**MOHAMMED BHADSORAWALA**

**FACIAL EMOTION DETECTION MODEL**

This project aims to create a **Machine Learning** model based on Deep Learning and **Convolutional Neural Networks** to detect emotions from images of human faces

Hence this project also involves use of image processing libraries like **OpenCV** and other helper libraries as **Matplotlib,Numpys,Pandas**

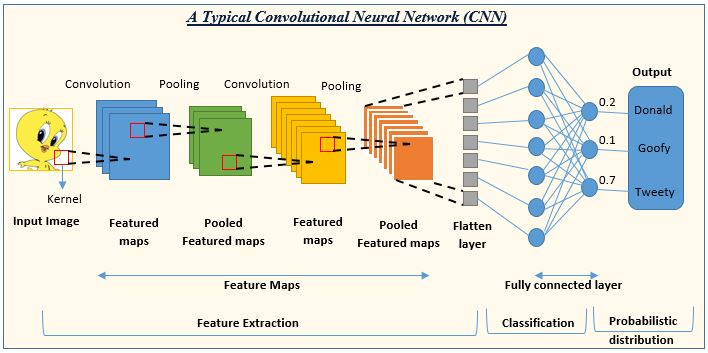
I used **FER 2013** dataset for using grayscale 48x48 images for training and testing the model.The dataset consists of approximately 35,000 images and hence is small for training a model based on human gesture.Hence accuracy of this model wasn't State of the Art but it still provides satisfactory results and learnings about several DL domains

Another reason was low resolution of images,as 48x48 pixels cant capture too many features for sharp distinction between several emotions.

**CNN:**

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

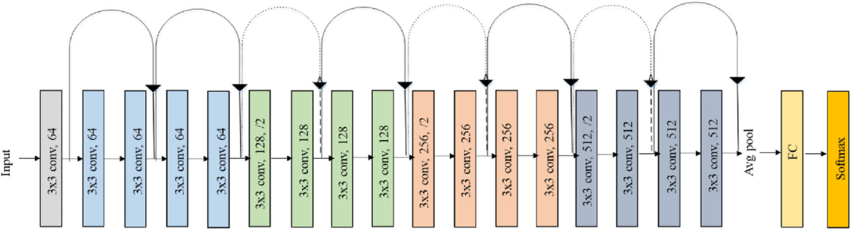
The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.



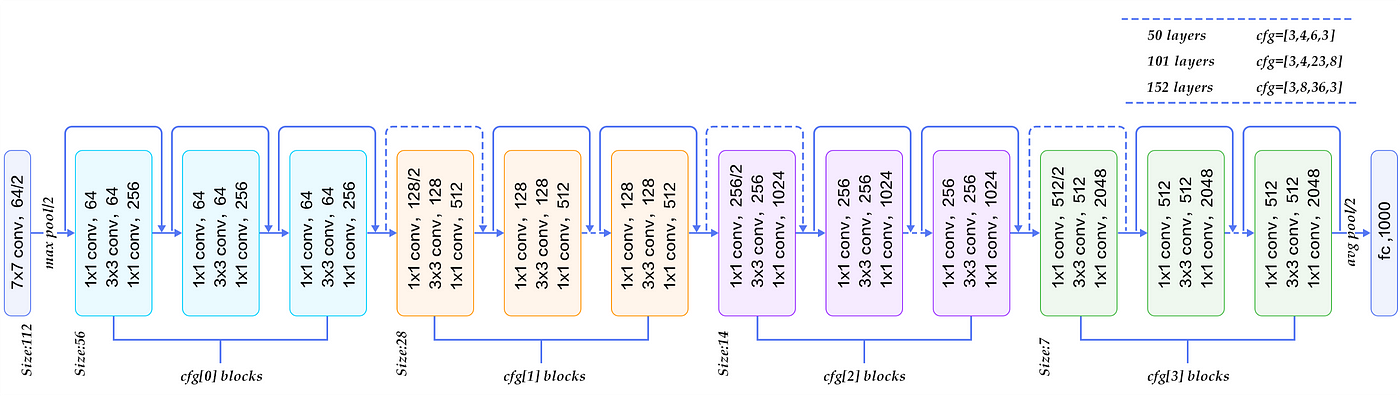
Several CNN architectures are available today,I used 2 of them namely **ResNet 18,ResNet 50**

A Residual Neural Network (a.k.a. Residual Network, ResNet) is a deep learning model in which the weight layers learn residual functions with reference to the layer inputs. A Residual Network is a network with skip connections that perform identity mappings, merged with the layer outputs by addition. It behaves like a [Highway Network](https://en.wikipedia.org/wiki/Highway_network) whose gates are opened through strongly positive bias weights. This enables deep learning models with tens or hundreds of layers to train easily and approach better accuracy when going deeper.

**RESNET 18 ARCHITECTURE**

****

**RESNET 50 ARCHITECTURE**

****

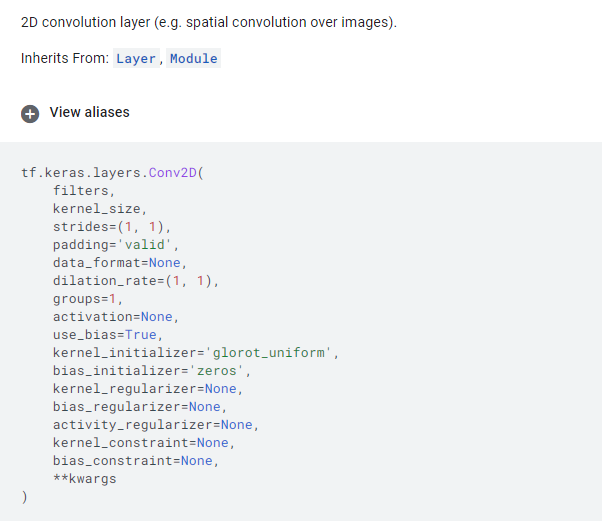
I preprocessed images into numpy arrays in terms of intensity of their grayscale values and fed such arrays into these models to generate the emotion as output.

For output i used **ONE HOT ENCODING** vectors as this is a multi class classification problem

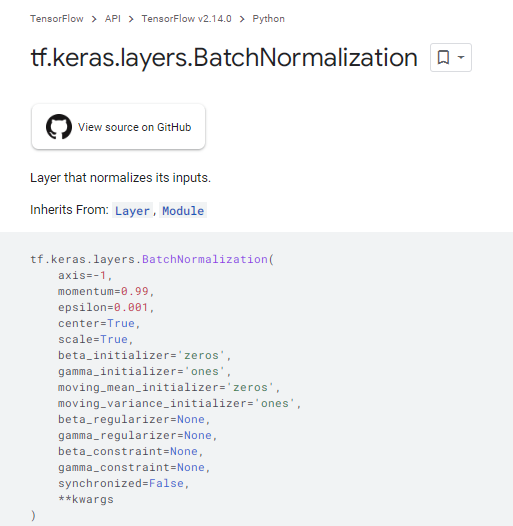
I used **Tensorflow,**a deep learning framework which consists of several layers and functions which need not be implemented from scratch

Also TF also care of backpropagation to update the weights in all layers so we just need to define the forward propagation and input and outputs for the model rest is taken care by the framework.

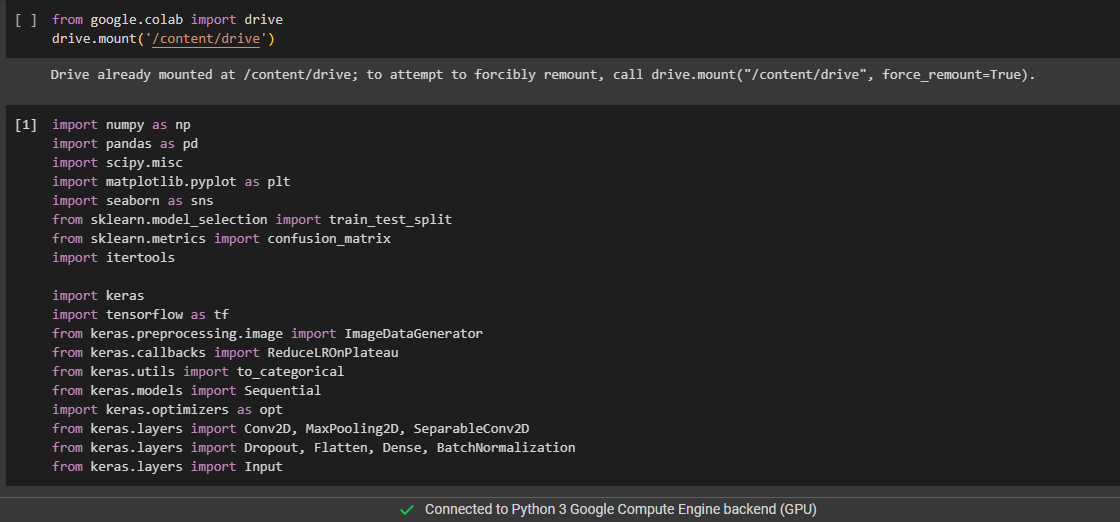
For main convolution, i used **tf.keras.layers.Conv2D**

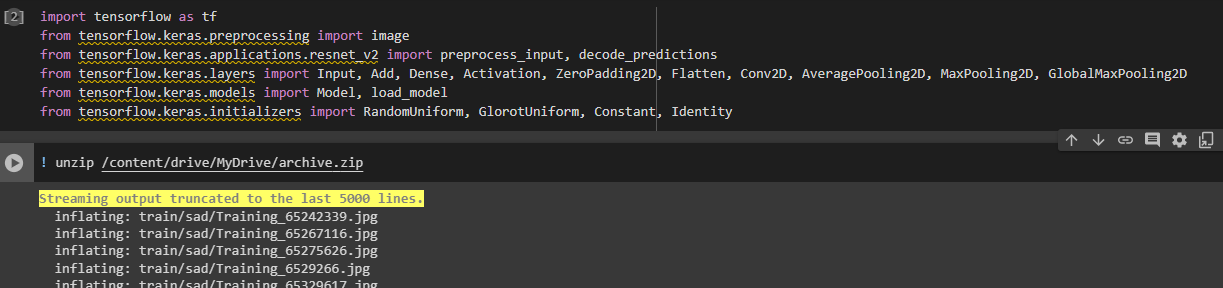
****

Another commonly used layer is **BatchNormalization,**a method used to make training of artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling.

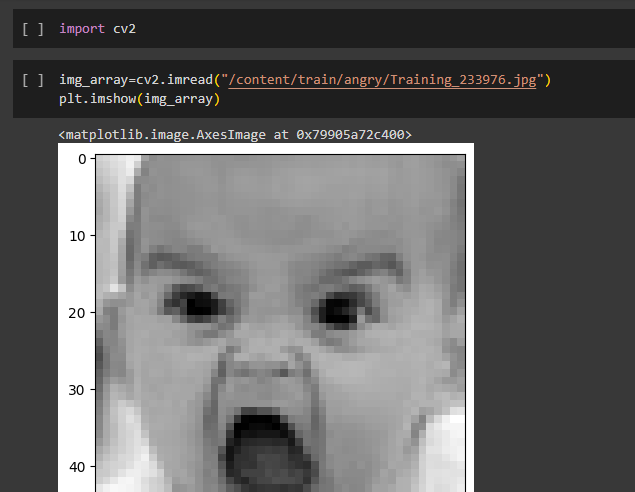


**CODE:**

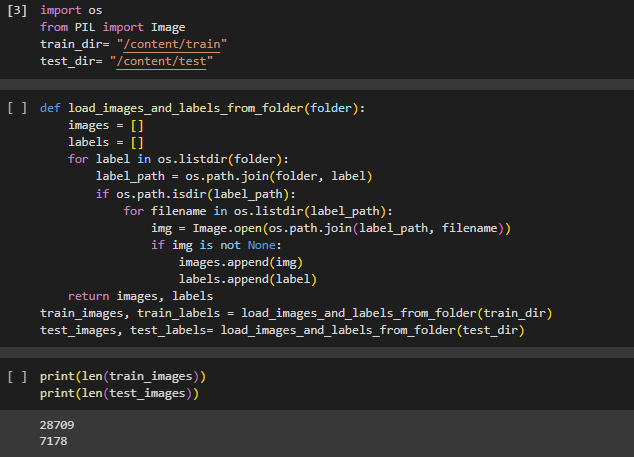
****

****

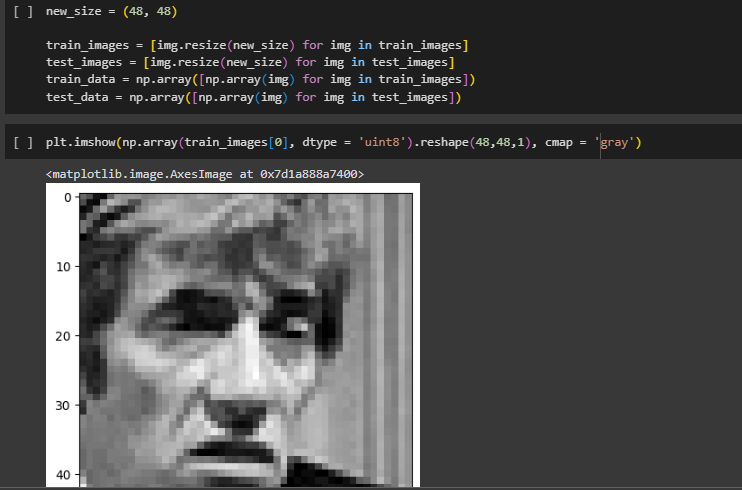
Importing necessary libraries and functions

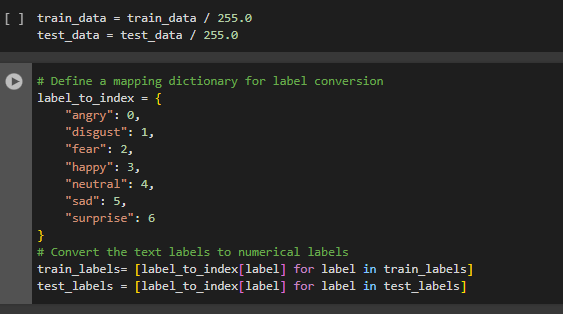
****

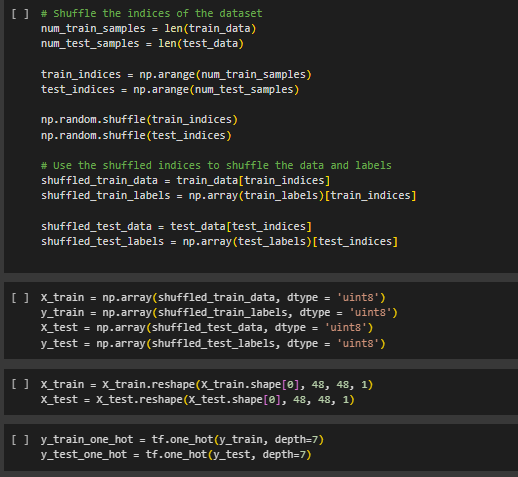
Displaying images using OpenCV

****

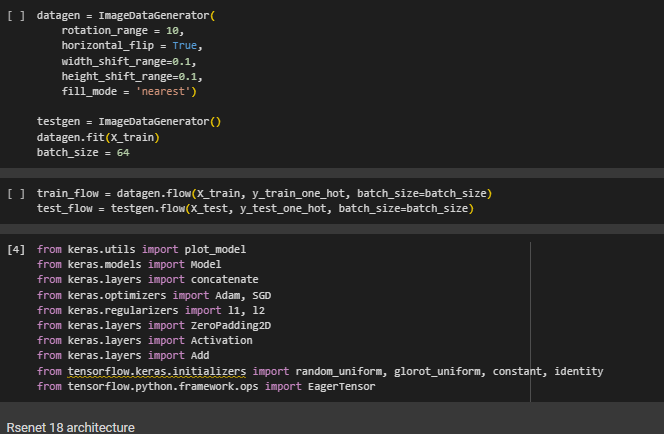
Converting images into numpy arrays to feed into the model

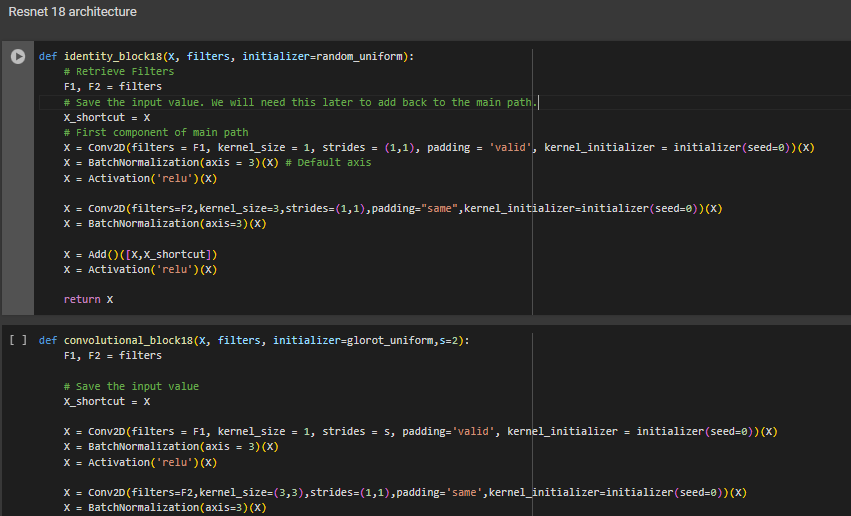
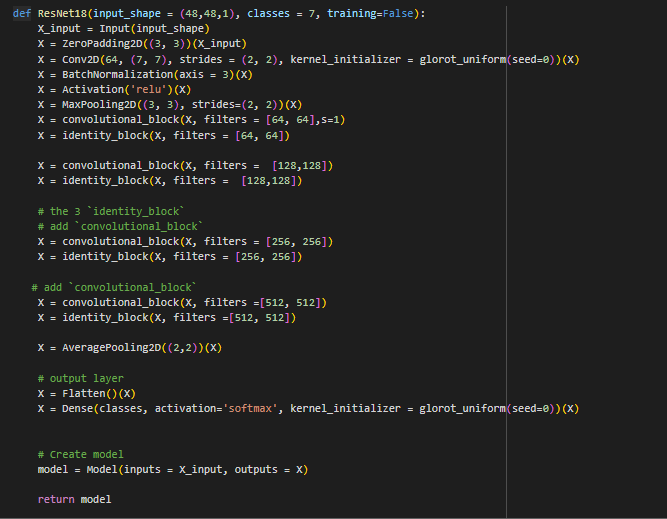
****

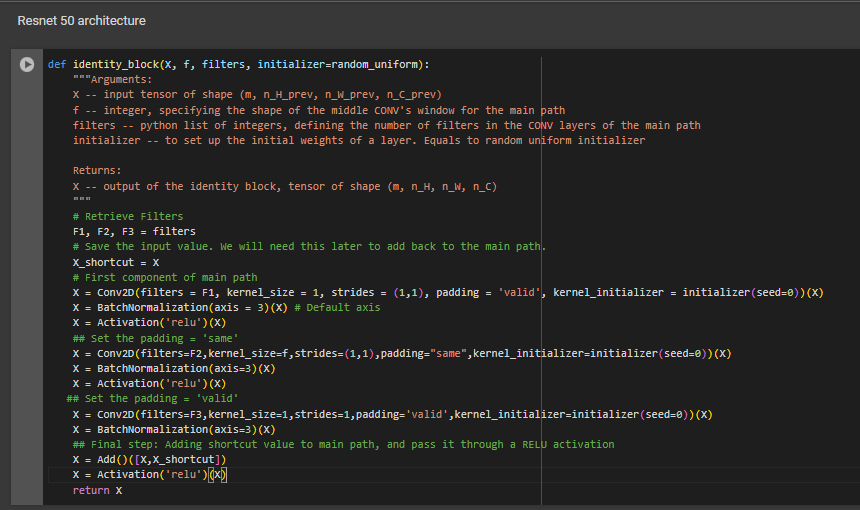
****

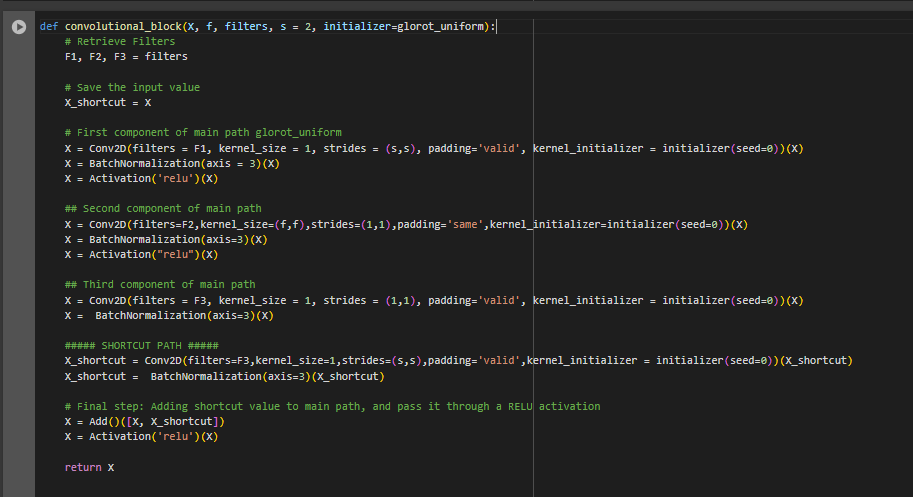
****

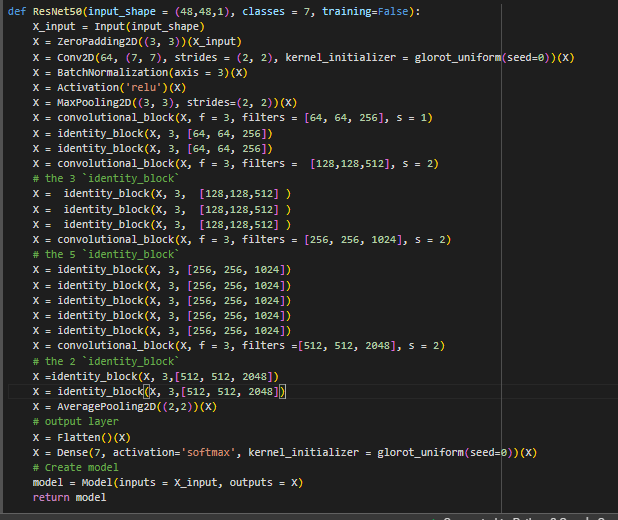
Preparing dataset for training

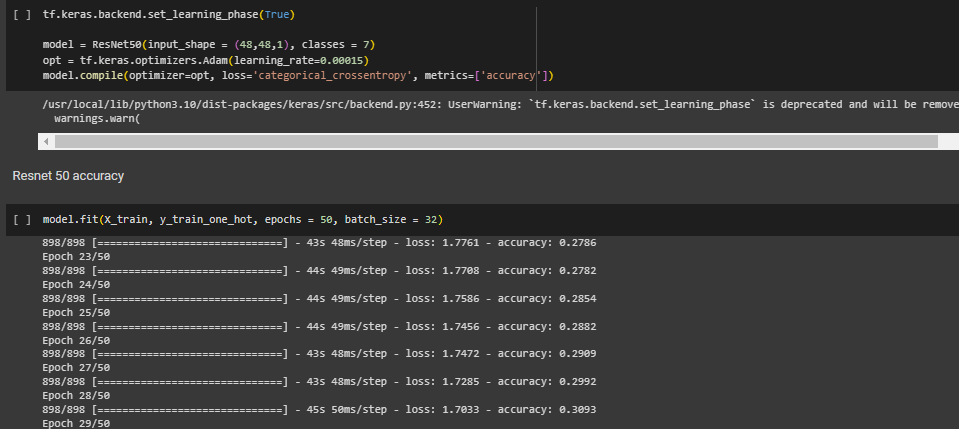
****

****

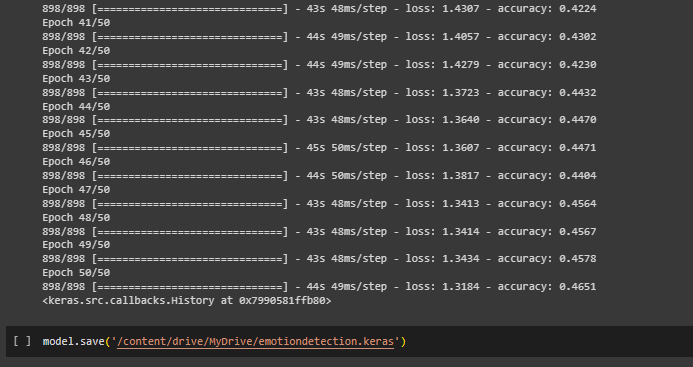


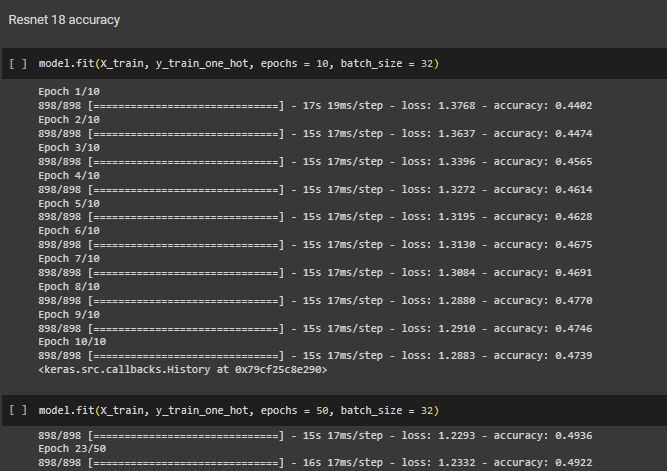


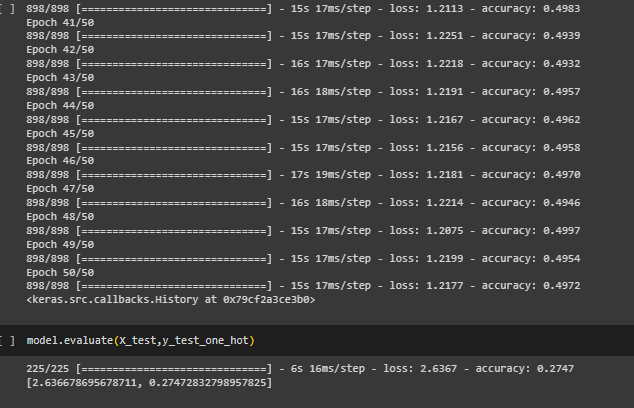




Training the model upto 50 epochs ie 50 traversals over entire train dataset

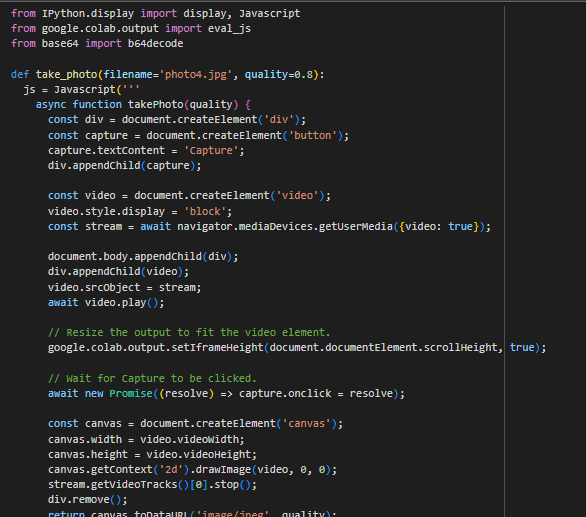


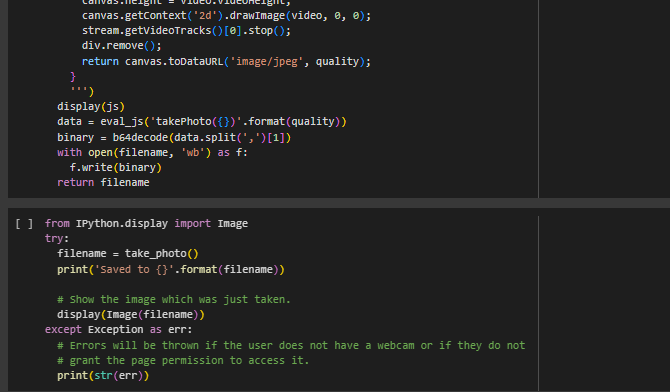




As seen from above both models give almost same accuracy and i tested ResNet 50 on test set

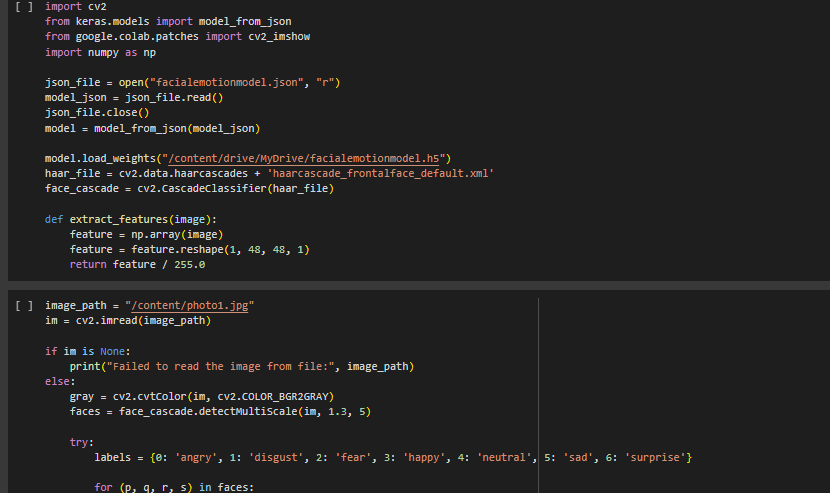
In order to test this model on real world data i imported script from colab to capture my image from camera

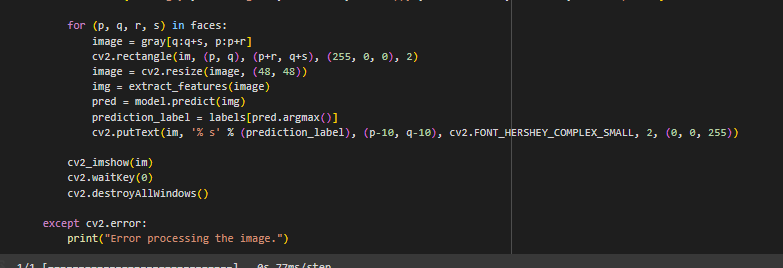


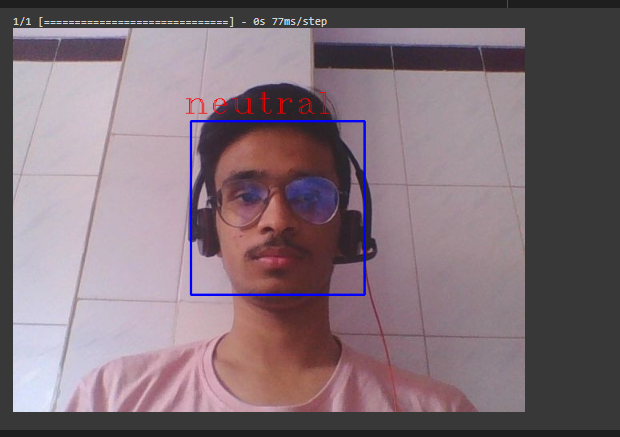


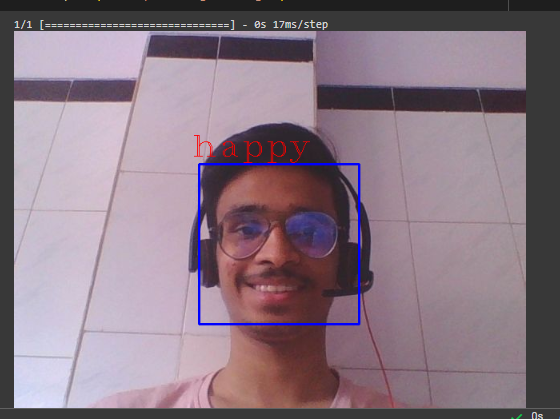
This script saves photo taken from camera

Later i used OpenCV to convert camera’s image into grayscale 48x48 for passing it to model and later pasting the prediction on the image as output









**Conclusion:** Size of dataset and size of image limits the accuracy of the model yet it provides satisfactory results on real world examples and this project serves its purpose of teaching about various DL models,CNN architectures and use of DL frameworks such as TensorFlow