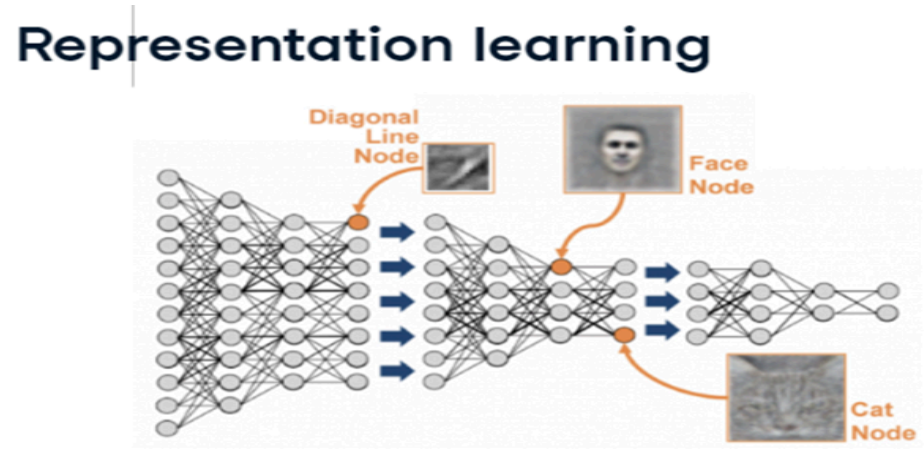


Representation learning

Deep networks internally build representations of patterns in the data.

Partially replace the need for feature engineering.

Subsequent layers build increasingly sophisticated representations of raw data.



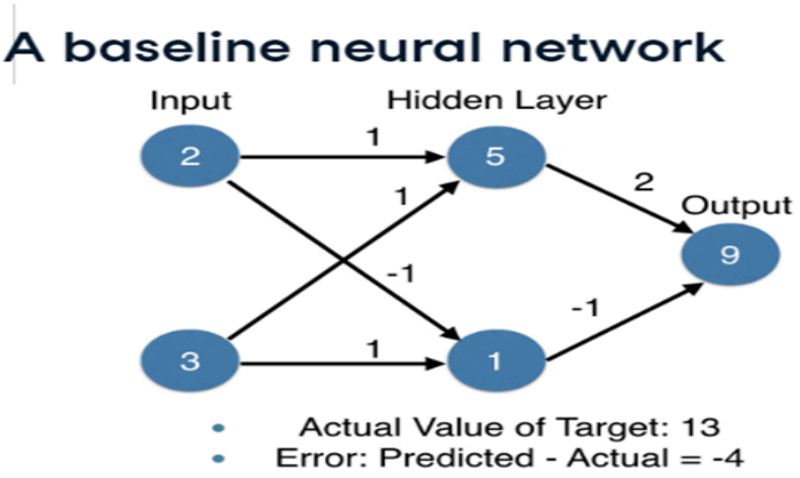
In image classification, the initial hidden layer of a neural network constructs basic patterns or interactions, (such as detecting diagonal, horizontal, or vertical lines, as well as identifying blurry regions, by analyzing groups of neighbouring pixels.)

After identifying diagonal, horizontal, and vertical lines, subsequent layers of the network integrate this information to detect larger patterns, such as squares.

As the network progresses, it may combine the positions of squares and other geometric shapes to recognize objects like faces, cars, or any other elements present in the image.

The model isn't explicitly instructed on interactions (like searching for diagonal lines); instead, through training, the network learns weights that identify relevant patterns, improving its predictive capability.

Optimization



Loss function

loss function: a function to aggregate all of the errors into a single measure of the model's predictive performance

The most common loss function is mean-squared error (square each error and take the average of that as a measure of model quality)

Example :

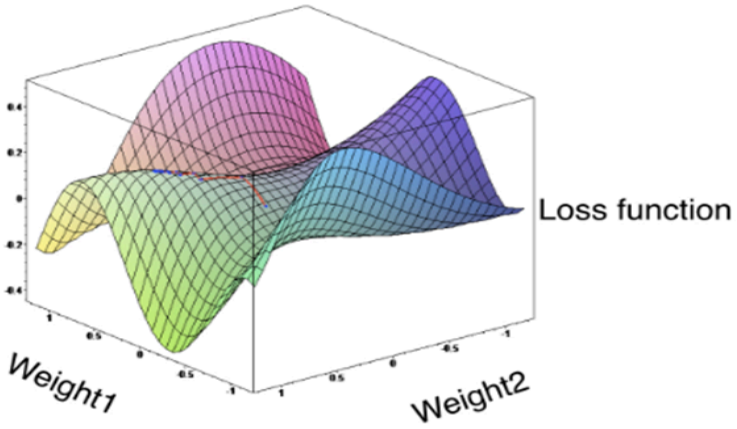
Squared error loss function

Prediction	Actual	Error	Squared Error
10	20	-10	100
8	3	5	25
6	1	5	25

- Total Squared Error: 150
- Mean Squared Error: 50

2 weights Loss function illustration: plotting the model performance for each set of weights

Loss function



lower loss function values means a better model performance

Goal: Find the weights that give the lowest value for the loss function

=> algorithm for this purpose called gradient descent

Gradient descent

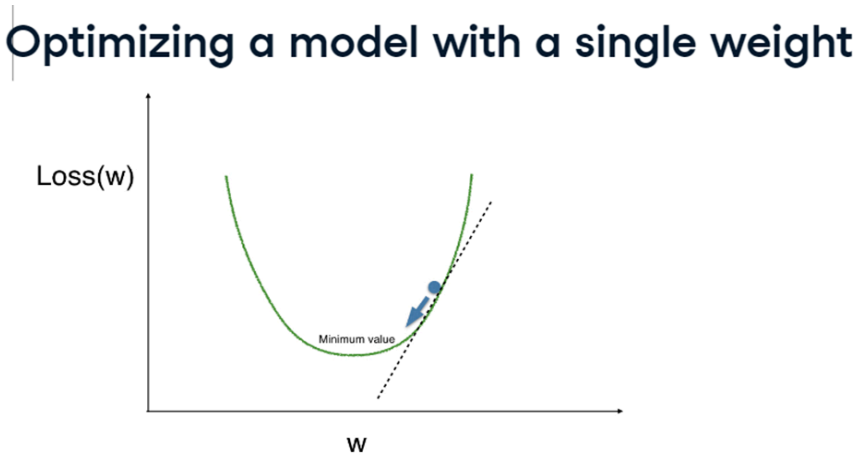
Gradient descent steps:

1- start at a random point

2- until you are somewhere flat:

*Find the slope

*Take a step downhill



If the slope is positive:

Going opposite the slope means moving to lower numbers

Subtract the slope from the current value

Too big a step might lead us astray

=> **Solution: learning rate**

=> Update each weight by subtracting learning rate * slope

NB: Learning rates are frequently around 0.01

The learning rate ensures that we take small steps, so we reliably move towards the optimal weights

Backpropagation

-Backpropagation process: takes the error from the output layer and propagates it backwards through the hidden layers towards the input layer

- used to update the weights and biases of the network based on the error between the predicted output and the actual output

-trying to estimate the slope of the loss function with respect to each weight

-Do forward propagation to calculate predictions and errors

2- Creating a keras model with deep learning

Model building:

Steps:

1- specify architecture: Determine the number of layers, nodes in each layer, and the activation functions to be used in each layer.

2-Compile the model: Define the loss function and specify how optimization will occur.

3- fit the model: Iterate through cycles of back-propagation and weight optimization using your data.

4-Use the model for predictions: Apply the trained model to make predictions on new data.

When constructing Keras models, the input layer's node count is determined by the number of columns in the input data.

A straightforward approach to model building is the sequential model specification, where each layer is directly connected to the one following it in the network diagram.

The predominant layer type is the dense layer, named for its dense interconnections between all nodes in the previous layer and those in the current layer.

During model compilation, specifying the optimizer and loss function is crucial. '

Adam' is a recommended optimizer due to its adaptive learning rate adjustment during gradient descent, promoting stable weight optimization.

For regression tasks, mean square error often serves as the preferred loss function.

When fitting the model, backpropagation and gradient descent are applied to update weights. Scaling the data before fitting can facilitate optimization.

Using models process :

1- Save the model after you've trained it

2- Reload that model

3- Make predictions with the model

3- Conventional neural networks

CNN: a type of artificial neural network architecture. They consist of multiple layers of interconnected nodes, or neurons, organized in a hierarchical manner.

Images are stored as arrays of numbers.

Color images are stored in a 3-dimensional array.

The first two dimensions correspond to the height and width of the image (number of pixels).

The last dimension corresponds to the red, green, and blue colours present in each pixel.

To examine the red, green, and blue data in a particular pixel, indexing is done into both spatial dimensions of the image.

The numbers in the array represent the intensity of the image in each pixel.

In black-and-white images, higher numbers represent brighter parts of the image, while lower numbers represent darker parts.

Classifying images

The objective of Classifying Images:

The goal is to develop an algorithm capable of distinguishing between different classes of images.

This task falls under classification in machine learning.

1-Training Phase:

During training, the algorithm is presented with samples from various classes.

Throughout the training phase, the algorithm adjusts its parameters to recognize patterns in the data that differentiate between different classes of images.

2-Evaluation of Classifier:

At the end of training, it's essential to assess how well the classifier performs.

To avoid an overly optimistic estimation due to overfitting, evaluation is done by testing the classifier on a separate portion of data set aside for this purpose.

= > Mathematical Representation: One-Hot Encoding:

One-hot encoding is a mathematical representation used to represent class data.

Each row in the encoding represents one sample, while each column corresponds to one of the classes.

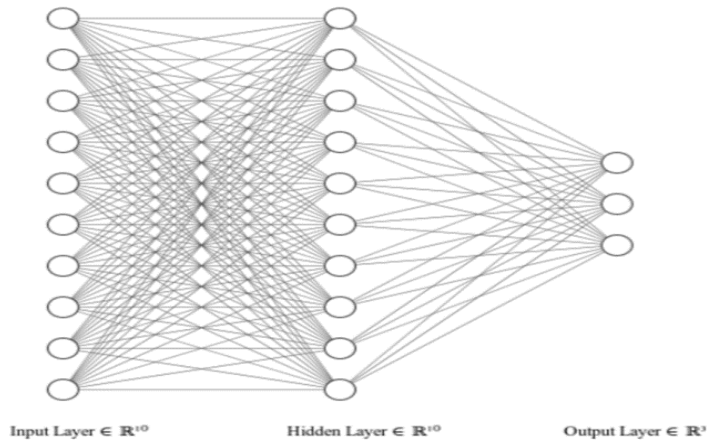
In each row, all values are set to 0 except for the column corresponding to the class of the image sample being represented.

we can use one hot encoding array to determine how many predictions are correct.

Image classification with Keras: start by using a fully connected network by importing a sequential model and initializing it

Our network will comprise densely connected layers, where each unit in a layer is linked to every unit in the preceding layer.

The initial layer of the network establishes connections with all the pixels in the input image.



This diagram shows the network and all its connections

Convolutions: in the neural network each unit in the first layer has a weight connecting separately with every pixel in the image. Pixels in most images are not independent of their neighbours, for example, images of objects contain edges and neighbouring pixels along an edge tend to have similar patterns.

=> How can we use these correlations to our advantage?

One dimension convolution

Example: an array that contains 5 zeros followed by 5 ones

This array contains an edge in the middle where the value goes from 0 to 1

The kernel defines the **feature** that we're looking for, in this case, we're looking for a change from a small value on the left to large values on the right. We start the result as all zeros then we slide the kernel along the array.

In each location, we multiply the values in the array with the values in the kernel and sum them up.

this is the result for that location :

What is a convolution?

```
array = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
kernel = np.array([-1, 1])
conv = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])
conv[0] = (kernel * array[0:2]).sum()
conv[1] = (kernel * array[1:3]).sum()
conv[2] = (kernel * array[2:4]).sum()
...
for ii in range(8):
    conv[ii] = (kernel * array[ii:ii+2]).sum()
conv
```

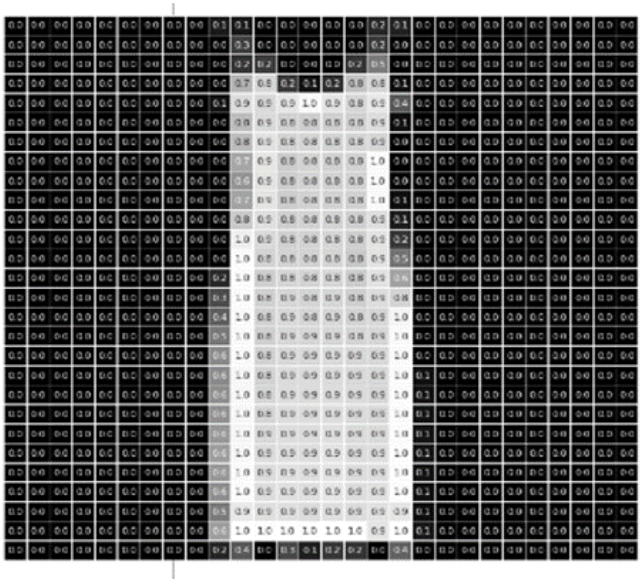
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0])
```

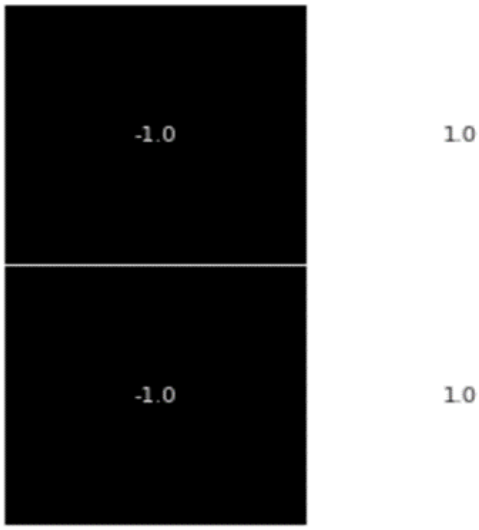
TWO dimension convolutions

Images 'convolutions operate in 2 dimensions.

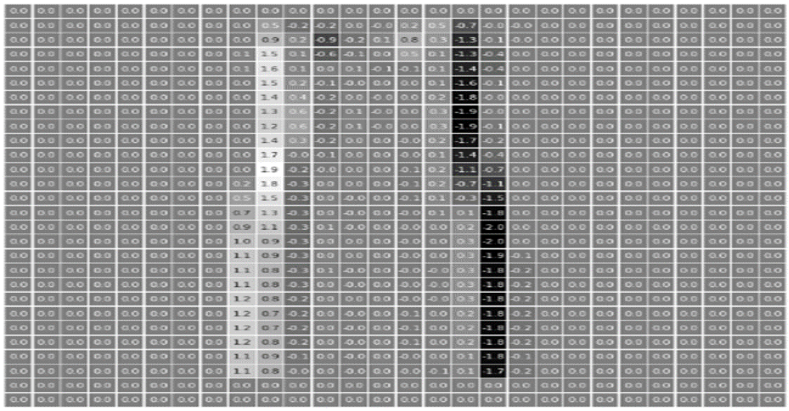
Dress example:

we convolve the image using a kernel designed to detect vertical edges on the left side. As a result, areas corresponding to left edges are accentuated while areas opposite to this kernel, such as the right side of a dress, exhibit a negative response during convolution.





Dress convolution graph operation:



4- Face detection Viola-Jones algorithm

This algorithm operates on 384 by 288-pixel images, faces are detected at 15 frames per second on a conventional 700 MHz Intel Pentium III.

Steps:

1-Integral image: a speedy new image representation enabling rapid feature evaluation. They can be computed from an image with minimal operations per pixel.

2-Classifier construction: involves selecting a limited number of significant features through AdaBoost. With a vast number of Haar-like features within each image subwindow, far surpassing the pixel count, fast classification requires the exclusion of most features, focusing on critical ones.

NB: Haar-like features are simple rectangular patterns used in object detection and feature extraction tasks in computer vision.

3-Combining successively more complex classifiers in a cascade structure: Cascading increasingly complex classifiers dramatically speed up detection by prioritizing attention to promising image regions. This focus-of-attention approach rapidly pinpoints potential object locations within an image.

Features:

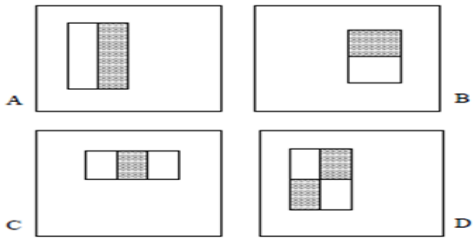


Figure 1: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

A primary benefit of using features instead of pixels is their ability to encode domain-specific knowledge, which may be challenging to learn directly from a limited amount of training data. Specifically, we employ **three types of features.**

1- The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions.

The regions have the same size and shape and are horizontally or vertically adjacent (see Figure 1).

2- A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle.

3- The four-rectangle feature computes the difference between diagonal pairs of rectangles.

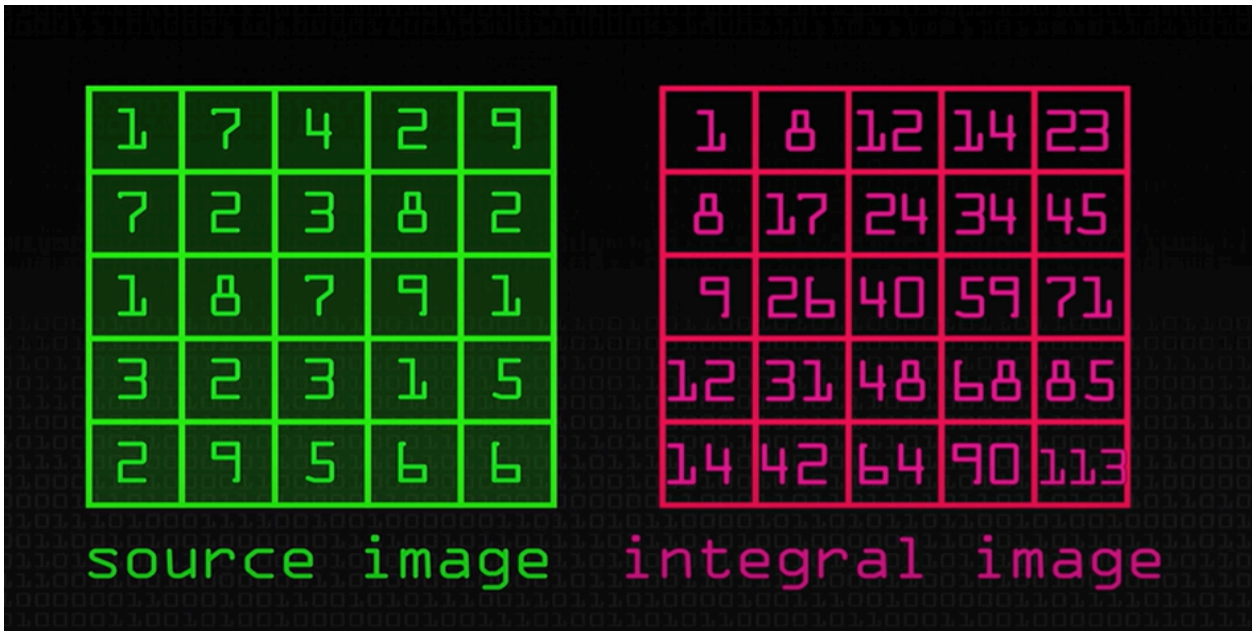
Given that the base resolution of the detector is 24x24, the exhaustive set of rectangle features is quite large, over 180,000.

Integral image:

Image integral calculation:

Each pixel (x, y) in the integral image contains the sum of all pixel values above and to the left of it in the original image, including the pixel itself.

This calculation is done in a single pass over the original image, and the resulting integral image has the same dimensions as the original image.



Rapid feature evaluation:

During the detection process, the Viola-Jones algorithm evaluates various rectangular features within image subwindows to distinguish between objects and backgrounds.

These rectangular features are defined by the difference in the sum of pixel intensities within adjacent rectangles.

Instead of recomputing the sums of pixel values for each feature at every location and scale in the image, the integral image allows for rapid computation of these sums.

By utilizing the integral image, the sums of pixel values within any rectangular region can be calculated efficiently using just four array references, regardless of the size or position of the region.

Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the integral image.[2] The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

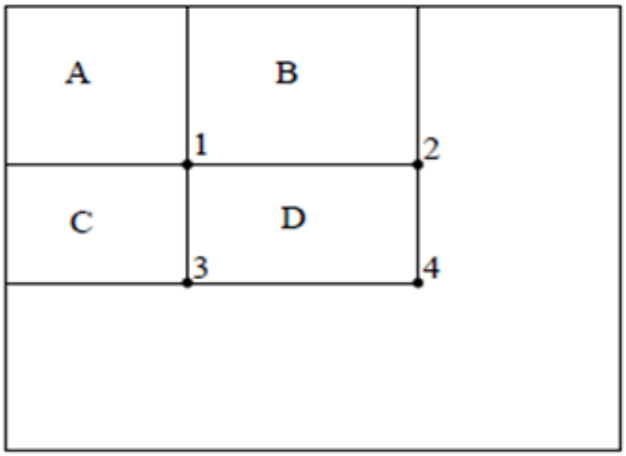


Figure 2: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image. Using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + i(x, y) \quad (1)$$
$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (2)$$

(where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$, and $ii(-1, y) = 0$) the integral image can be computed in one pass over the original image.

With the integral image, computing any rectangular sum requires only four array references. Calculating the difference between two rectangular sums entails eight references. For two-rectangle features, this computation requires six references; for

three-rectangle features, eight references; and for four-rectangle features, nine references. (see figure 2)

Classification function:

Given a feature set and a training set of positive and negative images, various machine-learning approaches can be employed to learn a classification function.

However, with over 180,000 rectangle features per image sub-window, far exceeding the number of pixels, computing the entire set is prohibitively costly.

NB: The image sub-window is a small rectangular region within an image

=> The idea is that a small subset of these features can effectively form a classifier.

To identify the features, the weak learning algorithm is tailored to select the single rectangle feature that best separates positive and negative examples.

For each feature, the weak learner determines the optimal threshold classification function to minimize misclassifications.

NB: the weak learning algorithm is a machine learning algorithm that produces a classification model.

Best features:

Figure 3: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

of the nose and cheeks (see Figure 3). This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

Binomial decision trees:

The decision tree makes decisions based on features extracted from image sub-windows.

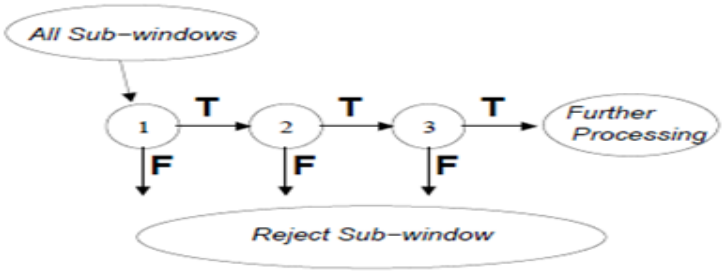
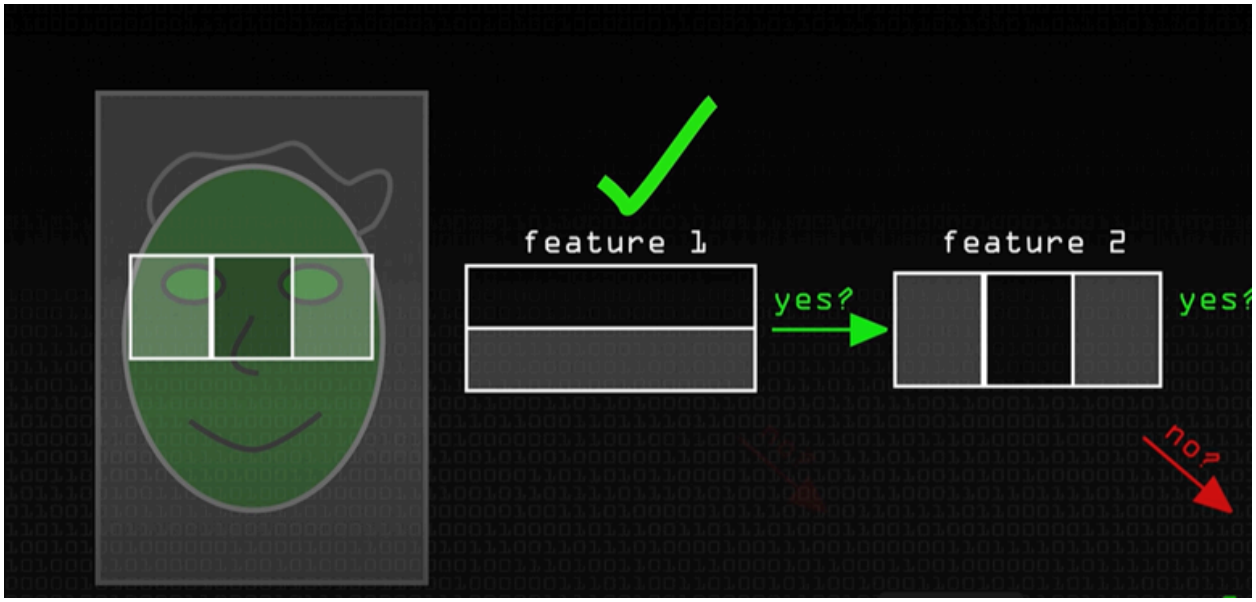


Figure 4: Schematic depiction of a the detection cascade. A series of classifiers are applied to every sub-window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages of processing the number of sub-windows have been reduced radically. Further processing can take any form such as additional stages of the cascade (as in our detection system) or an alternative detection system.



Face recognition:

Triplet loss function: is a technique used to train neural networks for embedding faces into a continuous space. It ensures that images of the same person are closer together in this space, while images of different people are farther apart. The aim is to minimize the distance between embeddings of images from the same person (anchor and positive) while maximizing the distance between embeddings of images from different people (anchor and negative)**.**

Anchor: It refers to the image of a face for which we want to compute the embedding.

Positive: It refers to another image of the same person as the anchor. The distance between the embeddings of the anchor and the positive image should be minimized.

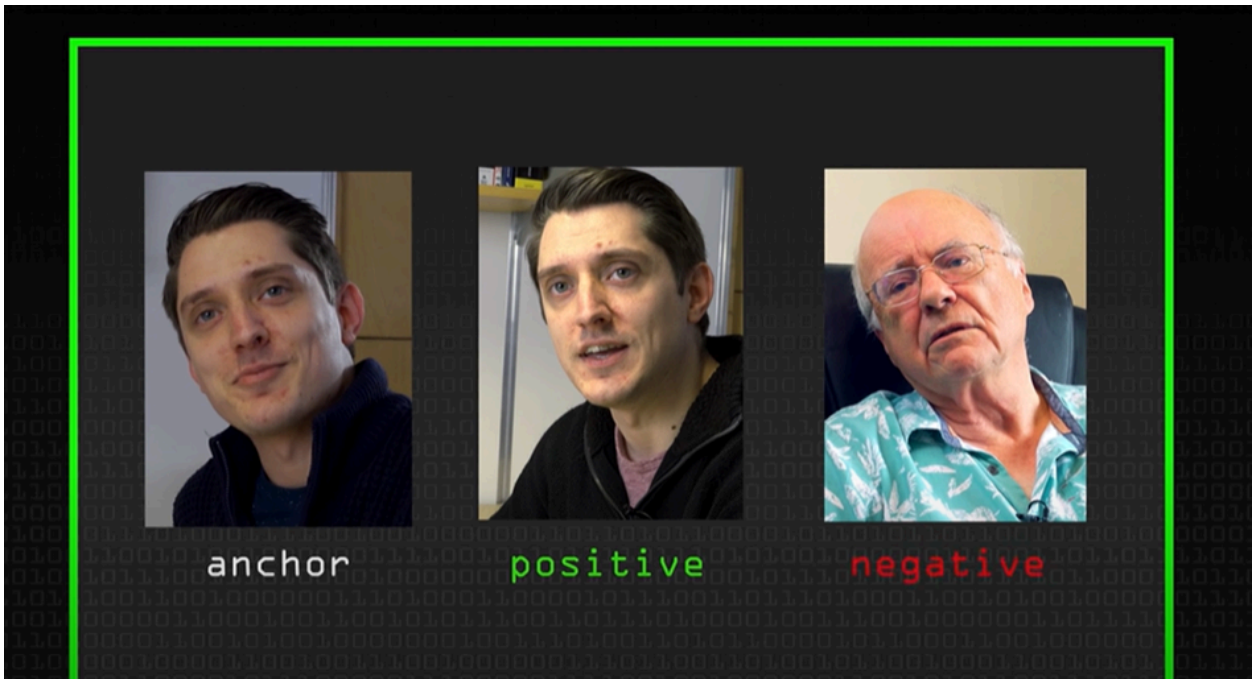
Negative: It refers to an image of a different person from the anchor. The distance between the embeddings of the anchor and the negative image should be maximized.

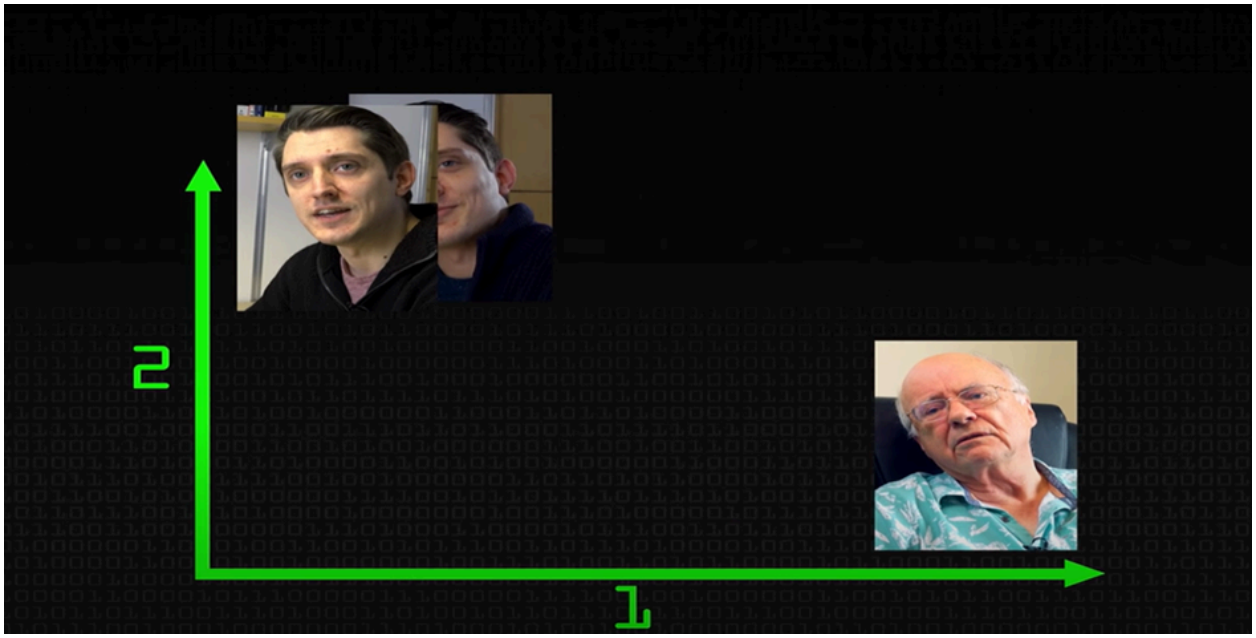
The loss function is formulated as follows:

$$L = \max(d(a, p) - d(a, n) + \alpha, 0)$$

Where:

- $d(a, p)$ is the distance between the embeddings of the anchor and positive images.
- $d(a, n)$ is the distance between the embeddings of the anchor and negative images.
- α is a margin that specifies a minimum difference between the distances. It prevents the model from trivial solutions by enforcing that the distance between the anchor and the positive is at least a certain margin smaller than the distance between the anchor and the negative.
- L is the triplet loss.





We repeat the process for different pairs of people with different positive and negative samples

This is actually not a classification rather than a dimensional reduction

Classification vs dimension reduction:

-Classification refers to the process of categorizing an image sub-window as containing the object of interest (e.g., a face) or not.

VS

- Dimensionality reduction involves reducing the number of features or dimensions in the data while preserving its essential characteristics.

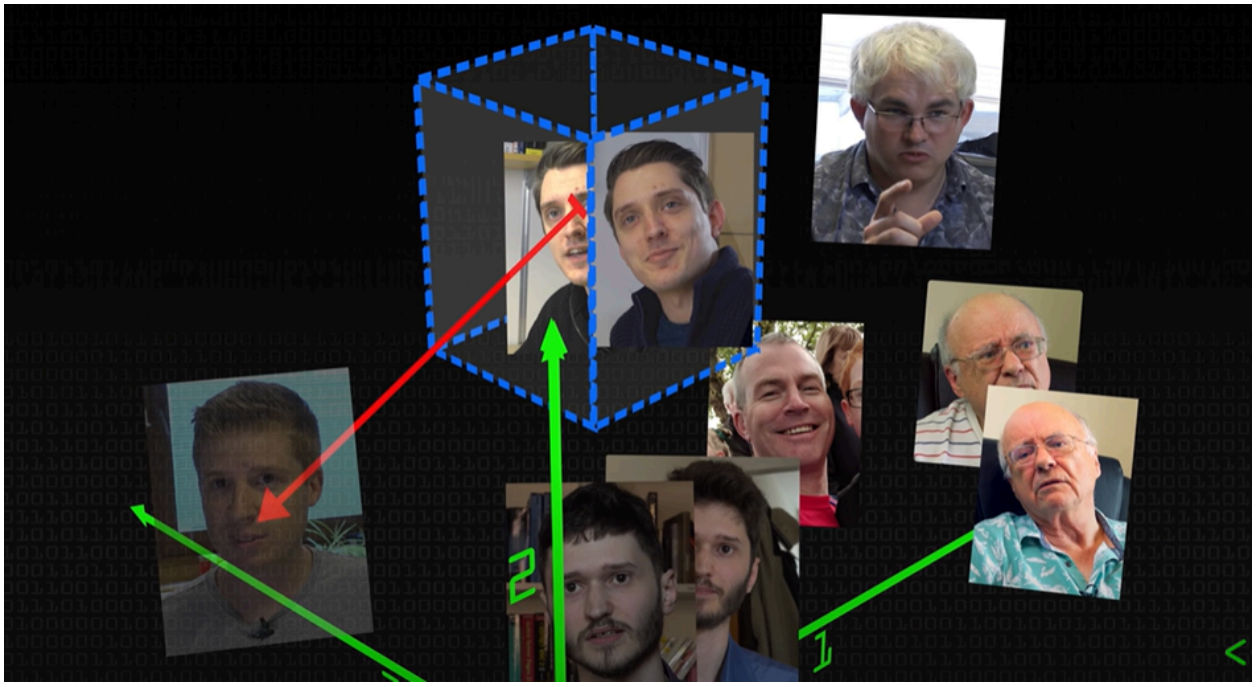
- Dimensionality reduction techniques are employed to reduce the feature space, making it more manageable without significantly sacrificing the ability to discriminate between objects and backgrounds.

One-shot learning:

One-shot learning problem: The one-shot face recognition problem refers to the scenario where a system is required to recognize a person with just one image of that person's face.

How do we convince a phone to let the person in after only seeing one picture of his face?

=> We don’t train the network to classify him we just use the existing network that we trained on thousands of people



We store the face's location in the matrix. When the user attempts to log in again if the new image aligns closely with the previous one in space, access is granted; otherwise, it's denied.

Standard requirements and rules

1- Standard requirements:

Accuracy and Performance Standards:

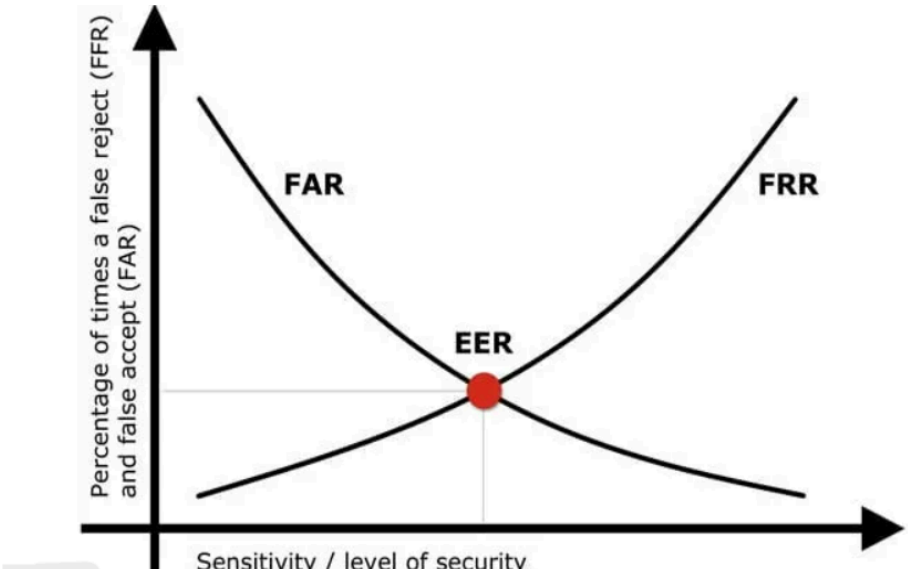
The accuracy and performance of biometric systems are expressed in terms of the following error probabilities:

1. **False Acceptance Rate (FAR):** the percentage of recognitions where someone is wrongly recognized (false acceptance).
2. **False Rejection Rate (FRR):** the percentage of recognitions where someone is wrongly not recognized (false rejection).

These rates are crucial metrics for evaluating the performance and reliability of biometric systems. The accuracy of a system is typically measured by its Equal Error Rate (EER), which is the point where FAR equals FRR.

$$(\%)\ FRR = \frac{\text{No. of false rejected individuals}}{\text{Total no of Individual Samples}} * 100\%$$

$$(\%)\ FAR = \frac{\text{No. of false acceptence Individuals}}{\text{Total no of Individual samples}} * 100\%$$



The National Institute of Standards and Technology (NIST) conducts periodic Facial Recognition Vendor Test (FRVT) programs. These programs provide benchmarks

and evaluations for facial recognition algorithms to assess their accuracy, speed, efficiency, and scalability. Vendors submit their algorithms for testing, and results are published publicly.

Industry Standards:

ISO/IEC 19794-5: This standard specifies the data interchange format for facial images and features, ensuring interoperability between different facial recognition systems.

ISO/IEC 30107-1: This standard establishes a foundation for Presentation Attack Detection (attempts to trick a facial recognition system by presenting a fake face instead of a real one), ISO/IEC 30107-1 acts as a shield by defining different types of presentation attacks that can target facial recognition (e.g., deepfakes, mask) and establishes a framework for detecting these attacks.

Data Privacy Standards:

Data privacy standards play a key role in governing the collection, storage, and processing of facial data to protect personal information.

Rules and Regulations:

General Data Protection Regulation (GDPR): The GDPR categorizes biometric data, which includes facial recognition information, as a special category of personal data due to its sensitivity. As such, facial recognition falls under this definition and within Article 9 of the GDPR. It clarifies that biometric data should not be used to identify a person unless an individual has provided explicit consent.

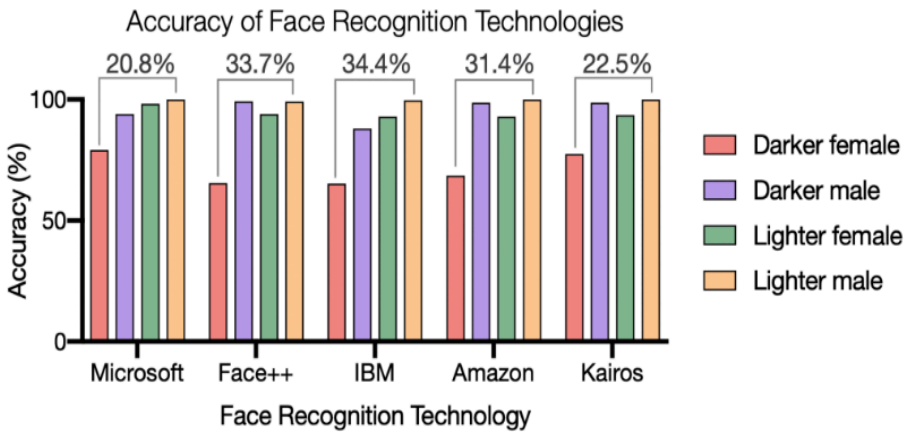
Convention 108+: Convention 108+ is a legal framework for data protection and guarding individual privacy in the digital age.

1. **Purpose Limitation:** Facial data can only be collected for clearly defined and legitimate purposes. Any further processing of the data must be compatible with the original purpose.
2. **Data Minimization:** Only the minimum amount of facial data necessary for the specific purpose can be collected and processed.
3. **Limited Duration of Storage**: A retention period for the data must be established. This period cannot exceed what's necessary for the intended purpose.

Ethical Guidelines:

Ethical guidelines for facial recognition ensure that the technology is used responsibly, transparently, and with consent while respecting personal information. Here's a breakdown of some key ethical considerations:

1. **Consent:** Explicit consent must be obtained from individuals before collecting or using their facial data.
2. **Transparency:** Transparency ensures that individuals are aware of how facial recognition is used and for what purpose.
3. **Algorithmic Fairness and Bias Mitigation:** Algorithmic fairness and bias mitigation are key factors in facial recognition. Bias can arise from skewed training data or flawed information, resulting in unfair outcomes. Many studies have found that facial recognition algorithms are often inadequate with darker-skinned and female faces, mainly due to the lack of diversity in training datasets. Addressing bias helps reduce unintended consequences in facial recognition. Therefore, the outcomes of facial recognition should not disadvantage certain individuals or groups based on demographic factors. Fairness ensures that all individuals are treated equally.



4. **Security:** Implement strong security measures to protect facial data from unauthorized access or misuse.

Report- Part2

IT 360

Security Project

Authors:

Jaouher ElMani

Jihene Gazzeh

Farah Jaouadi

Nour Elloumi

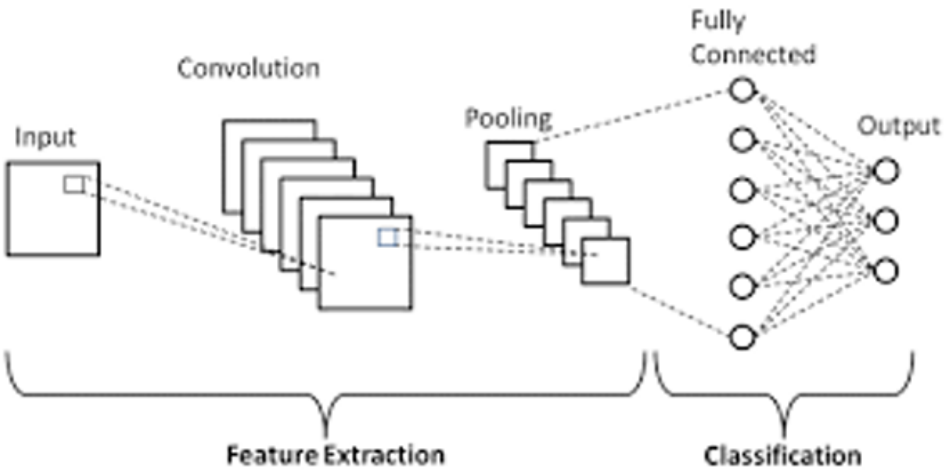
Contents

Facial recognition using deep learning

Main characteristics:

1- Face image dataset: a repository of a large number of face images to recognize

2- Pre Trained CNN model: Convolutional neural networks are designed to learn spatial hierarchies of features from raw input data in an automatic and adaptive manner



3- feature extraction:

INPUT: Initially, the input image of a person is provided.

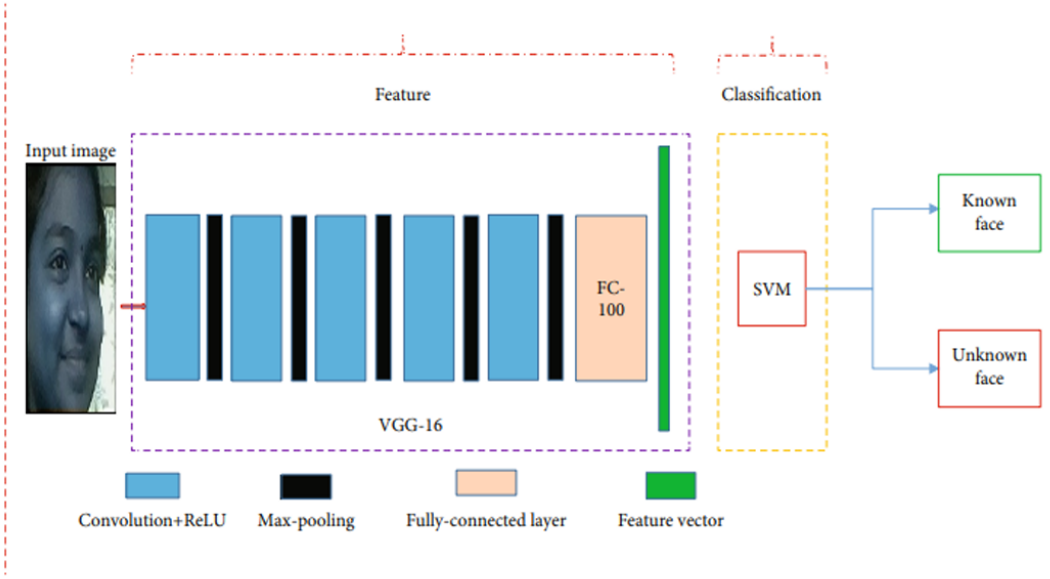
CONVOLUTION: In CNN-based face comparison, convolution mathematically combines two sets of face information. Each face undergoes feature extraction independently using the same filters, ensuring consistent face representation regardless of the comparison.

POOLING: Pooling is the step in which features are extracted from the image output of a convolutional layer. This involves reducing the dimensionality of the image by extracting its key features, and combining the resulting output of the previous layer into a single one.

4-Classification:

FULLY CONNECTED LAYER: Following pooling, a fully connected layer predicts the image's label by flattening the previous layer's output and passing it through FC layers. Multiple FC layers are used for better performance. This marks the classification stage.

OUTPUT: The classified and extracted images are then outputted. Here, a pre-trained ResNet50 CNN is employed for feature extraction and classification.



-Resnet-50 with SVM model

ResNet-50, a deep CNN, extracts features for face recognition, while **SVM** classifies these features. Combining them enhances accuracy. Here's how it works:

1-Face Detection: Use Haar Cascade or similar methods to locate faces in input images.

2-Feature Extraction: Employ ResNet-50 to extract robust facial features.

3-Feature Encoding: Convert extracted features into fixed-length vectors for comparison.

4-SVM Classification: SVM learns to classify faces based on encoded features.

5-Face Recognition: Compare new face images with labeled ones using the same pipeline, returning the closest match's identity.

-Support vector machines: SVM (Support Vector Machines) is a Machine Learning algorithm used for classification and regression. In face recognition, SVM categorizes faces based on extracted features. Here's how it works:

1-Input image undergoes face detection to extract the face region.

2-Local Binary Patterns (LBP) extract features from the face region.

3-Features represent the face in a feature space for comparison.

4-SVM learns to classify faces by maximizing margin between classes using a hyperplane.

5-During recognition, new face image passes through the same pipeline, with SVM outputting the closest match's class label based on feature vector distance.

5- Train the model:

Haar cascade classifiers are object detection algorithms used to detect objects, such as faces, in images or video. They analyze images at various scales to identify characteristic features of the object. Positive and negative examples are used to train a cascade of classifiers. The algorithm learns to distinguish relevant features, such as eyes and nose, from irrelevant ones. Once trained, the cascade is applied to an image by scanning it with a sliding window, checking for the presence of detected features to identify objects like faces.

Here’s an explanation of how a Haar cascade classifier works for face detection:

1. Training Stage: During the training stage, the algorithm is trained on a large dataset of images containing faces (positive samples) and images not containing faces (negative samples).

2. Feature Extraction: The algorithm extracts features from each image in the dataset. Haar-like features are used to capture the differences in intensity between adjacent regions of an image.

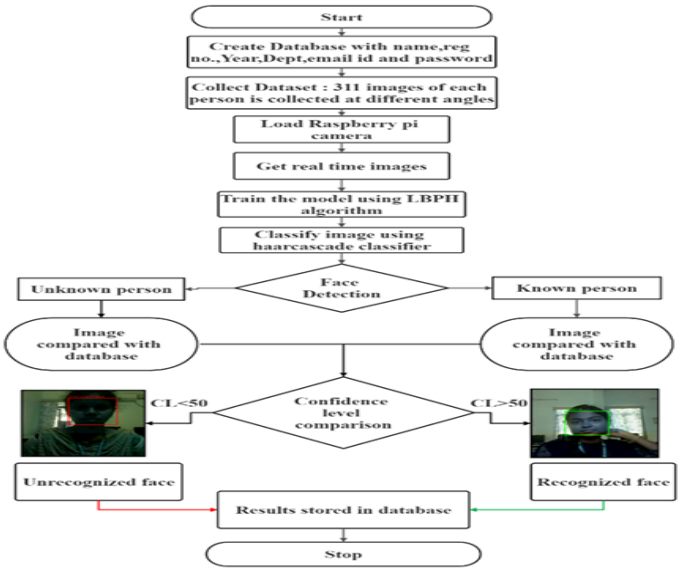
3. Adaboost Learning: Adaboost is a machine learning algorithm that is used to combine multiple weak classifiers into a strong classifier. Each weak classifier is trained

to detect a specific feature in the image, and Adaboost selects the best features to create the strong classifier.

4. Cascade of Classifiers: The final strong classifier is a cascade of multiple classifiers. Each classifier in the cascade is trained to detect a different set of features in the image. The cascade works by rejecting regions of the image that are unlikely to contain the object being detected, while passing promising regions on to the next classifier in the cascade.

5. Detection: During the detection stage, the cascade is applied to a new image. The algorithm scans the image at different scales and locations, and at each location, the features of the image are compared to the features in the cascade. If the features match those of a face, then the algorithm identifies the region as a face.

- Local binary pattern histogram: LBPH is a face recognition algorithm. The algorithm's workflow is depicted below and comprises five steps. It relies on four parameters: radius, neighbors, grid X, and grid Y.



1. Radius: The algorithm calculates the radius of the face image by measuring the distance between the eyes, nose, jaws, lips, and forehead.

2. Neighbors: A graph is created by connecting the central points of facial features such as eyes, nose, lips, and cheeks.

3. Grid X: The width of the face is measured horizontally.

4. Grid Y: The height of the face is measured vertically.

5. Training the Algorithm: training the algorithm using a dataset that contains images of the individuals we wish to recognize. Each image in the dataset must be assigned a unique identifier, which can be a number or name of the person. The algorithm utilizes this information to recognize an input image and produce an output. Images of the same person should have the same identifier. Once the training dataset is prepared, we can observe the computational steps involved in LBPH.

6. Applying the LBP operation: the initial computational stage involves applying the LBP operation to generate an intermediary image that accentuates the facial features and better represents the original image. This is achieved through the implementation of a sliding window approach that takes into account the radius and neighbors parameters.

7. Extracting the Histograms: With the image produced in the previous step, we can partition it into several grids by employing the Grid X and Grid Y parameters.

8. Performing face recognition: At this stage, the algorithm has undergone training. The resulting histograms are utilized to represent each image from the training dataset.

Advantages:

1. User-Friendly and Fast Deployment

- Deep learning frameworks provide user-friendly APIs, simplifying model creation and deployment with minimal code.
- Ideal for various applications including facial recognition due to ease of implementation and rapid prototyping capabilities.

2. Multiple Backend and Modularity

- Support for multiple backends such as TensorFlow, PyTorch, and MXNet offers flexibility for different project requirements.
- Modular design allows for seamless integration of different components and facilitates experimentation.

3. Pretrained models

- The availability of pre-trained deep learning models with pre-trained weights accelerates model deployment and feature extraction, reducing training time and computational resources.
- Pretrained models come with robustness and generalization capabilities, having been trained on large datasets like ImageNet.

4. Multiple GPU Support

- Deep learning frameworks offer built-in support for training on single or multiple GPUs, leveraging parallel processing for faster training and improved performance.
- Effective handling of large datasets enhances scalability and efficiency.

Limitations:

1. Problems in low-level API

- Occasional low-level backend errors disrupt workflow and hinder development, especially when performing advanced operations or customizations beyond the framework's capabilities.
- Limited backend customization options pose challenges for specific requirements or optimizations.

2. Need improvement in some features

- Data preprocessing tools lack the versatility of standalone libraries like scikit-learn, requiring additional implementations for tasks like clustering or principal component analysis (PCA).
- Lack of certain features, such as dynamic chart creation, limits visualization options and analysis capabilities.

3. Slower than its backend

- Deep learning frameworks exhibit slower computation speeds compared to their underlying backends, particularly when running on GPUs.
- Trade-offs between user-friendliness and computation speed may need to be considered depending on the specific application requirements.

4. Defined Size of the Image:

- Some deep learning models, including those used for facial recognition, have predefined input image sizes (e.g., 256x256), restricting adaptability to varying input dimensions.

5. Loss of Spatial Information:

- Facial images are summarized into singular output values, potentially leading to loss of spatial information crucial for accurate recognition tasks.

Facial Landmark Detection Algorithms

Facial landmark detection algorithms automatically identify key points on a face, such as the nose tip, eye corners, eyebrows, and chin tip, from images or videos. These algorithms also localize distinctive facial features, including fiducial points like the nose's

tip, providing a reliable method for analyzing and understanding facial expressions and structures. (Fiducial points serve as reference points within a scene, against which other objects can be measured or related.)

Main characteristics:

Facial landmark algorithms involve two crucial steps to identify key points on a face:

1- Face detection:

In landmark detection, face detection is an essential step. It involves recognizing facial regions in each frame of a video or image. The process precisely locates the human face in an image then returns a value that is the location of the face's bounding rectangle (box). This rectangle pinpoints the area within which facial landmarks, such as the eyes, nose, and mouth, will be detected and localized. A common method that can be used for face detection is the Viola-Jones algorithm.

2- Landmark detection:

Once a face is detected, landmark detection focuses on pinpointing specific key points on that face. Different approaches exist for identification, but all share a common goal: to locate and categorize the following anatomical landmarks: right eyebrow, left eyebrow, right eye, left eye, nose, mouth, and jaw.

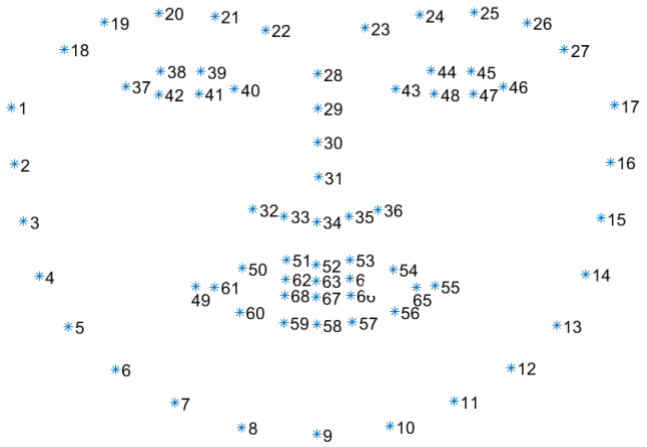
It exists in two key approaches that can be taken: Traditional Approaches and Modern Deep Learning Approaches.

- **Statistical Models (Traditional Approach):**
  Statistical models like *Active Shape Models (ASMs)* and *Active Appearance Models (AAMs)* use a training dataset of labeled facial images with corresponding landmark locations. These models statistically analyze the training data to learn typical variations in facial shapes and appearances.

Active Shape Models (ASM): The Active Shape Model (ASM) is an efficient and standard method for facial landmark detection. Here's a breakdown of the key steps involved:

1- Initialization:

The process starts by aligning a "mean shape" (average facial shape from the training data) to the detected face region. This mean shape serves as a reference point for the subsequent steps. Facial shapes are represented as vectors, where each element combines the x and y coordinates of a specific landmark (typically around 68 points encompassing eyebrows, cheeks, eyes, mouth, and nose).



68 facial landmark points

2- Iterative Refinement:

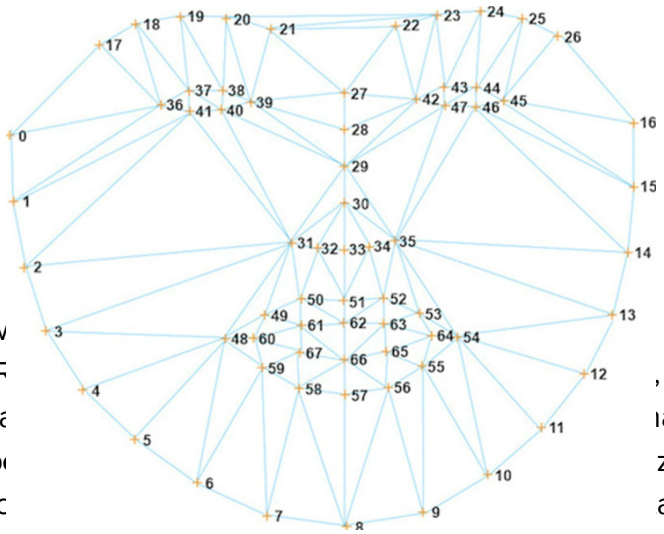
In each iteration two key processes happen:

2-1- Proposing a Temporary Shape: The model analyzes the detected face region. Based on the image texture and learned patterns from the training data, it proposes an initial estimate for the location of each facial landmark. This initial guess is called the "temporary shape."

2-2- Refining the Shape: The proposed temporary shape is then compared to a "universal shape model." This model is a statistical representation of the variations in facial shapes learned from the training data. The ASM adjusts the temporary shape to better conform to the universal shape model, essentially refining the estimated landmark locations. The iterative process continues until convergence.


Initial Position | After 1 iteration | After 2 iterations | Convergence

into smaller areas using triangles, with vertices being the ASM landmark points, as shown below:



Here's a breakdow , and translation to ensure all shapes a ialized training shapes.

1- Normalization: F

2- Principal Comp zed training shapes to capture the signific a. This analysis yields:

Mean Shape: Similar to ASM, this represents the average facial shape from the training data.

Shape Bases: These are a set of eigenvector directions that capture the most prominent variations in facial shapes from the training data.

3- Building the Appearance Model:

Image Warping: Each training image is warped to align with the mean shape obtained in step 2. This focuses on analyzing appearance variations independent of pose or scale.

Second PCA for Appearance: PCA is applied again, but this time on the warped images, capturing the variations in appearance (grayscale intensity patterns) across the training data. This results in:

Mean Appearance: Average grayscale intensity pattern for a face after pose and scale variations are removed.

Appearance Bases: Eigenvector directions capturing prominent appearance variations like wrinkles, freckles, etc.

Once the shape and appearance models are built, the AAM can be used for facial landmark detection in new images.

Initial 3 its 8 its 11 its Converged Original

- **Deep Learning (Modern Approach):**
  In facial landmark algorithms, deep learning uses CNNs for feature extraction. The fundamental concept is CNN's ability to automatically learn features directly from the image data. These features capture various aspects of a face, such as edges, corners, and textures, that are crucial for identifying landmarks. To train a CNN for facial landmark detection, a massive dataset of images containing faces is required. Each image needs to be labeled with the corresponding location of facial landmarks (e.g., eyes, nose, mouth). The CNN analyzes the image, extracts features using its convolutional layers, and tries to predict the landmark locations based on the learned features. Once trained, the CNN can be used to detect faces and localize landmarks in new images.

OpenCV

The OpenCV library is popular for real-time computer vision tasks, enabling the detection of landmarks on detected faces in videos or images. This application first detects faces in a current video frame or image and then finds their facial landmarks. OpenCV employs both traditional methods by applying pre-trained AAMs and offers functions like *cv2.face.fitShapeModel()* for landmark detection in new images. Additionally, it utilizes modern deep learning methods, such as landmark detection with Deep Neural Networks (DNNs). Functions like *cv2.dnn.detectLandmark()* within the DNN module allow you to leverage pre-trained deep learning models for more accurate landmark detection, especially in challenging scenarios.

Advantages:

High Precision: Deep learning models offer high precision landmark localization, crucial for facial recognition.

Real-time Processing: The real-time processing abilities of facial landmark detection make it suitable for video analysis or interactive applications. Both AAM and deep learning models can achieve real-time performance.

Robustness: Facial landmarks are robust against variations in lighting conditions, facial expressions, and textures making them versatile.

Limitations:

Sensitivity to Occlusions: Facial landmark algorithms might struggle to maintain accuracy when the face is obscured by objects like glasses, hats, or even facial hair.
Training Data Dependency: AAM and deep learning rely heavily on training data. Poor training data can significantly impact performance and decrease accuracy.
Privacy Concerns: Tracking and analyzing facial landmarks raises ethical concerns, especially without user consent.

Viola-Jones algorithm

The Viola-Jones algorithm is a **machine-learning technique** for object detection that was proposed in 2001 by Paul Viola and Michael Jones. It was primarily conceived for face detection.

The Viola-Jones algorithm uses a set of features similar to **Haar wavelets**, which are a set of square-shaped functions. The feature value is computed as the difference between the sum of the pixel values in the white areas and the sum of pixel values in the black areas.

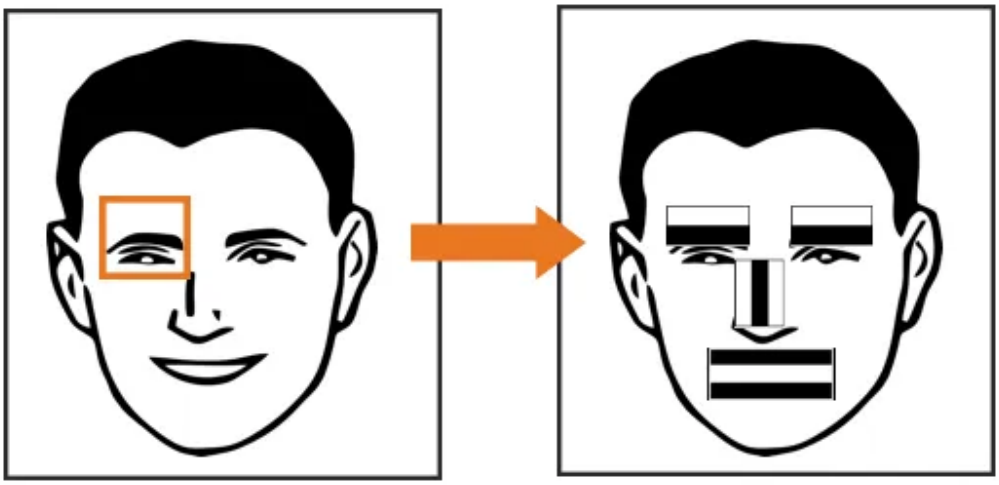
The two most common are Edge Features and Line Features:

Edge Features

So let's say for example you want to detect part of a face, in this case an eyebrow, naturally the shade of the pixels on an eyebrow in an image will be darker and abruptly gets lighter (skin).

Line Features

Now let's say you want to detect a mouth: naturally the shape of the lips region on your face goes from light to dark to light again. For this, Line features prove to be the best.



It also uses **integral images** for accelerating the feature computation, Each point in the integral image is a sum of the pixels above and left of the corresponding pixel in the source image

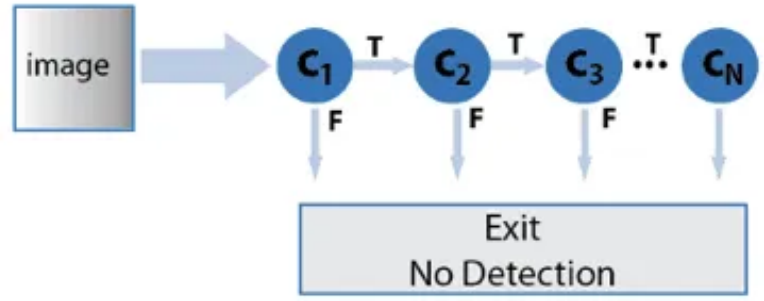
1	3	7	5
12	4	8	2
0	14	16	9
5	11	6	10

Source Image

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

Integral Image

AdaBoost learning for feature selection, and a **cascade of classifiers** for fast rejection of windows without faces.



Here are some of its advantages and limitations:

Advantages:

1. **Efficiency**: The Viola-Jones algorithm is an efficient solution for resource-constrained devices. It has relatively high detection speed and accuracy, and low computational power requirements.
2. **Real-time Detection**: Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection.
3. **Versatility**: It can be learned to detect any object, not just human faces.
4. Detection is very fast
5. Simple to understand and implement
6. Less data needed for training than other ML models
7. No rescaling of images needed (like with CNN's)
8. Much more interpretable than contemporary models

Limitations:

1. **Accuracy**: Despite its efficiency, the Viola-Jones algorithm has lower accuracy than modern face detection methods based on Convolutional Neural Networks (CNNs).
2. **Training Time**: This algorithm is very slow to train.
3. Restricted to binary classification
4. Mostly effective when face is in frontal view
5. May be sensitive to very high/low exposure (brightness)
6. High true detection rate, but also high false detection rate

Report- Part3

IT 360

Security Project

Authors:

Jaouher ElMani

Jihene Gazzeh

Farah Jaouadi

Nour Elloumi

Contents

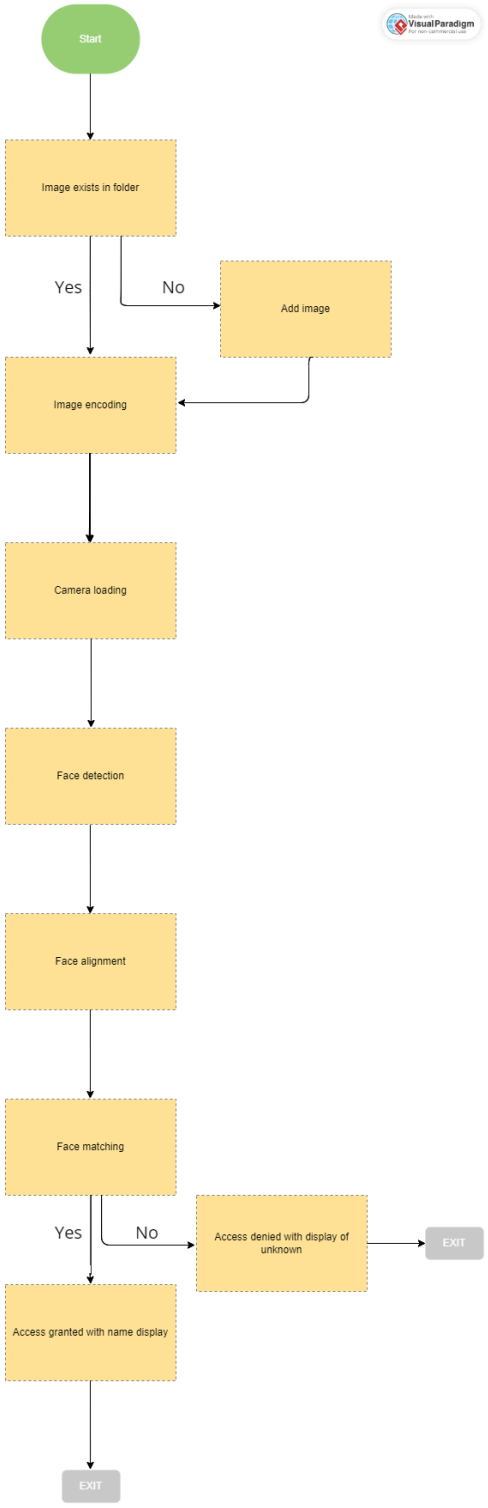
1- Purpose of our solution

The aim of our facial recognition project is to be implemented in an employee check-in system to enhance security and streamline the check-in process.

Using facial recognition, employees can simply scan their faces at checkpoints to gain access/authorization to specific areas of the building. Thus, facial recognition provides a secure, convenient, and efficient solution for employees, improving workplace efficiency.

This approach eliminates the need for physical keys and reduces the risk of unauthorized access.

2- Design


Start
Image exists in folder
Yes
No
Add image
Image encoding
Camera loading
Face detection
Face alignment
Face matching
Yes
No
Access denied with display of unknown
EXIT
Access granted with name display
EXIT

3- Steps/Main components

1. Start:

2. Images exist in the folder?: The system verifies the presence of an image in a designated folder.

3. Add Image: In the absence of an image in the folder, the system waits for an image to be added by the user

4. Images Encoding:Upon the presence of an image, the system undertakes an encoding process. This process involves the transformation of raw image data into a format that can be utilized for comparison. Key facial features are identified and numerically represented.

5. Camera Loading: This refers to the system accessing a live feed from a camera for real-time facial recognition.

6. Face Detection: detect the presence of a face. This is typically done using machine learning algorithms that have been trained to identify faces.

7. Face Alignment: Following the detection of a face, the system aligns the face to a standard format. This generally involves the rotation and scaling of the face image to ensure consistent positioning of features such as the eyes, nose, and mouth. This alignment enhances the accuracy of the recognition step.

8. Face Matching: The system compares the aligned and encoded face image with a database of known faces.

- **Access Denial**: If the face is not recognized (i.e., it does not match any faces in the database), access is denied. The system may also display a message indicating the detection of an unknown individual.
- **Access Grant**: If the face is recognized (i.e., it matches a face in the database), access is granted. The system may also display the name associated with the recognized face.

Report- Part4

IT 360

Security Project

Authors:

Jaouher ElMani

Jihene Gazzeh

Farah Jaouadi

Nour Elloumi

Contents

1- Tools

- **Programming Languages**: Python is the primary language for coding and scripting.
- **Libraries and Frameworks**:
    - CV2: OpenCV (Open Source Computer Vision Library) is a popular library for computer vision tasks. It provides a wide range of functions for image and video analysis, including object detection, facial recognition, feature detection, and image processing.
    - OS: The os module in Python provides a way of interacting with the operating system. It allows the user to perform various tasks such as navigating file systems, creating and deleting directories, executing system commands, and accessing environment variables.
    - Numpy: NumPy is a fundamental package for scientific computing in Python. It provides support for large multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
    - Face recognition: Face Recognition provides tools for face detection, face recognition, and face manipulation tasks. It is built on top of other libraries like dlib and OpenCV, and it offers pre-trained models for face recognition tasks.
    - Glob: The glob module in Python provides a way to search for files that match a specified pattern. It allows to find files using wildcard characters (* and ?) and provides functions for traversing directories and obtaining lists of file paths that match the specified pattern.
- **Integrated Development Environments (IDEs)**: PyCharm
- **Version Control**: Git is employed for managing code changes and facilitating collaboration.

2. Development Phases

- **Requirement Analysis:** The primary objective of our facial recognition project is to enhance security and streamline the check-in process in an employee check-in system.

- **Design**: Utilizing Visual Paradigm, we construct a functional flow diagram to provide a clear and concise visual representation of the system's operation.
- **Coding**: This phase involves the actual creation of the system, where we transform our designs and plans into functional code.
- **Evaluation and Testing**: We assess the performance of the trained model on a separate test dataset. Key metrics such as accuracy, precision are used for evaluation.
- **Deployment**: The trained model is integrated into the target application or system.
- **Maintenance and Updates**: We monitor system performance over time, collect feedback, and periodically retrain the model with new data. This allows us to improve accuracy and adapt to evolving use cases and environmental conditions.