Facial Recognition Using Machine Learning

Team Members :
Pankaj Katkar
Shivam Kumar
Manika Khare

October 3, 2018

Introduction

- Face recognition is a software application capable of uniquely identifying or verifying a person
- Widespread technology
- Google's FaceNet and Facebook's DeepFace have achieved accuracy of 99.6% and 97.5% respectively.

Motivation

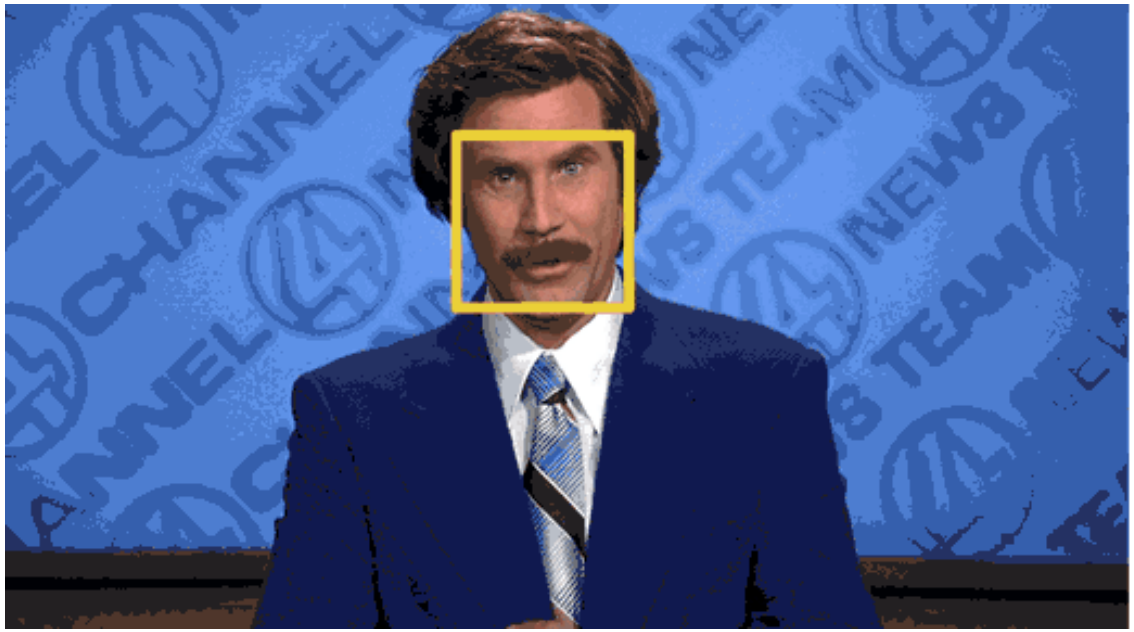
- Used for securing data through biometric authentication.
- Technology has significantly contributed in the domain of investigation and crime detection.
- Used by some organization to track the attendance of the employees
- Facebook uses it to tag persons automatically and Google uses it to group photos of different persons.

Face Recognition

Face Recognition

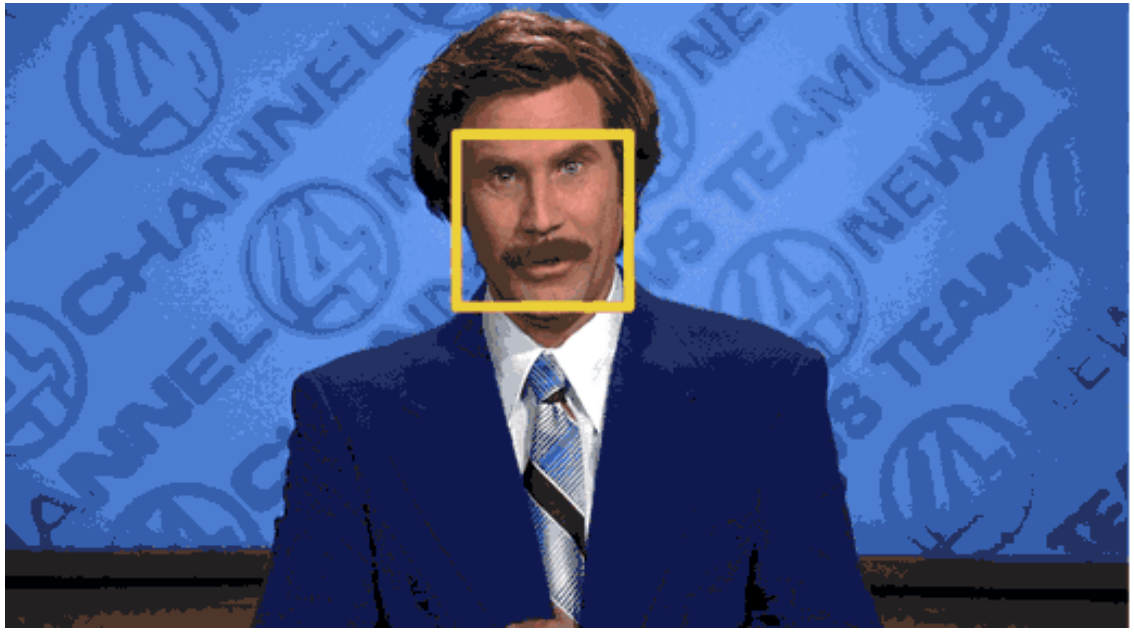
1. Find face in image

Face Recognition



1. Find face in image

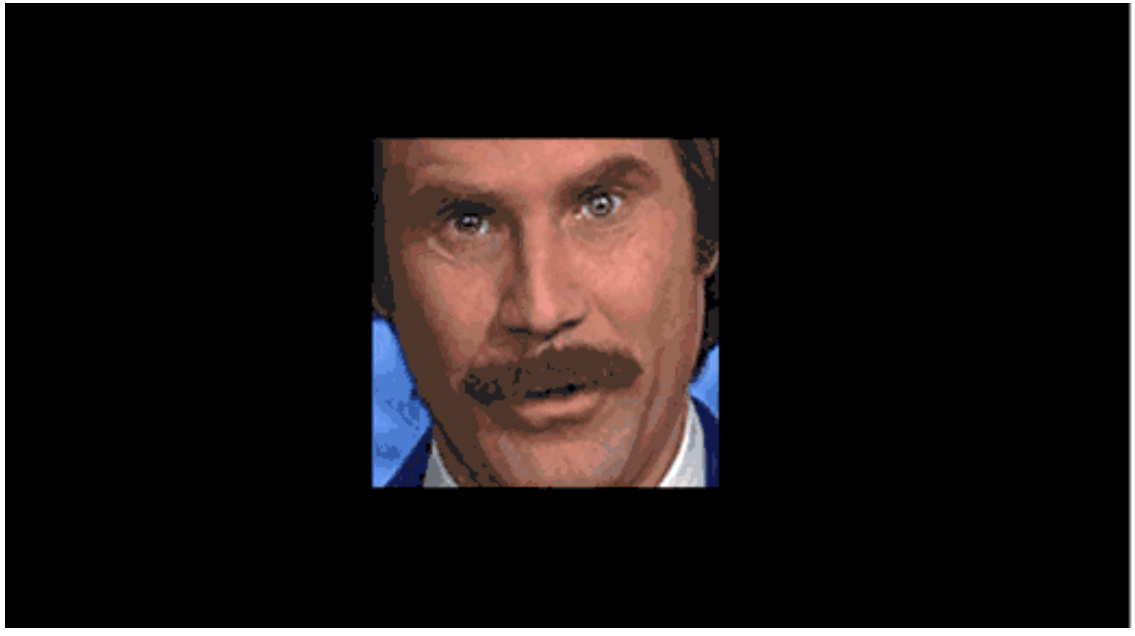
Face Recognition



1. Find face in image

2. Analyze facial features

Face Recognition



1. Find face in image

2. Analyze facial features

Face Recognition

1. Find face in image

2. Analyze facial features

Face Recognition

1. Find face in image

2. Analyze facial features

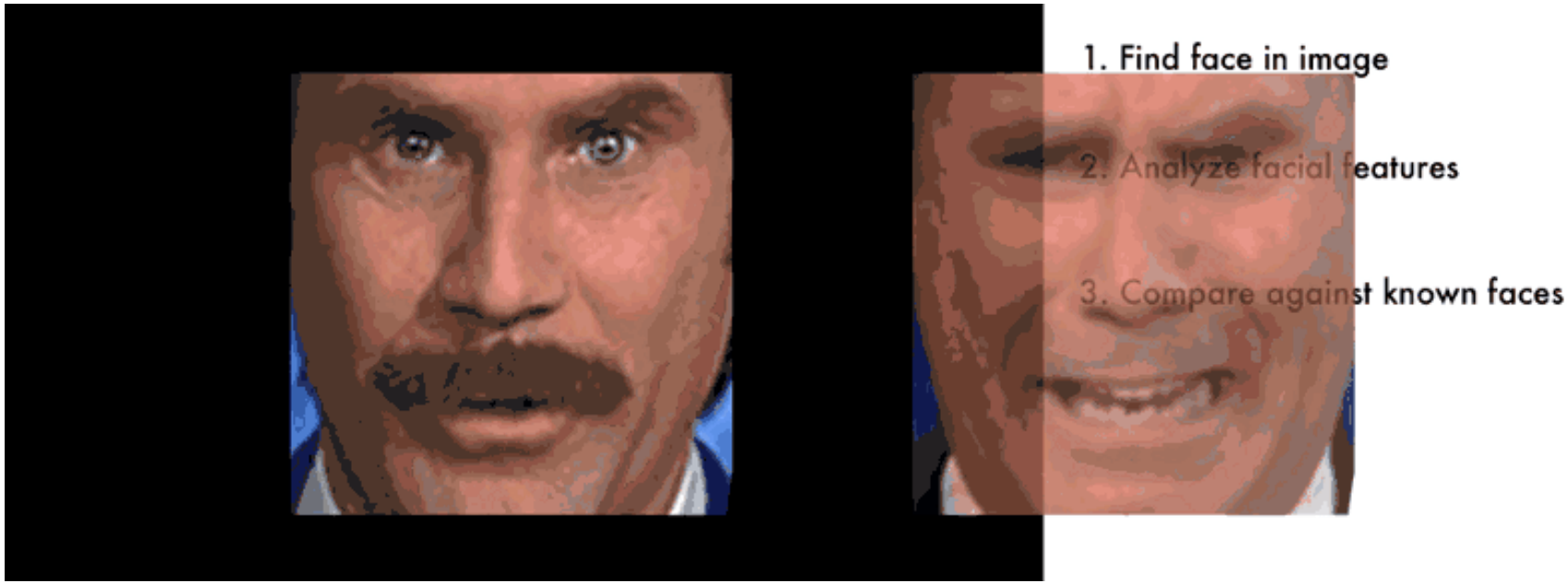
Face Recognition

1. Find face in image

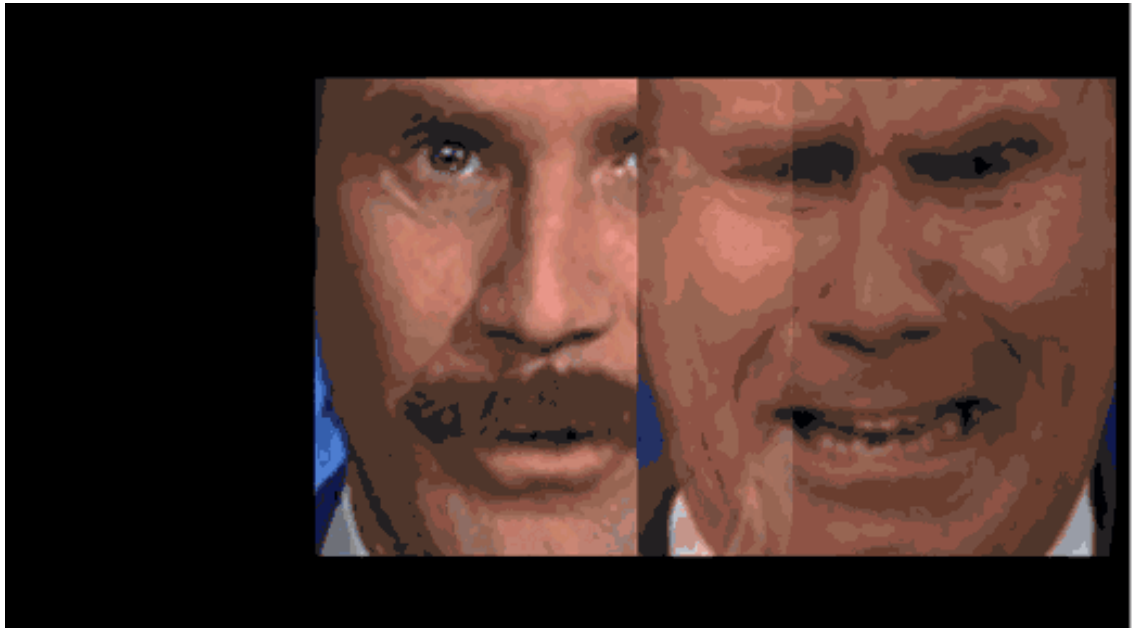
2. Analyze facial features

3. Compare against known faces

Face Recognition



Face Recognition

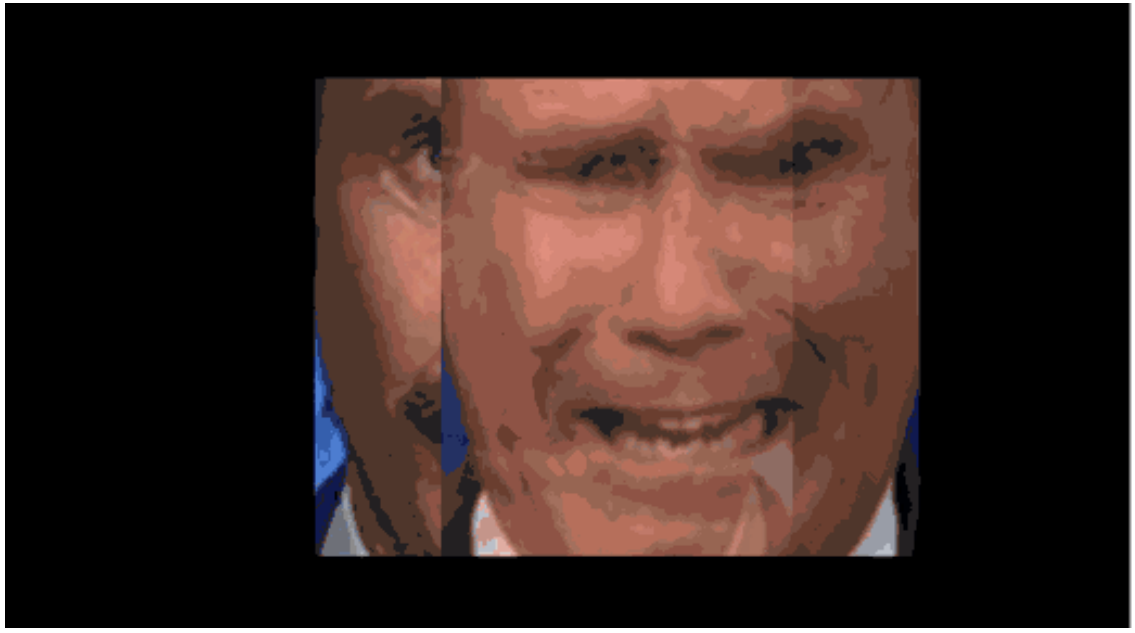


1. Find face in image

2. Analyze facial features

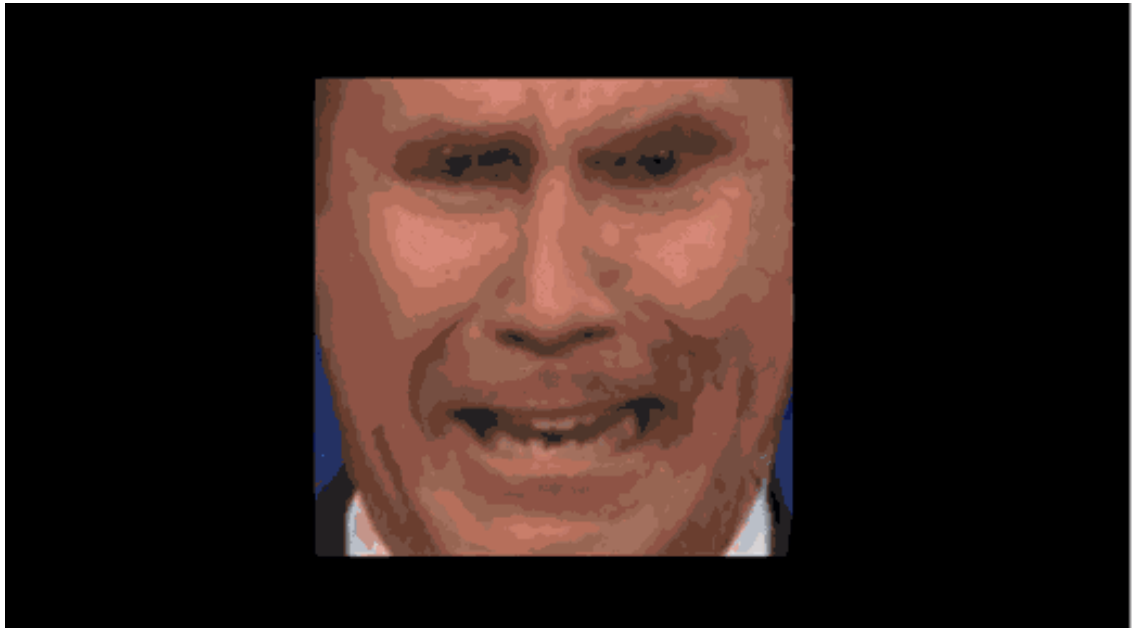
3. Compare against known faces

Face Recognition



1. Find face in image
2. Analyze facial features
3. Compare against known faces

Face Recognition



1. Find face in image

2. Analyze facial features

3. Compare against known faces

Face Recognition

1. Find face in image

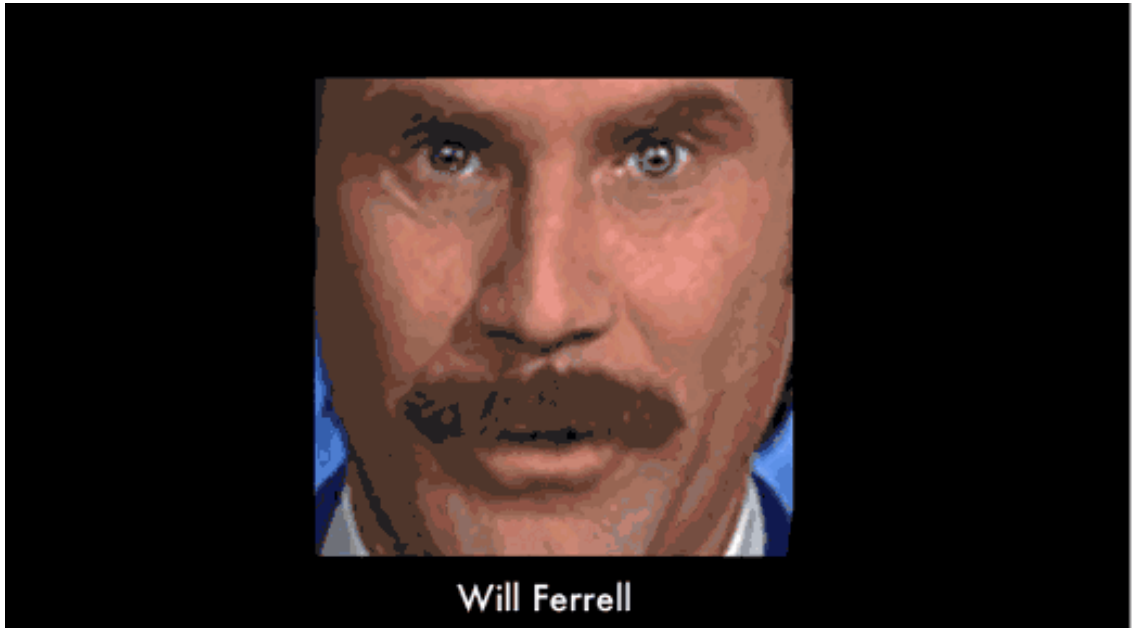
2. Analyze facial features

3. Compare against known faces

Face Recognition

1. Find face in image
2. Analyze facial features
3. Compare against known faces
4. Make a prediction

Face Recognition



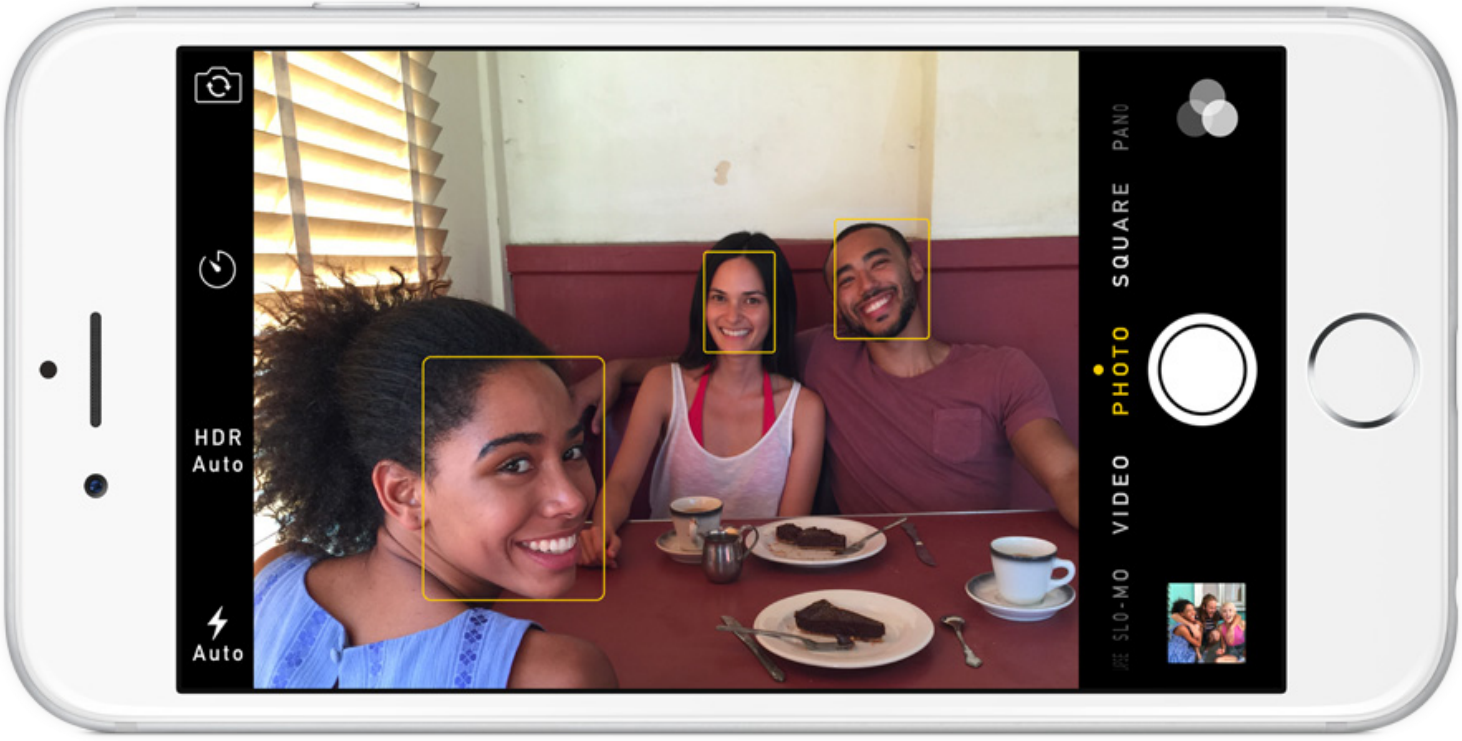
1. Find face in image
2. Analyze facial features
3. Compare against known faces
4. Make a prediction

Algorithms used

- Histogram of Oriented Gradients (HOG)
- Face Landmark Estimation
- Deep Convolutional Neural Network

Finding all Faces in Image

Finding all the Faces in Image



How to do this ???

- Using the method **Histogram of Oriented Gradients (HOG)**

Histogram of Oriented Gradients (HOG)

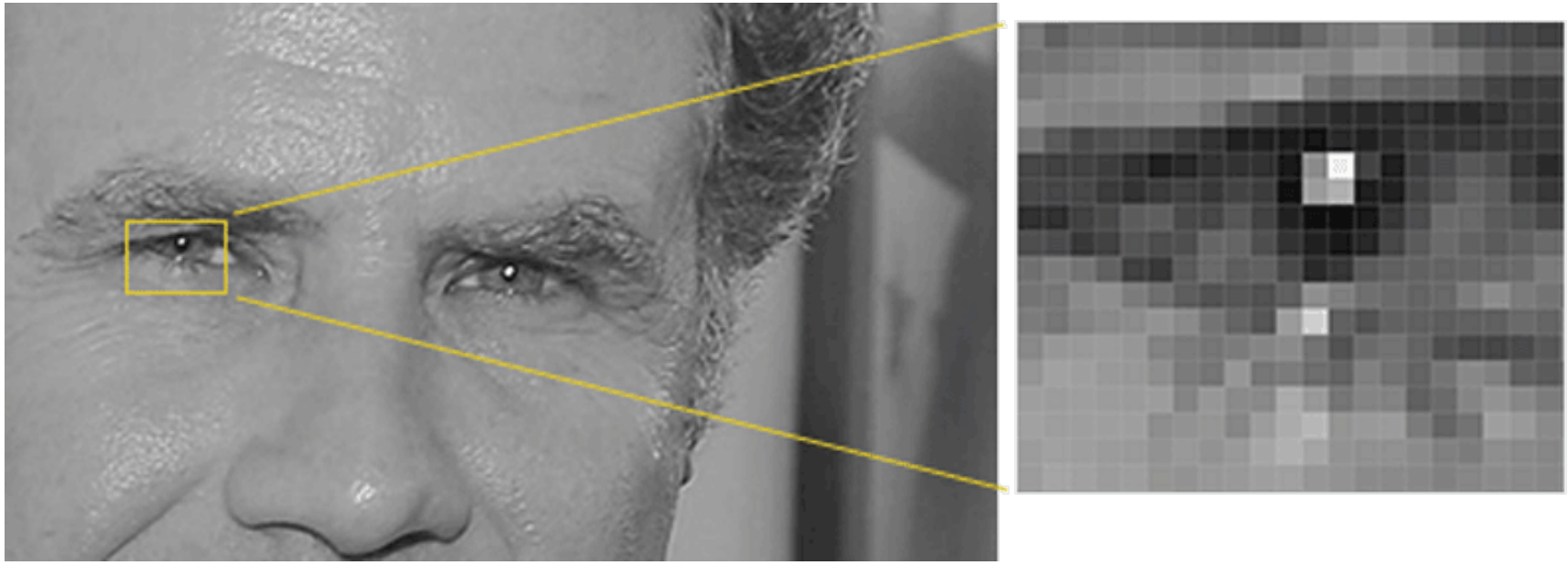
Step 1

- Convert Color Image To Black And White Image

Histogram of Oriented Gradients (HOG)

Step 2

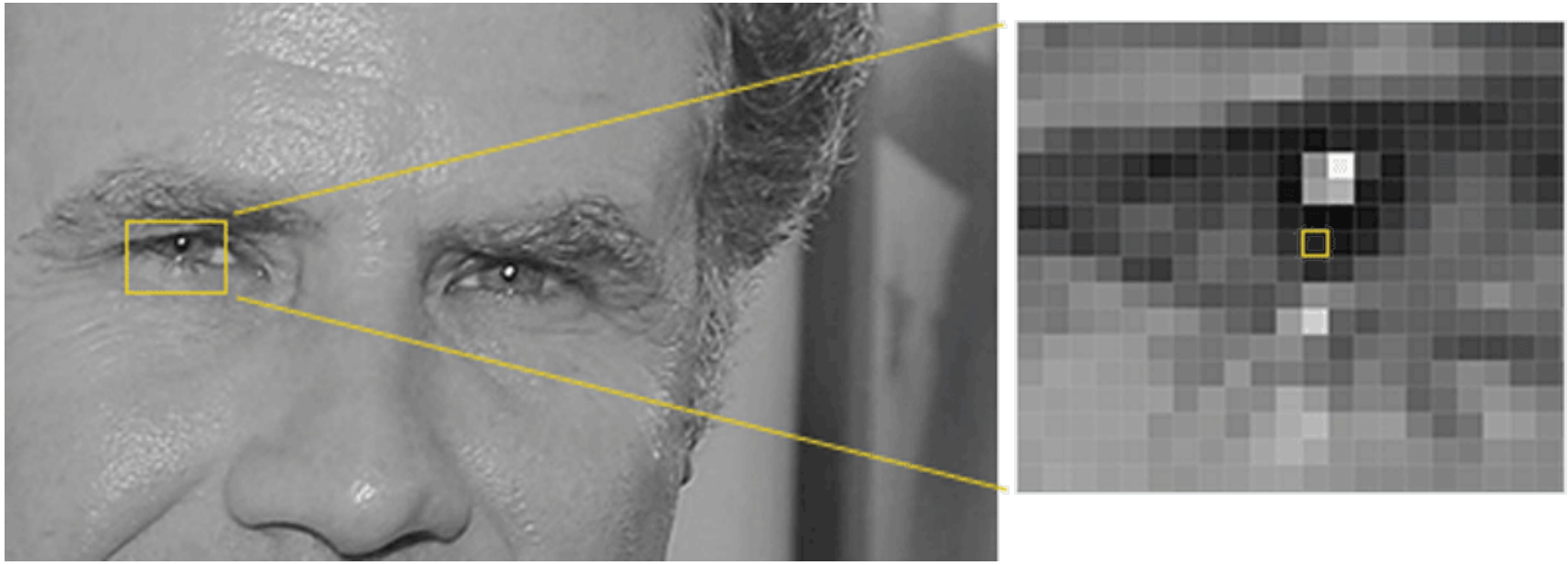
- For every single pixel, we look at the pixels that directly surround it



Histogram of Oriented Gradients (HOG)

Step 2

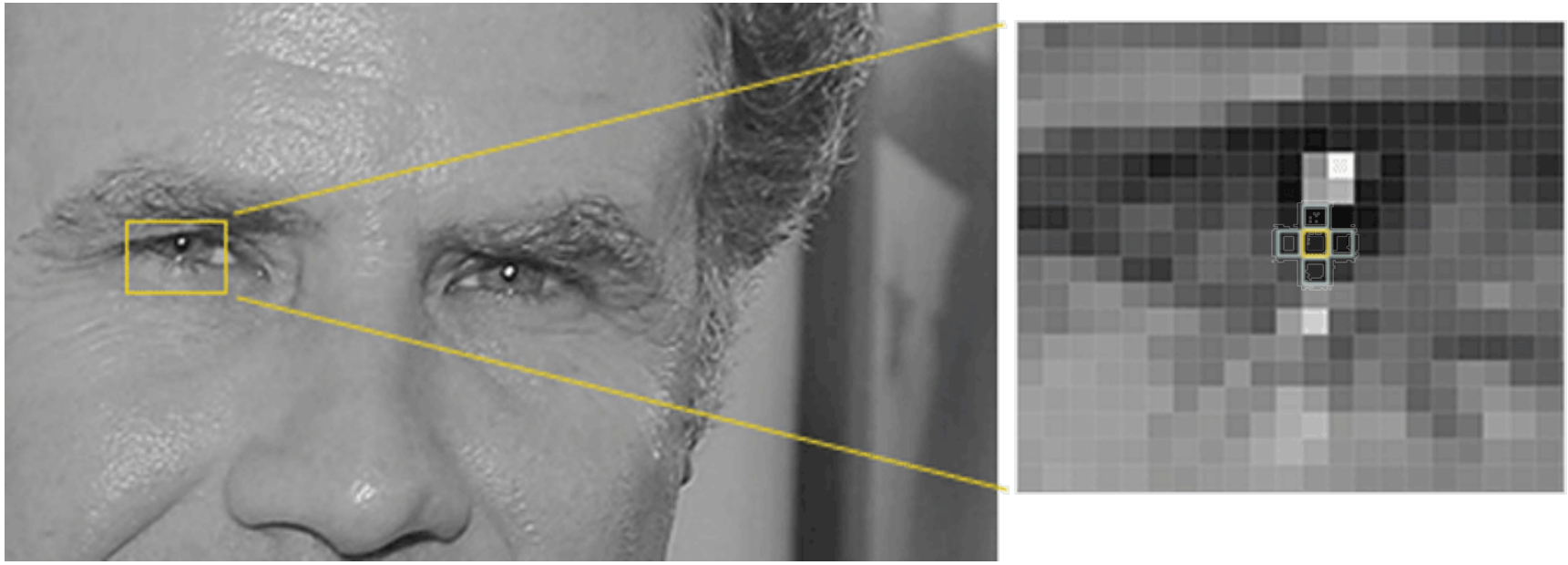
- For every single pixel, we look at the pixels that directly surround it



Histogram of Oriented Gradients (HOG)

Step 2

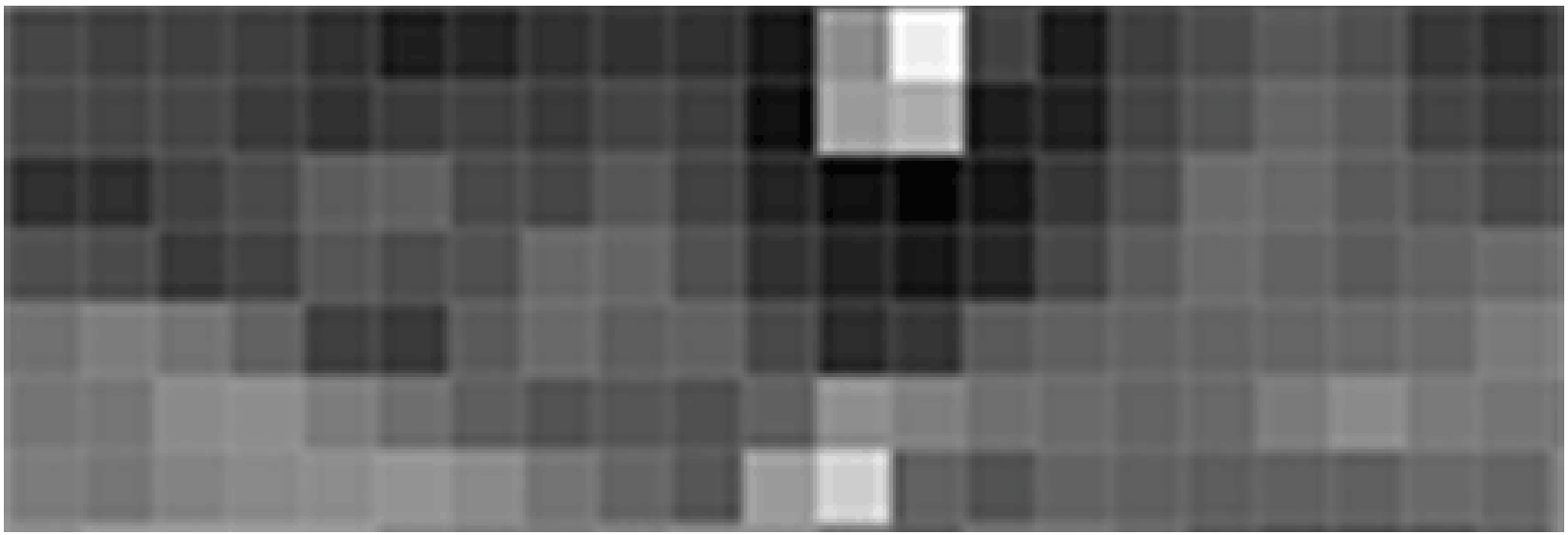
- For every single pixel, we look at the pixels that directly surround it



Histogram of Oriented Gradients (HOG)

Step 3

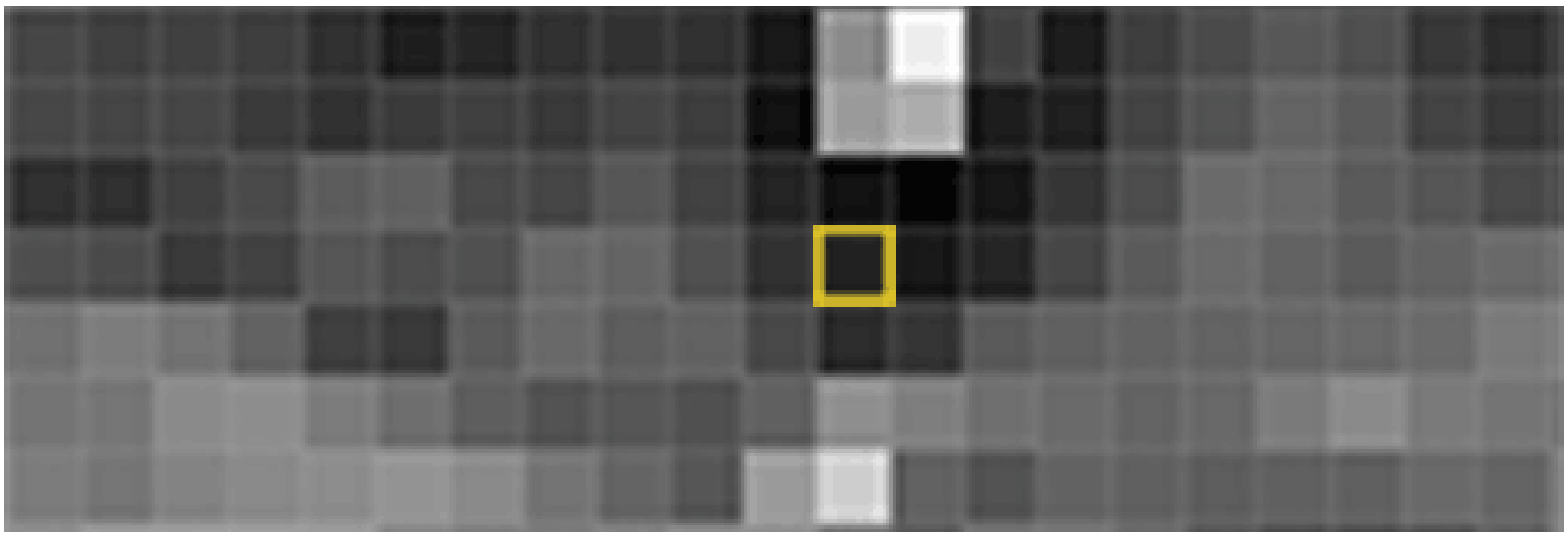
- Then we draw an arrow showing in which direction the image is getting darker .These arrows are called gradients



Histogram of Oriented Gradients (HOG)

Step 3

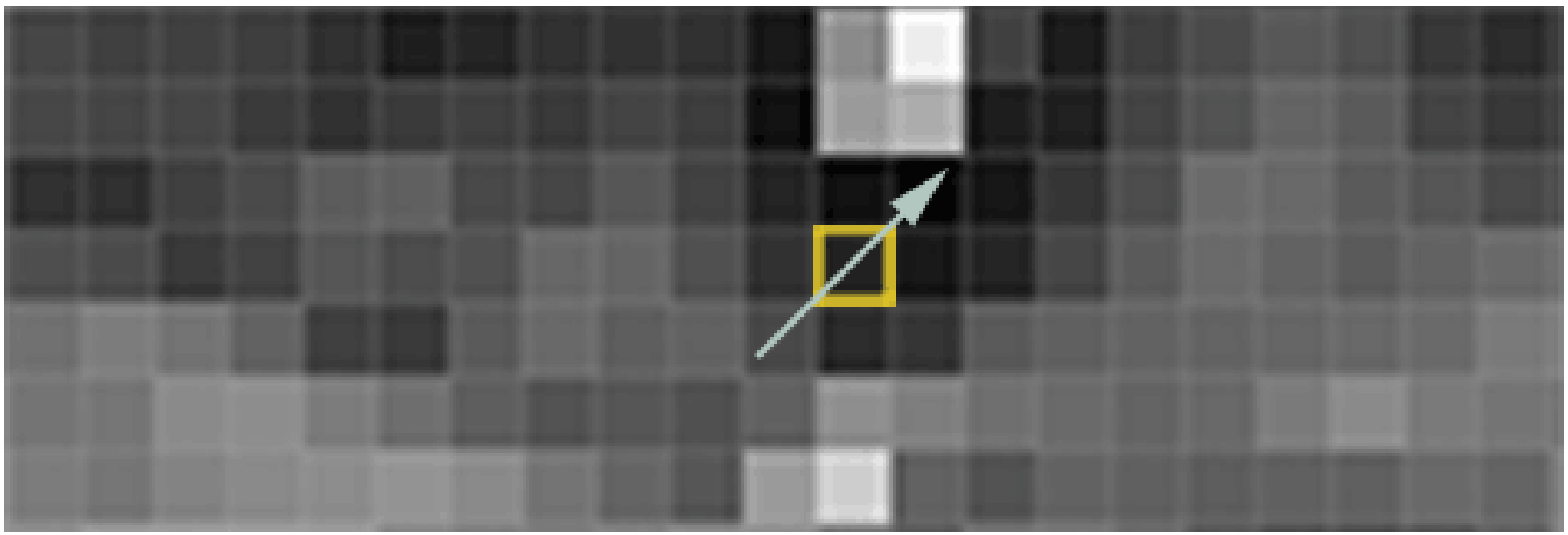
- Then we draw an arrow showing in which direction the image is getting darker .These arrows are called gradients



Histogram of Oriented Gradients (HOG)

Step 3

- Then we draw an arrow showing in which direction the image is getting darker .These arrows are called gradients

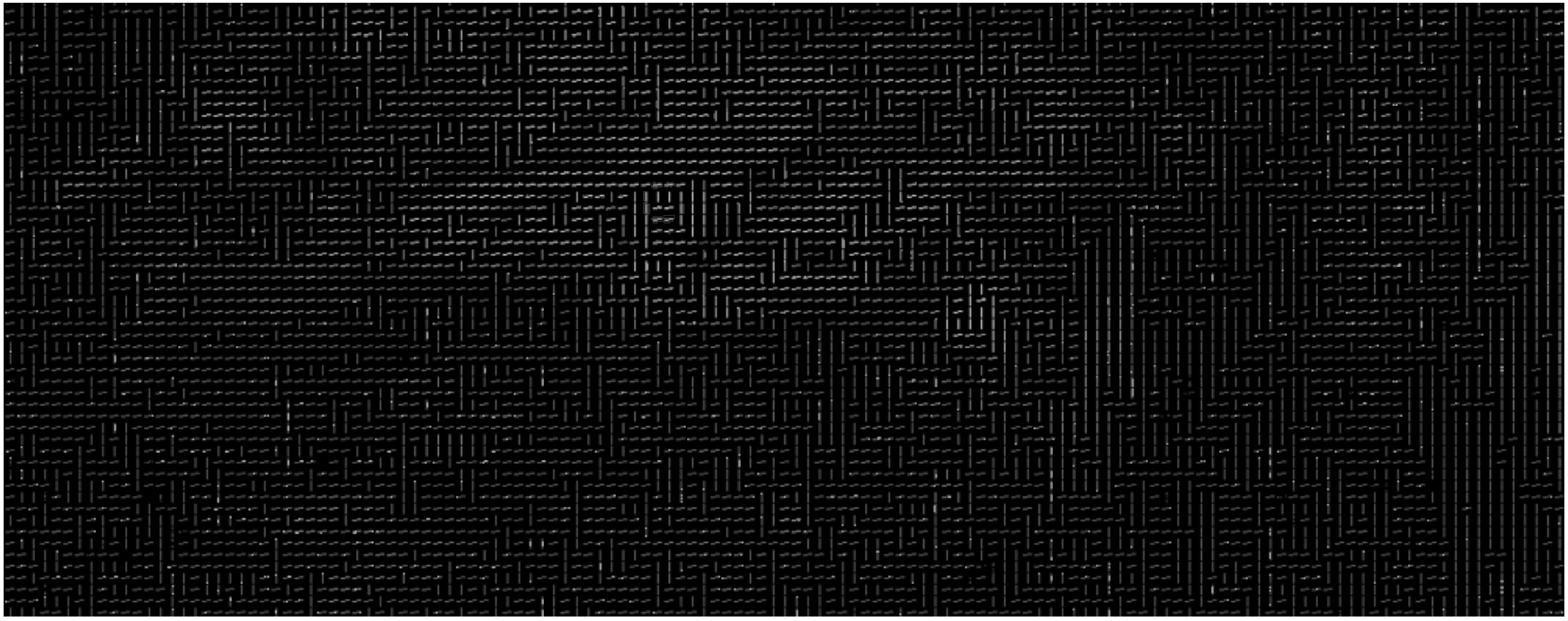


Histogram of Oriented Gradients (HOG)

Result

Histogram of Oriented Gradients (HOG)

Result



Histogram of Oriented Gradients (HOG)

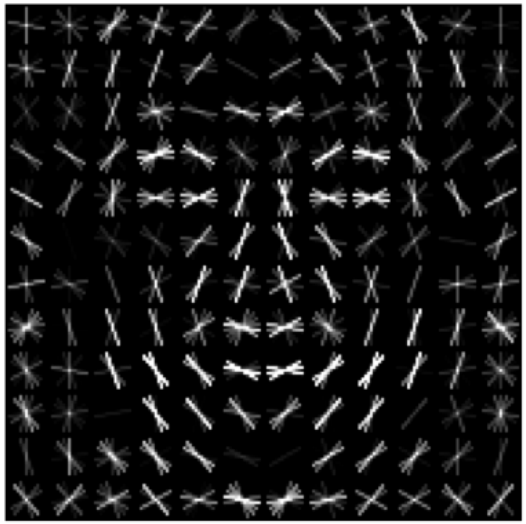
Step 4

- Saving the gradient for every single pixel gives us way too much detail
- We will break up the image into small squares of 16x16 pixels each.
- Then for each square, we will count up how many gradients point in each major direction (how many point up, point up-right, point right, etc).
- Then we will replace that square in the image with the arrow directions that were the strongest.

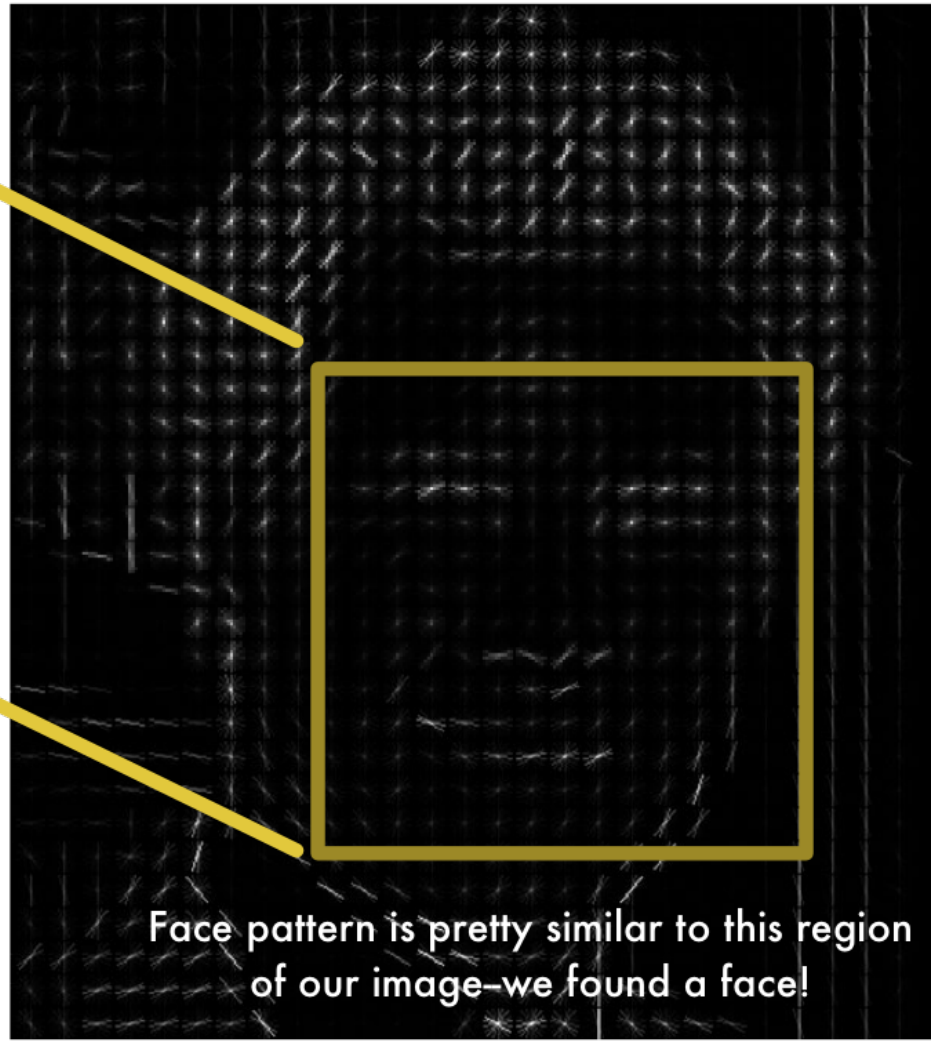
Histogram of Oriented Gradients (HOG)

Finding Faces in Image

HOG face pattern generated from lots of face images

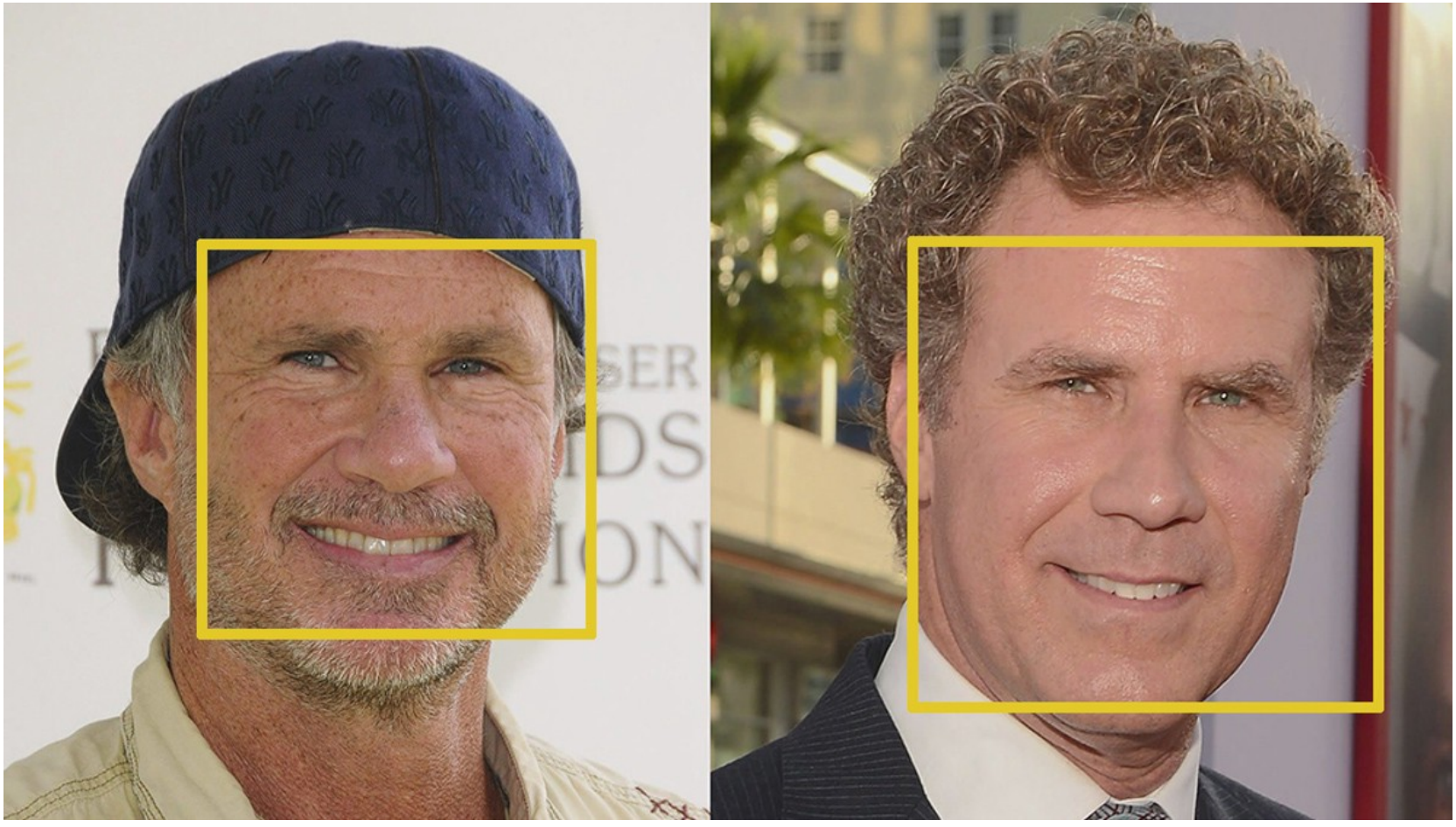


HOG version of our image



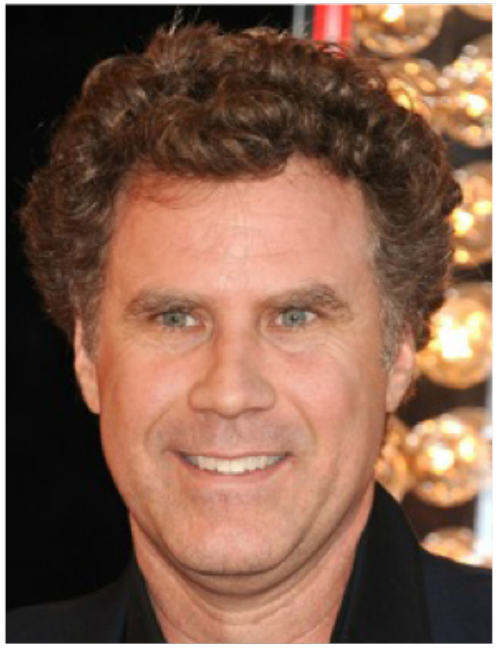
Histogram of Oriented Gradients (HOG)

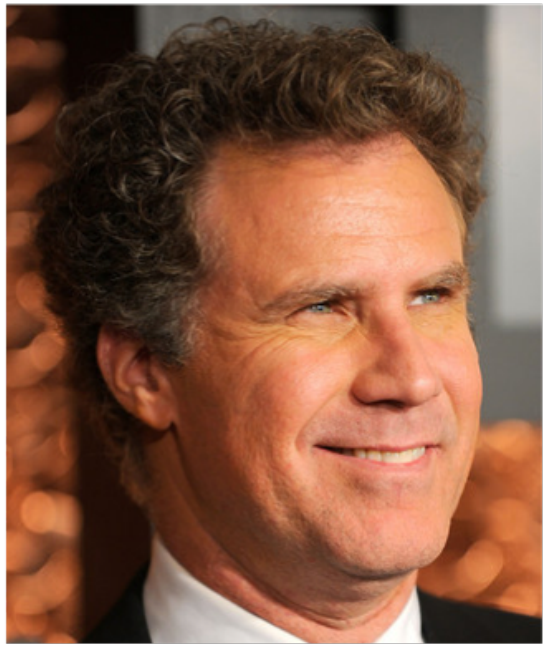
Now we can detect Faces in Image



Posing and Projecting Faces

Posing and Projecting Faces





Why to do this ???

- To deal with the problem that faces turned in different directions look totally different to a computer

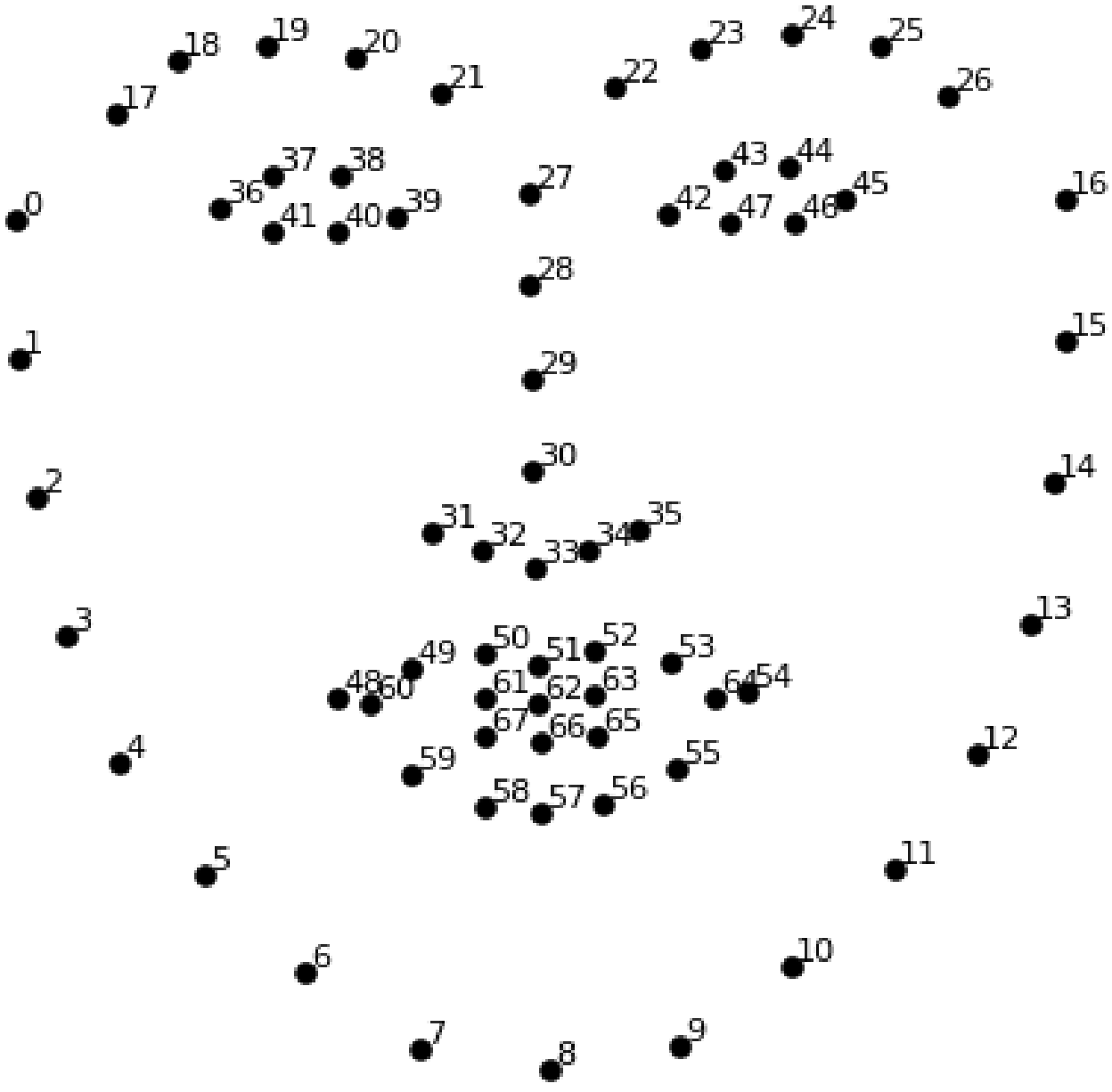
How to do this ???

- Using face landmark estimation algorithm

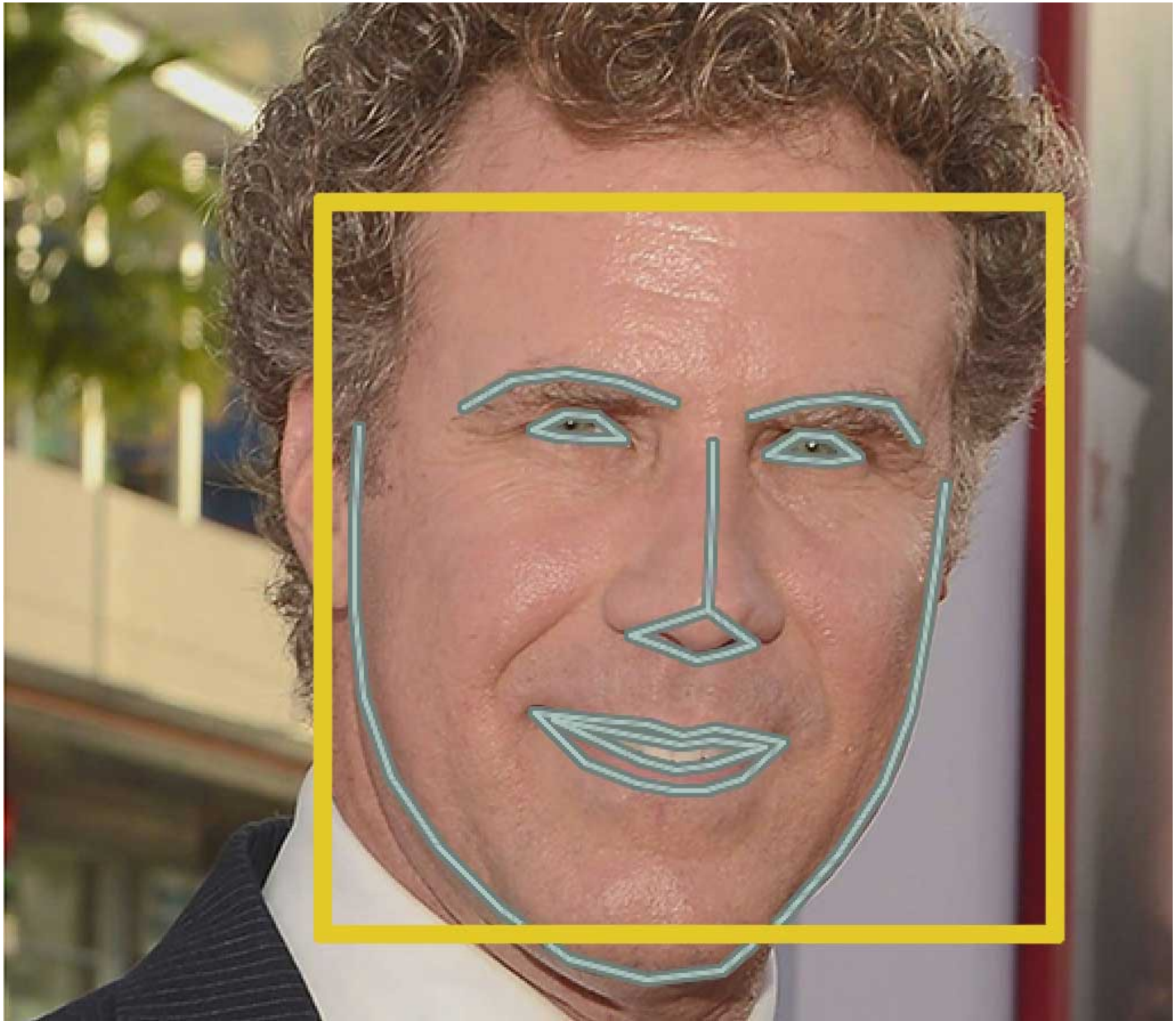
Face Landmark Estimation

- The basic idea is we will come up with 68 specific points (called landmarks) that exist on every facethe top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc.
- Then we will train a machine learning algorithm to be able to find these 68 specific points on any face

68 specific points on any face

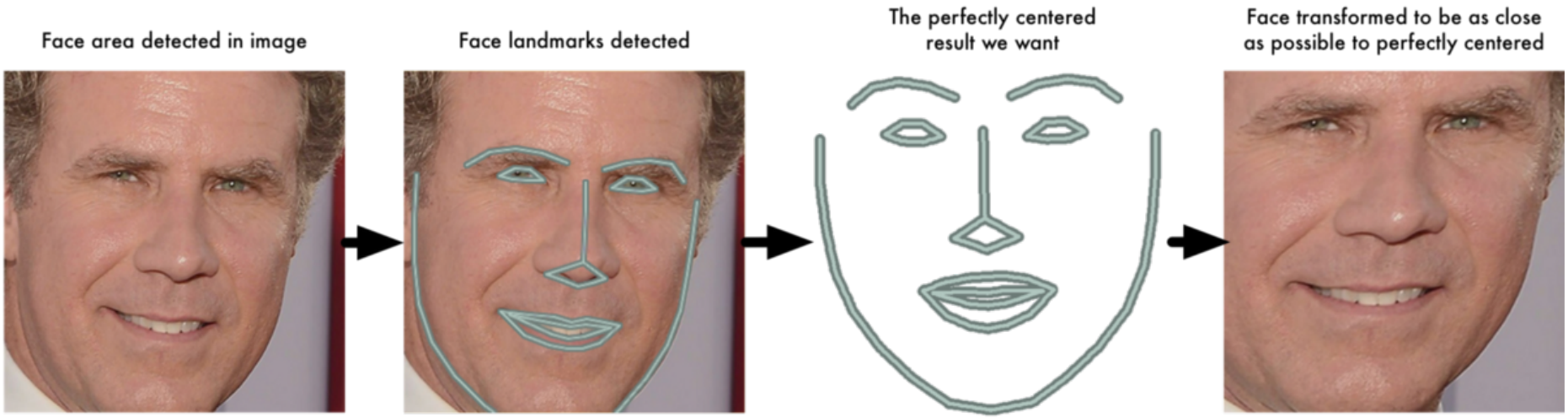


Result of locating the 68 face landmarks on our test image



Face Landmark Estimation Continued...

- Now that we know were the eyes and mouth are, well simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible
- We wont do any fancy 3d warps because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines



Encoding Faces

Encoding Faces

- The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged.

Seems like a pretty good idea, right??

- A site like Facebook with billions of users and a trillion photos cant possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long

Then, How to do this ??

How to Identify People Given their Faces

Figure: Person 1

Figure: Person 2

One-shot Learning

Learning from one example to recognize the person again.

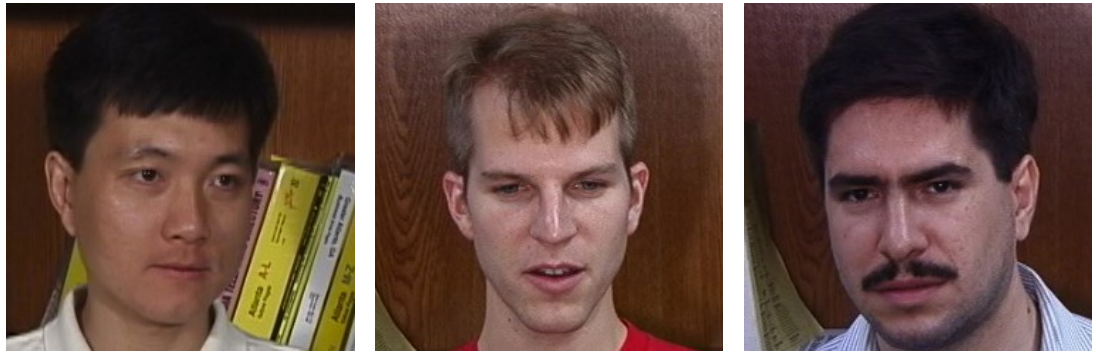


Figure: Training Pictures with Name/ID

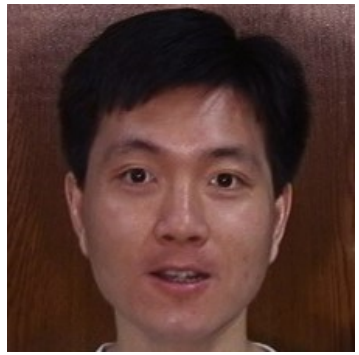


Figure: Identify the Person

A Smaller Problem

Face Verification

Figure: Same Person

Figure: Different Person

Face Verification vs Face Recognition

Verification

- Input : image, name/id
- Output : whether the input image is that of that claimed person.

Recognition

- Has a database of k persons
- Input : image
- Output : ID if the image is any of the k persons.(or "not recognized")

Learning a "similarity" function

- $d(img1, img2) =$ degree of difference between images
- if $d(img1, img2) \leq Threshold$: Same Person
- if $d(img1, img2) > Threshold$: Different Person

Figure: d(img1,img2) $\leq$ Threshold

Figure: d(img1,img2) $>$ Threshold

Naive Approach

compute distance
pixel per pixel

if less than threshold
then y=1

database image

input image

$$Considering\ image\ 1\ as\ matrix\ M^1\ and\ imge\ 2\ as\ M^2$$

$$\left(Both\ M^1\ and\ M^2\ are\ of\ dimension\ m \times n\right)$$

$$d(image1, image2) = \sum_{1 \leqslant i \leqslant m} \sum_{1 \leqslant j \leqslant n} \left| M^1_{i,j} - M^2_{i,j} \right|$$

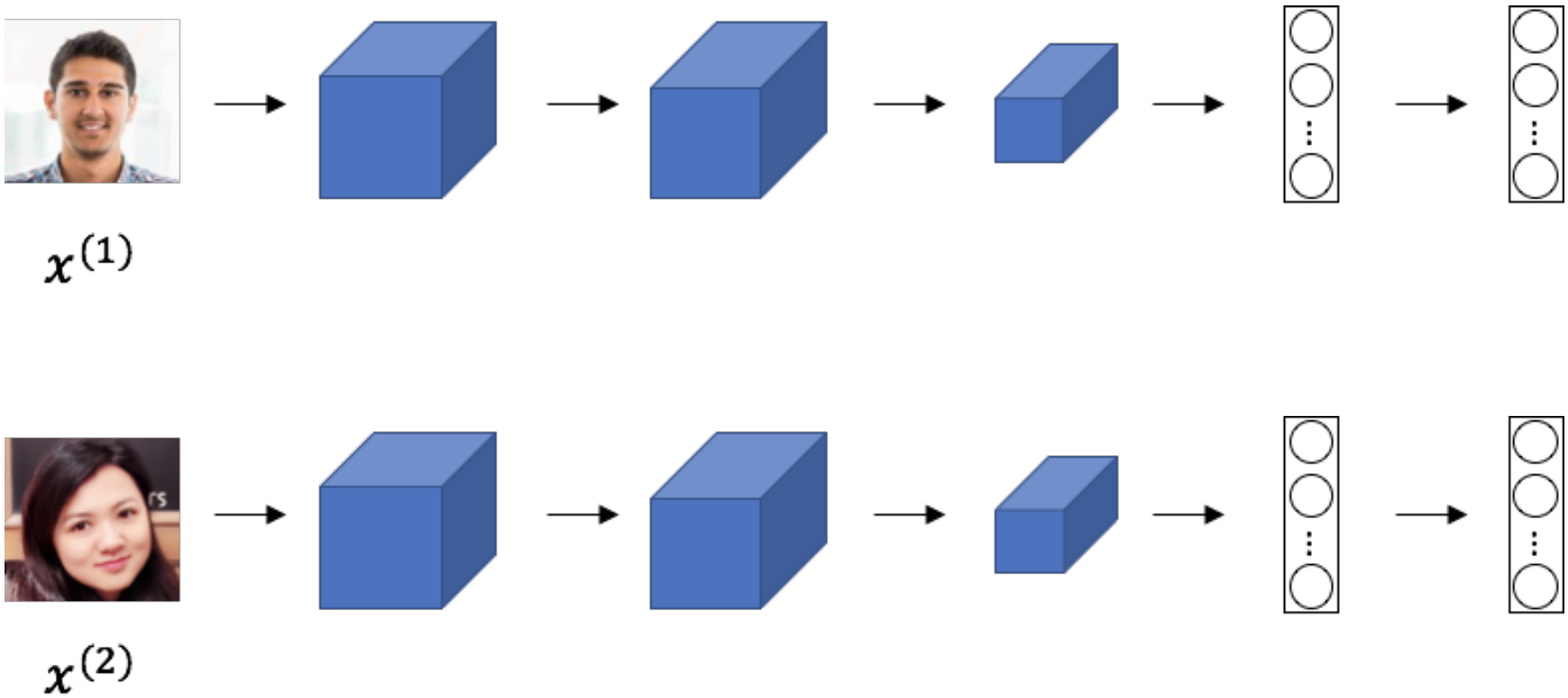
Shortcomings of this Similarity Function

Figure: Distance between these two photos will be **less**

Figure: Distance between these two photos will be **more**

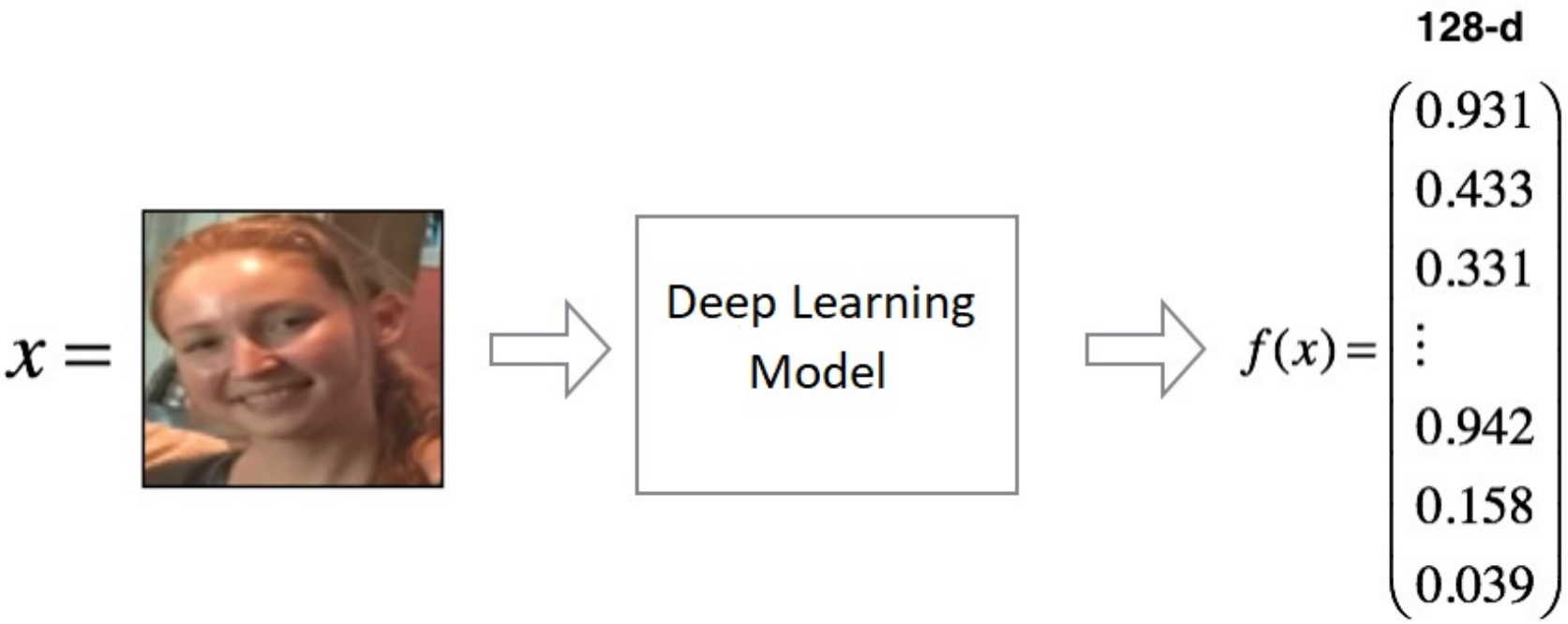
Should be the other way around

Better Approach : Siamese Network



- A Deep Learning Model
- Consists of many Convolutions layers and Dense layers
- Produces Encoding for Input Image

Encoding Images



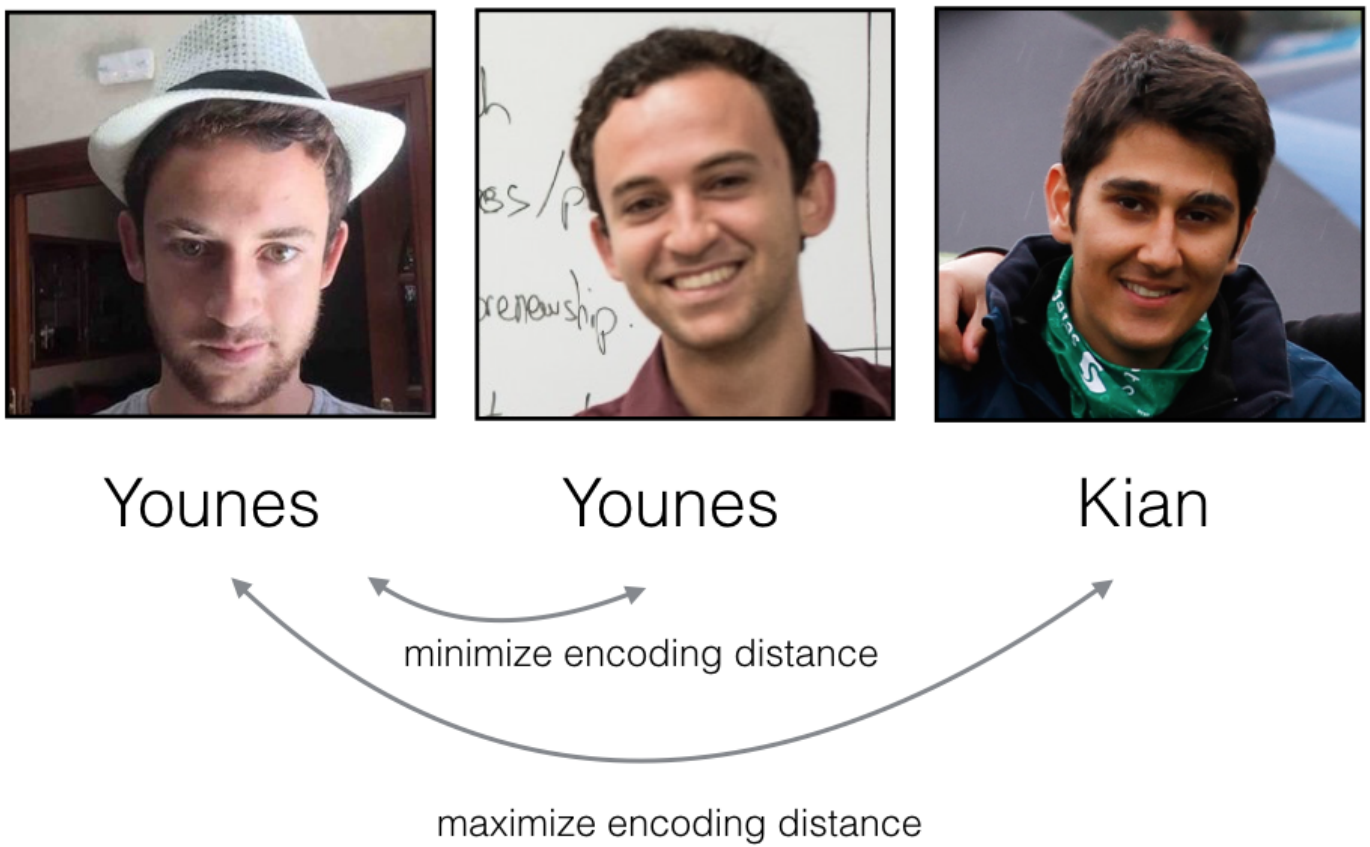
- Converts Image into 128-dimensional Vector
- $f(x)$: This Deep Learning Model

Training Our Model

- Parameters of NN(Neural Network) define our encoding function $f(x)$

Learn Parameters so that:

- if x^i, x^j are the same person, $||f(x^i) - f(x^j)||^2$ is small
- if x^i, x^j are the different person, $||f(x^i) - f(x^j)||^2$ is large



Learning Objective

Objective :

- $||f(A) - f(P)||^2 \leq ||f(A) - f(N)||^2$

Problem :

- $f(x) = 0$ satisfies this objective.

Solution (Improved Objective) :

- $||f(A) - f(N)||^2 - ||f(A) - f(P)||^2 \geq \alpha$
- α is a positive number(margin).

Loss Function

- $L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0)$
- $loss\ function,\ J = \sum_{1 \leqslant i \leqslant m} L\left(A^i, P^i, N^i\right)$

Choosing the triplets A,P,N

- Choose triplets that're "hard" to train on.
- In other words,
  $||f(A) - f(P)||^2 \approx ||f(A) - f(N)||^2$

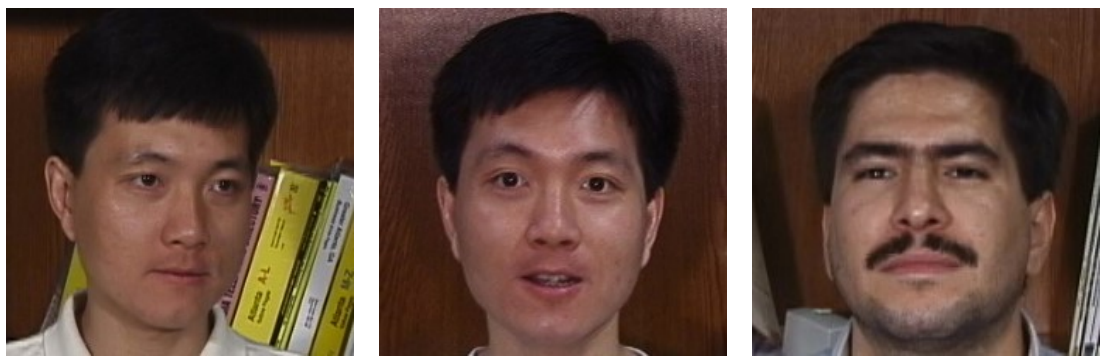


Figure: A training example for triple loss

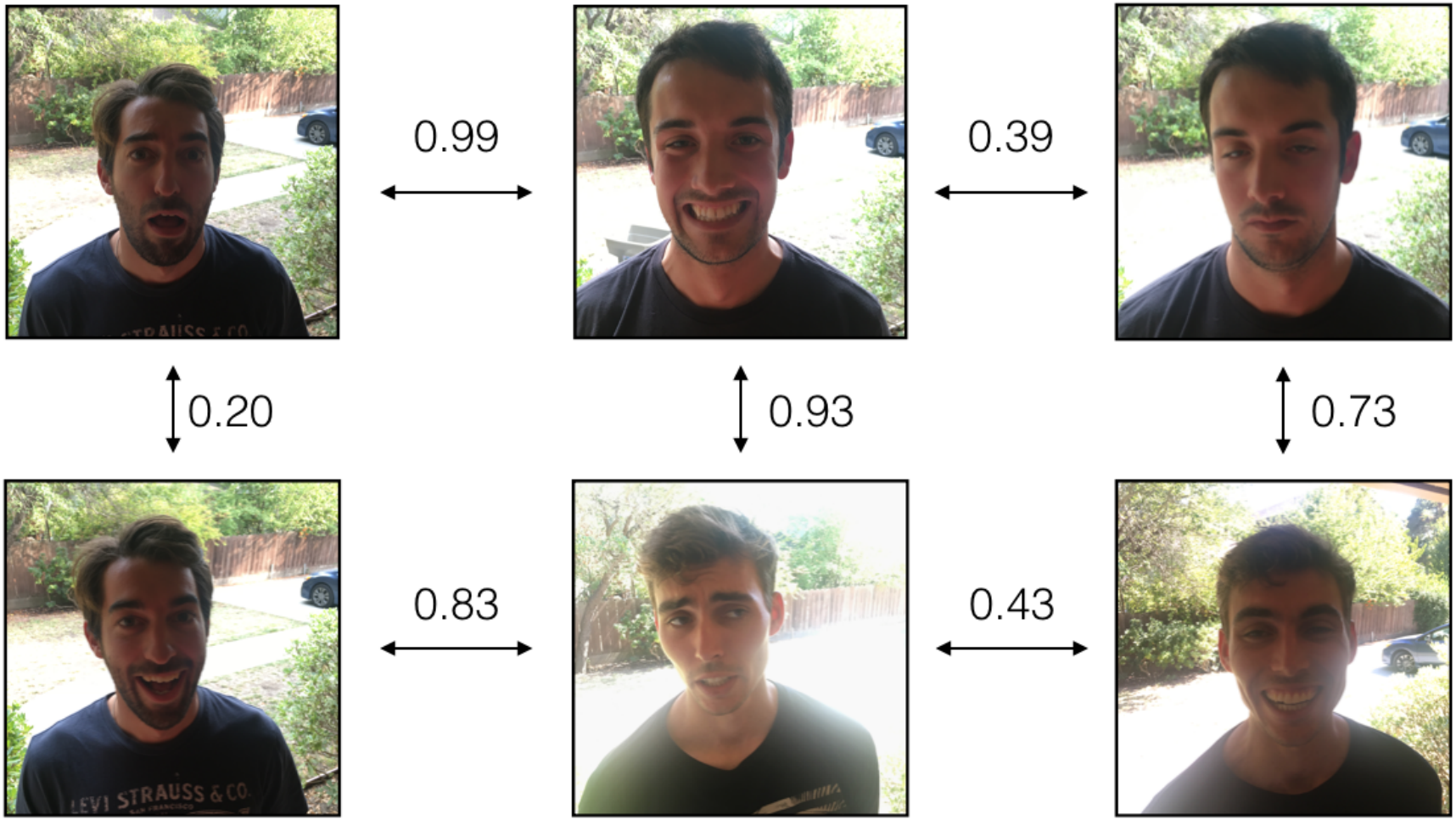
Property of Encoding After Training

Figure: Distance between $f(x^1)$ and $f(x^2)$ will be **more**

Figure: Distance between $f(x^1)$ and $f(x^2)$ will be **less**

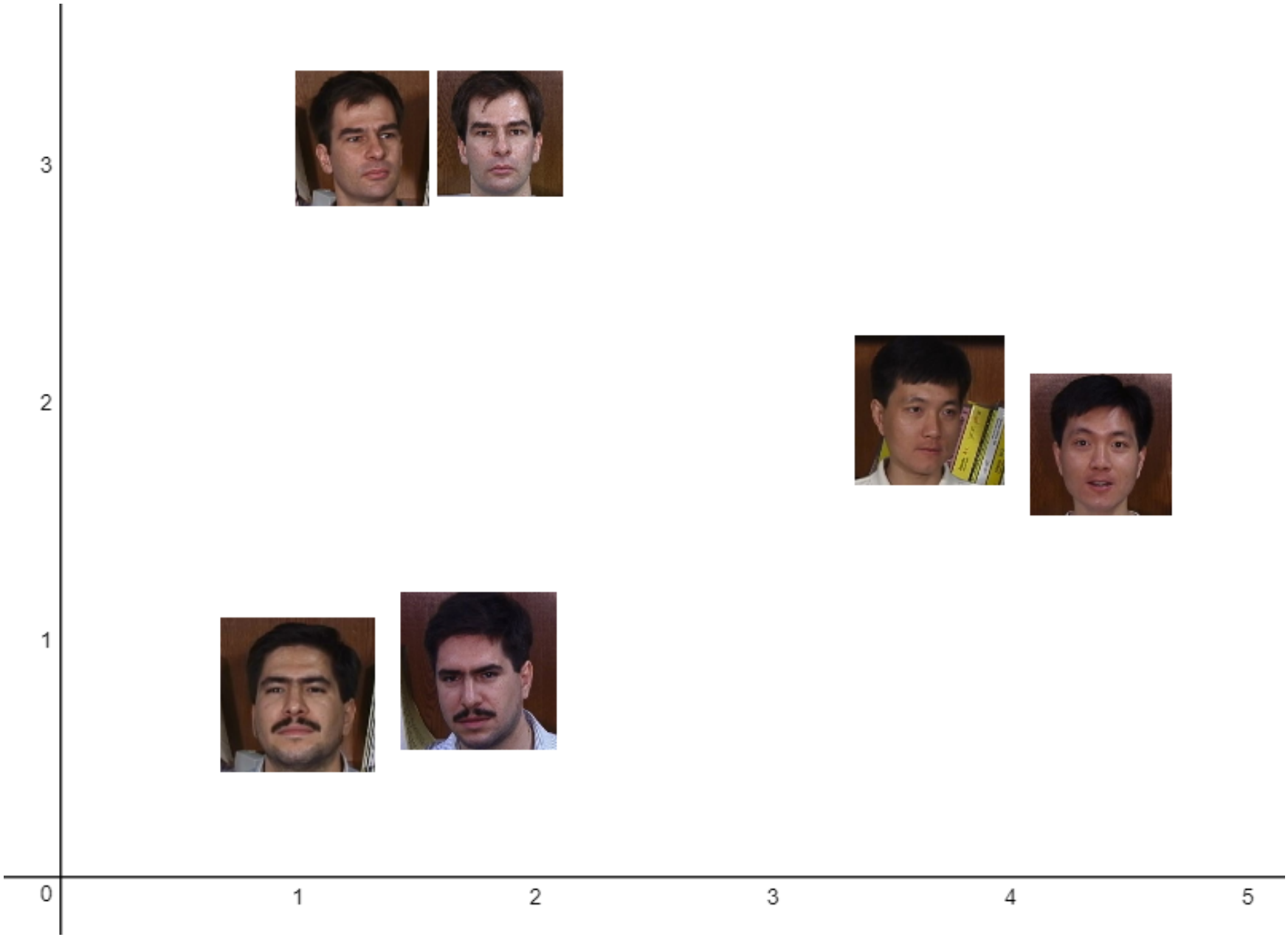
Encoding of images of the same person will be closer

Example of Encoding

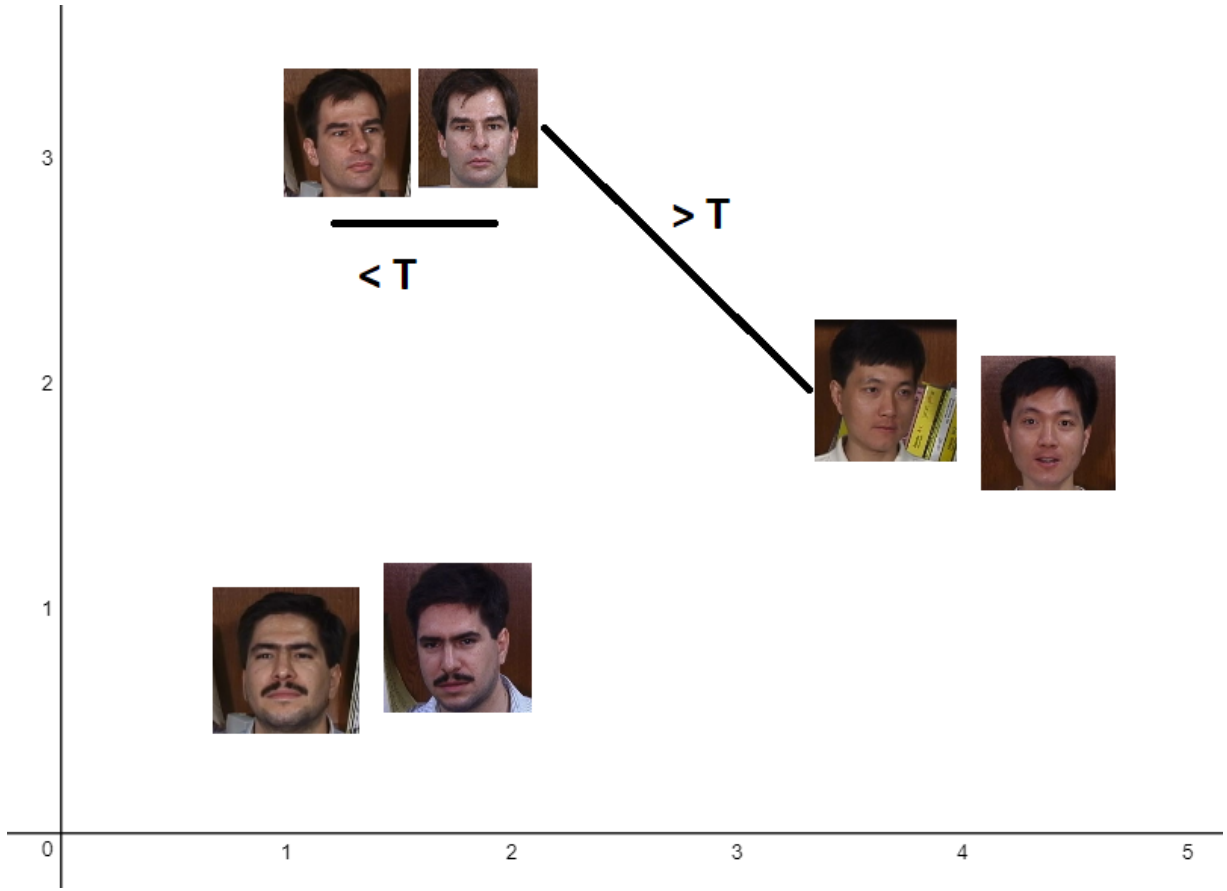


In this case, Setting threshold to 0.5 will verify all images correctly.

For Better Visualization, Assume our Encoding is 2D

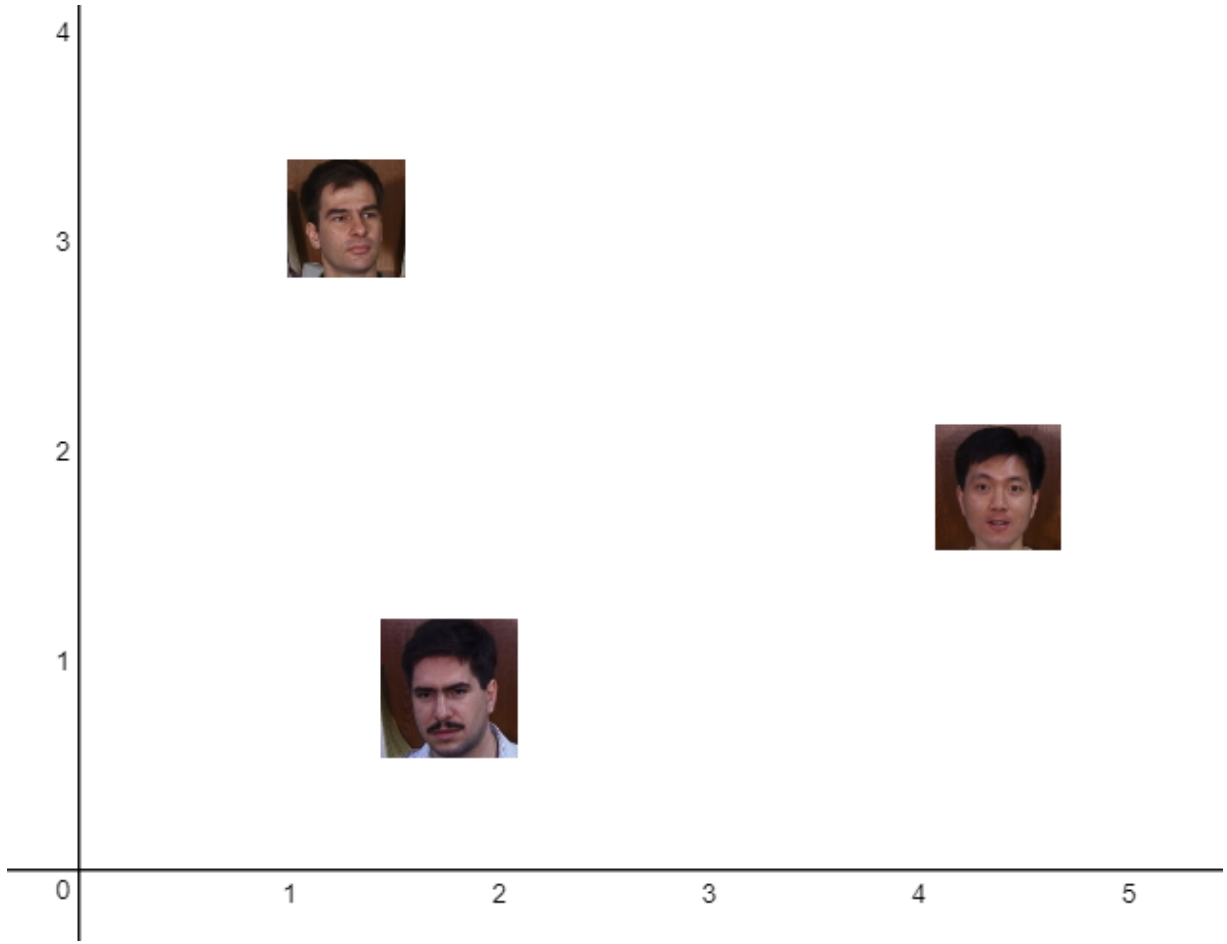


How to Verify Face



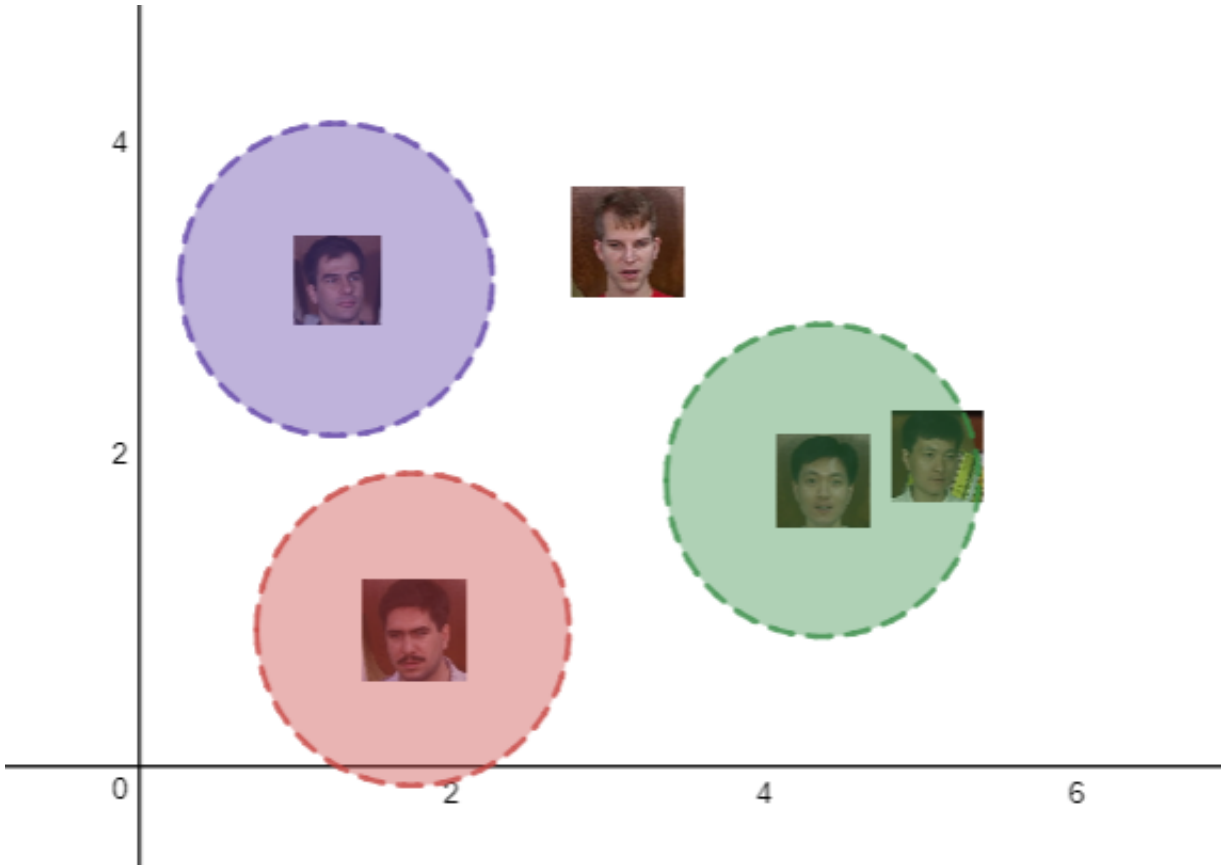
- If Distance between $f(x^1)$ and $f(x^2) \leq T(Threshold)$: Declare Same Person
- If Distance between $f(x^1)$ and $f(x^2) > T$: Declare Different Person

How to Recognize Face



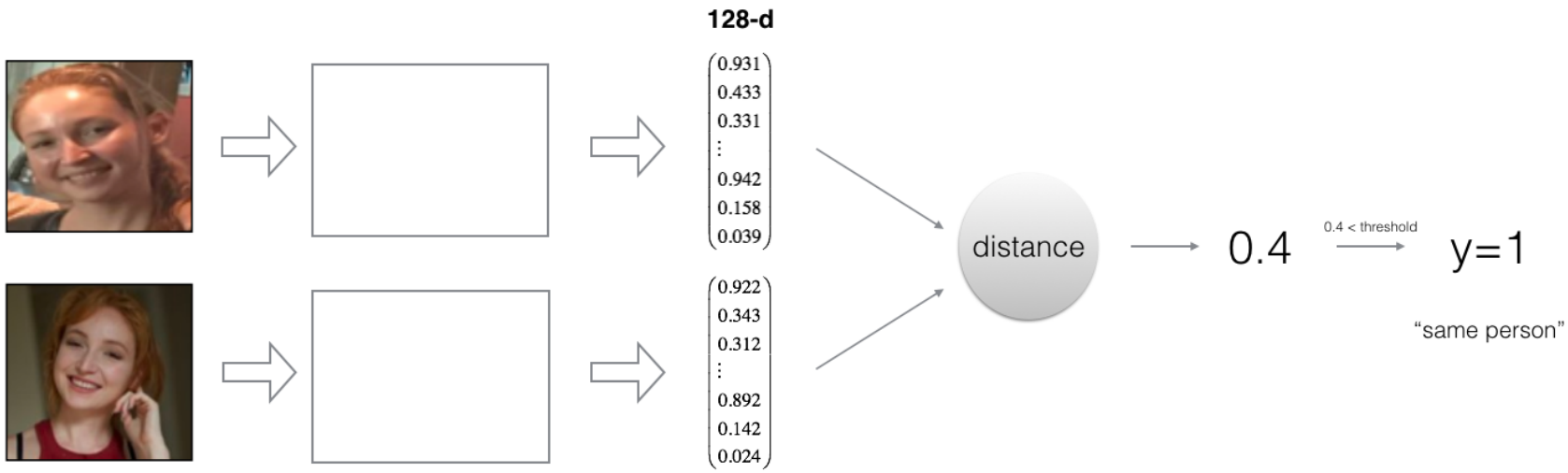
- Get encoding of training images and save them

How to Recognize Face



- Compare the encoding of new image to each encoding in our database
- If distance with any encoding is less than threshold : return ID of that encoding (else 'Not recognized')

When our Encoding is 128-Dimensional



- Our method will remain same irrespective of number of dimensions.
- Same things will be done for 128-d,355-d,etc.

Limitations and possible improvements

Limitations :

- For now face recognition is not that secure. It's not reliable.
- For training the machine about faces takes huge amount of space and time!!!

Improvements :

- By combining information from multiple cameras
- By using high quality images.

Thank you!

manika.khare.16002@iitgoa.ac.in
shivam.kumar.16001@iitgoa.ac.in
pankaj.katkar.16001@iitgoa.ac.in

References

- Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning
- Convolutional Neural Networks - Coursera