ML Task 3: Synopsis

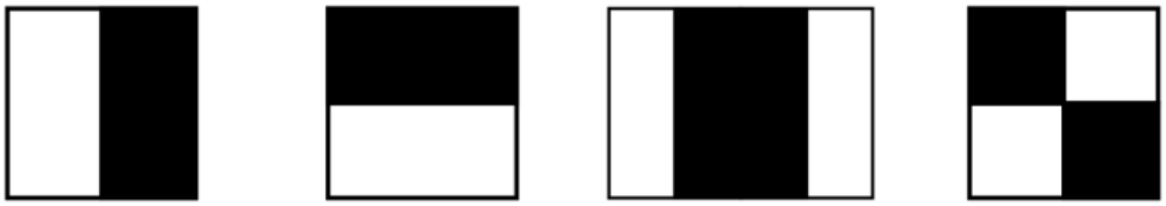
Preprocessing

- Grayscale conversion: To reduce the given dataset of images (usually) in RGB to grayscale to reduce the number of pixels. This reduces computational complexity and helps in reducing the time taken to run the model. However, OpenCV interprets images as BGR, so in the code executed to complete this task, RGB has simply been converted to BGR.
- Standardization: Images may be of a different size. Without scaling them to a size that can fit to screen, the images can cut off by a window that is too small in size or they can be too small to be properly seen. Hence, scaling is done with a requisite scale on the frame that can eliminate these issues. This is termed standardization. In the accompanying executed code, the images are quite large in size, so they are scaled down appropriately.
- Data Augmentation: These are minor alterations performed on a frame to prevent the neural network from learning irrelevant features about that frame. Some of these include:
    1. Horizontal or Vertical Flipping: It interchanges the axes (rotates the image by 90^o).
    2. Shifting: Pixels can be shifted horizontally or vertically.
    3. Cropping: Used to focus on a particular feature of a frame.
    4. Smoothing: This is used to blur an image, which reduces the noise in a frame, allowing the neural network to focus on what the programmer wants it to focus on. Here, Gaussian Blur is used to smooth the input image.

Modelling Techniques

In FER library, two techniques can be used to detect emotions. They are:

- Haar Feature Modelling: An image is a matrix of numbers to a computer. These numbers range from 0 to 255, where 0 represents black and 255 represents white. These numbers are called pixels. Haar features are manually determined, specially designed matrices that are multiplied across the matrix of numbers of image, emphasizing some features and smoothing others. Some examples of Haar features are:



And their 3 x 3 matrix representations respectively are:

-1	-1	5
-1	-1	5
-1	-1	5

5	5	5
-1	-1	-1
-1	-1	-1

-1	5	-1
-1	5	-1
-1	5	-1

5	-1	-1
-1	5	-1
-1	-1	5

The first two Haar features are “edge features”, which are used to detect edges. The third is a “line feature”, and the fourth is a “four rectangle feature”, used to detect slanted lines.

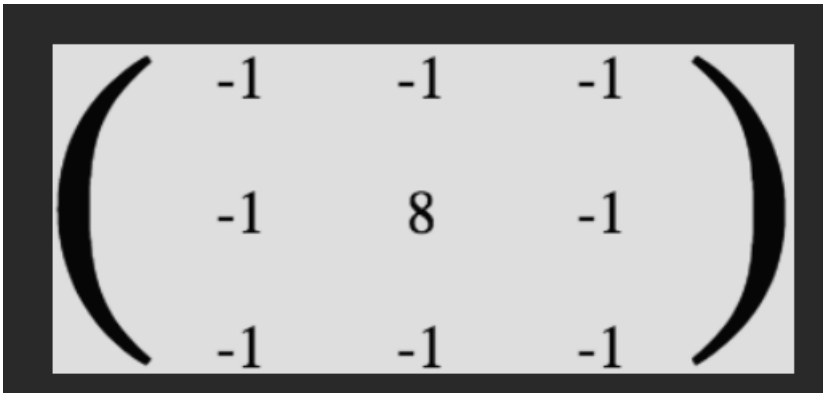
The large matrix of pixels (say, 64 x 64) are divided into smaller 9 x 9 matrices, and the Haar features move across the image, multiplying themselves to these matrices and yielding a result. The numerical value of this result explains how the now “processed” image will look.

For example, an eye can be found out by the line feature (shown below), which is dark on top and light otherwise.

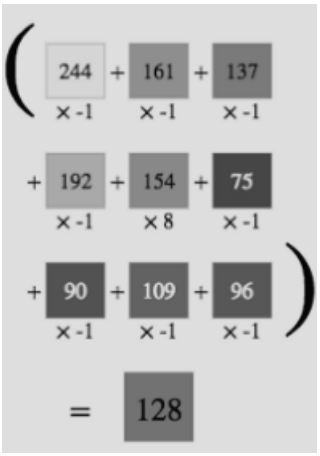
These Haar features are manually introduced to the neural network, so we basically create a classifier with a relatively small dataset. This allows us to train the computer well without many face datasets. This also helps in reducing computations, thus giving a quicker execution speed.

- MTCNN Modelling: Multi Task Cascaded Convolution Neural Networks (MTCNN) is one of the most effective ways used for face analysis in frames. Convolution Neural Networks (CNN) consist of an input neuron layer, several hidden layers in between and an output neuron layer. We Each layer of neuron has a particular purpose: one layer determines the outline of an image, another one determines certain shapes or features, and finally the last layer gives us the output (in the executed code accompanying this synopsis, the output layer gives the emotions of a person).

  So, in simplified mathematical terms, in each layer a kernel is used to manipulate the image. A kernel is a 3 x 3 (or any other ordered) matrix that moves across the matrix of image (9 pixels at a time) and via scalar multiplication modifies it (as it did in Haar features). For example, lets say our convolution kernel is:



If this matrix is multiplied to 3 x 3 matrix of 9 pixels, it will look like this:



A new value is obtained, and this value is assigned to that layer's matrix of numbers for the image. This scalar multiplication continues until a full output image is created for that layer. This happens at each neural layer of the CNN. For our code, in the final output layer, the transformations applied are such that we get 7 probabilities as output, one for each emotion recognized in the FER library.

The difference between Haar classifier and CNN is that CNN uses training to determine the values in a kernel (matrix), whereas Haar classifier uses manually determined set of values.

Conclusion

In the executed code, MTCNN model has been used because it has a higher degree of freedom, meaning it can also detect partially covered faces and their emotions depending on the quality of training datasets. It can also detect a large variety of faces, which is useful in face analysis. Haar classifiers, while not completely obsolete, only function well if the input image has faces that are fully frontal without any coverings. Faulty outputs can be obtained if faces are tilted, covered etc., as only features with clear lines and edges could be determined due to the nature of manually determined kernels being used. However, in CNN as the machine decides the value

of kernels, the CNN learns more parameters and thus is simply a better judge of human emotions in a frame.