Attendance

outline

- [x] Identifying Faces in an Image
- [x] [Posing and Projecting Faces](#2. Posing and Projecting Faces)
- [x] [Face Recognition Embeddings](#3. Face Recognition Embeddings)
- [x] Finding Person's name from the encoding

Others

- [x] How many images per person
- [x] [Liveness detection](#Liveness Detection)
- not mandatory
  - [ ] Making Clusters for unknown people
  - [x] Saying the name of the person

1. Identifying Faces in an Image

HOG

we first make the picture gray-scale as we don't need colors to identify a face. then we replace each individual pixel with an arrow showing which direction the image is getting darker (gradients showing the flow from light to dark across the entire image).

- why not pixels directly:
  - different images may have very bright and very dark pixels for the same person.
  - but when using the direction of brightness changes, both very dark images and really bright images will end up with the same exact representation.
- Seeing the forest for the trees
  - Saving the gradients of every single pixel gives us way too much detail.

- We only need to see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image
  - Break image into small squares 16x16, count the gradients pointing in the major directions (up-right, down-left, ...), then replace the whole square with the arrow directions that were strongest.

Then to find faces in this HOG image:

- just find the part of the image that looks the most similar to a known HOG pattern that was extracted from other training faces.

Pros

1. Fastest method on CPU
2. Works very well for frontal and slightly non-frontal faces
3. Light-weight model as compared to the other three.
4. Works under small occlusion

Basically, this method works under most cases except a few as discussed below.

Cons

1. The major drawback is that it does not detect small faces as it is trained for minimum face size of 80×80. Thus, you need to make sure that the face size should be more than that in your application. You can however, train your own face detector for smaller sized faces.

2. The bounding box often excludes part of forehead and even part of chin sometimes.
3. Does not work very well under substantial occlusion
4. Does not work for side face and extreme non-frontal faces, like looking down or up.

CNN

MMOD: Max-margin object detection algorithm with CNN instead of HOG

Pros

1. Works for different face orientations
2. Robust to occlusion
3. **Works very fast on GPU**: but do u have large batches
4. Very easy training process

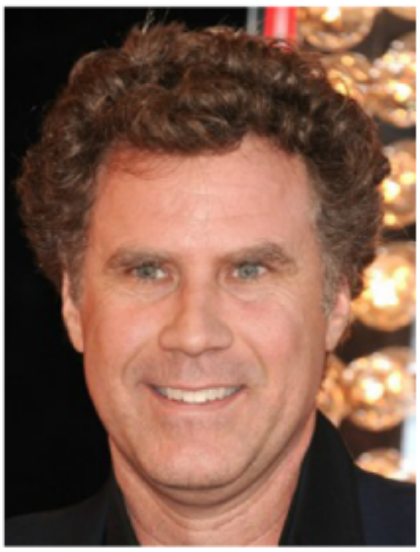
Cons

1. Very slow on CPU
2. **Does not detect small faces as it is trained for minimum face size of 80×80. Thus, you need to make sure that the face size should be more than that in your application. You can however, train your own face detector for smaller sized faces.**
3. The bounding box is even smaller than the HoG detector.

Conclusion

- Dlib based methods are pretty fast but fall apart when the size of the face is small, smaller than 70x70.

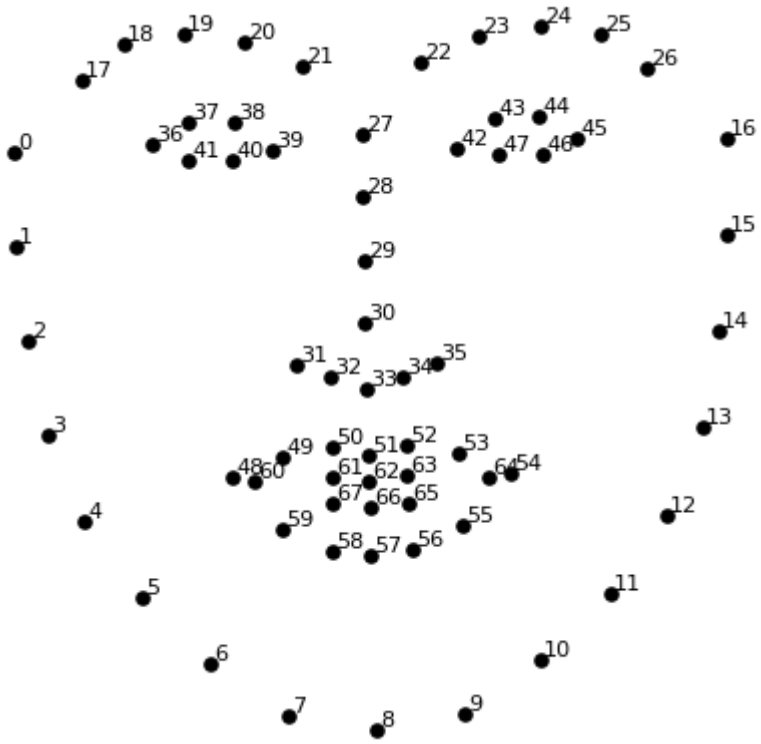
2. Posing and Projecting Faces



these two pictures look the same for humans but have totally different values for the computers to see.

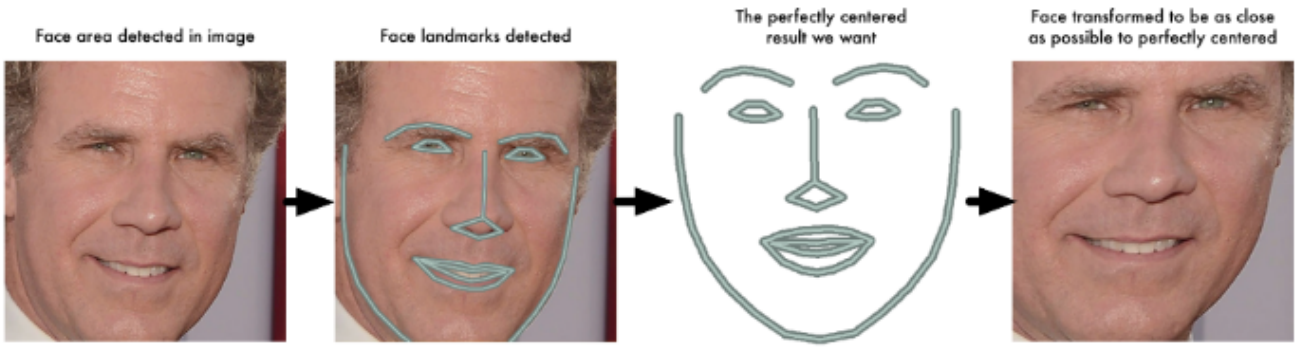
Face landmark Estimation

68 specific points (landmarks) using ML



Affine Transformations

using basic image transformations like rotation and scale that preserve parallel lines.



Now no matter how the face is turned, we are able to center the eyes and mouth are in roughly the same position in the image. This will make our **Face Recognition** more accurate.

3. Face Recognition Embeddings

Deep Metric Learning (Similarity Learning)

- Supervised
- The goal: A similarity function measures how similar or related two objects are.

Ranking Similarity Learning

- Given are triplets of objects (X, X+, X-) whose relative similarity obey a predefined order: X is more similar to X+ than to X-.
- The goal: learn a function f such that for any new triplet of objects, it obeys $f(x_i, x_i^+) > f(x_i, x_i^-)$

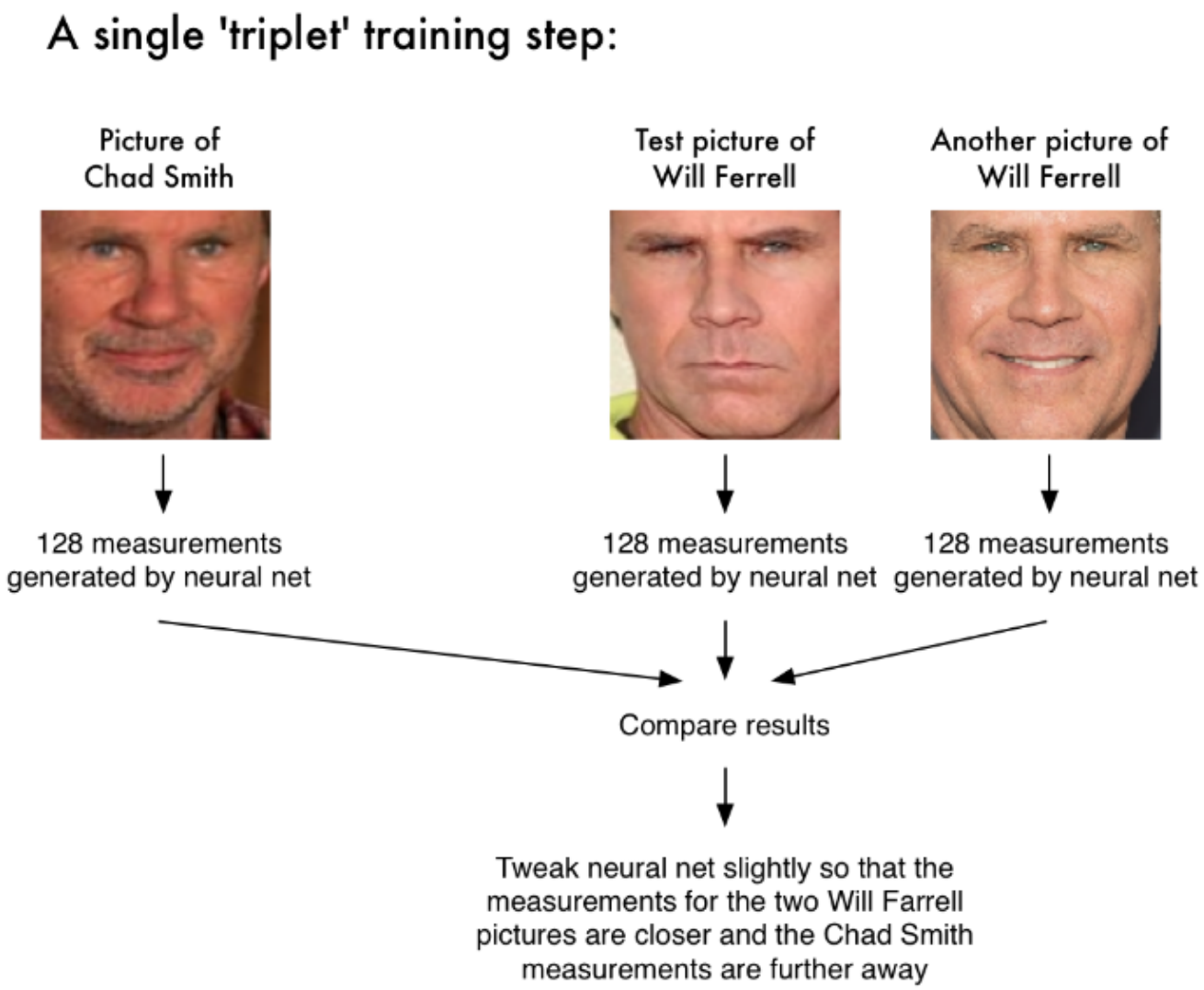
Measuring the face features

The Training works by looking at 3 face images at a time:

1. Load a training face image of a know person
2. Load another picture of the same known person
3. Load a picture of a totally different person

The Embeddings

then the algorithm generates measurements for each face, making sure that the measurements it generates for #1, #2 are slightly closer while making the measurements for #2, #3 are slightly further apart.



Liveness Detection

- Get data

- Train a simple model (small model will be enough)

Edge Cases

- [x] Hijab, Khimar
  - [x] using default implementation
  - [x] Using Multiple images
- [x] People with darker skin color

Test Cases

- [x] face_encoding directly or face location and then encoding
  - as we will need the later to pass to liveness detection