In any ML project in general, we go through a data pre-processing or cleaning step before feeding the required data to the ML model. While working with OpenCV, since the data is in the form of images, the objective of this step is to feed data that is easier to analyse and process by the ML model. It can be simple work like resizing the image or something like color transformation. When we open a image in openCV using cv2, BGR(Blue, Green, Red) color is the default color space to display images. Many times, color isn't necessary to recognize and interpret an image. Grayscale can be good enough for the same. Color images contain more information than black and white images and hence take up more space in memory. Grayscale reduces the number of pixels that need to be processed.

Pre-processing Steps :

1. Reading an image using img=cv2.imread().
2. Displaying the read image using cv2.imshow(' ', img)
3. Writing an image using cv2.imwrite
4. Changing colored image to grayscale
5. Histogram plotting
6. Edge detection using Canny Algorithm

Now, preprocessing in face recognition involves

a) Histogram Equalization(HE) and Histogram Specification(HS):
   Histogram Normalization is one of the most commonly used methods for preprocessing. In image processing, the idea of equalizing a histogram is to stretch and redistribute the original histogram using the entire range of discrete levels of the image, in a way that an enhancement of image contrast is achieved. The most commonly used histogram normalization technique is histogram equalization where one attempts to change the image histogram into a histogram that is constant for all brightness values.

b) LOG :
   LOG is another frequently used technique of gray Scale transform. It simulates the logarithmic sensitivity of the human eye to the light intensity.

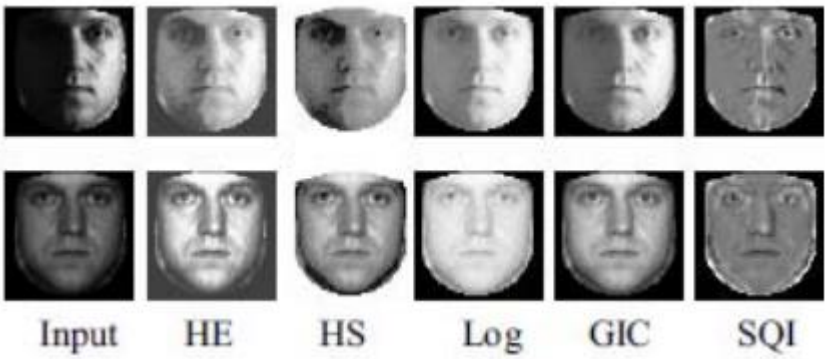
c) Gamma Intensity Correction(GIC) :
   The Gamma Intensity Correction (GIC) corrects the overall brightness of a face image to a pre-defined

canonical face image. Thus the effect of varying lighting is weakened.

d) SQI SQI is based on the reflectance-illumination model: I = RL, where I is the image, R is the reflectance of the scene and L is the lighting. The lighting L can be considered as the low frequency component of the image I and can be estimated by a low-pass filter F , i.e., L ~ F * I . Thus we can get the self-quotient image as :-

$$R = \frac{I}{F * I}$$

Examples :



The results show that HE,HS and GIC are better than the other two methods. However, the above example shows that the preprocessing approaches don't always work well on different datasets. Some approaches may also hurt the recognition of face images with normal lighting.

We can combine the strengths of gamma correction, DOG filtering and contrast equalization to utilize the net effect.

Classification

Classification is the process of recognising, understanding and grouping ideas and objects into preset categories. Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. In short, classification is a form of pattern recognition.

Different types of classification algorithms are –

1. Logistic Regression

Logistic regression is a calculation used to predict a binary outcome: either something happens, or does not. This can be exhibited as Yes/No, Pass/Fail, Alive/Dead, etc. Independent variables are analyzed to determine the binary outcome with the results falling into one of two categories. The independent variables can be categorical or numeric, but the dependent variable is always categorical.

Written like this:

$$P(Y=1 \mid X) \text{ or } P(Y=0 \mid X)$$

It calculates the probability of dependent variable *Y*, given independent variable *X*.

This can be used to calculate the probability of a word having a positive or negative connotation (0, 1, or on a scale between). Or it can be used to determine the object contained in a photo (tree, flower, grass, etc.), with each object given a probability between 0 and 1.

2. Support Vector Machines(SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called

as support vectors, and hence algorithm is termed as Support Vector Machine.

TYPES OF SVM :

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

3. Convolutional neural networks (CNN)

We use filters when using CNNs. Filters exist of many different types according to their purpose. Filters help us exploit the spatial locality of a particular image by

enforcing a local connectivity pattern between neurons.

Convolution basically means a pointwise multiplication of two functions to produce
a third function. Here one function is our image pixels matrix and another is our filter. We slide the filter over the image and get the dot product of the two matrices. The resulting matrix is called an “Activation Map” or “Feature Map”.

4. Local Binary Patterns Histograms(LBPH) :

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.

Steps involved in LBPH :

1. **Parameters**: the LBPH uses 4 parameters:

- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the

higher the computational cost. It is usually set to 8.

- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the

algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids.

5. Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **euclidean distance**, **chi-square**, **absolute value**, etc. In this example, we can use the Euclidean distance

(which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

The Uniform LBP formula is :

$$LBP(gp_x, gp_y) \sum_{p=0}^{P-1} S(gp - gc) \times 2^p$$

gc- *the intensity value of the central pixel*

gp- *the intensity of the neighboring pixel with index p*

the function S can be expressed as:

$$s(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & if \quad x < 0. \end{cases}$$

P- *number of sampling points on a circle of radius R(circular neighborhood).*

P- *controls the quantization of the method.*

R- *determines the spatial resolution of the method or operator.*

The gray values of neighbors which do not fall exactly in the center of a pixel(block) are estimated by interpolation.

5. Haar Cascade Algorithm

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features.

Calculating Haar Features

The first step is to collect the Haar features. A **Haar feature** is essentially calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums.

These features can be difficult to determine for a large image. This is where **integral images** come into play because the number of operations is reduced using the integral image.

Creating Integral Images

Integral images essentially speed up the calculation of these Haar features. Instead of computing at every pixel, it instead creates sub-rectangles and creates array references for each of those sub-rectangles. These are then used to compute the Haar features.

Adaboost Training

Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination

of “weak classifiers” to create a “strong classifier” that the algorithm can use to detect objects.

Conclusion

From all the above mentioned algorithms and many more, LBPH is the one of the easiest face recognition algorithms. It can represent local features in the images. It is possible to get great results(mainly in a controlled environment). LBPH is robust against monotonic gray scale transformations.