URL: https://github.com/serengil/deepface

A Python library (**deepface**): This is a popular open-source Python library named "deepface" that wraps various state-of-the-art face recognition models, including the original DeepFace developed by Facebook (if available) along with VGG-Face, FaceNet, OpenFace, ArcFace, Dlib, SFace, and GhostFaceNet. It provides a user-friendly interface to access and utilize these models for various facial recognition tasks like:

1. Face detection and recognition
2. Facial attribute analysis (age, gender, emotion, race)
3. Facial landmark detection
4. Face Anti Spoofing analysis (Given image is real or fake)

https://visagetechnologies.com/HTML5/latest/Samples/ShowcaseDemo/ShowcaseDemo.html

Research idea:

- Enhancing person recognition through multimodal technologies (thumb, face, iris, signature)

Application of CONVNet

Face Recognition

1. **What is face recognition?**

- **Face recognition** system identifies a person's face. It can work on both images or videos.
- **Liveness detection** within a video face recognition system prevents the network from identifying a face in an image. It can be learned by supervised deep learning using a dataset for live human and in-live human and sequence learning.
- Face verification vs. face recognition:
    - **Verification:**
        - Input: image, name/ID. (1 : 1)
        - Output: whether the input image is that of the claimed person.
        - "is this the claimed person?"
        - Face verification system received two intputs, image and ID/name.... system use ID/name of person and match save image corresponding to the input image.
        - Verification (ID, image):
            - matched the saved image corresponding to input ID
            - if match return true else false
    - **Recognition:**
        - Has a database of K persons
        - Get an input image
        - Output ID if the image is any of the K persons (or not recognized)
        - "who is this person?"
        - Recognition (image):
            - Matched the input image with the K saved image
            - If the person is matched return ID else Not matched message

Examples:

Face Verification (1:1 Comparison):

- **Unlocking your smartphone:** When you use Face ID or similar facial recognition features to unlock your phone, your face is compared to a stored template (your selfie) on the device. This confirms you are the authorized user (verification).
- **Secure online payments:** Some banks or online retailers might use facial verification during online transactions. You might capture a live selfie to verify your identity matches the one associated with your account (verification).
- Secure Access Control **/Building access control:** In some office buildings, facial verification systems might be used to grant access. Your face is compared to a pre-approved database of authorized personnel (verification).
- In this scenario, the input consists of a face image and an ID. The verification system uses the ID to locate the stored face image corresponding to it and then matches the input image with the stored image to grant or deny access.

Face Recognition (1:N Comparison):

- **Attendance Systems**:
  - **Example**: Educational institutions using facial recognition to automatically record student attendance.

- **Surveillance Systems**:

  - **Example**: Law enforcement agencies using facial recognition technology in CCTV footage to identify and track individuals in public places.

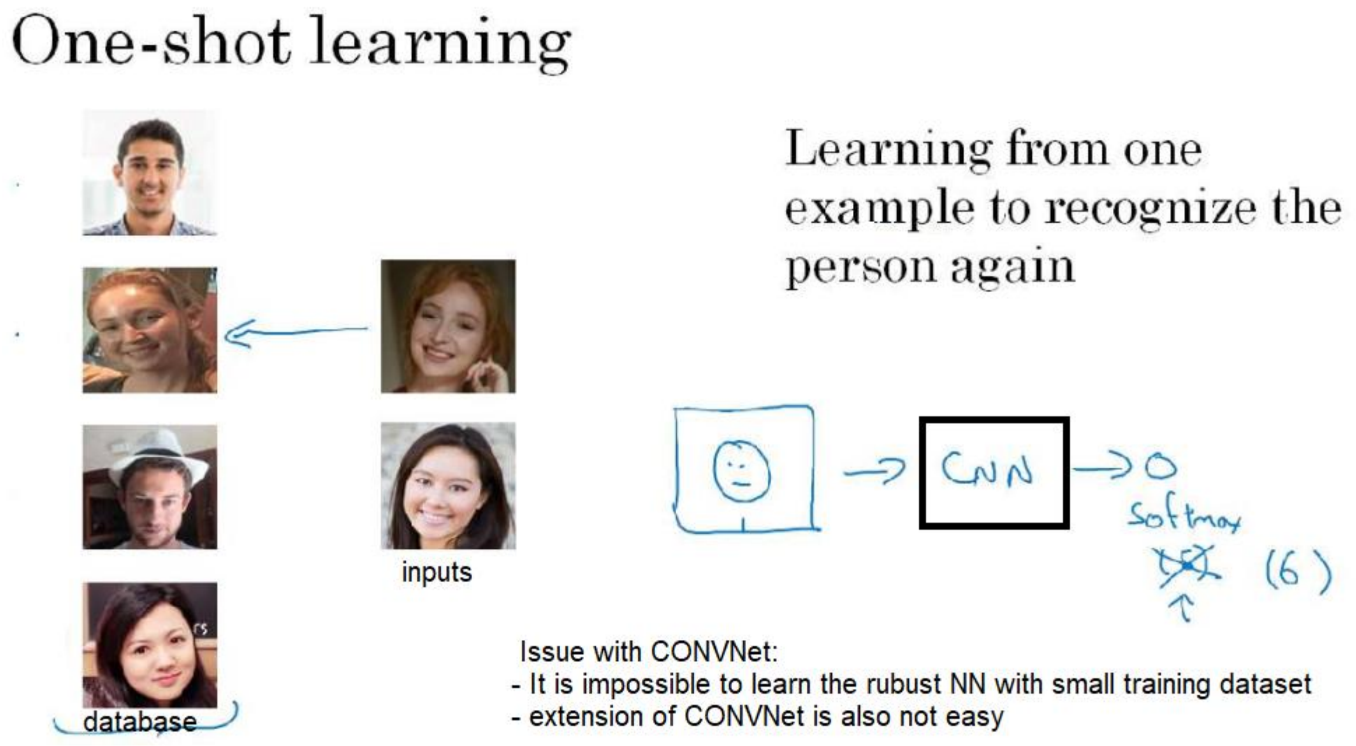
- **Social Media Tagging**:

  - **Example**: Facebook's automatic photo tagging feature, which recognizes and suggests tags for people in uploaded photos.

Common thing is the matching between the embeddings of input and stored faces

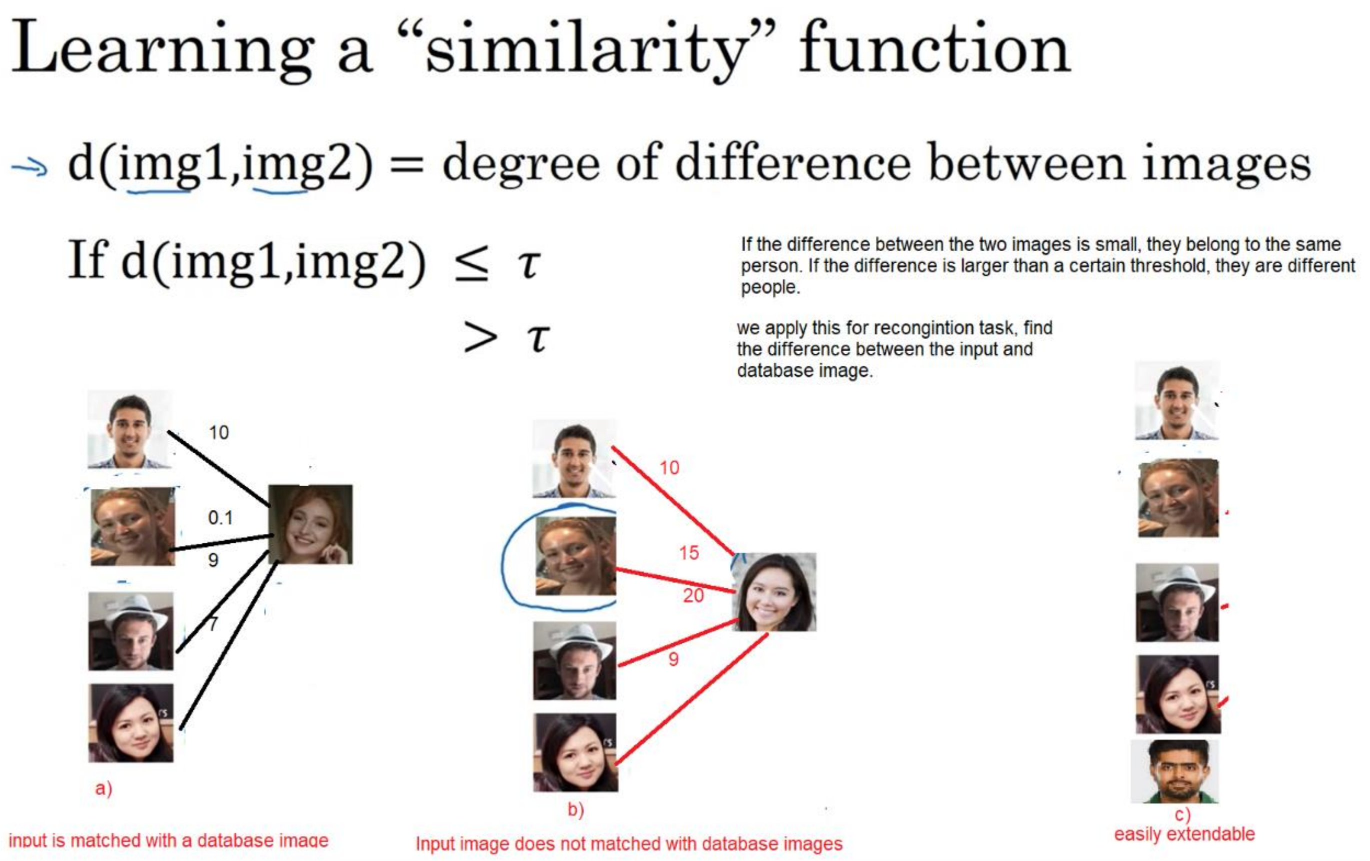
2. One Shot Learning problem:

a. Multiple class classification of fac recognition



Issue with CONVNet:
- It is impossible to learn the rubust NN with small training dataset
- extension of CONVNet is also not easy

b. Solution of one-shot learning: learn similarity function



Learning a “similarity” function

→ $d(\text{img1},\text{img2})$ = degree of difference between images

If $d(\text{img1},\text{img2}) \leq \tau$

$> \tau$

If the difference between the two images is small, they belong to the same person. If the difference is larger than a certain threshold, they are different people.

we apply this for recongintion task, find the difference between the input and database image.

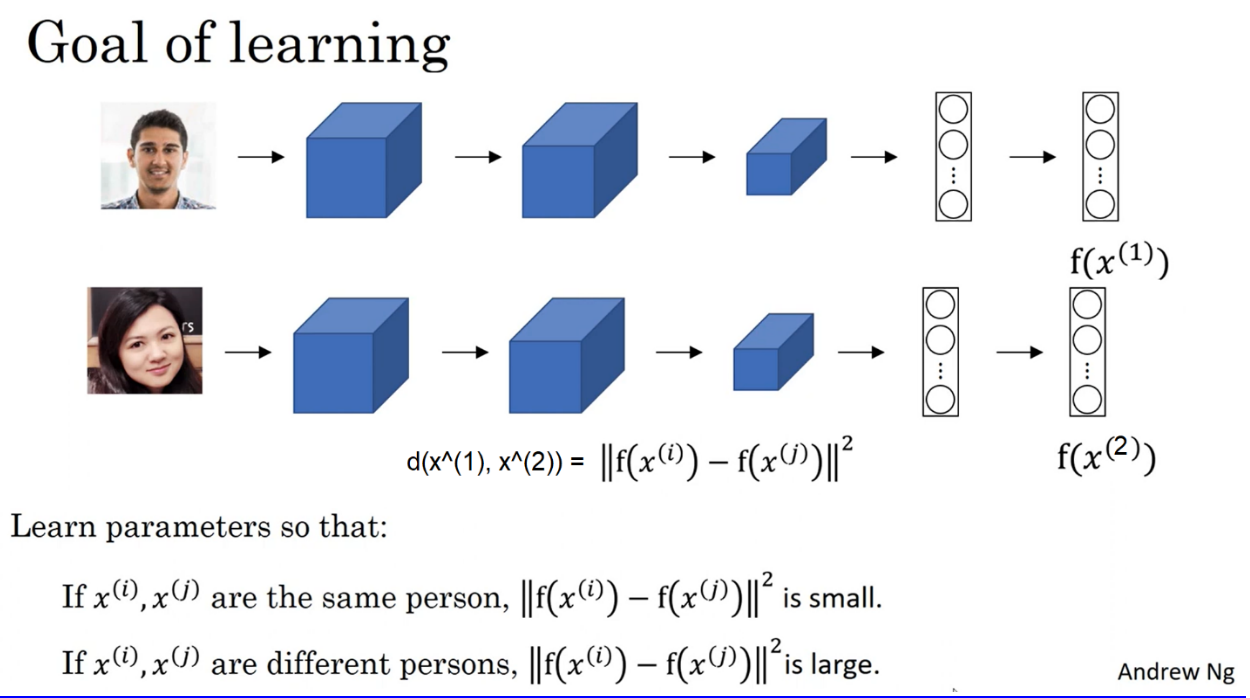
10
0.1
9
7
a)
input is matched with a database image
10
15
20
9
b)
Input image does not matched with database images
c)
easily extendable

- One of the face recognition challenges is to solve one shot learning problem.

I. **Training time:** The model learns from a large dataset of generic faces.
II. **Recognition time:** The model uses one image per new person to perform recognition.
III. It's called **one-shot learning** because the recognition system only needs one example to correctly identify new individuals, leveraging the pretrained model.

4. Siamese Network

- [Taigman et. al., 2014. DeepFace closing the gap to human level performance]



Goal of learning

d(x^(1), x^(2)) = $\|f(x^{(i)}) - f(x^{(j)})\|^2$

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 different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

Andrew Ng

- We implement the **similarity function** using a type of NNs called Siamease Network in which we can pass multiple inputs to the two or more networks with the same architecture and parameters.

5. Pyorch implementation … a sample code:

```
import torch
from torch import nn
from torch.nn import functional as F

class SiameseNetwork(nn.Module):
  def __init__(self):
    super(SiameseNetwork, self).__init__()
    # Define the shared network architecture
    self.cnn = nn.Sequential(
        nn.Conv2d(1, 8, kernel_size=4, stride=2),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=2),
        nn.Conv2d(8, 16, kernel_size=4, stride=2),
        nn.ReLU(inplace=True)
    )
    self.fc1 = nn.Linear(16 * 7 * 7, 128)  # Assuming input image size is 
28x28

  def forward(self, x1, x2):
    # Pass each input through the shared network
    output1 = self.cnn(x1)
    output1 = output1.view(-1, 16 * 7 * 7)
    output1 = F.relu(self.fc1(output1))

    output2 = self.cnn(x2)
    output2 = output2.view(-1, 16 * 7 * 7)
    output2 = F.relu(self.fc1(output2))

    # Calculate the distance between the embeddings (Euclidean distance)
    euclidean_distance = F.pairwise_distance(output1, output2)
    return euclidean_distance
// NOTE: during forward pass a computation graph of siamese/ twin network 
is build and backward pass compute gradient on computation graph of the 
siamese/ twin network
```

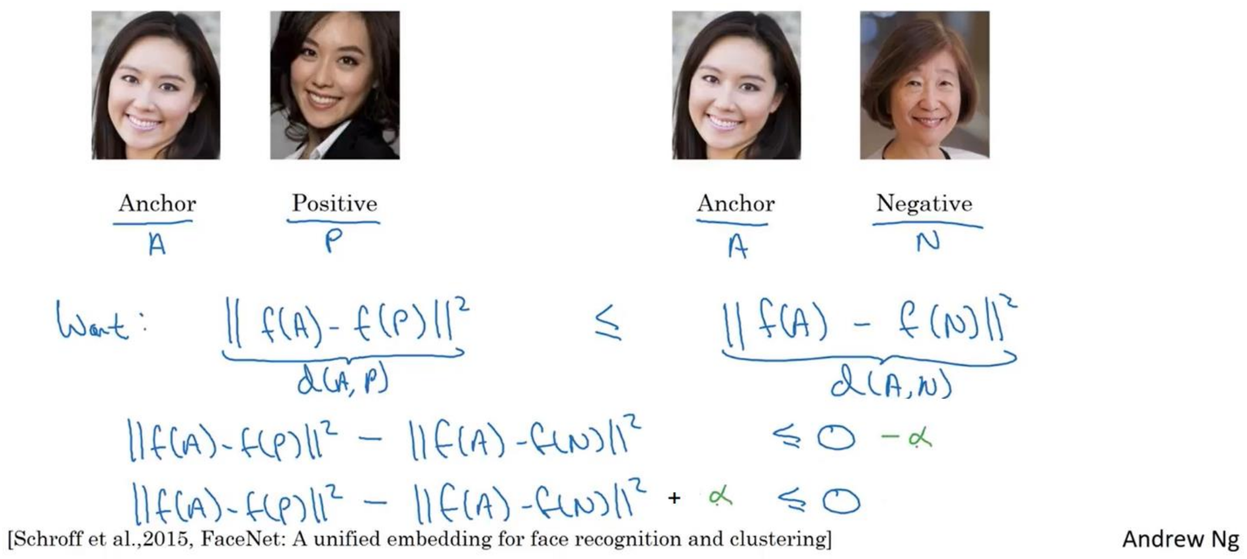
- Siamese network architecture are as the following:
    - We make 2 identical conv nets which encodes an input image into a vector. In the above image the vector shape is (128)

Example:○

Distance between $[1.0, 2.0]$ and $[2.0, 3.0]$ is:

$$\sqrt{(1-2)^2 + (2-3)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.4142$$

6. Triplet loss:



Anchor A Positive P Anchor A Negative N

Want: $\underbrace{||f(A)-f(P)||^2}_{d(A,P)} \leq \underbrace{||f(A)-f(N)||^2}_{d(A,N)}$

$$||f(A)-f(P)||^2 - ||f(A)-f(N)||^2 \leq 0 - \alpha$$

$$||f(A)-f(P)||^2 - ||f(A)-f(N)||^2 + \alpha \leq 0$$

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering] Andrew Ng

Lets,

$$\boldsymbol{d(A,P) = ||f(A) - f(P)||^2}$$
$$\boldsymbol{d(A,N) = ||f(A) - f(N)||^2}$$

$$\boldsymbol{d(A,P) + \alpha \leq d(A - N)}$$

$$\boldsymbol{d(A,P) + \alpha - d(A - N) \leq 0}$$

In this case we can easily satisfied this equation, To avoid the NN to avoid this trivial solution we add – alpha(margin) term. this alpha prevent the NN to produce the trivial solution

Example:

if alpha = 0.2, d(A,P) = 0.5, and d(A,N) = 0.51,

0.5 + 0.2 < = 0.51

0.7 is not lesser or 0.51

It clearly does not meet the specified condition (Triplet loss), resulting in a higher loss. Which enforce the network to learn the parameter that make the difference between A and P is smaller (**lesser than 0.5)** while the difference between A and N is larger (**larger than 0.51).**

8. Triplet loss funtion:

We need to minimize the following loss funcion:

For single trainng examples,

$$\boldsymbol{L(A,P,N) = max\,(d(A,P) + \alpha - d(A-N)\,,0)}$$

For m training examples,

$$\mathbf{J} = \sum_{i=0}^{m} \boldsymbol{L(A^{(i)}, P^{(i)}, N^{(i)})}$$

Case1:

- if alpha = 0.2, d(A,P) = 0.5, and d(A,N) = 0.51,
- L (A, P, N) = max (0.5+0.2 – 0.51, 0) = max(0.19,0) = 0.19,
  - which indicates a significant error. This means we need to adjust the neural network parameters to reduce the loss.

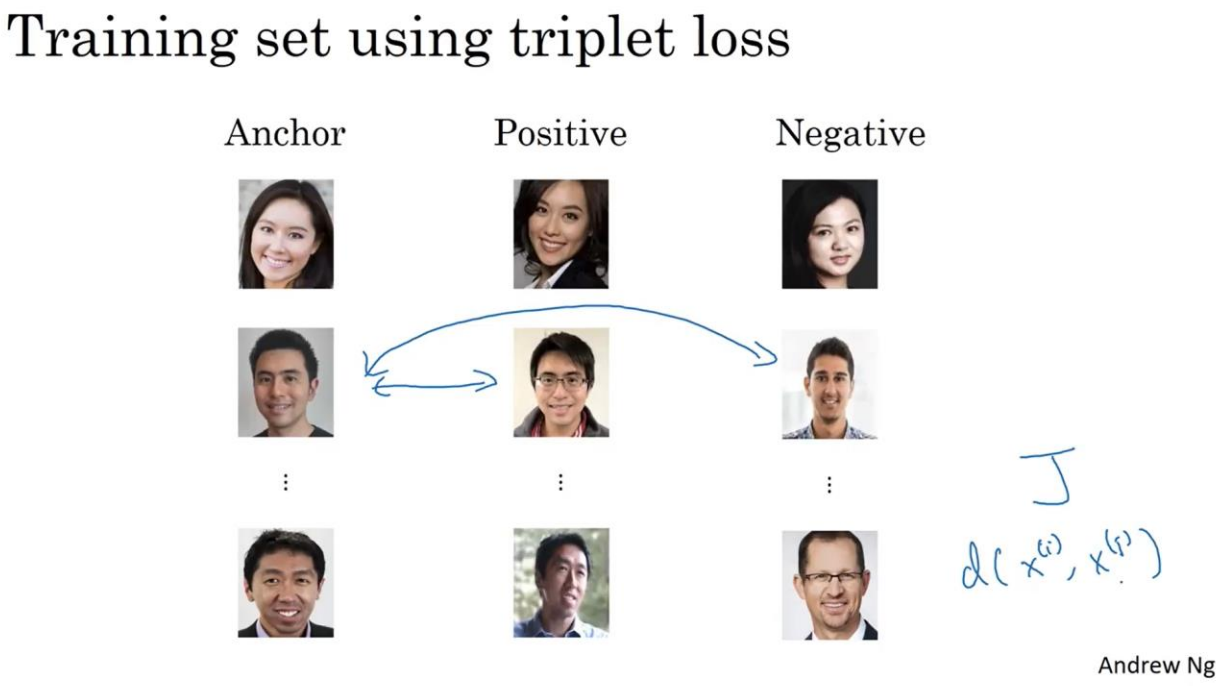
Case2:

- if alph = 0.2, d(A,P) = 0.5, and d(A,N) = 0.9,
- L (A, P, N) = max (0.5+0.2 – 0.9, 0) = max(-0.2, 0) = 0,
  - Network works well

8. Dataset size:

- For training single example/person is not work, so if we have the 1000 persons than 10 images per person is required (10K) to trained the network

9. Selection of triplets A, P, N: to ensure effective learning



If we select random triplets, the network may not learn effectively, as the triplet loss could be zero (i.e., **d(A, P) + alpha + d(A, N)**) . This would mean the Siamese network is not being challenged and therefore learns nothing. To ensure effective learning, we need to choose hard examples by selecting these difficult triplets, we can ensure that the loss is a positive number, which drives the network to learn the correct parameters.Parameters that make d(A,P) smaller and d(AnN) larger in order to minimzed the loss.

10. Process of One-Shot Learning in Face Recognition:

I. Training Phase:

- A large face dataset (e.g., CASIA-WebFace, VGGFace2, or MS-Celeb-1M) is used to train a model like FaceNet, ArcFace, or similar networks.

- The model learns to map face images to a high-dimensional **embedding space**, where the embeddings (vector representations) of similar faces are close, and embeddings of different faces are far apart.

- Loss functions like **triplet loss** or **contrastive loss** are used during training to optimize the embeddings.

II. Database Creation (Enrollment):

- For every known person, provide **one image** (or a few images if desired) of the person to the pretrained model.

- Extract the **embedding** of this image using the pretrained model.

- Save these embeddings in a database along with their corresponding identity (e.g., person A → [embedding_A]).

III. Testing (Recognition):

- When a new input image is presented, pass it through the same pretrained model to generate its **embedding**.

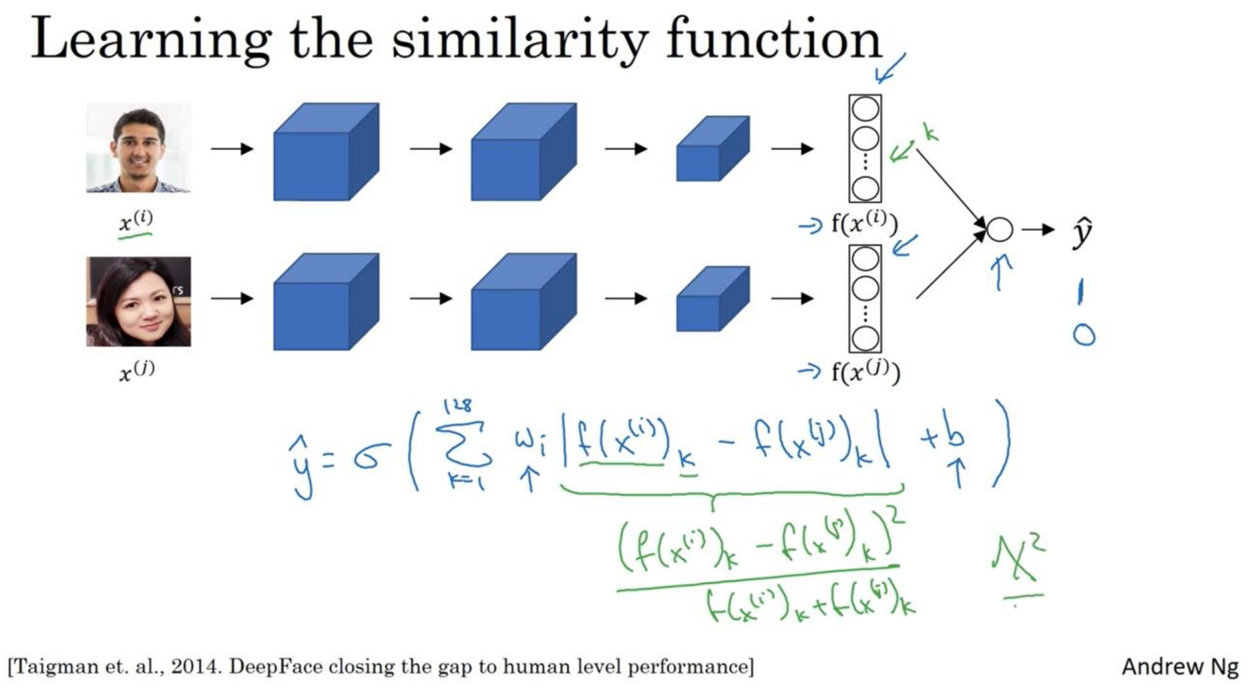
- Compare this new embedding with all saved embeddings in the database using a **similarity metric** (e.g., cosine similarity, Euclidean distance).

- Identify the person with the **closest matching embedding** (distance below a predefined threshold).

Advantages of This Approach:

- Scalable: Adding new people requires only saving their embeddings, without retraining the model.
- Efficient: You only need one or very few images per person during enrollment.
- Memory-Efficient: Only embeddings (vectors, e.g., 128-dimensional) are stored, not the raw images.

11. Face recognition as binary classification problem

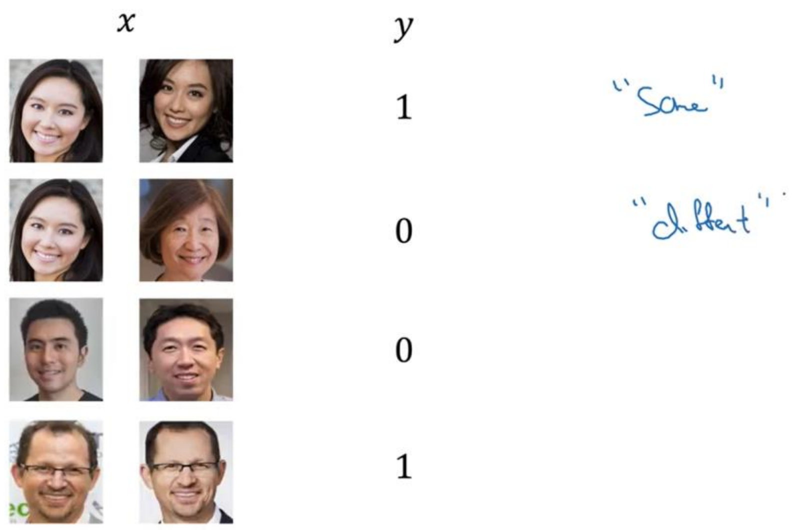


Learning the similarity function
$x^{(i)}$
$x^{(j)}$
$f(x^{(i)})$
$f(x^{(j)})$
$\hat{y}$
1
0
$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$
$\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$
χ^2
[Taigman et. al., 2014. DeepFace closing the gap to human level performance]
Andrew Ng

- Find the L1 loss between the 128 dimensional vectors and attached it with the single neuran with sigmoid function, binary cross entropy function is used for training of siamese network
- If the pair image of same person then y is 1, for pair of input images belongs to two different persons then y is 0.

12. Dataset preparation for DeepFace

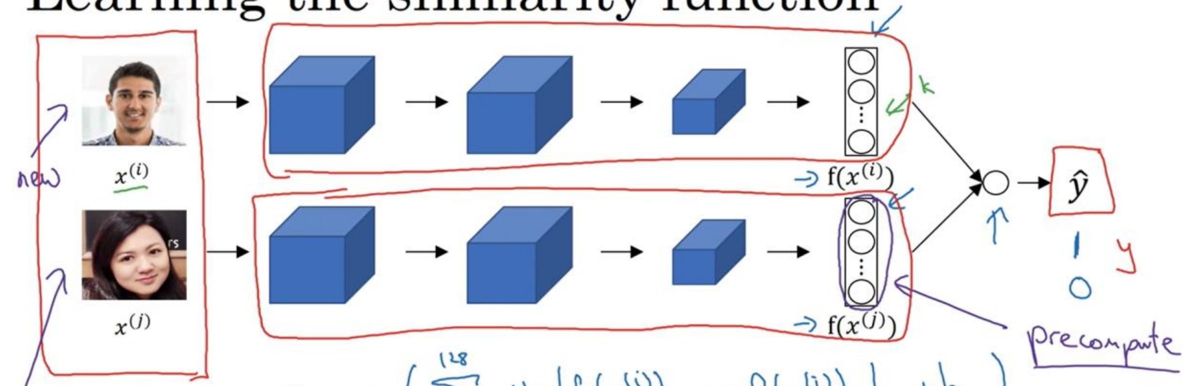
Face verification supervised learning



[Taigman et. al., 2014. DeepFace closing the gap to human level performance] Andrew Ng

13. Computation saving by computing encoding earlier

Learning the similarity function



$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \underbrace{\left| f(x^{(i)})_k - f(x^{(j)})_k \right|}_{\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}} + b \right)$$

χ^2

[Taigman et. al., 2014. DeepFace closing the gap to human level performance] Andrew Ng

14. Reason to trained or finetuned the model:

Training a face recognition model like FaceNet on your specific dataset instead of using a pre-trained model trained on a large, generic dataset of faces is useful in several scenarios:

1. Domain-Specific Applications

- If your face recognition application involves a very specific domain (e.g., medical images, facial scans under specific conditions like infrared or thermal imaging), the characteristics of the faces may differ significantly from the generic datasets used to train models like FaceNet.
- In such cases, training on your own dataset ensures the model learns the specific features relevant to your application.

2. Limited Set of Known Individuals

- If the dataset consists of a relatively small number of individuals (e.g., employees of a company or members of an organization), training the model on that specific dataset allows it to optimize for better recognition of those individuals.

3. Different Demographic Distribution

- If the demographics of the individuals in your dataset (e.g., age, ethnicity, or cultural factors) are very different from those in the large generic datasets, the pre-trained model might not generalize well. Fine-tuning or training on your dataset can help improve performance for your specific demographic group.

4. Customized Feature Embedding

- If you require embeddings that are fine-tuned to align with your system's requirements, such as specific clustering or comparison tasks, training or fine-tuning the model on your data ensures that the embeddings are tailored to the application.

5. Unique Conditions

- If the images in your dataset were captured under unique conditions (e.g., specific lighting, background, camera angles, or resolutions) that deviate from those in the generic datasets, the pre-trained model may not perform well. Training on your dataset helps adapt the model to those conditions.

6. Privacy Concerns

- For applications where privacy or data security is a concern, you may want to avoid relying on generic pre-trained models and instead train a model from scratch or fine-tune it on your own secure dataset.

7. Fine-Grained Recognition

- If your application involves fine-grained face recognition where subtle differences matter (e.g., recognizing twins or individuals with very similar facial features), training on your dataset ensures the model captures these distinctions.