# Dataset

The dataset is a collection of images of different facial expressions that depict the following emotions:

* Anger
* Disgusted
* Fearful
* Happy
* Neutral
* Sad
* Surprised

The dataset is taken from Kaggle. The dataset already divided into a train and test set during the time of download. The number of images for each emotion in the train and test dataset are inconsistent. The images in the dataset are of the size 48x48 i.e., 2304 pixels per image. Since there are three 3D matrices associated with each image one of each Red, Green, Blue (RGB), to make it easier to manage the dataset and to train the algorithm faster and more efficiently, the images are converted into grayscale. The reduces the 3 2D matrices for each image to one. Then we convert the grayscale image into a Numpy array and each pixel of the image is treated as a separate element of the array. Since, each element of the array is treated as a separate feature, therefore, there are 2304 features for each instance in the dataset along with the 2305th feature which is the label.

This dataset already comes with a separated folder for each training and testing dataset. The number of images for each emotion in both the training and test sets are as followed:

Training Set

* Anger: 3995 images
* Disgusted: 436
* Fearful: 4097
* Happy: 7215
* Neutral: 4965
* Sad: 4830
* Surprised: 3171

Test Set

* Anger: 958
* Disgusted: 111
* Fearful: 1024
* Happy: 1774
* Neutral: 1233
* Sad: 1247
* Surprised: 831

# Machine Learning

The objective of this project is to create a web app that can capture the facial expression of a person based on a stimulus provided by a slideshow on the web app. The images captured will then be forwarded to the algorithm where it will classify the image and save it in the local directory.

The project will have two modules; the frontend module that will be made using flask and jinja, and the machine learning module made using a Convolutional Neural Network from PyTorch. This document will only cover the machine learning aspect of the project as I will handle the software aspect myself.

The backend of the project consists of a Convolutional Neural Network created using the Keras ML library for Python. The algorithm is specifically chosen for this research because the dataset for this research is of the image format and CNNs are very popular for image classification. The CNN used in this code has the following configuration:

Input Layer: 32 Nodes with ReLU activation function and input shape of (48, 48, 1).

Hidden Layer 1: 64 Nodes with ReLU

Hidden Layer 2: 128 nodes with ReLU

Hidden Layer 3, 4: 512 nodes each with ReLU and Regularization

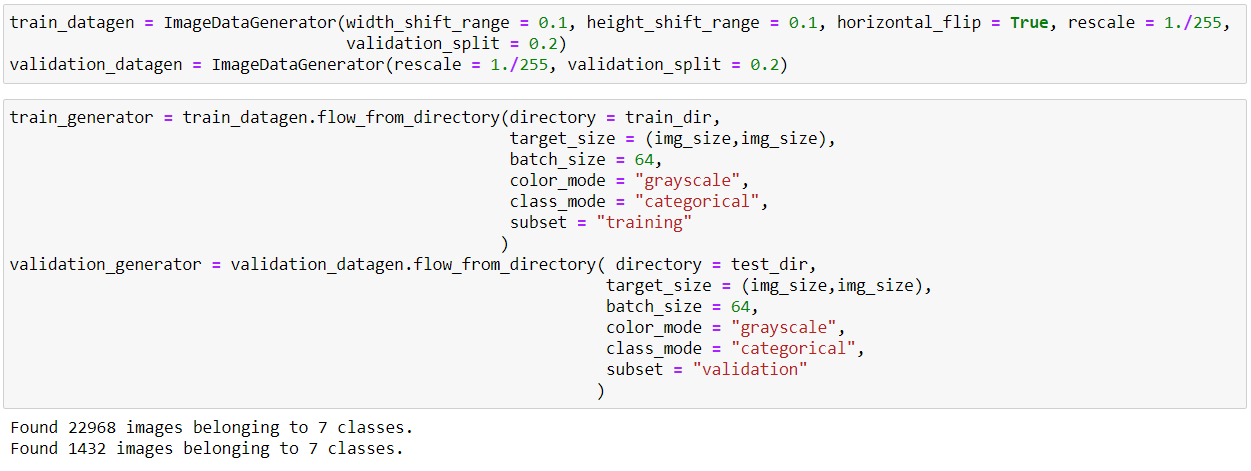
Hidden Layer 5: Meant for Flattening the input to the next Hidden layer since the next hidden layer is an ANN not a CNN. It has 256 nodes with ReLU

Hidden Layer 7: ANN 512 Nodes with ReLU

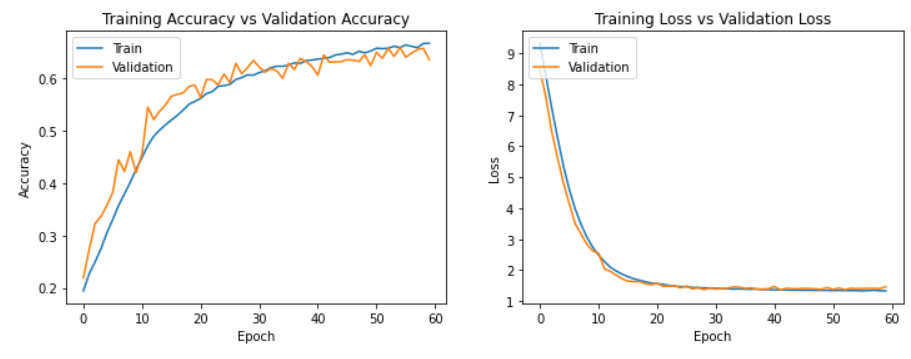
Output layer: 7 nodes with Softmax

The algorithm runs for 60 epochs. The batch size of images for each layer is a fixed 64. The kernel size for only the 3 layer is a 5x5 matrix. Rest all the other CNN layers have the 3x3 matrix as the kernel. The model is also equipped with the Adam Optimizer.

After the training for the CNN is complete, the training accuracy is compared with the validation accuracy. The following code snippet shows the processing of the training and the validation datasets for the algorithm.



The validation split for both the datasets in the ImageDataGenerator function from Keras is 0.2. The Training loss is also plotted against Validation loss for the algorithm. The performance graphs are as follows:



The x-axis represents the number of epochs in both the graphs. As you can see, there is a higher degree of fluctuation in the validation accuracy graph when compared to the loss graph. The decrease in loss is consistent for both the training and validation sets although there is a bit of fluctuation for validation loss while training loss is completely smooth. The same goes for the accuracy graph as well. These fluctuations can be the result of less data in the validation set. Lesser data in the validation set has brought about greater magnification in pattern identification which is seen as greater fluctuation in the graph. I have also employed the inbuilt evaluator function for performance measurement in percentages. The following code snippet reveals that.

