Project Report for Facial Expression Recognition, Classifier Model

By :

Kunchanapalli Manohar

Bennett University, Greater Noida

Project repo : <https://github.com/manohar2000/face_emotion_recog>

Deployment mode : Via Flask backend [ runs on the localhost]

The steps to run the model has been mentioned in the github repository README.md file

The python libraries required for this project is listed in the requirements.txt file.

To install the dependencies create a new environment, and run the following command

“pip install -r requirements.txt”

For the backend part we have used Flask, for model inference in real-time, we have made of use Tensorflow backend. In case of the GPU support available in you system, you could fasten up the inference time.

The Working of the project could be done by running the “main.py” file with the command

“python main.py”

This will create a local server using Flask and once the server gets active, you can visit

“localhost:5000/” and you can see the responsive site active.

Upon clicking the Task Link button, we could observe the internship project task details.

Upon clicking the “Start Live Demo” button, we would be redirected to a page wherein the rendered frames

1. with the bounding box [ detecting the face ] and
2. Text showing the emotion of the frame after getting classified with the help of our model.

For the facial detection problem, we made use of OpenCV techniques, using haar cascades : “haarcascade\_frontalface\_default.xml”

This helps us in drawing the bounding box around the detected face. The captured live frames are then sent to the model for prediction and the output label gets displayed on the screen.

Steps of making the prediction :

1. Visit the web application
2. Click on the Start Live Demo button.
3. Live video capture starts.
4. The frames are captured.
5. The haarcascade filters helps in drawing a bounding box across the detected human face.
6. The bounding box region is cropped : creating a ROI containing just the face.
7. This cropped face frame is sent to the Machine Learning Model for prediction.
8. The model classifies the frame into one of the 7 categories :

1) Angry

2) Disgust

3) Fear

4) Happy

5) Neutral

6) Sad

7) Surprise

9) The predicted output label is shown on the screen.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Machine Learning Model Making :

Following was the distribution of the images belonging to the various classes :

3995 angry images

436 disgust images

4097 fear images

7214 happy images

4965 neutral images

4830 sad images

3171 surprise images

This is a classic example of Skewed distribution and using this distribution straight away would lead to

Biased results.

Hence there is a need to create synthetic data for “Disgust” class. To make this possible we have made use of Data augmentation techniques, primarily Horizontal flipping.

Model :

We have built a custom model having 4 convolutional layers and we have used dropout of 25 percent and

Also Batch Normalization.

After the 4 steps of Convolutional Layers followed by Max Pooling layers.

We have flattened the output and used 2 fully connected layers after that.

The final output layer consists of 7 neurons and since at a time we have to predict only one output variable, we make the use of “Softmax” Activation function.

Total params: 4,478,727

Trainable params: 4,474,759

Non-trainable params: 3,968

After training the model for 50 epochs, we observe an accuracy of 66 percent finally.

We write out the model configurations to “model.json” file which contains the meta data about the model.

We also save the weights values so that later it becomes easier to load the model and make the predictions.

In the backend file “ camera.py”, we have accessed the live camera and made the prediction by loading the saved model.