**Facial Recognition and Emoji Generation**

1. **Image Pre-processing**

Since images exist in different formats, i.e., natural, fake, grayscale, etc., we need to take into consideration and standardize them before feeding them into a neural network.

1. **Grayscale conversion**

Grayscale is simply converting images from colored to black and white. It is normally used to reduce computation complexity in machine learning algorithms.

Since most pictures don’t need color to be recognized, it is wise to use grayscale, which reduces the number of pixels in an image, thus, reducing the computations required.

1. **Normalization**

Also referred to as data re-scaling, it is the process of projecting image data pixels (intensity) to a predefined range (usually (0,1) or (-1, 1)). This is commonly used on different data formats, and you want to normalize all of them to apply the same algorithms over them.

Normalization is usually applied to convert an image’s pixel values to a typical or more familiar sense.

Its benefits include:

* Fairness across all images - For example, scaling all images to an equal range of [0,1] or [-1,1] allows all images to contribute equally to the total loss rather than when other images have high and low pixels ranges give strong and weak loss, respectively.
* Provides a standard learning rate - Since high pixel images require a low learning rate and low pixel images high learning rate, re-scaling helps provide a standard learning rate for all images.

1. **Data Augmentation**

Data augmentation is the process of making minor alterations to existing data to increase its diversity without collecting new data.

It is a technique used for enlarging a dataset. Standard data augmentation techniques include horizontal & vertical flipping, rotation, cropping, shearing, etc.

Performing data augmentation helps in preventing a neural network from learning irrelevant features. This results in better model performance.

Standard data augmentation techniques include horizontal & vertical flipping, rotation, cropping, shearing, etc.

There are two types of augmentation:

* Offline augmentation - Used for small datasets. It is applied in the data pre-processing step. We will be covering this augmentation in this tutorial.
* Online augmentation- Used for large datasets. It is normally applied in real-time.

Data Augmentation also includes:

* Flipping: This reverses the rows or columns of pixels in either vertical or horizontal cases, respectively.
* Rotation: This process involves rotating an image by a specified degree.
* Changing Brightness: This is the process of increasing or decreasing image contrast.
* Cropping: This is the process of creating a random subset of an original image which is then resized to the size of the original image.
* Scaling: An image can be scaled either inward or outward. When scaling an image outward, the image becomes more significant than the original and vice versa.

1. **Standardizing Images**

**Standardization** is a method that scales and pre-processes images to have similar heights and widths. It re-scales data to have a standard deviation of 1 (unit variance) and a mean of 0.

Standardization helps to improve the quality and consistency of data.

1. **Techniques for Classification**
2. **Supervised ML**
   * **Linear Regression:** Linear Regression is not suitable for classification as Linear regression deals with continuous values and classification deals with discrete values.
   * **Ridge and Lasso Regression:** Ridge Regression is used to get a generalized model of low bias and low fitting (Prevent Overfitting). **Cost Function = (yi-y)2 + a(slope)2**. Lasso Regression is used for feature Selection.

**Cost Function = (yi-y)2 + a|slope|**

* + **Logistic Regression:** Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).
  + **KNN Algorithm (K Nearest Neighbour):**  It classifies the data point on how its neighbour is classified. It uses the Eucledian Distance by using the standard distance formula, or the Manhattan Distance by **|(x2-x2) + (y2-y1)|**
  + **Decision Tree:** DecisionTree classifies the input data set into predefined classes. The purity is defined using Purity Split, Entropy and Gini Impurity. The features are selected using Information Gain.
  + **Random Forest Classifier:** It gives equal opportunity to all the features by selecting randomly. In Decision tree, there is low bias and high variance, this high variance is converted to low variance using Random Forest. Random Forest is not impacted by outliers. This algorithm is increasingly being applied to satellite and aerial image classification.
  + **Adaboost:** Boosting is a sequential combination of models (many weak learners sequentially combined to form a strong learner). Adaboost gives weight to all categories (overall=1). Divide new weight by sum of new weight to get normalised weight. This classification method is used for brain tumour type.
  + **XgBoost Classifier:** CNN is used to obtain features from the input, and XGBoost as a recognizer produces results to provide more accurate output. **Similarity weight = sum(Residual)2/sum[Pr(1-Pr)] + a**

1. **Unsupervised ML**
   * **K Means Clustering:** The algorithm aims to minimize the squared Euclidean distances between the observation and the centroid of cluster to which it belongs. We initialize k number of centroids. The average is computed to all centroids. All centroids should be very far.
   * **Hierarchical Clustering:** You need to find longest vertical line that has no horizontal line passing through it. It takes a lot of time and is thus suitable only for a small dataset.
   * **Validate Clustering Models (Silhouette Clustering and DB Scan Clustering):** These cannot be used for image classification.
2. **Convolutional Neural Networks**

* When we see an image, we automatically divide it into many small sub-images and analyse them one by one. By assembling these sub-images, we process and interpret the image. How can this principle be implemented in a Convolutional Neural Network?
* The work happens in the so-called **convolution layer**. To do this, we define a filter that determines how large the partial images we are looking at should be, and a step length that decides how many pixels we continue between calculations, i.e. how close the partial images are to each other. By taking this step, we have greatly reduced the dimensionality of the image.
* The next step is the **pooling layer**. From a purely computational point of view, the same thing happens here as in the convolution layer, with the difference that we only take either the average or maximum value from the result, depending on the application. This preserves small features in a few pixels that are crucial for the task solution.
* Finally, there is a **fully-connected layer**, as we already know it from the normal neural networks. Now that we have greatly reduced the dimensions of the image, we can use the tightly meshed layers. Here, the individual sub-images are linked again in order to recognize the connections and to carry out the classification.

**Code:**

The pre-processing steps for this task include installing an IDE (Jupyter Notebook or VSC preferred). Certain libraries like **DeepFace** and **Opencv** should be installed in the IDE. DeepFace is used for deep learning models implementation and Opencv is used for performing Computer Vision.

* **Matplotlib** is imported so that the image can be plotted appropriately.
* The image is assigned to the variable img using the imread() function.
* The image is initially in BGR and is thus converted to RGB.
* With the help of DeepFace.analyse(img), we can get the details about dominant emotion, gender, age, etc.
* One can then see the percentage of all the emoji’s in that photo by indexing the predictions’ emotions.
* The output is given in the form of a dictionary, thus we can get the dominant emotion using that index.
* In order to draw a rectangle across the face, we can use CascadeClassifier, where we need to input an XML file.
* The XML file is a haarcascade\_frontal\_default file which contains all the face recognition algorithms.
* The image is then grayscaled. The advantage of this is that it is the only partition of the plane that is not removed. The value of each pixel is related to the number of bits of data used to represent the pixel.
* In order to make a rectangle,

*for (x, y, w, h) in faces:*

*cv2.rectangle(img, (x, y), (x+w, y+h), (0,255,0), 2)*

Here (x,y) are the coordinates and (w,h) are the width and the height repectively. The color of the rectangle so formed is green using the BGR color plate and the box-sizing is 2.

* Show the image with the imshow() keyword. A rectangle will be displayed across the face.
* For the text around the rectangle, one can select font specifications according to their choice.
* putText() function is used to put text over an existing image. The dominant emotion is displayed in the form of text.
* One can give specific colors for positive (green), negative (red) or neutral (yellow) emotions.
* We can put this entire process under a while loop, so that we do not need to code it multiple times for multiple images.
* One can import the **Keras** (used for data augmentation) library to convert these dominant emotions into relatable emojis.

1. **Why this Algorithm?**

This Algorithm was selected by me because I have already taken part in a **Multicon Event** organized by Thakur College of Engineering, where we learnt about Image Processing – **Cropping, Blurring, Image Classification Methods and MNIST datasets**. I Found it fascinating and would like to get more knowledge for the same.