Specialist Diploma in Artificial Intelligence

AI Applications with Deep Learning

Build Your Own Face Recognition Model

Sub topics

- Facial Tasks
- How It Works
- Facial Detection Algorithms
- Face Embedding Algorithms
- Classification Algorithms
- Similarity Measures
- Putting It Altogether

Facial Recognition

FACIAL RECOGNITION TASKS

Tasks

- Facial tasks:
  - **Facial Verification**
    Given an image of two faces, tell if the two faces belong to the same person. Example use cases:
    - Airport security check using passport and a photograph of your face
    - Logging into bank ATM terminals with card + face as a second factor. https://www.bbc.com/news/business-54266602

Tasks

- Facial tasks:
  - **Facial Recognition**
    Given an image of a face, identify the person. Example use cases:
    - Scanning photographs of people in a train station and identifying who they are
    - Scanning photographs in photo gallery (in mobile phone) to tag the people in the photo

China Tech: 5 ways China is using Facial Recognition

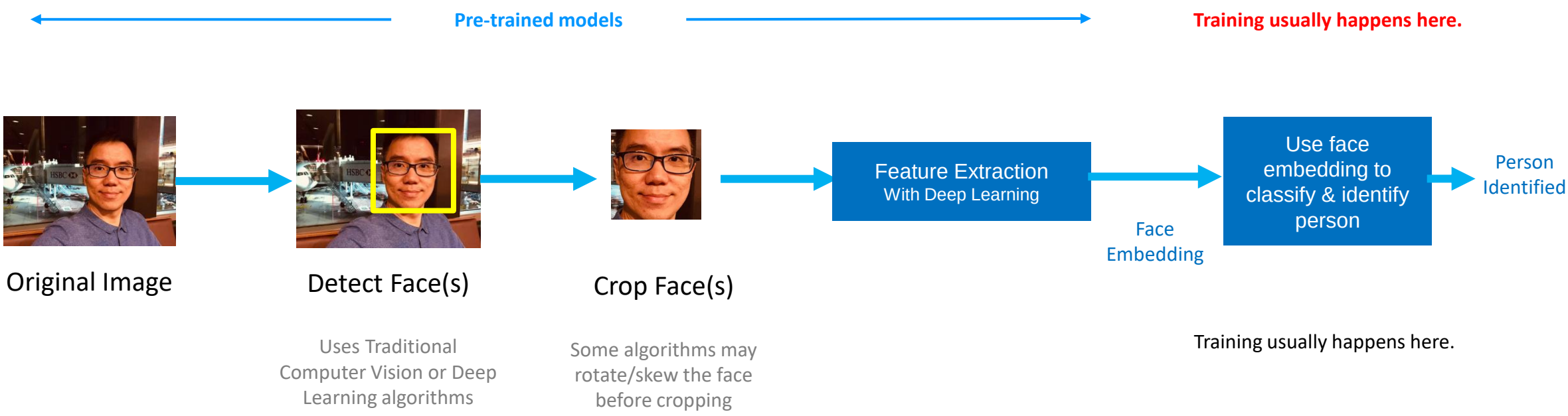
https://www.youtube.com/watch?v=Ib5Nv-l2Z4g

Facial Recognition

HOW IT WORKS

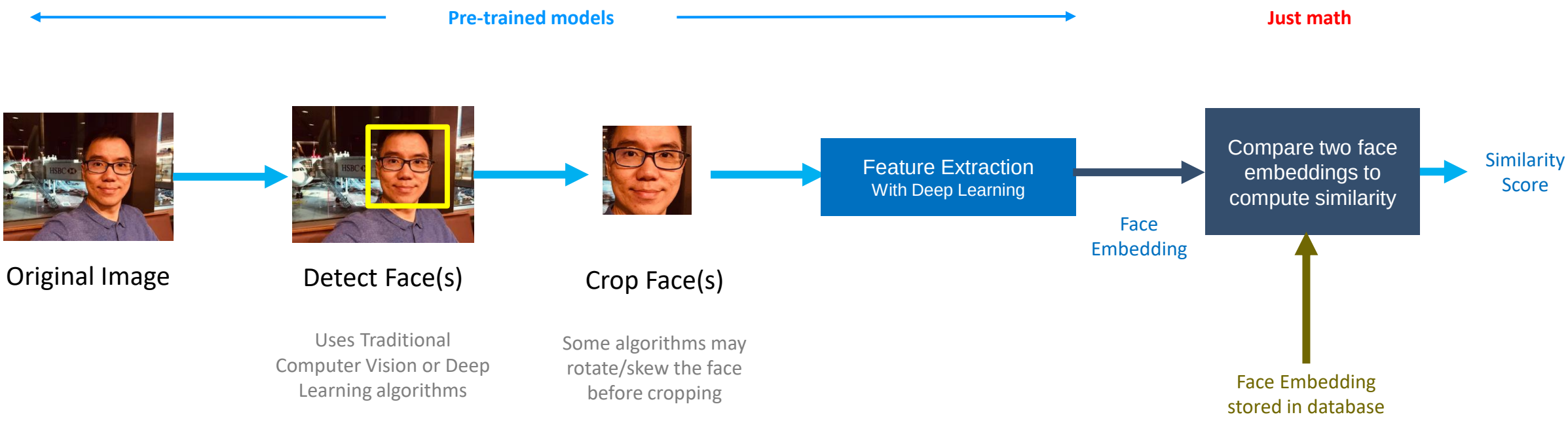
How it Works – Facial Recognition

- A typical facial recognition works in the following way:



How it Works – Facial Verification

- A typical facial recognition works in the following way:



Face Embedding

Cropped Face

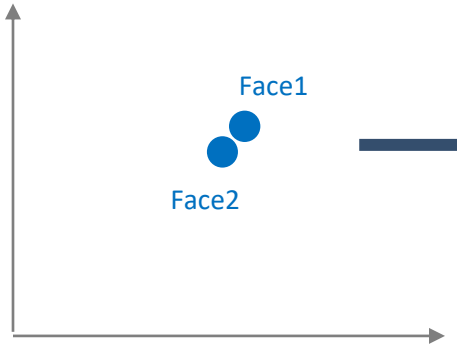
Feature Extraction
With Deep Learning

Face Embedding

Reference Image

Feature Extraction
With Deep Learning

Face Embedding stored in database



Compare two face embeddings to compute similarity

Use one of these determine how close they are:

a. **Euclidean Distance** (smaller is better)
b. **Cosine Similarity** (larger is better)

Face Embedding

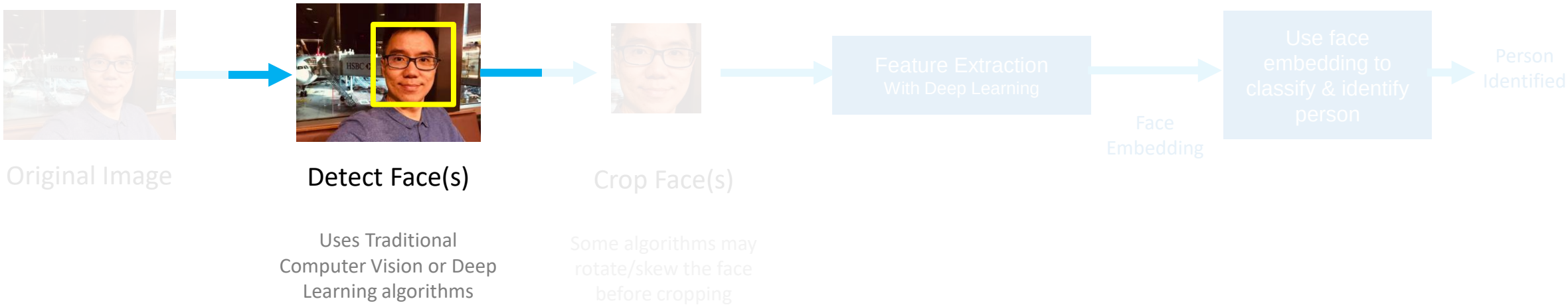
- Like word embeddings, a face embedding is a n-dimensional vector representation of a person's face in an image.

- Imagine the vector as a "point" in n-dimensions so that:
  - 2 photographs of the same person will produce two "points" that are very "near" in the n-dimensional space
  - 2 photographs of a different person will produce two "points" that are very "far" in the n-dimensional space

Facial Recognition

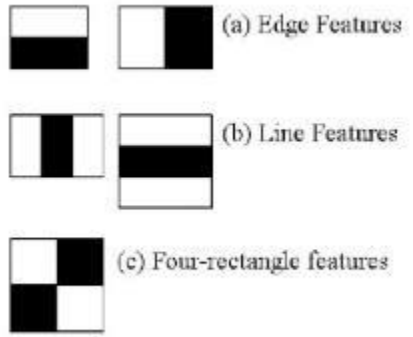
FACE DETECTION ALGORITHMS

Facial Detection



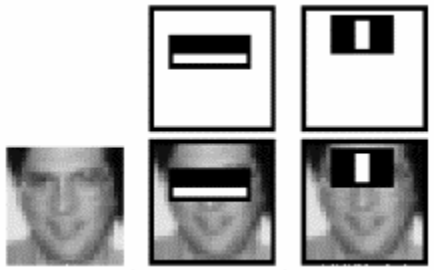
Haar Cascades

- Haar Cascades is an effective object detection ML algorithm based dark / light areas of an image.



- Haar features are like convolutional filters (light area is -1, dark area is 1) multiplied onto areas of an image

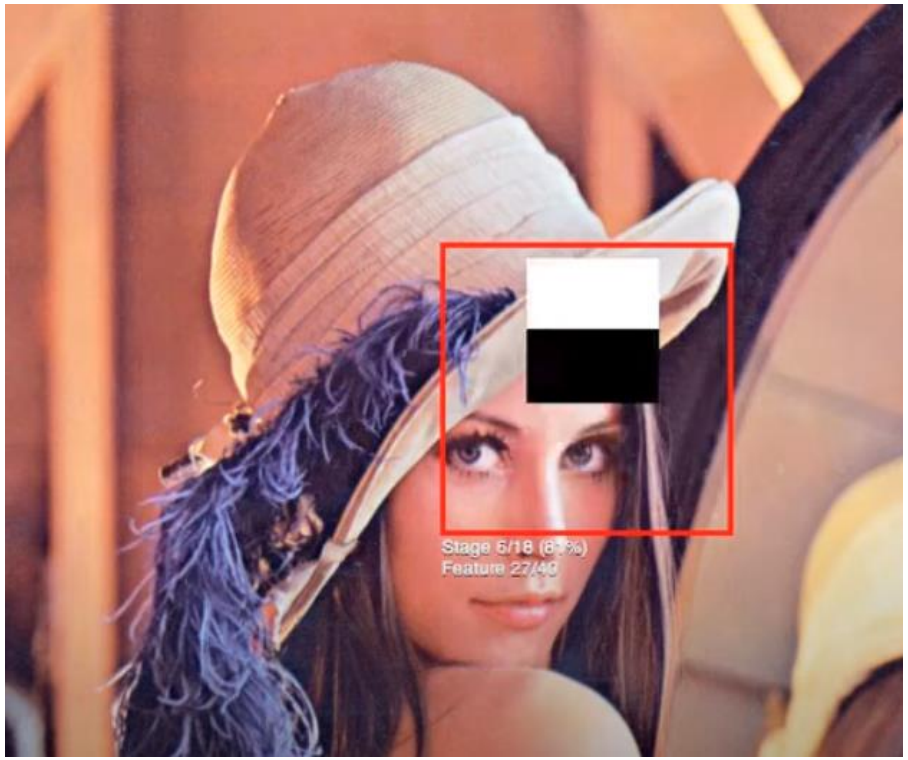
- Uses a cascade of Haar features for classification



- Fastest, least accurate

https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

Visualizing Haar Cascades



https://www.youtube.com/watch?v=hPCTwxF0qf4

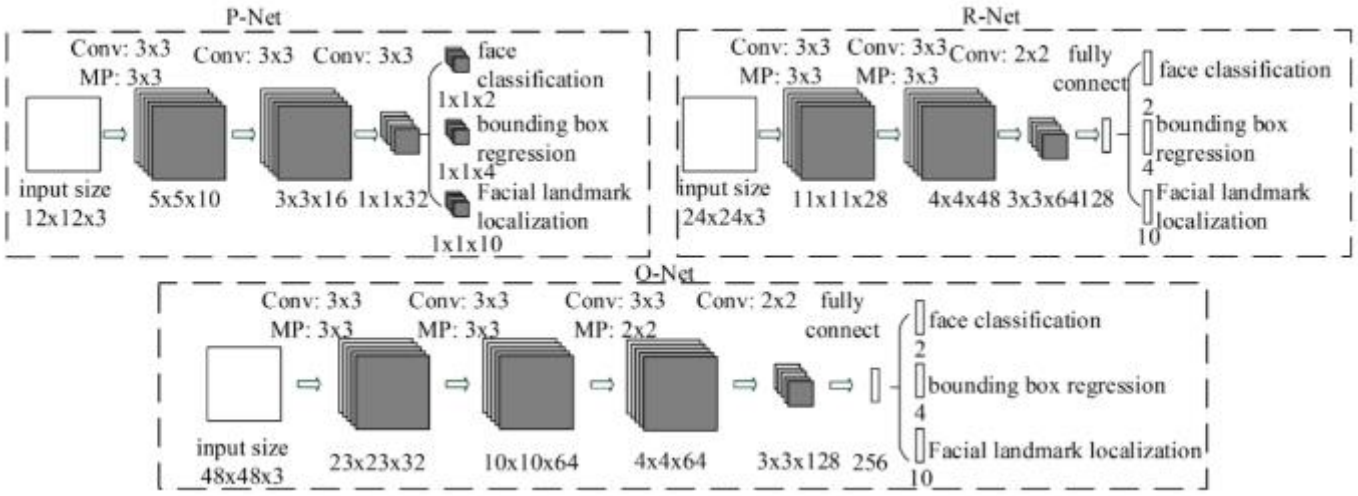
DLib

- Uses Histogram of Oriented Gradients (HoG).
- Divides the image into small square boxes and computes the histogram of edges in different directions.
- Use SVM to classify face / not face.

https://sklin93.github.io/hog.html

MTCNN

- Multi-Task Cascading Convolution Networks (state-of-the-art)

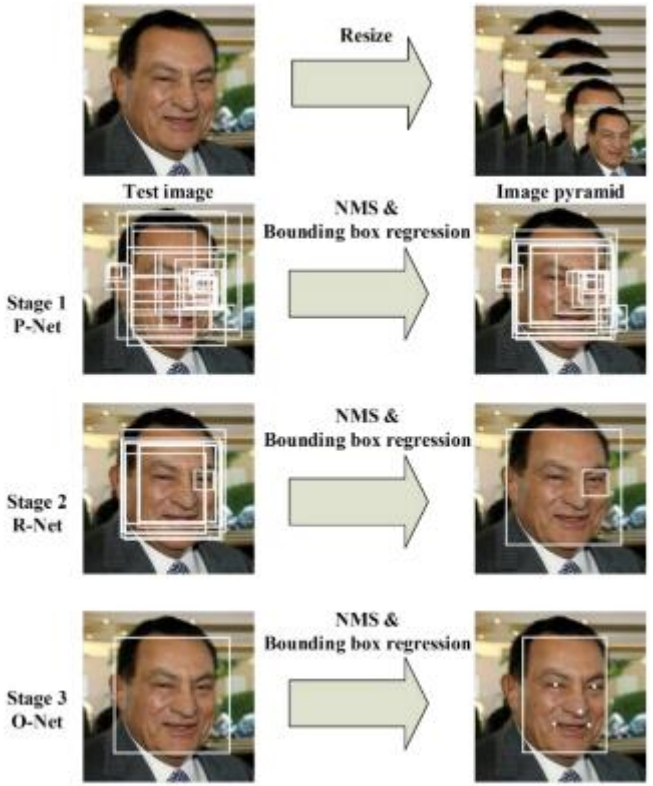


https://arxiv.org/abs/1604.02878
https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff

MTCNN

- Utilizes 3-stage cascading CNNs to determine face position and facial landmarks (eyes, nose, mouth)

- Most accurate, slowest



https://arxiv.org/abs/1604.02878
https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff

Comparison Between Haar and MTCNN

- Based on this author:
  - https://datawow.io/blogs/face-detection-haar-cascade-vs-mtcnn

- Benchmark Dataset: UTK Face (24,111 faces)

	Precision	Recall	Time
Haar Cascade	95.2%	82.6%	25 images / sec (CPU only)
MTCNN	98.0%	89.9%	3 images / sec (CPU only)

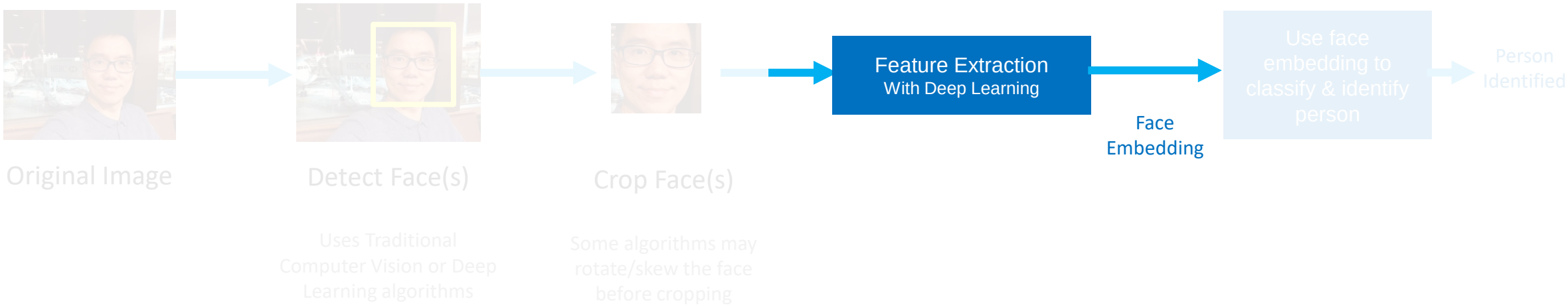
Comparison Between Haar, HoG, MTCNN

https://www.youtube.com/watch?v=GZ2p2hj2H5k

Facial Recognition

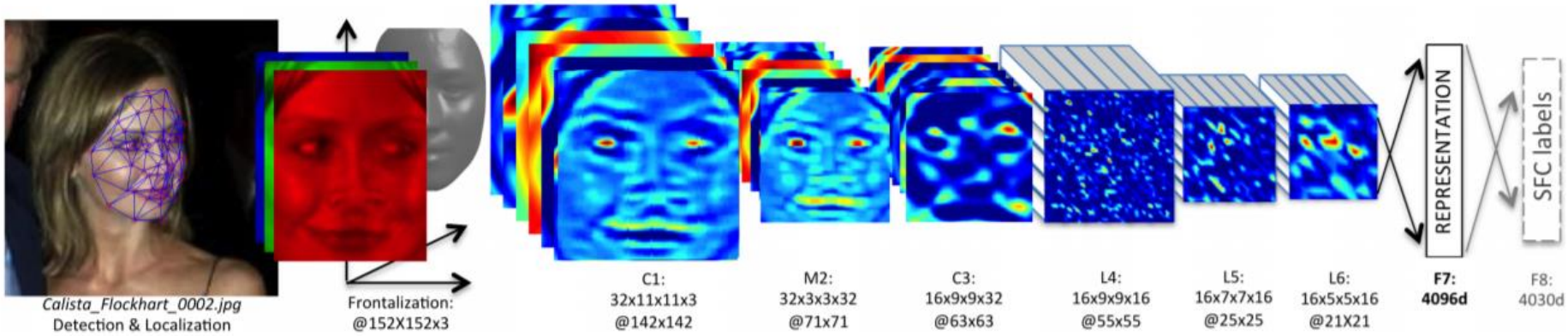
FACE EMBEDDING NETWORKS

Facial Detection



DeepFace

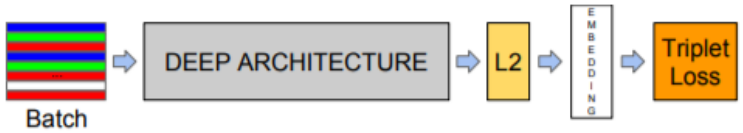
- Published by Facebook in 2014
- Improves accuracy by using 3D facial modelling to ‘align’ face before recognition



https://research.fb.com/wp-content/uploads/2016/11/deepface-closing-the-gap-to-human-level-performance-in-face-verification.pdf

FaceNet

- Published by Google 2015,
- FaceNet uses a Deep Convolutional Neural Network

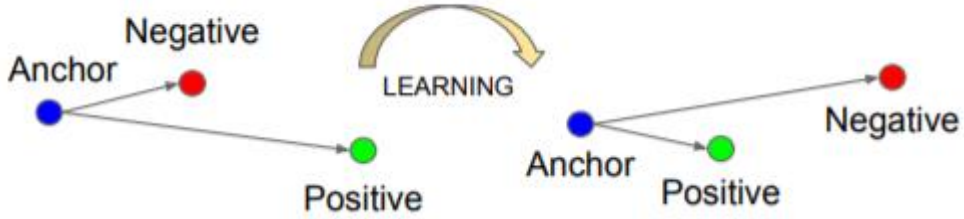


- The inputs are batches of **cropped** faces
- The “Deep Architecture” is Inception / Zeilur & Fergus architecture
- During training, FaceNet uses **Triplet Loss**
- During normal use, FaceNet produces the Face Embedding of an image of a person’s face

https://arxiv.org/abs/1503.03832

FaceNet Training with Triplet Loss

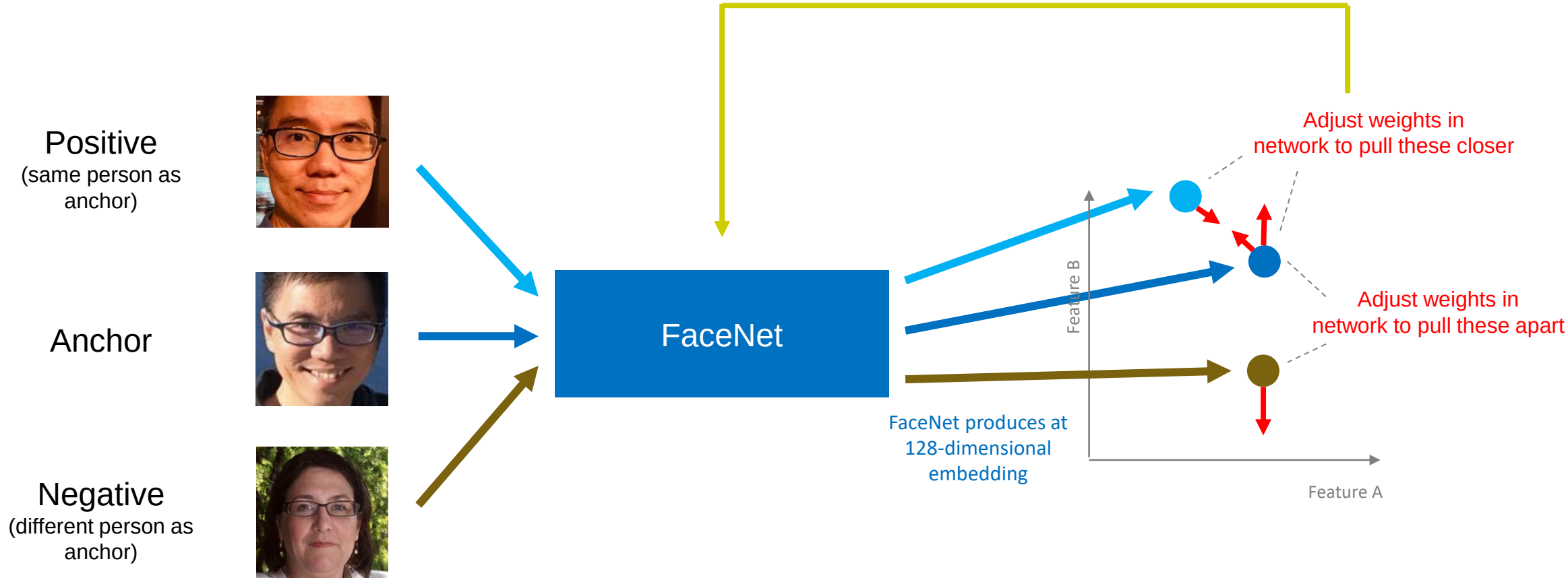
- Triplet Loss:



- For each training, randomly pick 3 faces:
  - 1 anchor
  - 1 positive example (another image of the same person in anchor image)
  - 1 negative example (another image of the different person in anchor image)

- Objective of Triplet Loss:
  - Maximize distance between (anchor to negative example)
  - Minimize distance between (anchor to positve example)

FaceNet



Comparison Between DeepFace and FaceNet

- Read comparison:
  - https://projectsflix.com/machine-learning/deeplearning/deepface-and-facenet-for-face-recognition/
  - Based on the papers provided in the previous slides

	Accuracy on LFW Dataset	Accuracy on YTF Dataset	Embedding
DeepFace	97.2%	91.4%	4096-dimensions
FaceNet	99.6%	95.1%	128-dimensions

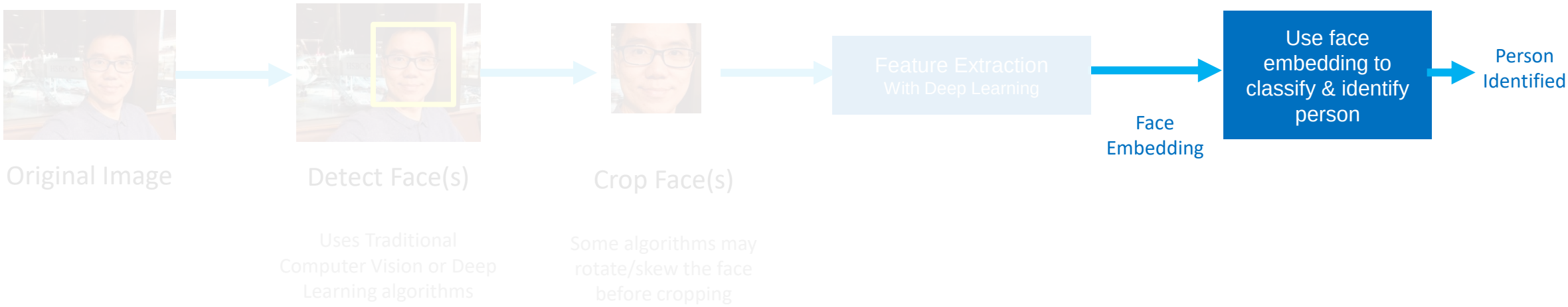
Other Materials

- Presentation of FaceNet's Triplet Loss by Andrew Ng:
  - https://www.youtube.com/watch?v=d2XB5-tuCWU

Facial Recognition

CLASSIFICATION ALGORITHMS

Facial Detection



Classification

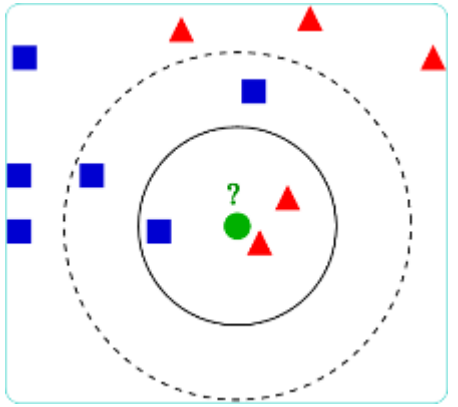
- A classification algorithm is required when you want to perform:
  - **Facial Recognition**

- The output from the deep learning network is a set of n-dimensional face embedding.

- Since it's a fixed set of output of n numbers, you can use any classification algorithm available in Scikit-Learn

K-NN Classification

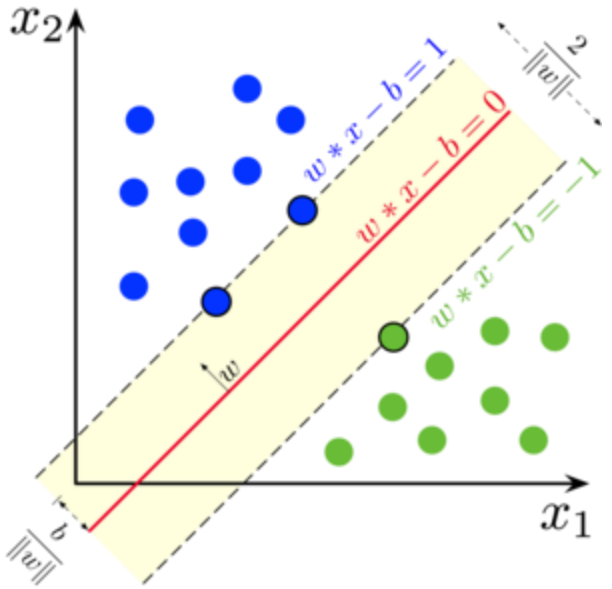
- *k*-Nearest Neighbours:
  - Look for all indexed face embeddings in your database
  - Find the *k* nearest indexed face embeddings (points) matching the new one that you've just obtained from your image
    NOTE: Set k = 1 to classify based on nearest point
- One-shot
  One photo per face; no training happens
- More data = slower inference



https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

SVM Classification

- Support Vector Machine:
  - Divides the face embedding space into one or more linear / radial hyperplanes

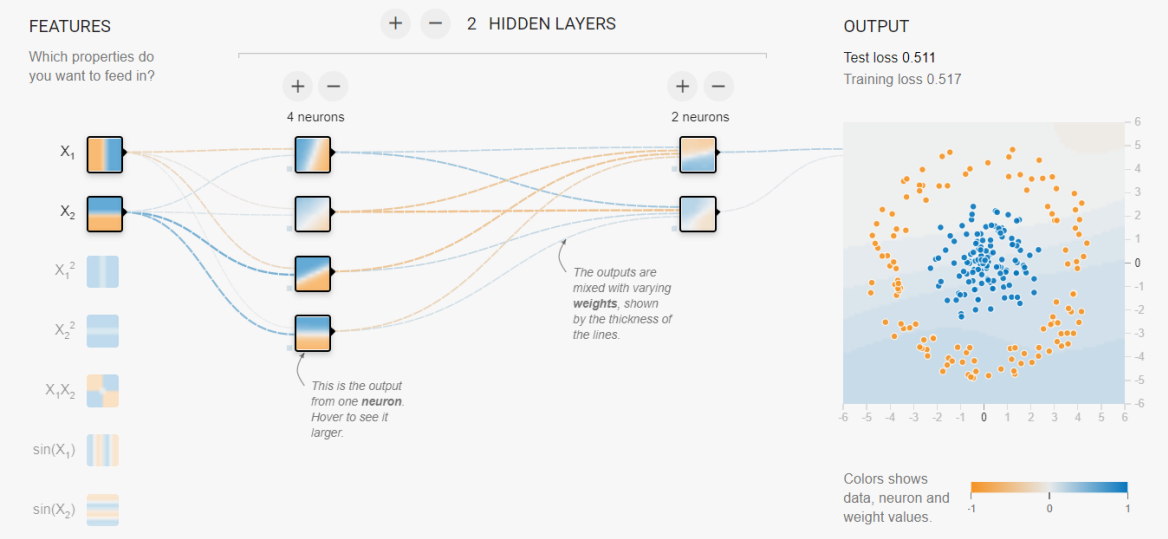


- Good with small number of data
- Training speed slow with more data
- Faster inference compared to kNN

https://en.wikipedia.org/wiki/Support-vector_machine

Neural Networks

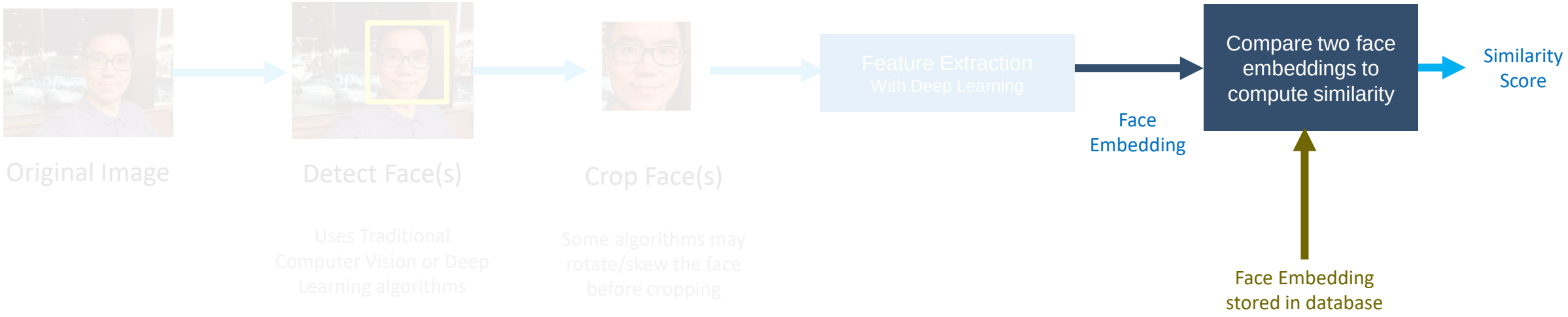
- Artificial Neural Networks:
  - Using neurons to classify face embeddings
- May require more than 1 photo per face for training



Facial Recognition

SIMILARITY MEASURES

How it Works – Facial Verification



Similarity Measure

- A similarity measure is required when you want to perform:
  - **Facial verification**

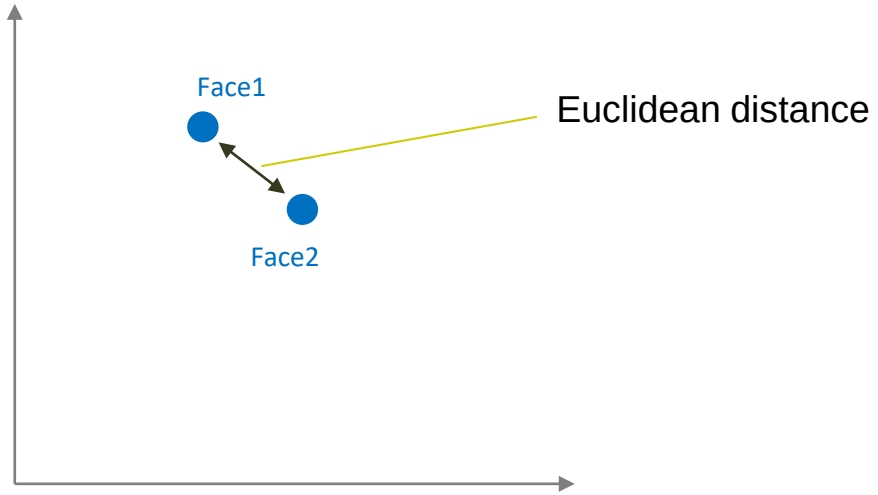
- Useful if you have a reference image of a person, and the live image of another person, and you want to compare to see if both are the same person.

- If the similarity of the face embedding of both images are within a threshold (decided by you) => both faces belong to same person

Euclidean Distance

- Distance between 2 points, p and q
- Computed with this formula: $$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

- Smaller value => more similar

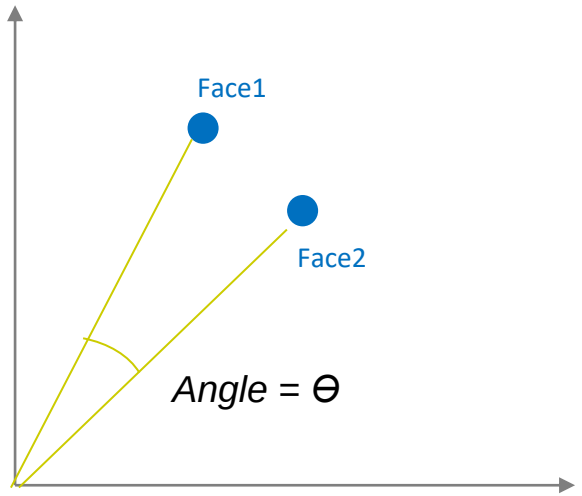


Cosine Similarity

- The cosine of the angle between 2 points, A and B from (0, 0)
- Computed with this formula:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

- Larger value => more similar

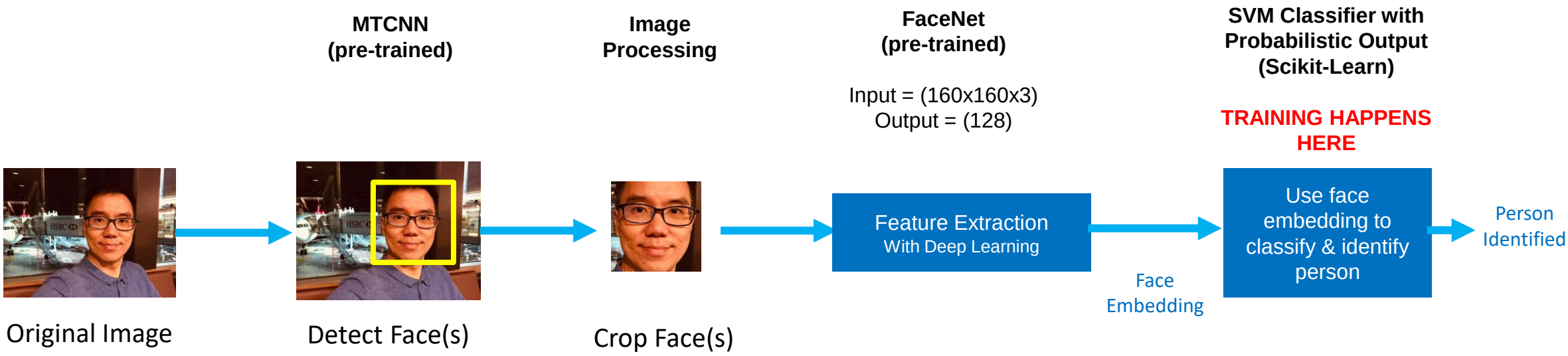


Facial Recognition

PUTTING IT ALTOGETHER

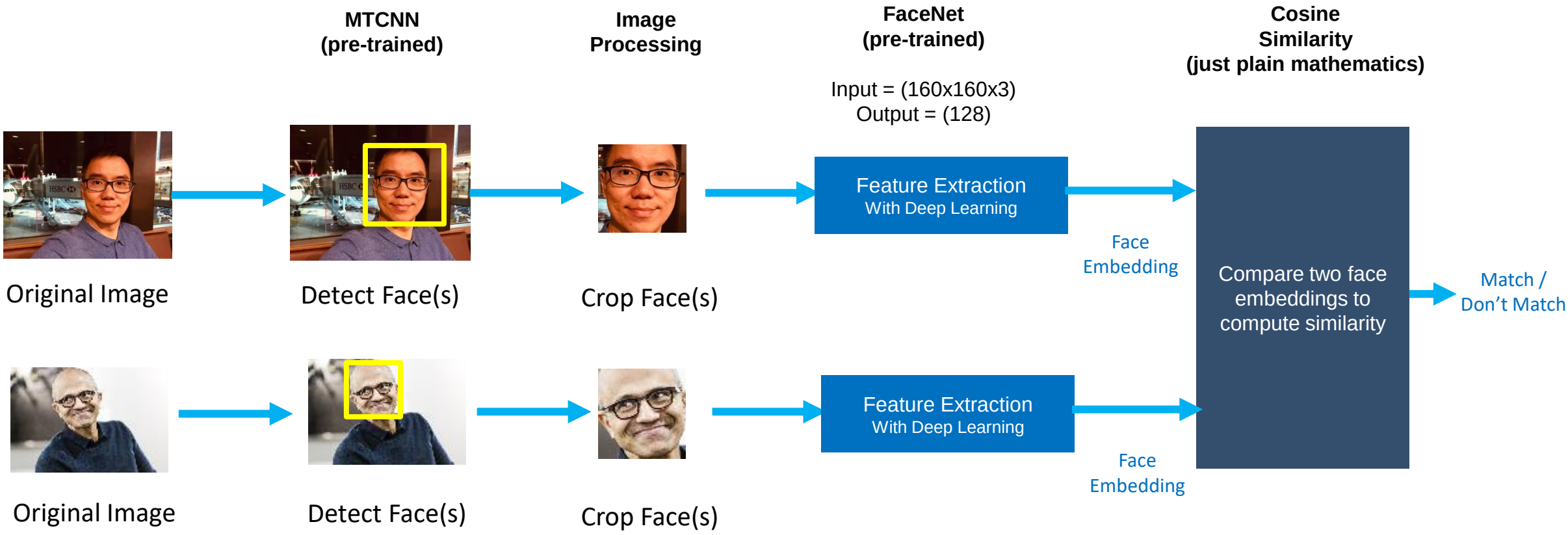
Putting It Together – Facial Recognition

- To put together a Facial Recognition engine, we must decide on the libraries at each step. For our practical, this is what we will use:



Putting It Together – Facial Verification

- To put together a Facial Verification engine, this is what we will use:



Putting It Together

- The libraries we use:
  - Pre-Trained MTCNN for Facial Detection https://github.com/ipazc/mtcnn
  - OpenCV for image loading, resizing, and other operations
  - Pre-Trained FaceNet https://github.com/nyoki-mtl/keras-facenet
  - Scikit-Learn for SVM Classifier
  - Numpy for mathematical computations

Summary

- Facial Tasks
- How It Works
- Facial Detection Algorithms
- Face Embedding Algorithms
- Classification Algorithms
- Similarity Measures
- Putting It Altogether