**RECOGNIZING SIMILAR FACES :**

*CONTRIBUTOR :*

*@THIYAGARAJAN R*

**Overview:**

**Recognizing faces in the Wild - The Smile lab is conducting a competition on Kaggle calling out the research-scholars to present a solution based on faces by kinship recognition by extracting the facial features !To be more precise , to spot out the persons whether they are blood-related which is solely based on the faces of the images provided!**

*Understanding more about data:*

The datasets which has been provided in the competition by Kaggle, It comprises of train files which consists of 2786 directories and in general 12379 image files which are wrapped in those directories! For Example F0002 directory ( to be precise it is a directory representing a family) which contains 3 sub directories named as MID1( 10 image files ), MID2( 9 image files) , MID3( 8 image files) and those sub-directories represent the Unique family members such as even either grand-parents , parents , and children , Our intention is to make them as blood-related if the feature matches else we should discard the information

They are provided a train.csv file display labels which conveys more meaningfully if father and mother are not kin relationship to each other but their children are kin relationship to parents! Using the dataFrame there are 3365 unique files with 2 columns relating that their kin relationship to parents/grandparents solely! For example (p1[0] and p1[1] are not related to each other but p2[0] is related to both p1[0] and p1[1] )

And to test the kin relationship they are provided test.zip files which comprises of 6282 Unknown image files and in which 2 images files are picked up to validate our model is bringing up the expected results as outcome is based on is\_related or not\_related

The submission of the file is to be made either 1 ( is\_related) or 0 ( not\_related) , the changes are reflected on sample\_submission.csv file

DATA EXPLORATION:

1. As a fore-prediction I have gone through the data in train.zip ( image directories) and train\_relationships.csv (their corresponding labels) there have been pseudo directory which is present in the csv labels but not existing in the train image directory seems to be meaningless!

2. With the help of Pandas , Data Explorations seems to be easier thereby removing the redundancy modifying the directories sorting out all the misconception data as a before-hand making easier for our model to process without a mess !

3. MatPlotLib PyPlot makes the work easier to display the images relating and bringing up the underlying meaning of train\_relationship.csv file and train.zip thereby providing the dataFrameColumn (p1[0]) fetches the corresponding image from train directory

4. Setting up a sample directory for validation while the rest was considered to be training directory for generalization

5. Processing the data by the index of the first train image and choosing the random image as second option ( since 2 images are required and must for validation though moreover the data should be wrapped up in a same class (family) )

***Defining Architecture:***

With the PyTorch Framework , it is quite efficient in adding up the sequential layers to construct a series of Convolution layers from input layers to 8 and then hidden layers which ranges from 16 , 32, 64 and adding ReLu activation function so as to perform the differentiation which makes the backpropagation easier while computing the gradient loss and performing optimization techniques on it

Why I opt for ReLu ?

Rectified Linear Units seems to be much more efficient than tanh ( as a recent case study proves that ReLu performs 6 times better on converging )

R(x) = max(0,x) if x<0 ,R(x) = 0 else if x>=0 R(x) = x and it assures the possibility of clearing

Vanishing Gradient problem

After the Convolutional layers adding Dropout layers which ignores all the neurons in the system , though it takes some neurons at random , the reason why we are considering only some of the neurons not all is **to prevent overfitting of neurons** ( the problem occurs if we proceed with whole system )

In the project I proceeded my model to go with a 75% neurons at first which means 25% are disabled , yes I have set dropout rate to 0.25 ( it conveys 25% of neurons are disabled during the training phase )

Advantages of using Dropouts :

* Time Efficient thereby running massive amount of epochs
* Convergence is the big deal achieved in Dropouts and therefore it looks as much as every learning steps as double manner than compared to single mannerism