Voila Jones Algorithm (Face Detection)

→ Face detection has several application.

→

Algorithmic Flow

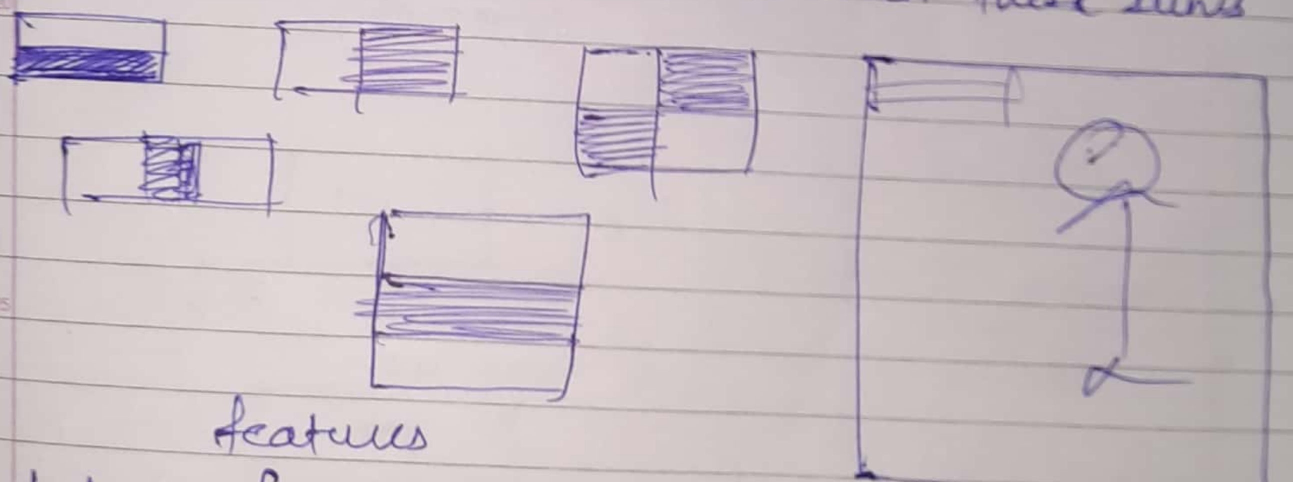
Input Image ⇒ Haar feature selection ⇒ Integral Image ⇒ Adaboost Training ⇓ Cascading Classifiers

Goal: Foal deatection not face recognition

Highly Robust.

Haar Features

- Adjacent rectangular regions at a specific location
- Sums up the pixel intensities in each region
- Calculate the difference between these sums



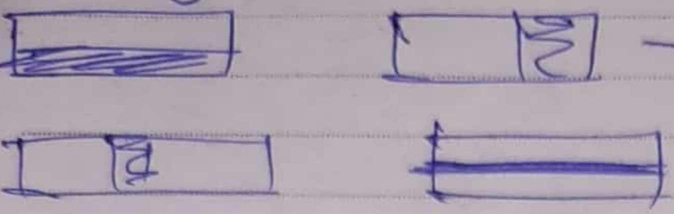
features

Integral feature

For each pixel add all the Intensities in the previous columns and row

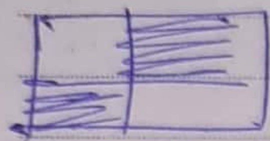
$$ii(x, y) = \sum_{x' \leq x,\ y' \leq y} i(x', y')$$

Boosting



→ Edge features

→ Line features



Four rectangle features

Standard

4960 facial data ⟷ 9544 Non-facial images

$F(x) = a_1 f_1(x) + a_2 f_2(x) + a_3 f(x)$

weights.

→ Not all features are created equal

→ Initializing each weight $w_i = \frac{1}{N}$

and then we can normalize weights

$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{N} w_{t,j}}$ → w_i is probability distribution

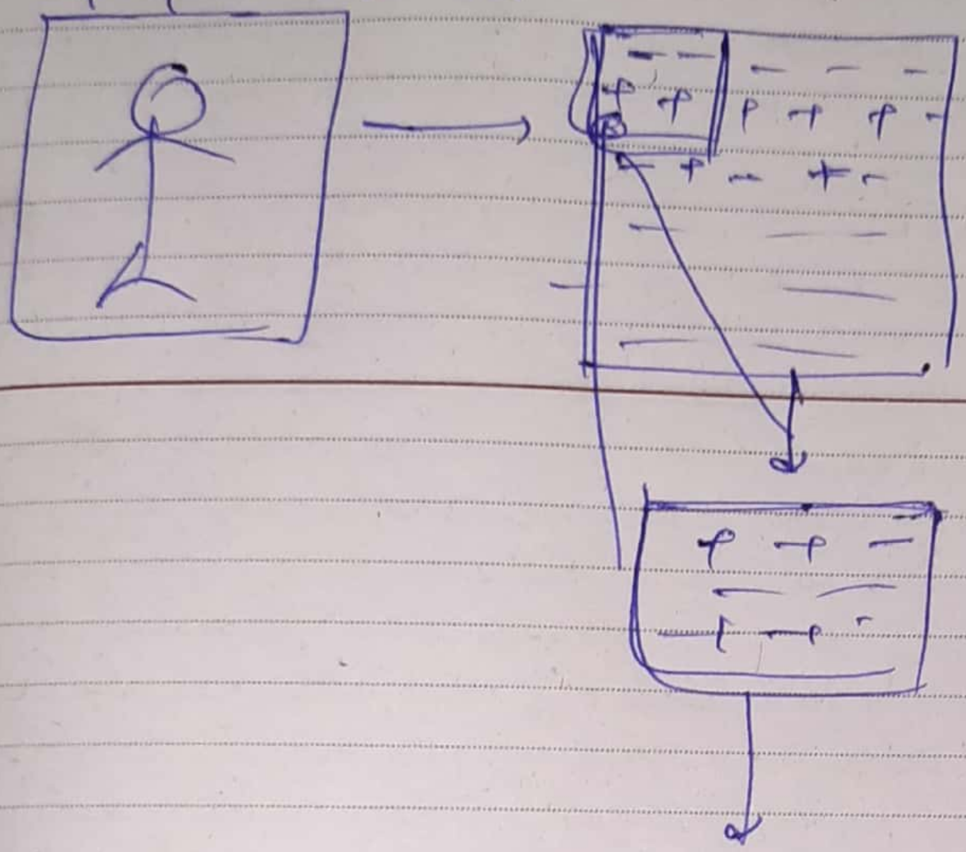
Cascading:

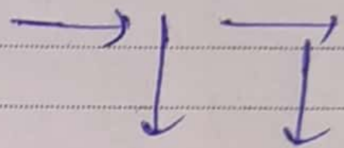
window → F1 $\xrightarrow{\text{Accept}}$ F2 $\xrightarrow{\text{Accept}}$

Reject ↓   ↓ Reject

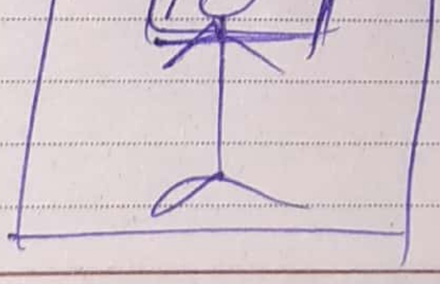
Trying all the features is time consuming but with cascading procedure get faster

Gray scale Image





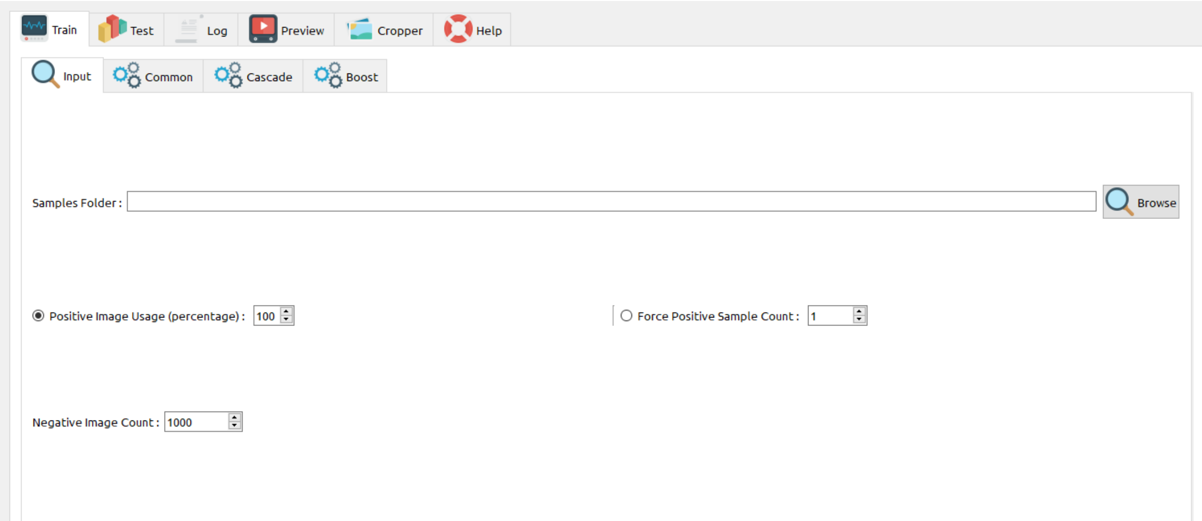
Face detected



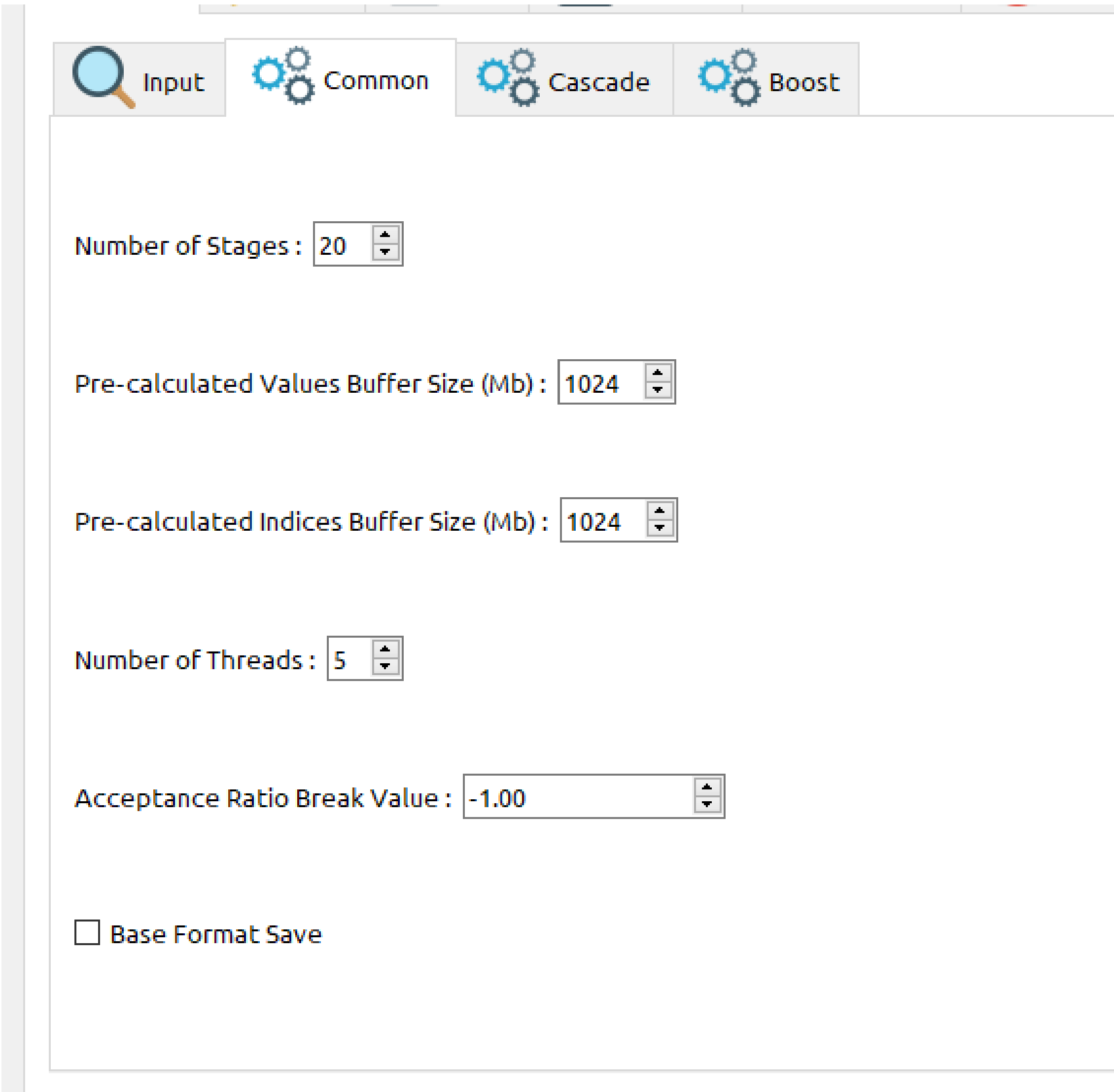
→ detection

For making cascade_final.xml document used following procedure used in viola jones algorithm :

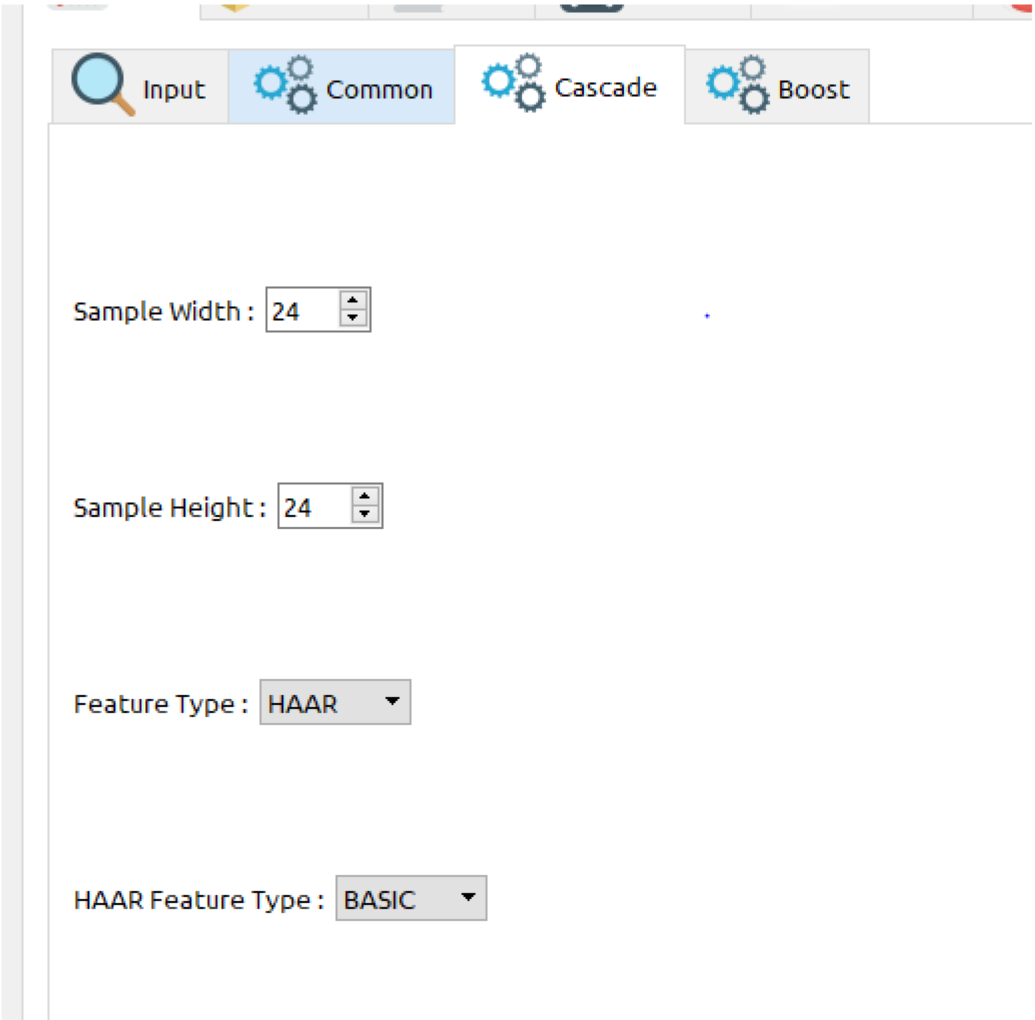
1) Created data set which had positive images (images which had the face) and negative images (images which don’t have faces)
2) After creating dataset, included the sample in below window,



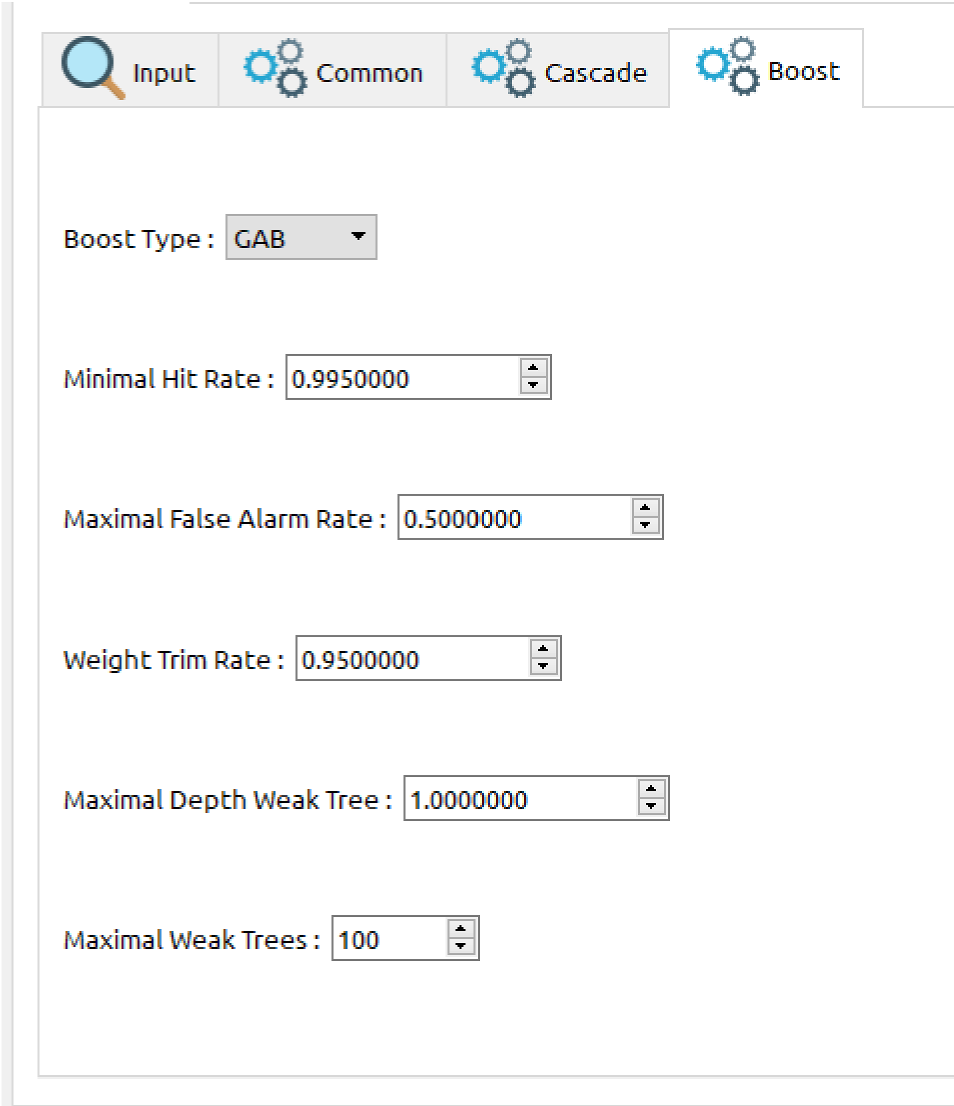
3) Selected some attributes



4) Now Cascaded the dataset in the following step in the software.



5) After filling the information in each steps



I have used dataset of having positive images value around 2000 images and negative images of around 4500 images, it required an overnight to make the final xml document.

Ran Viola – Jones Algorithm on the Avengers picture and result is as follows:

I further calculated the max_height and max_width for which the image can be cropped, actually resizing the image increases the accuracy in face recognition. But we can do the step that we crop the face only and hence it will further increase our model accuracy and also help us to reduce complexity of the model.

Credits for XML document : Cascade Trainer GUI

References: https://www.youtube.com/watch?v=uEJ71VlUmMQ

https://www.youtube.com/watch?v=LopYA64KmdE&t=602s

Camlin Page
Date / /

Mind Map

Detection → Classification of Faces

Integrating Face detection and Recognition system in real time.

① We will do face detection using OpenCV and will provide cropped image to the training of the recognition system.

Now in training Procedure;

Training

⇒ Suppose we have M images that are of dimension N×N, we first convert all images into vectors of dimension $N^2 \times 1$. Each of it is denoted by Γ_i

⇒ We next find 'mean face', i.e the mean of all the M vectors as Ψ

$$\Psi = \Sigma \Gamma_i$$

⇒ Now we will find offsets of image from the mean face by substracting every image as

$$\phi_i = \Gamma_i - \Psi$$

⇒ Next want to find the covariance matrix of A where

$$A = [\phi_1 \ \phi_2 \ \cdots \ \phi_M] \rightarrow \text{offsets of all the images.}$$

$$C = \frac{1}{M} \sum_{n=1}^{M} \phi_n \phi_n^T = AA^T \quad (N^2 \times N^2 \text{ matrix}).$$

where $A = [\phi_1 \ \phi_2 \ \cdots \ \phi_M]$ $(N^2 \times M \text{ matrix})$

⇒ There are procedures / methods here to find eigenvectors and eigen values of covariance matrix

Case 1 : when N is small

if is N is small, then for $N^2 \times N^2$ covariance matrix is not greater ~~....~~ and can be obtained in python

Case 2 : when N is large
$N \uparrow$ Computation power $\uparrow$
to calculate eignvector

Hence we calculate eigen vector of small covariance matrix

$$\boxed{C_S = A^T A \quad (M \times M) \text{ matrix.}}$$

Calculate eigen vector for this small matrix so to find eigen vector of covariance matrix. we can multiply that vector to A

$$\boxed{u_i = A v_i}$$

u_i : eigen vector of covariance matrix

v_i : eigen vector of small covariance matrix

⇒ Now next step is to calculate weight matrix

$$\hat{\phi}_i - \text{mean} = \sum_{j=1}^{k} w_j u_j \qquad (w_j = u_j^T \phi_j)$$

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \vdots \\ w_k^i \end{bmatrix} \qquad i = 1, 2, \ldots M$$

Ω_i represents the weight matrix of M train images.

⇒ Now, Instead of taking all the weights we have taken best K weights.

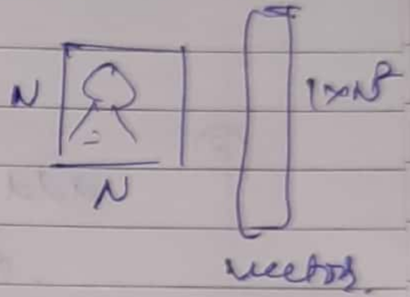
↳ Corresponding to that K weights we have eigen vectors.

⇒ Now, training ends, here for testing

Testing

⇒ ① Read and convert the image into vector.

⇒ Subtract the mean from the vector



$$\Phi_i = \Gamma_i - \Psi$$

get this offset

↳ Now from this offset calculate weight matrix

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \qquad (w_i = u_i^T \Phi)$$

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

⇒ Calculate minimum distance bet Ω_{Train} and Ω_{test}

$$e_r = \min \|\Omega - \Omega^{l}\|$$

e_r we will get which image is close to which set of person.

Defining $\boxed{\text{Accuracy} = \dfrac{\text{Correctly matched}}{\text{Total Test Images}} \times 100}$

Mind Map (Train)

①

N×N Image ⇒ vector

$N^2 \times 1$

⇓

② add them continoously in a loop and obtain the average Image (Ψ)

$$\phi_i = \Gamma_i - \Psi$$

$$A = [\phi_1 \ \phi_2 \ \text{---} \ \phi_M] \quad (N^2 \times M) \text{ Matrix}$$

③

$$C = \frac{1}{M} \sum_{n=1}^{M} \phi_n \phi_n^T = AA^T \quad (N^2 \times N^2) \text{ Matrix}$$

where $A = [\phi_1 \ \phi_2 \ \text{---} \ \phi_m]$ $(N^2 \times M)$ Matrix

Calculate eigenvectors and eigenvalues directly

$C_s = A^T A$ $(M \times M)$ Matrix

$U_i = A v_i$ → small covariance matrix

eigenvector

(4) $\hat{\phi}_i - \text{mean} = \sum_{j=1}^{k} w_j^i u_j \quad (w_j^i = u_j^T \phi_i)$

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \vdots \\ w_k^i \end{bmatrix} \qquad i = 1, 2, \ldots M$$

Now Test (Map)

Read the Image $\Rightarrow$ Convert into vector form $\Rightarrow$ Subtract Mean from flattened Image

$\phi_i = F_i - \psi$

$\Downarrow$

Calculated minimum distance (er) $e_r = \min \|\Omega - \Omega_i\|$ $\rightarrow$ finally get Accuracy of overall model.

→ Important Note :-

Here value of k → complexity of Model

In Machine Learning Models, we have to keep complexity and accuracy both in mind. For increase in some amount of accuracy we can't increase our complexity.