Deep Learning For Attendance

Btech Project

Akash Dhasade, Saket Joshi

Problem Statement

YES - Configure the best available facilities of face detection and face recognition for the purpose of marking attendance. Create a pipeline of processes that gives best results for classroom attendance.

NO - Build a CNN for face detection and face recognition that gives best results.

Dataset

Self generated dataset: We took several images and videos in the process of building the pipeline. The results are yet to be verified on a larger dataset.

Literature Review

Face detection

1. Viola Jones
2. Trained CNN's
3. Histogram of oriented gradients

Face recognition

1. Use CNN's that generate face encodings. Match people by comparing their face encodings.

Viola Jones Algorithm

Haar features

↓

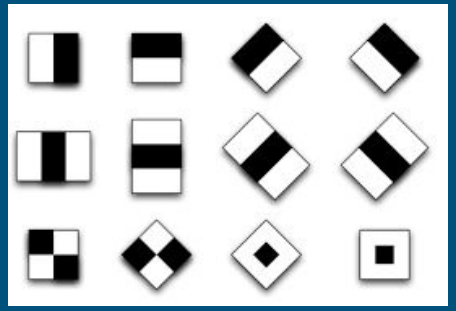
Integral Image

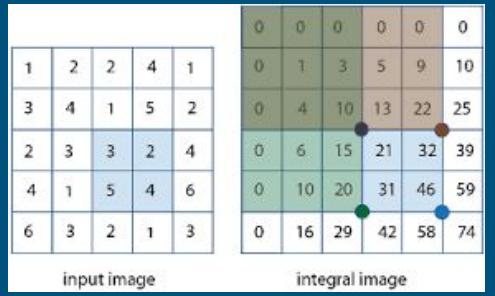
↓

Adaboost algorithm

↓

Cascade of classifiers





Histogram of Oriented Gradients

Gradient Image

↓

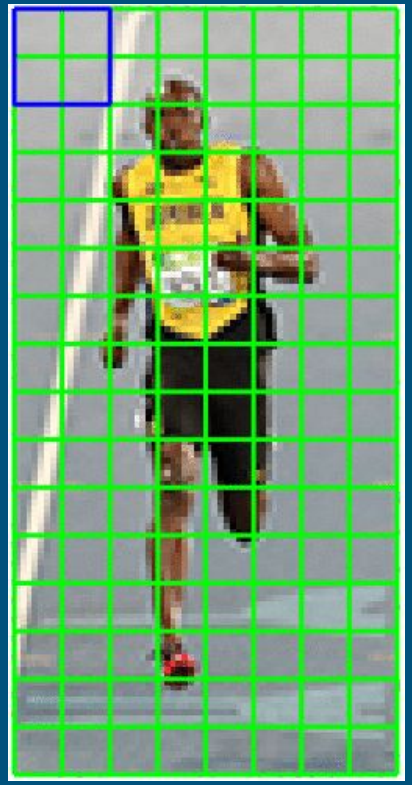
Calculate histogram of gradients in 8 X 8 cells

↓

16 x 16 block normalization

↓

Final feature vector



Face recognition

Traditional approach: Train a CNN to identify persons.

1st layer of CNN → Image

Output Layer of CNN → Identity of person

Approach we choose: Get encodings from faces that can best distinguish them.

Build a CNN/neural network that can generate encodings.

Train classifier based on encodings

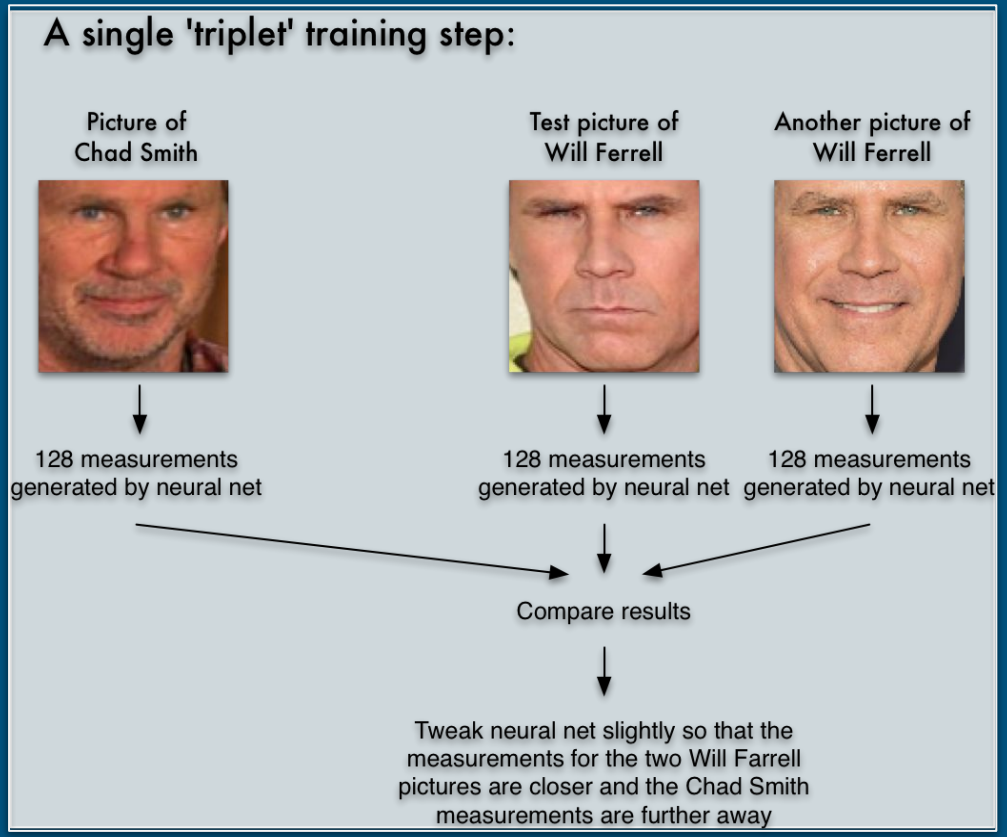
What features to choose?

How training works?

What features to choose?

Let the computer figure out!

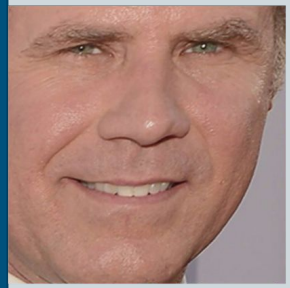
Train a deep CNN to generate measurements for face.



Using trained networks

Lucky for us, the fine folks at OpenFace already did this and they published several trained networks which we can directly use.

Input Image



128 Measurements Generated from Image

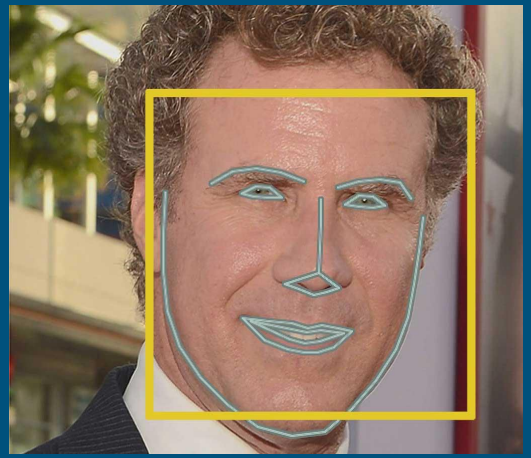
0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.116387888778918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

The network is so awesome that just Euclidean distance works!

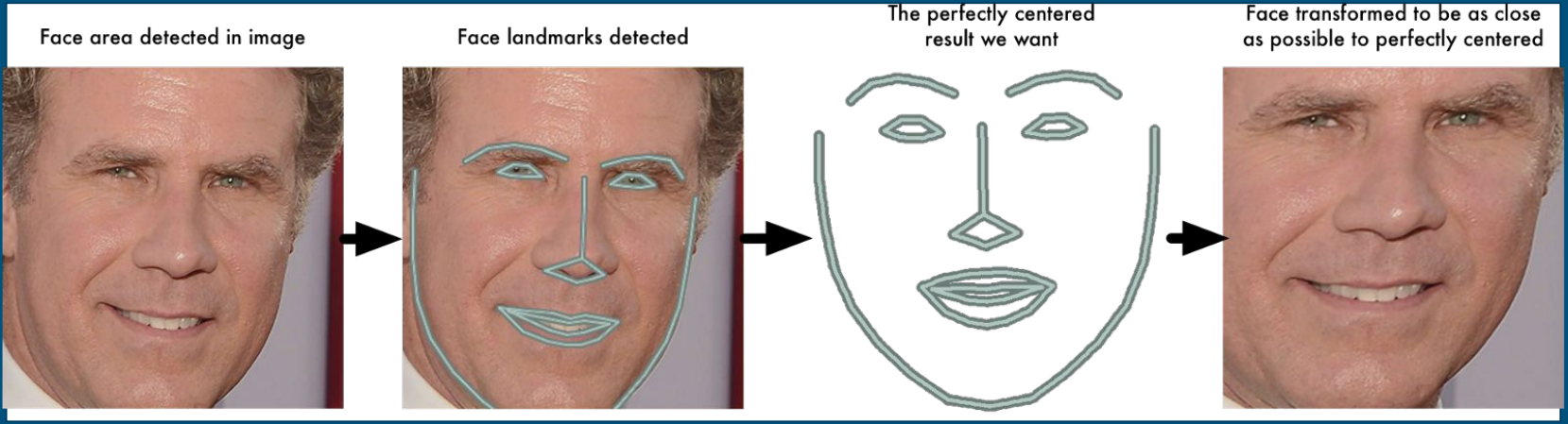
Posing and projecting faces

- We use an algorithm called face landmark estimation.
- A trained machine learning model is able to identify 68 specific points (called landmarks) given a face. These include - top of chin, inner edge of eyebrow, etc.
- Knowing where these different points are, we can shear, rotate or apply similar affine transformation to make the eyes and mouth as centred as possible.

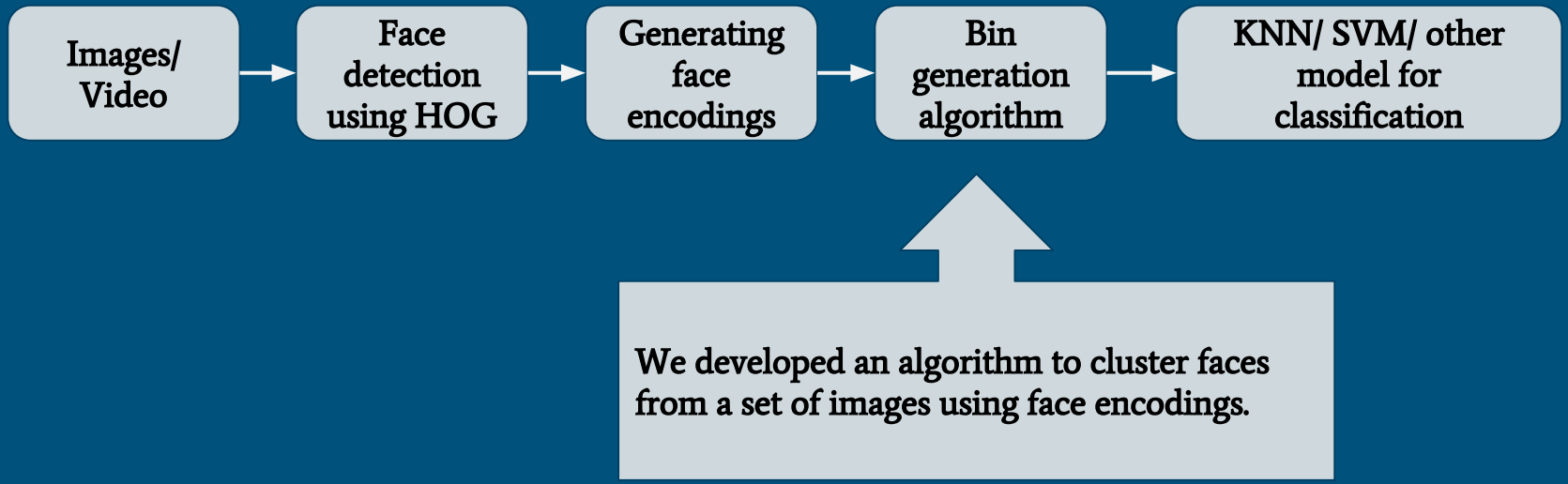
Shown next



Making faces perfectly centred



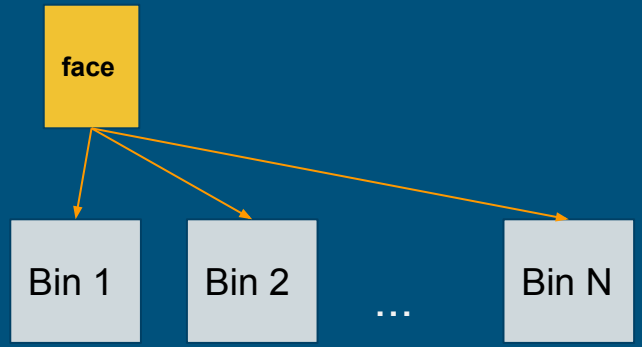
Our pipeline



Algorithm to cluster faces, create Dataset

Input - Set of images taken from a video stream, face encodings of faces detected in the images

Output - Bins of images one for each unique person



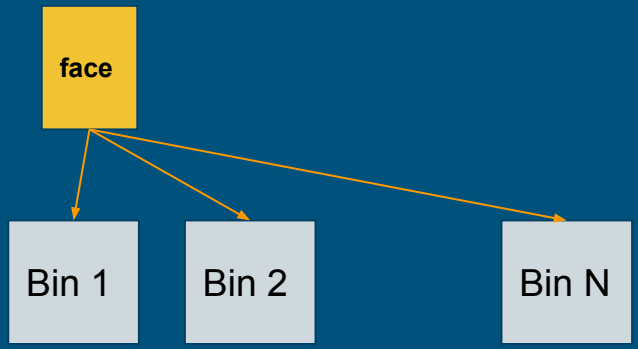
Given each bin belongs to a different person, where do we place a face such that the error of incorrect placements in minimized?

Note that we do not know any such bin apriori !

The problem with Clustering algorithms

One approach to create an initial database for training is to

- take frames from video feed
- Detect faces and pass to a clustering algorithms.



Limitations:

- Number of people is required as an input
- This method neglects an important piece of information: faces in the same image are different.

Possible solutions: ...

- limitation 1: Use a threshold to decide number of clusters
- Limitation 2:
  - Use constraint-based clustering
  - We use a custom algorithm based on prioritized recognition which incorporates this information.

```
images : [Image]
faces : [Face]
bins: [Bin]

//bin initialization
Let I be the image with maximum number of detections
for f in I:
  create_bin(f)

//method to find bin score
def bin_score(f):
    bin_scores = []
    for bin in bins:
      sum = 0
      for face in bin:
          sum += get_score(f, face)
      sum /= size(bin)
      bin_scores[bin] = sum
      return bin_scores
```

Algorithm - Part 1

```
// main loop

while not empty(faces):
  all_scores = []

  for face in faces:
    //get bin score for the face
    bin_scores = bin_score(face)

    // find max bin score and its bin index
    max, bin_index = get_max(bin_scores)

    //record this score for the face
    all_scores[face] = (max, bin_index)

  face_index = index of face with max score in all_scores

  //We choose this as the next face to be put
  next_face = faces[face_index]

  // remove this face from the list
  faces = faces - faces[face_index]

  // add this face to its corresponding bin
  bin[bin_index] += face

  recalculate_scores(faces, bin_index)
```

Algorithm - Part 2

Our results: Solvay conference

Test Image 1

All 16 students detected !

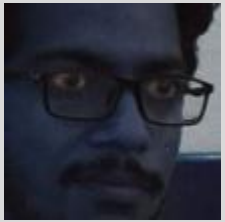
Bin Generation results!

Bin 1




...

Bin N



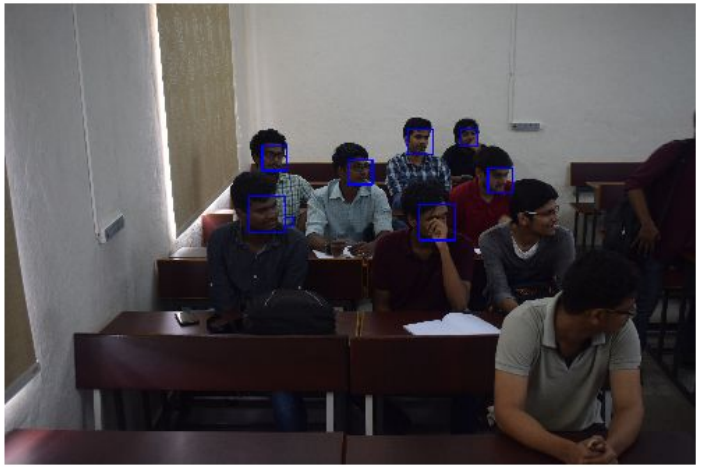


Test Image 2

12 faces detected !

Test Image 3

Robustness of face detection !
(Side profiles)



Test Image 4

Robustness of face detection !

Face Detection and recognition Evaluation 1

Evaluation 1:

Input:
Frames from a video
31 frames x 12 people = 372 faces

Face detection Accuracy = 57.3%

Face recognition (binning)
Accuracy = 100%

Observations:
It was found that images had some blur since random frames were extracted from the video feed.

False positives = 0 implies a lower threshold may be used

Failure cases are dominated by side profiles and back benches.

We expect better results when the camera is mounted.

Although in separate images, all people had more than one face detected.

Face Detection and recognition Evaluation 2

Evaluation 2:

Input:
Frames from a video
6 Images x 16 people = 96 faces

Mean Accuracy = 77.3%
Median Accuracy = 84.5%

Recognition (Binning) Accuracy = 100.0%

Observations:

Dataset used is small, there is need for better evaluation. We find that a single image with very low detection impacted the overall accuracy. Median accuracy is better.

False positives = 0

Failure cases are dominated by side profiles and back benches.

We expect better results when the camera is mounted.

All people had more than one face detected.

What remains to be done?

Quantifying results on a large dataset!

We look forward to quantify results on a large dataset as soon as possible with better imaging facilities in place.

Final touch to make it ready-to-use !

The system is almost ready. We just need to give finishing touches for proper storage and retrieval of data.

Can you show a demo by taking a picture from today's class?

Why not !

By Saket and Akash

Akash Dhasade

TCS15B031, Dept of CSE, IIT Tirupati

Saket Joshi

TCS15B034, Dept of CSE, IIT Tirupati

References

1. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78
2. https://cmusatyalab.github.io/openface/
3. https://www.learnopencv.com/histogram-of-oriented-gradients/

Thank You!