Amazon Campus Mentorship Series (ACMS)

LIVE - FACE RECOGNITION

Guided By :
Rohan Bathla

Team :
Deepti Chamoli
Devyani Bajaj
V Dhanya Akhila

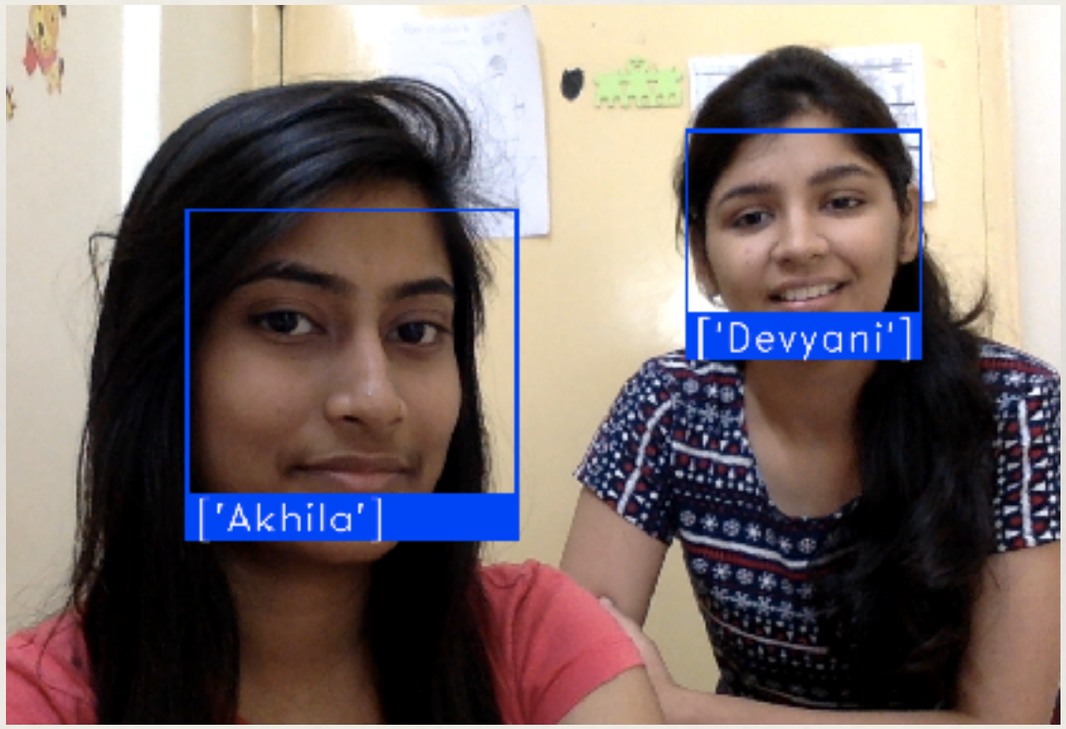
College :
IIIT Bangalore

CONTENTS

INTRODUCTION

Problem Statement

To develop an application that recognises people from live stream using camera by implementing Machine Learning and Deep Learning techniques.



Github Url : https://github.com/Jolig/Live_FaceRecognition

MAJOR STEPS INVOLVED

Live Face Recognition is a series of several related problems such as

- **Face Detection**
  - Look at the picture and find all the faces in it.

- **Posing and Projecting Faces**
  - Focus on each face and be able to understand that even if the face is turned in a weird direction and is still the same person.

- **Encoding Faces**
  - Pick out unique features of the face that we can use it to differ one face from other like - How big eyes are, How long the face is, etc.

- **Finding Person's Name from Encodings**
  - Compare the unique features of that face that you can to all the people machine already knows to determine person's name.

- **Making this process Live with Camera**

https://drive.google.com/file/d/1qgH0uw8x3MRM3nfHUJoaEe-lU1kJfE2O/view?usp=sharing

---

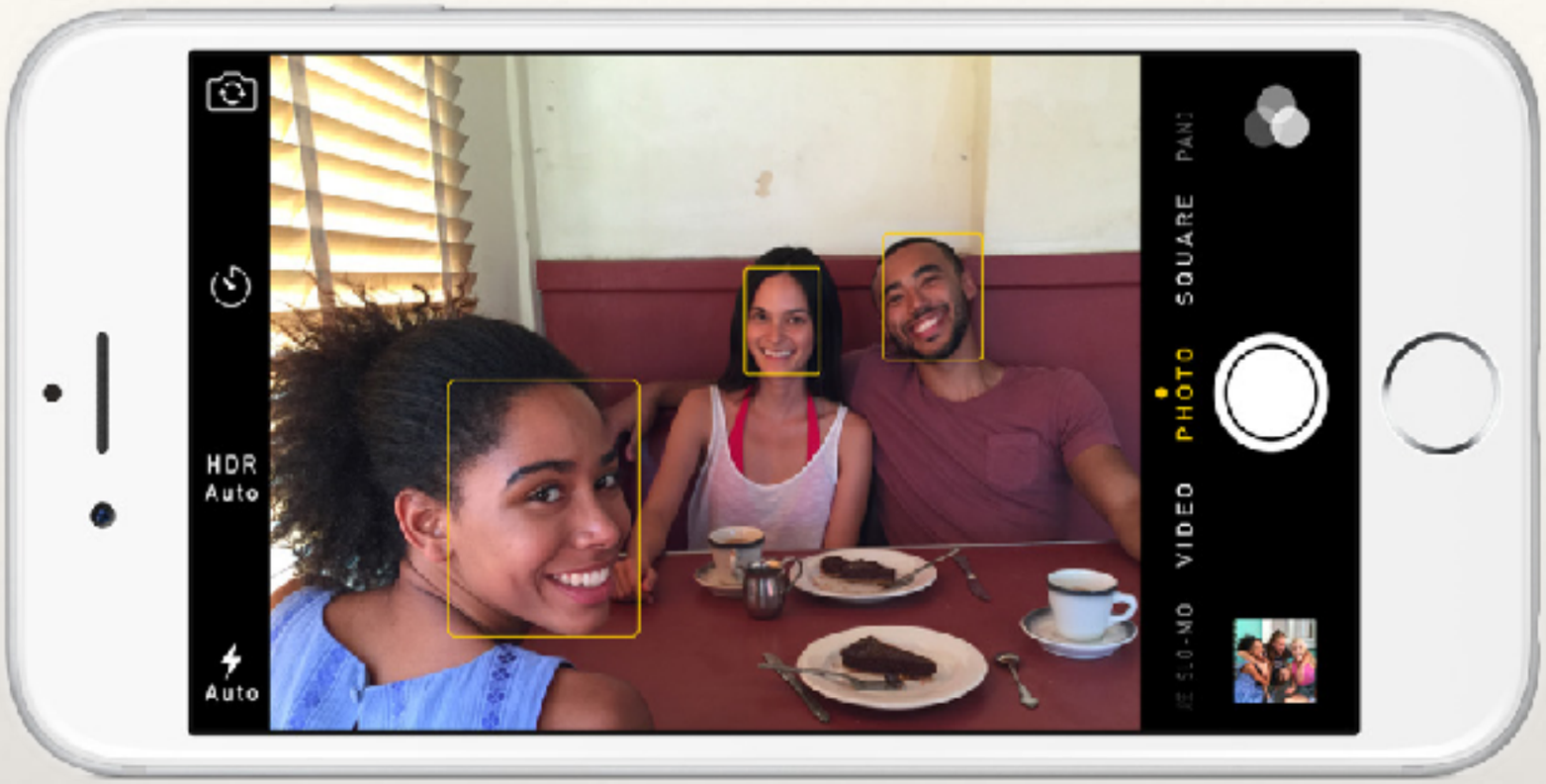
Complete Process of Face Recognition involving all the steps mentioned in the previous slide.

Sample Data

- For detailed explanation of each step we have used these pictures as sample data.

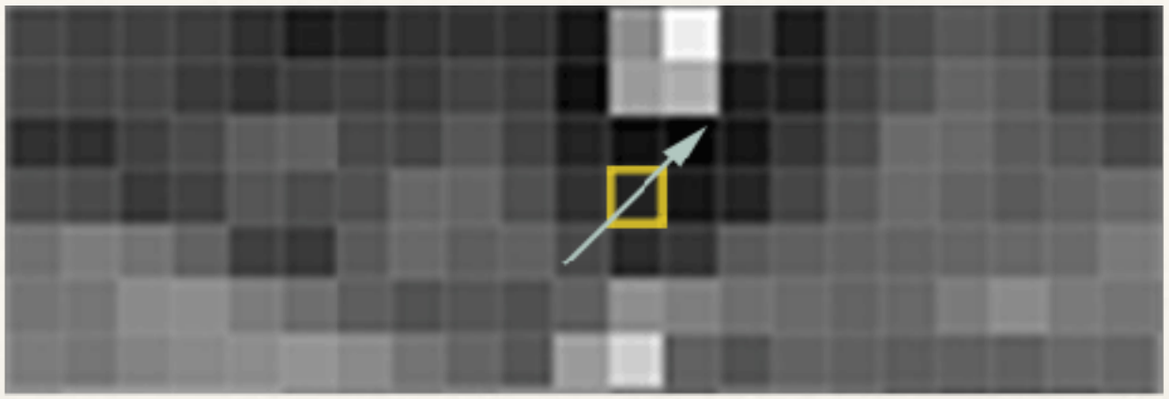
- All the face recognition steps are applied on these and finally determines the names of the people in these images.

1- Face Detection



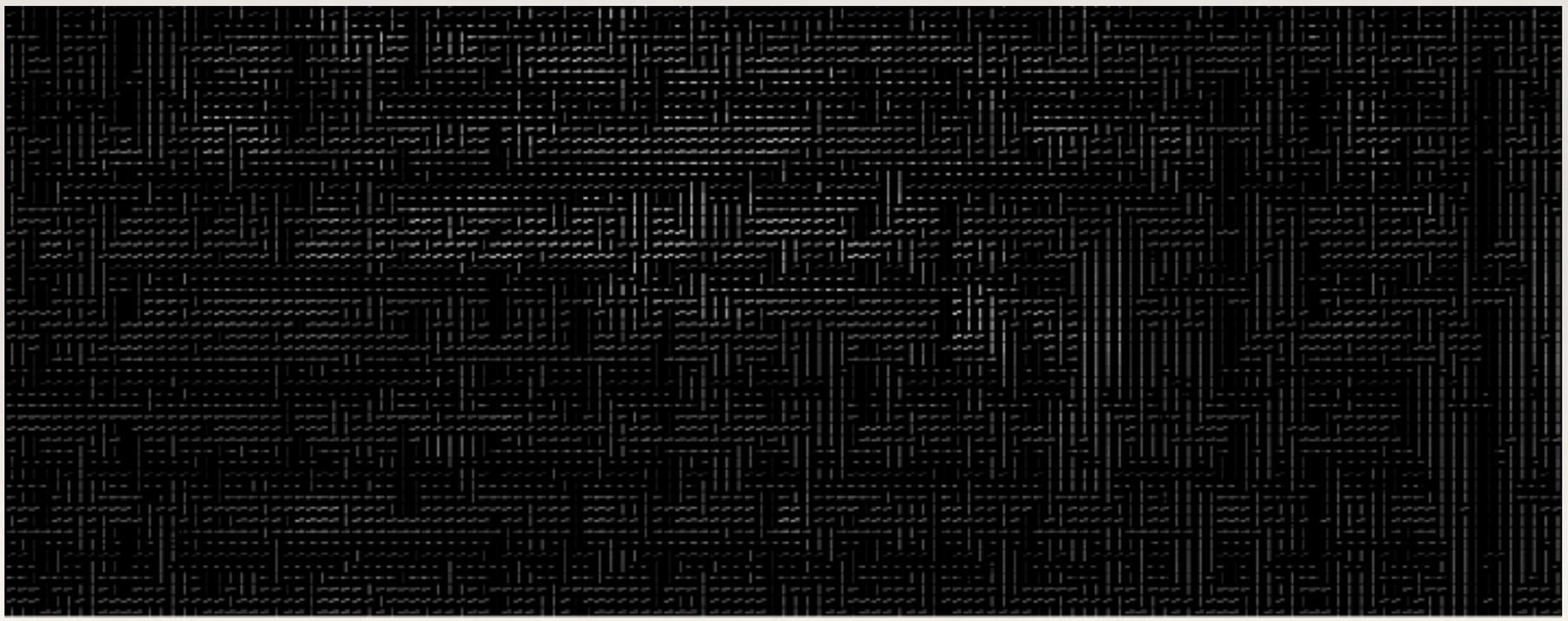
- Our First Step is to detect all the faces in the image.
- **HOG(Histogram of Oriented Gradients)** method of face detection.
- It looks at every pixel of the image one at a time and for each single pixel it looks at the surrounding pixels for comparison of DN values.

It figures out how dark the current pixel is compared to it's neighbours and draws an arrow in such a direction where image is getting dark.



Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right.

It Repeats the above process for all the pixels in the image and every pixel is replace by an arrow



These arrows are called as gradients and show the flow from light to dark across the image.

Advantages of HOG(over other techniques)

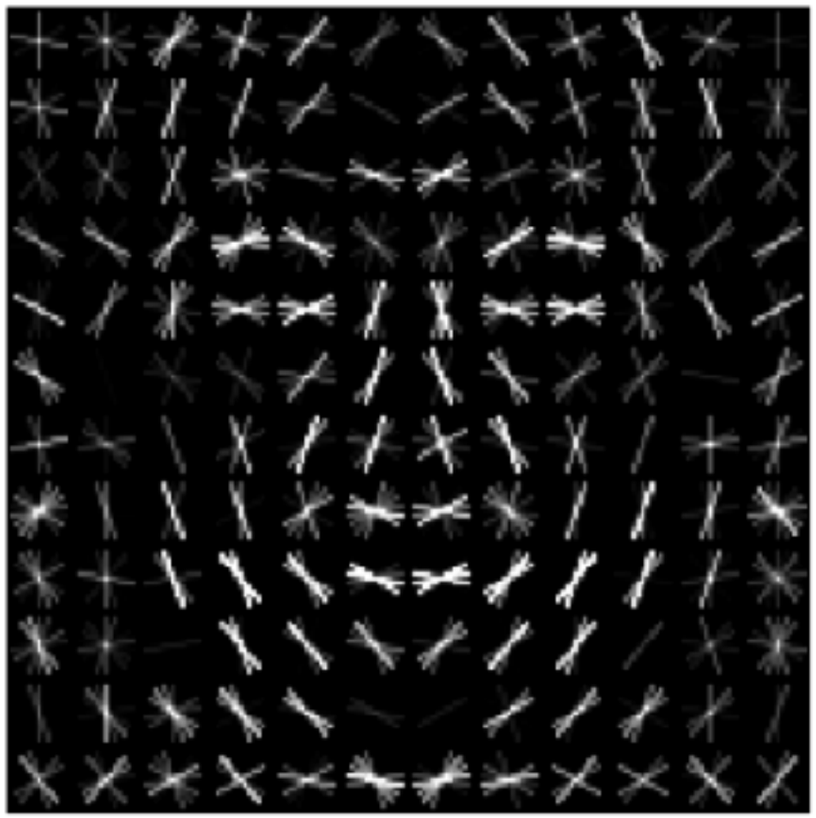
- If we analyse the pixels directly really dark and really light images of the same person will have totally different values.

- But by considering the direction in which the brightness changes both dark and bright images will end up with same representation.

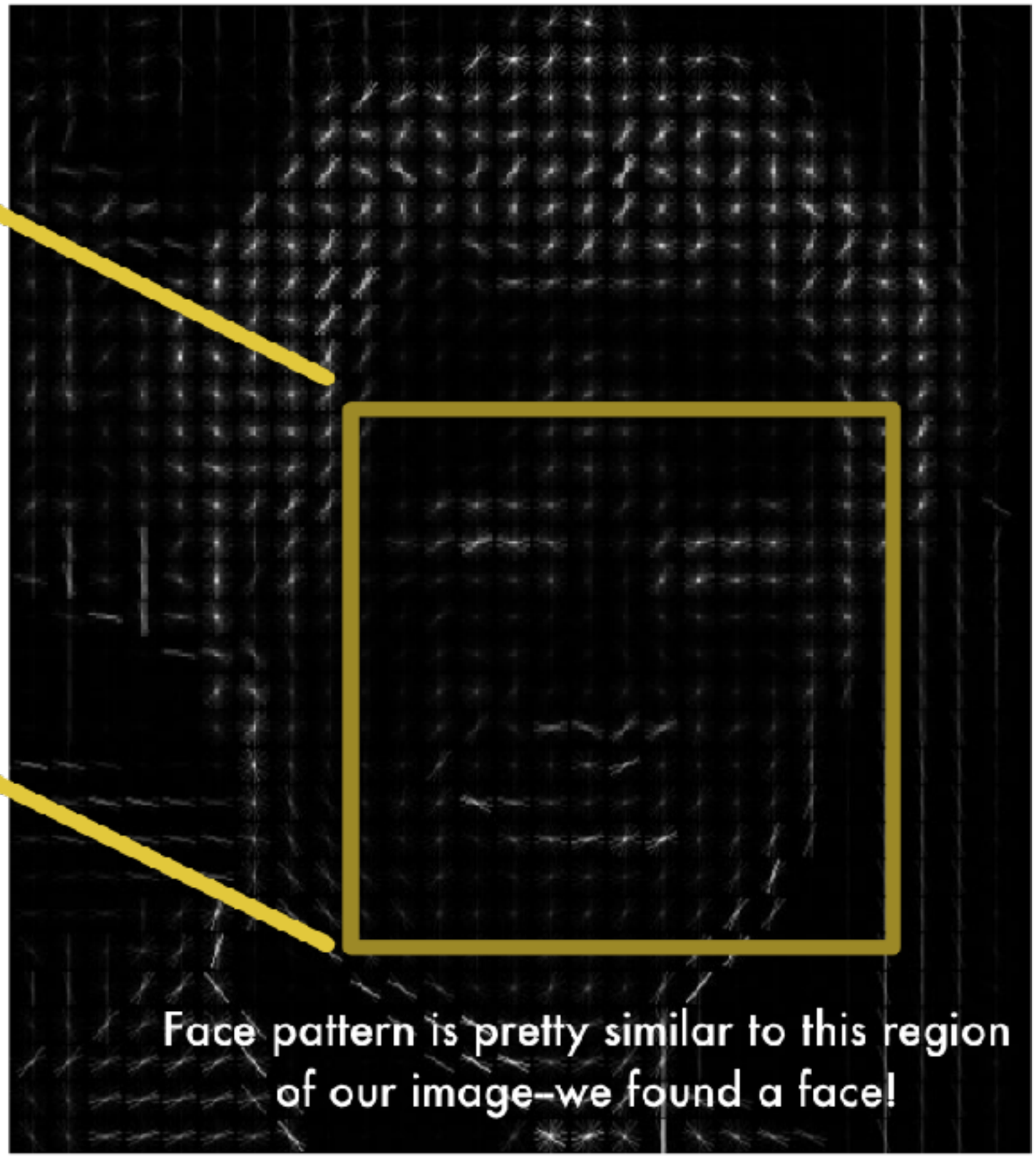
- Saving gradient for each pixel might be very messy.

- So we will break up the image into 16*16 pixels each. In each square we count the gradients in all major directions and replace that square in the image with the arrow directions that were the strongest.

HOG face pattern generated from lots of face images

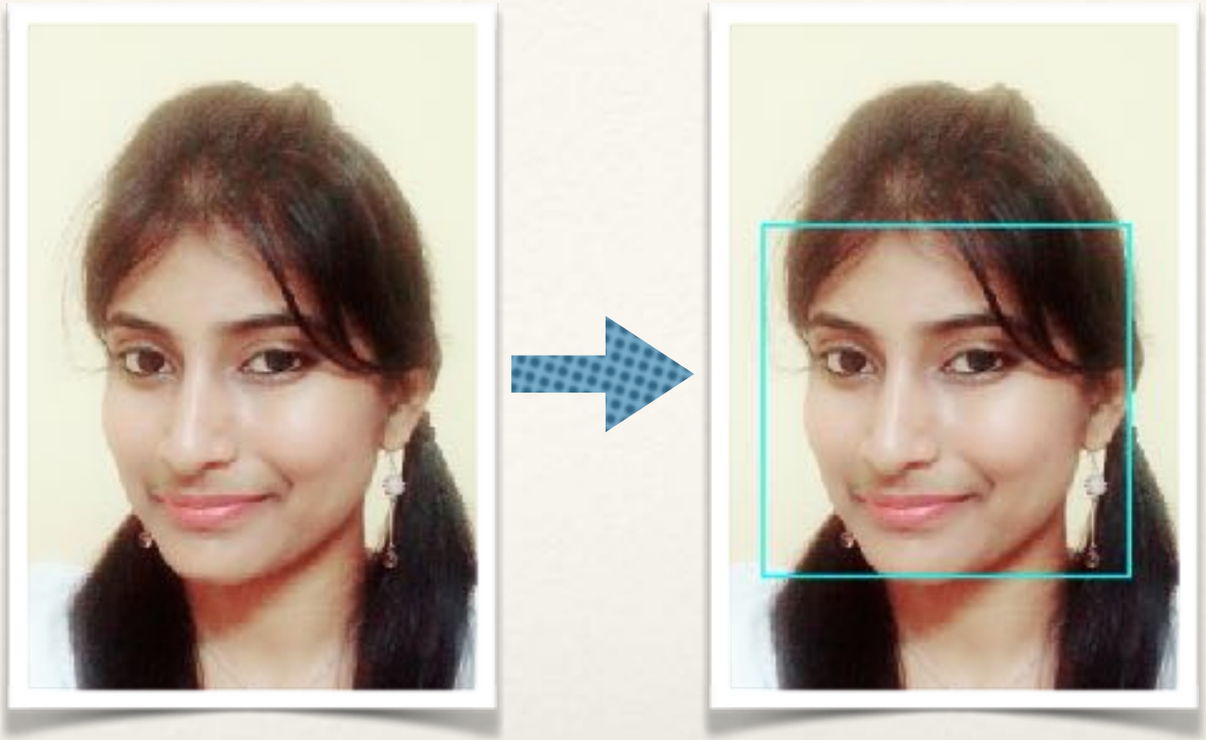


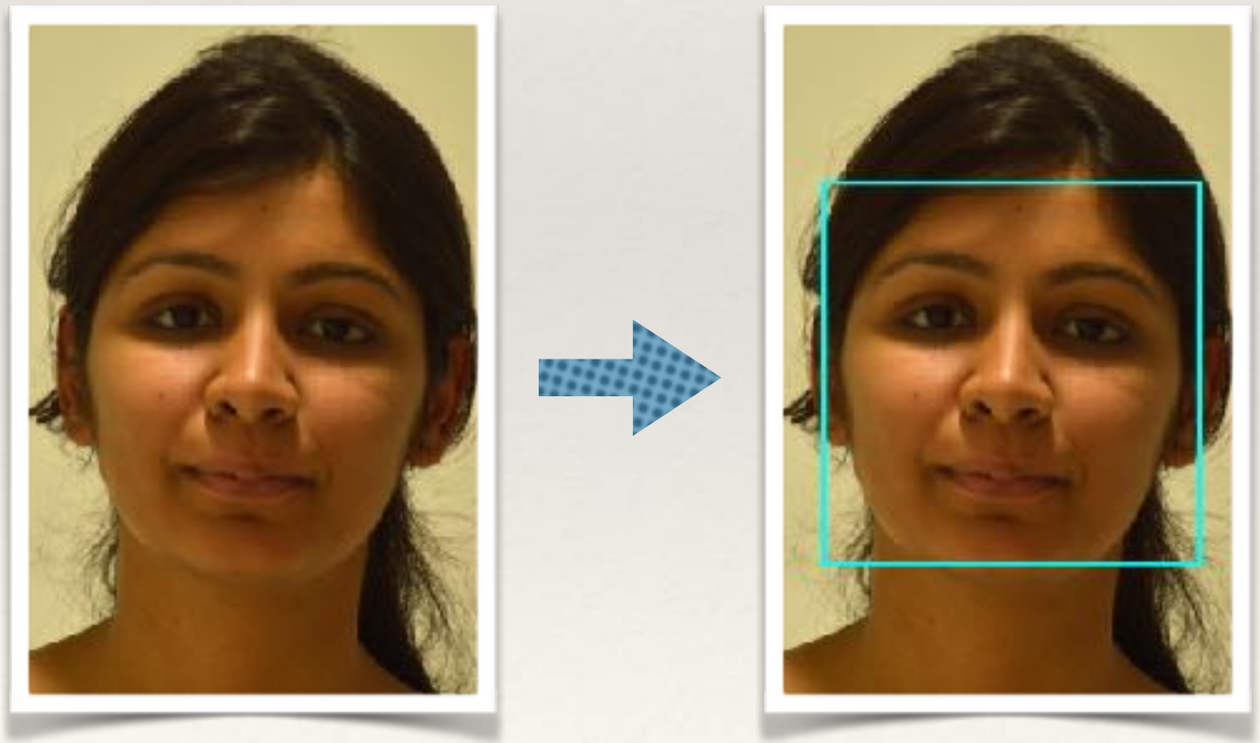
HOG version of our image



Face pattern is pretty similar to this region of our image–we found a face!

RESULT of FACE DETECTION STEP





2- Posing and Projecting Faces

- Deals with the problem that faces turned in different directions look totally different to computer.
- Method of **Face Landmark Estimation.**
- Extracts 68 specific points(called Landmarks) that exist on each face and then trains ML algorithm to predict these points in any face.

- Now simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible.

- This makes our next steps more accurate.

Face area detected in the image

Face landmarks detected

Perfectly centered desired result

Transform very near to desired result.

3- Encoding Faces

SO WHAT COULD BE THE SIMPLEST APPROACH TO RECOGNISE????

- Let us take a use case of

- when it needs to recognise a friend to automatically tag, directly compare the unknown face we found in Step 2 with all the pictures we have of friends that are already tagged.

- When we find a previously tagged face that looks very similar to our unknown face, it must be the same person.

Seems like a pretty good and simple idea, right????

BIGGEST CATCH : With billion of users and trillions of photos it's really obtuse to loop through previous faces and compare with the newly updated Pictures…..

3- Encoding Faces

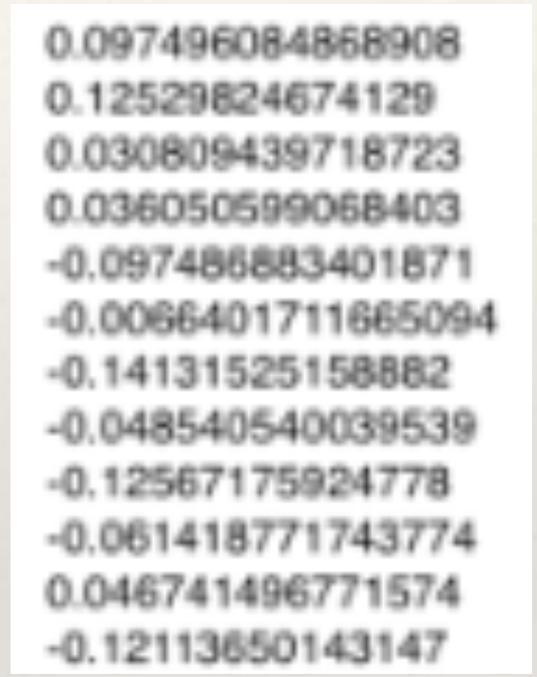
SOLUTION :

- we need a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements.

- Measurements that look obvious to humans don't really make sense to a computer looking at individual pixels in an image.

- **Deep learning** does a better job than humans at figuring out which parts of a face are important to measure.

- These measurements are called as the **Embedding** of each face.



Picture 1-1

128 measurements generated by resnet

Picture 2-1

128 measurements generated by resnet

Picture 2-2

128 measurements generated by resnet

COMPARE RESULTS

Tweak the neural network slightly so that the measurements for the two pictures(right) are closer and the one(left) measurements are further away..

128 measurements generated from image

Input Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.000521632784511 89
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.00595575105398 89
-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	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

4- Finding the Person's Name

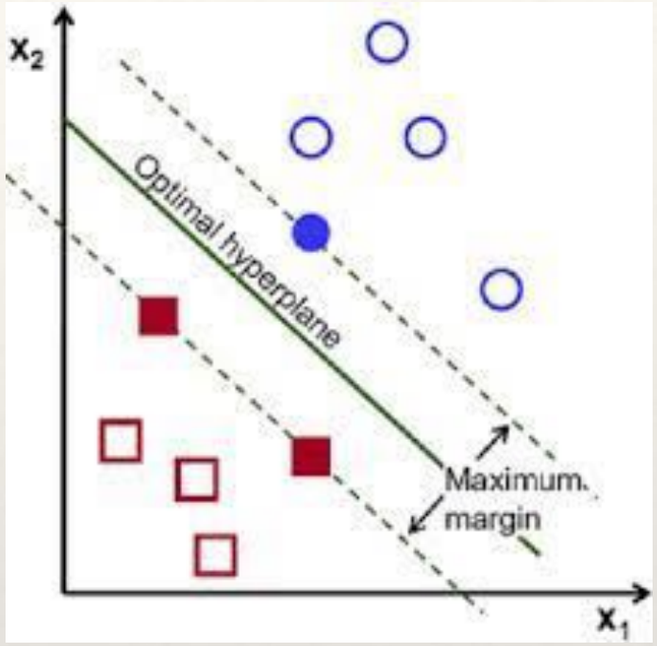
- Find the person in our database of known people who has the closest measurements to our test image.
- This can be done by using any Machine Learning algorithm.
- In our project we have used **SVM Classifier**.
- Train the SVM classifier that can take in the **Embeddings** from a new test image and tells which known person is the closest match. Result would be the name of the person..!!

What SVM Does?

- Given labeled training data, the algorithm outputs an optimal hyperplane (with maximum margin) which categorises new examples.

- In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

- For **multi class problems**, it reduces to the problem of multiple binary classification problems. It can be categorised into '***one -vs–rest***' and '***one-vs–one***'.



Why SVM?

- Works well for Multi Class Problems.
- Easy to implement than ANN(because of resource constraints).
- Fast and efficient.

5- Making the process live

Non - Live Face Recognition

Input : Plain Image

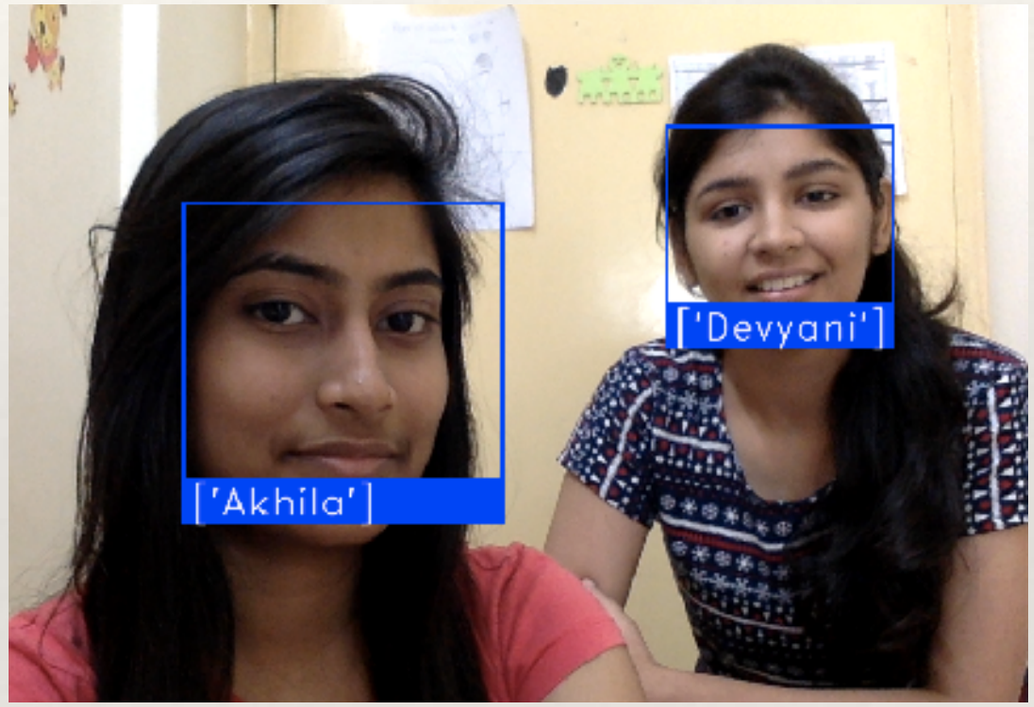
Output : Names of the persons in the image

Live Face Recognition

Input : Live Stream

Output : Names of the persons in the stream

- **openCV** - Interface to capture live stream with camera.
- It captures video from the camera(we have used laptop cam in the application)
- It further captures each frame of the video and pass that as an instance of image and all those instances are taken as input and Non-live part is followed.



IMPLEMENTATION

ABOUT DATA

Data Set:

http://cswww.essex.ac.uk/mv/allfaces/index.html

- Total number of individuals: 397(395+2)
  - 2 classes data were added manually
- Number of images per individual: 20
- Total number of images: 7940
- Gender: contains images of male and female subjects

TECHNOLOGIES USED

- **Programming Language:**
  Python

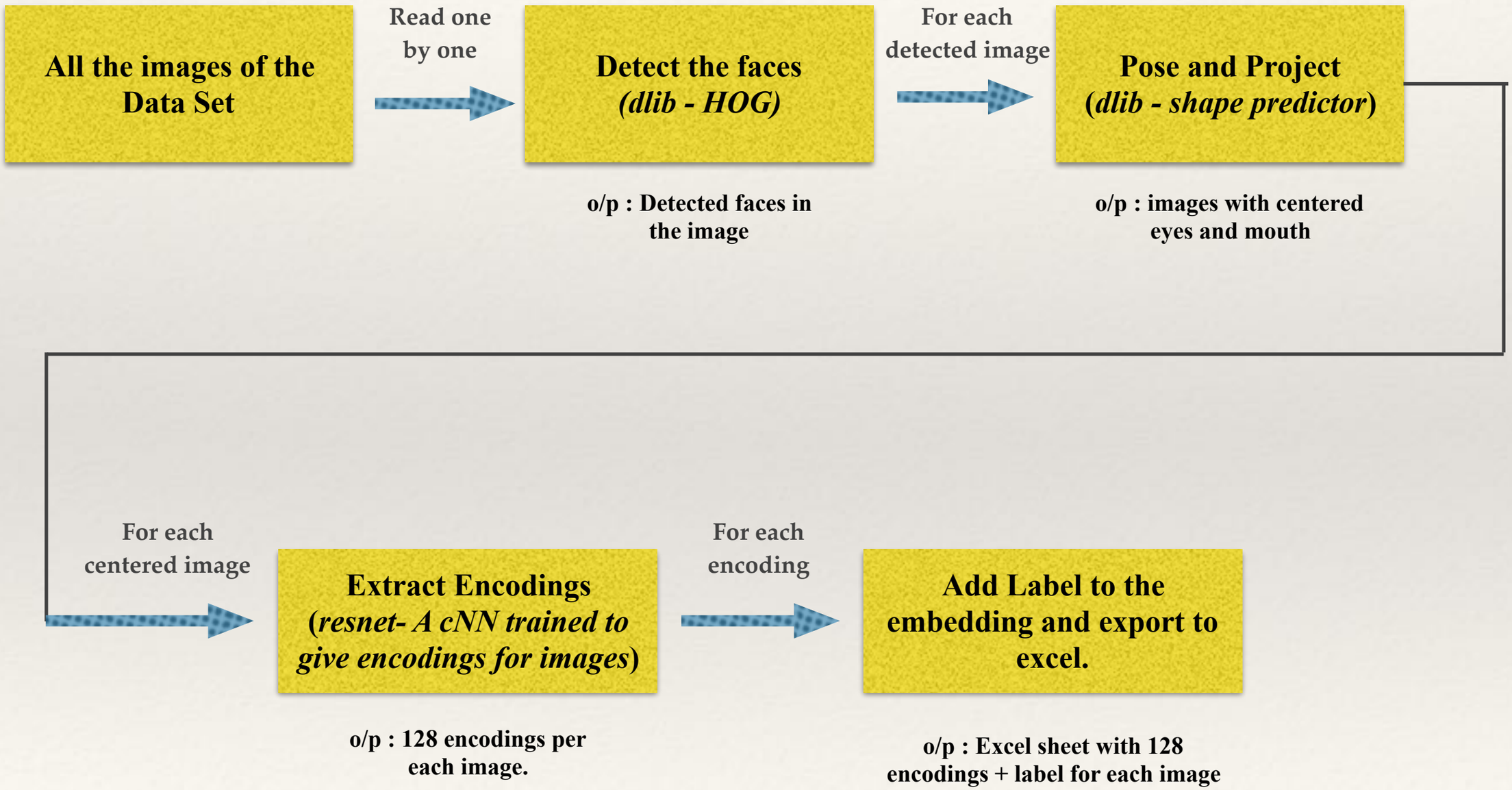
- **IDE :**
  Pycharm

- **MODELS :**
  Shape Predictor(for landmarks of Step2)
  Resnet(for embeddings of Step 3)
  SVM Classifier(trained with embeddings for step 4).

- **LIBRARIES:**
  openCV, dlib, numpy, pandas, scikit-learn, etc.

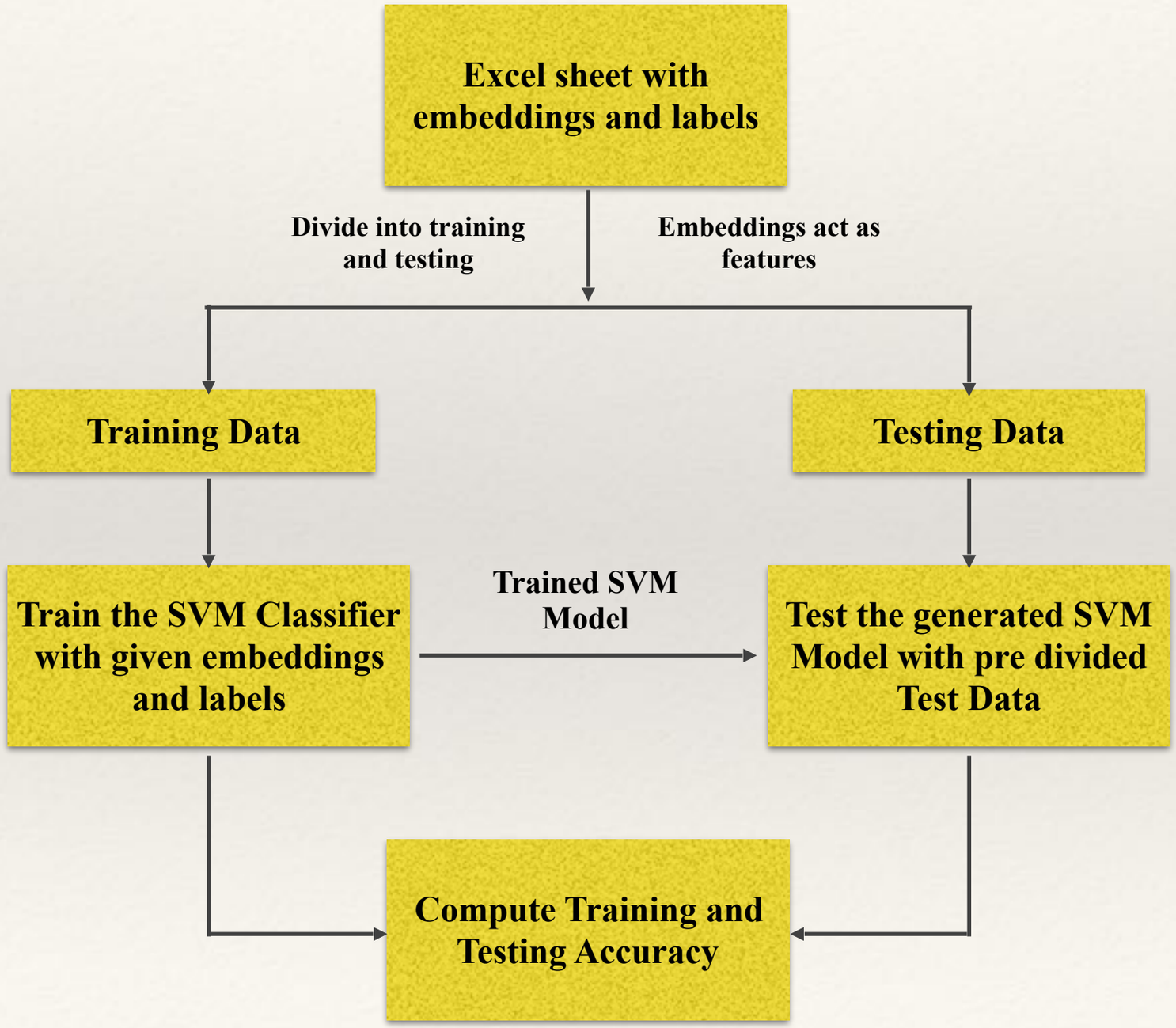
FLOW OF THE PROJECT

PART - I(Steps 1 - 3)



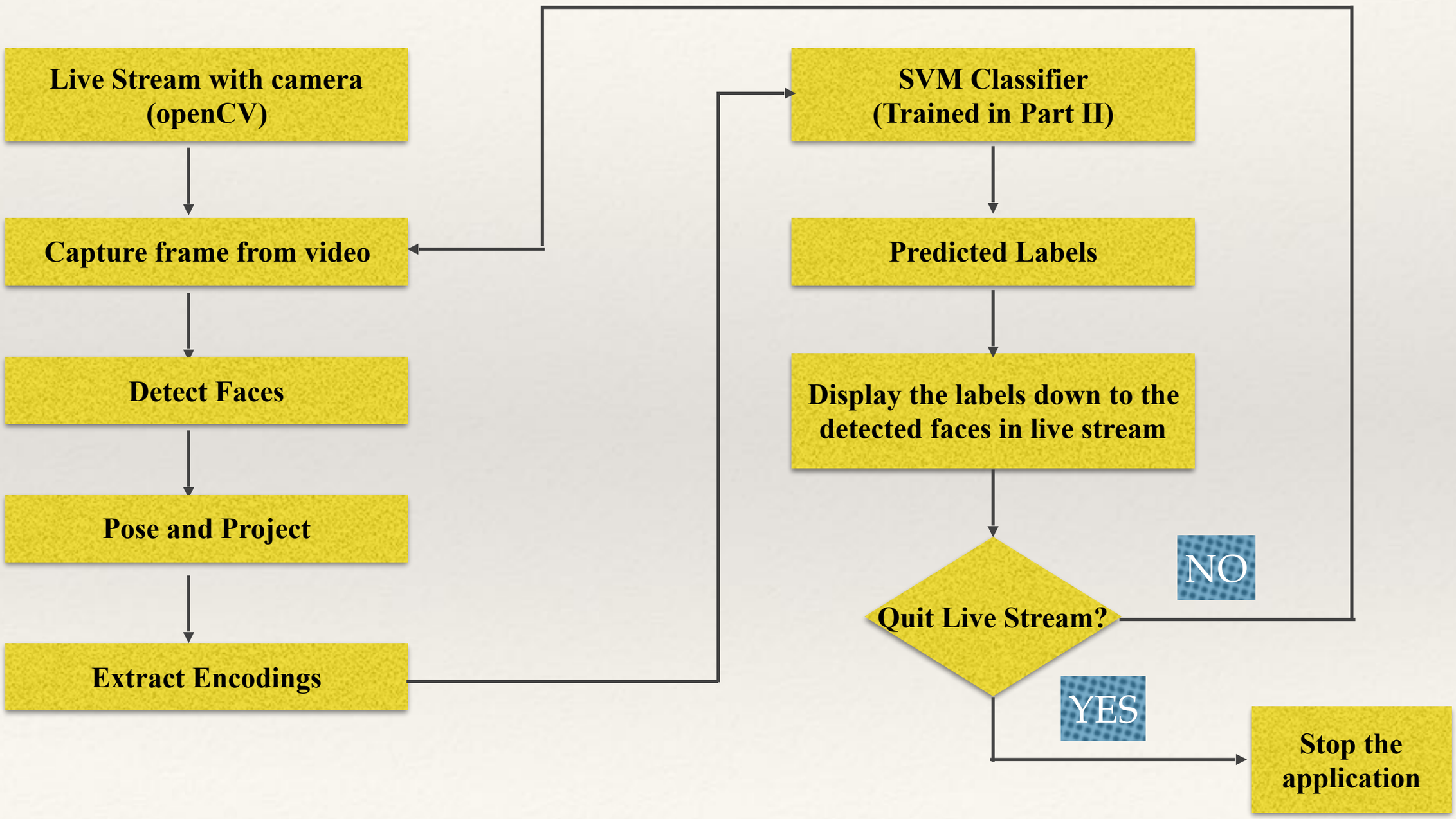
FLOW OF THE PROJECT

PART - II(Step 4)



FLOW OF THE PROJECT

PART - III(Step 5)



ACCURACY ASSESSMENT

Total Training Time:

No. of features : 7940

No. of classes : 397

Time : around 8 min

Accuracy (SVM):

Model Score

(Training Accuracy) : 0.9937

Testing Accuracy : 0.9904

CONCLUSION

- The project implements face recognition from live stream with laptop camera using machine learning and deep learning techniques.
- It takes frames from camera video and detects, centres the faces from which face encodings are extracted using a pre trained convolutional neural network.
- This project uses SVM classifier to get trained by the generated encodings to recognise faces from those encodings.
- Instead of SVM any other multi-class classifier can also be used, since our model accuracy is quite good with SVM we did not try any other classifier.
- The system reveals around 99% accuracy for the specified data set.


Thank
you