CIS731- ARTIFICIAL NEURAL NETWORK CODE SUBMISSION REPORT

Chetanraj Kadam (250256631)

PROJECT DESCRIPTION

The project is about detection of human emotion from facial expressions using convolutional neural networks (CNNs).

I am submitting following implemented modules in **project_ckadam.zip:**

- Code

    1. DataProcessing.ipynb : Code for data visualization, data augmentation and mean center
    2. Shallow_CNN.ipynb: Code for shallow CNN model and results
    3. Deep_CNN.ipynb: Code for Deep CNN model and its results
    4. VGG16.ipynb: Code for VGG16 both as feature extracter and fine-tuned. Corressponding results.
    5. Models Folder: This folder contains trained models in .h5 format and images of model structures.
    6. Data Files Folder: Saves .npy data files in this folder after data processing stage

- Dataset:

    1. fer2013.csv : Kaggle Facial Expression Recognition dataset

RESULTS

Following are results I got at each stage and for each model:

- **Data Preprocessing:**

- Visualization from original dataset and corresponding emotions from code:

- Performed data augmentation by flipping images horizontally. So original training set of 28709 images become of 57418 images.
  Following can show effect of data augmentation:

  Image no. = 0

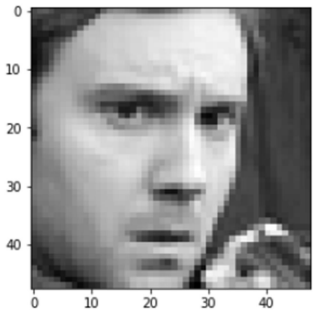
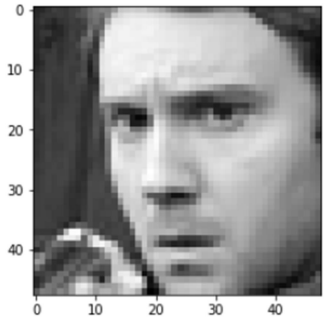
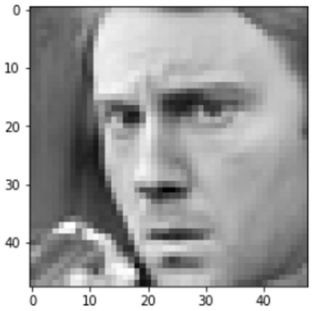


Image no. = 28709



So like this all images got flipped and added to training set

- Then performed mean centering and normalization on dataset
  After this above image becomes like this:

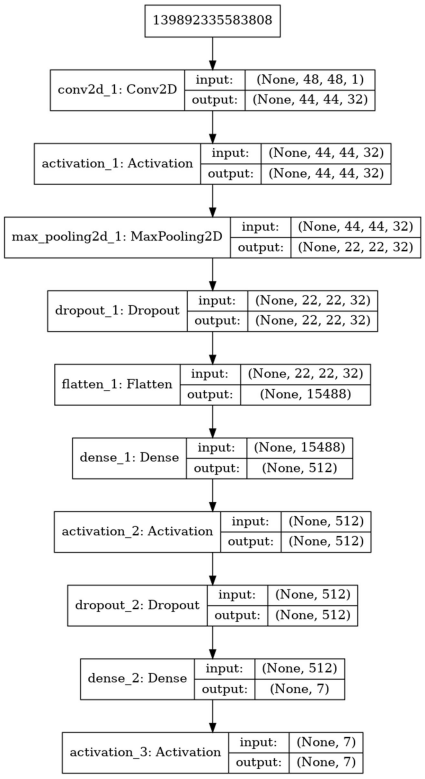


After pre-processing done I saved train, test and validation sets in .npy files in model folder for later use of all models.

- **Shallow CNN model:**

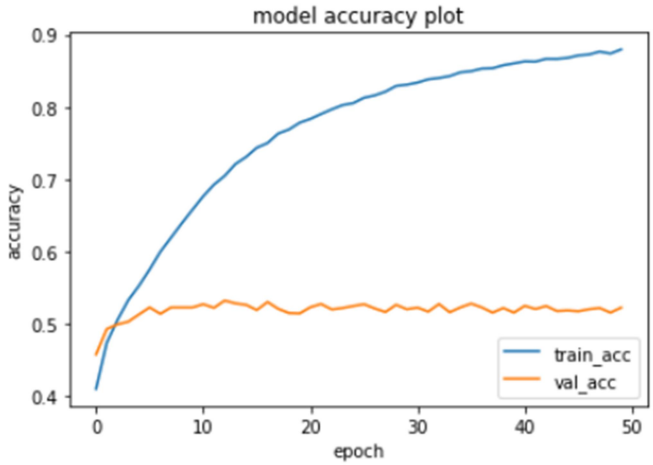
I implemented shallow model with only one 5x5 convolution layer, one 2x2 max pooling layer and two fully connected layers. Results I got for this model is as following:

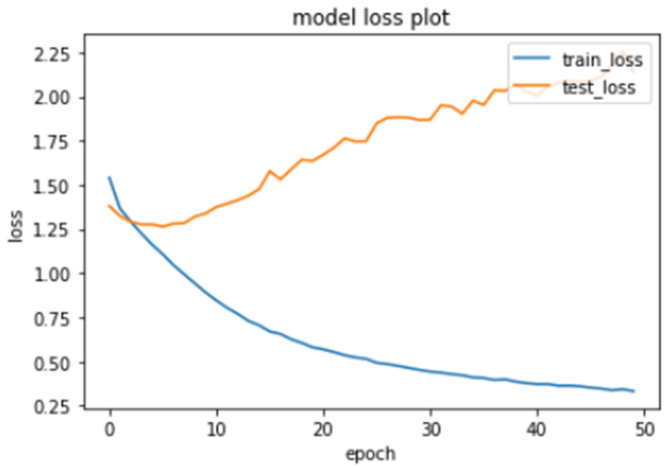
Structure of this model obtained using plot is as follows:



Results I obtained after running this model are as follows:

```
===> Results <===

Training Accuracy: 87.95%
Validation Accuracy: 52.24%
```

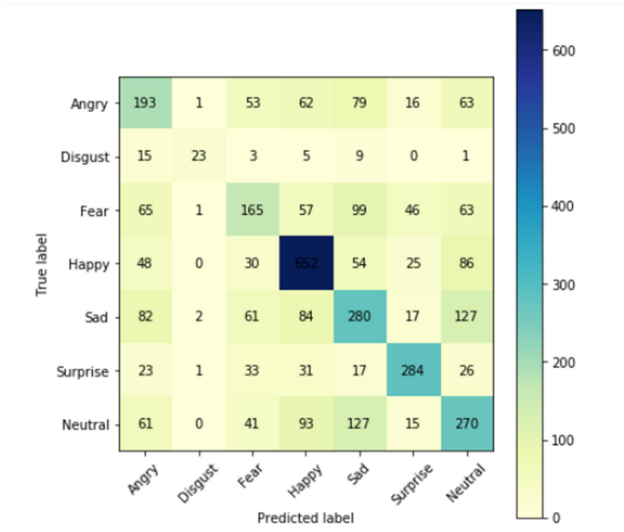




When evaluated on test data I got following results: **Accuracy : 52.02%**

```
Evaluating model on test data
3589/3589 [==============================] - 0s 51us/step

Test Accuracy: 52.02%

Loss: 2.19
```

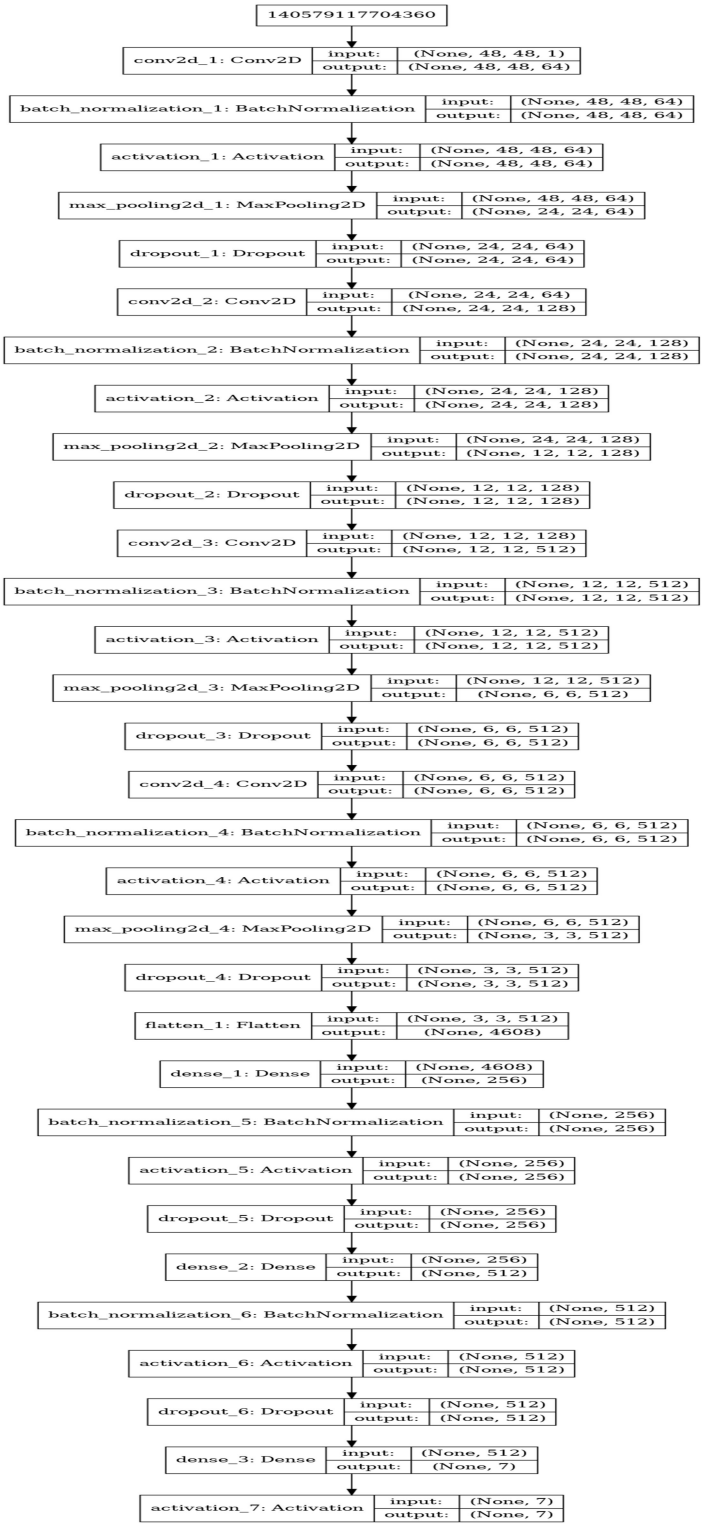
Confusion Matrix for above results is as follows:



- **Deep CNN model:**

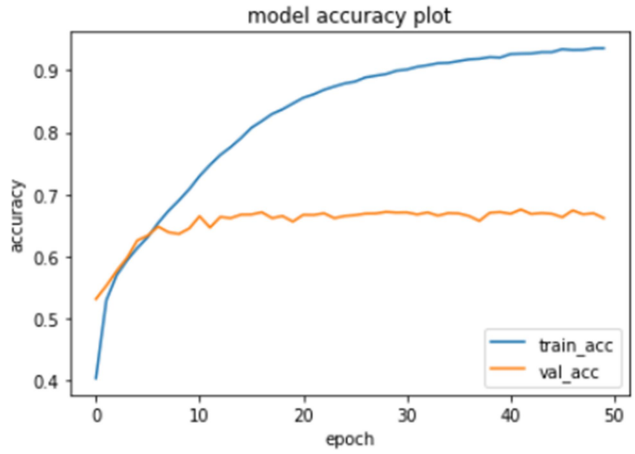
I implemented deep model as [Conv – Relu - MaxPooling]* 4 + [Dense - Relu] *2 + FC-softmax.

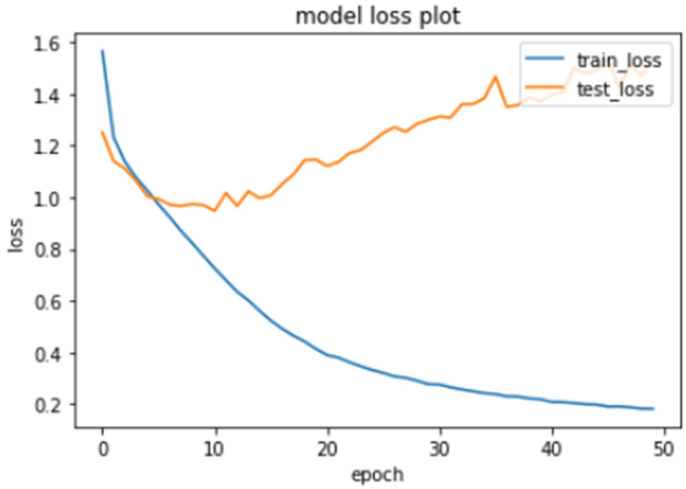
Structure of this model obtained using plot is as follows:



Results I got for this model are as following:

```
===> Results <===

Training Accuracy: 93.57%
Validation Accuracy: 66.15%
```



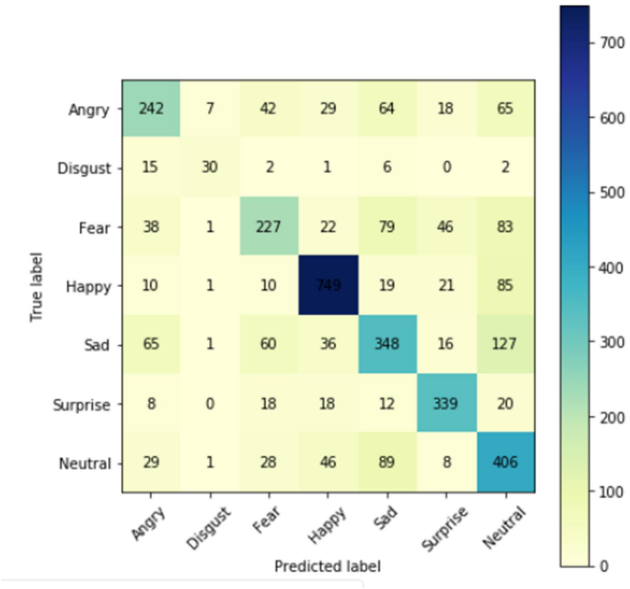


When evaluated on test data I got following results:

Accuracy : 65.23%

```
Evaluating model on test data
3589/3589 [==============================] - 1s 169us/step

Test Accuracy: 65.23%

Loss: 1.64
```

Confusion matrix for above results:



- **VGG16 model:**

I implemented VGG16 for implementation of transfer learning. . I used Keras to load built in pretrained weights of VGG16.

- **As Feature Extracter:**
- First implementation I used VGG16 as feature extractor and added own dense layers. Results I got for this model are as following:

```
===> Results: VGG16: Feature Extracter <===

Training Accuracy: 77.48%
Validation Accuracy: 48.04%
```

When evaluated on test data I got following results:

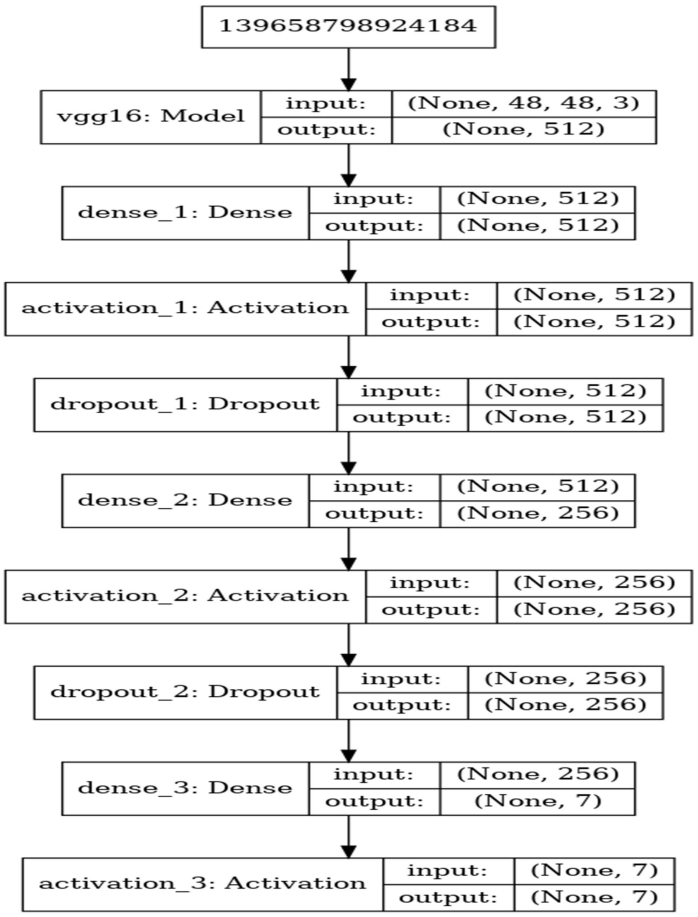
Accuracy : 49.51%

```
Evaluating model on test data
3589/3589 [==============================] - 1s 292us/step

Test Accuracy: 49.51%

Loss: 1.93
```

- **Fine Tuning on our dataset:**

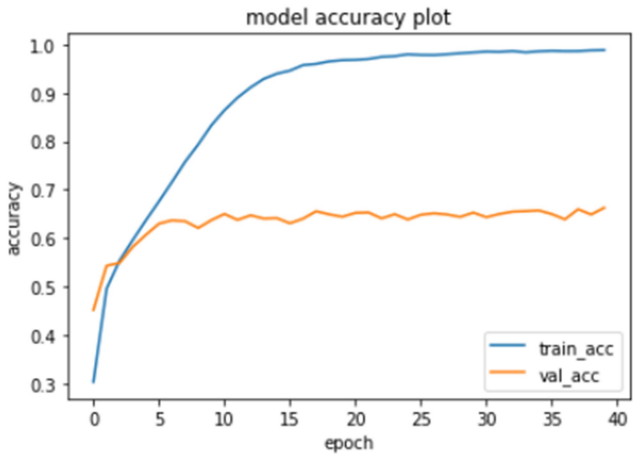
- In second implementation I downloaded Vgg model and then added own dense layers. And then I fine-tuned whole network using our dataset.

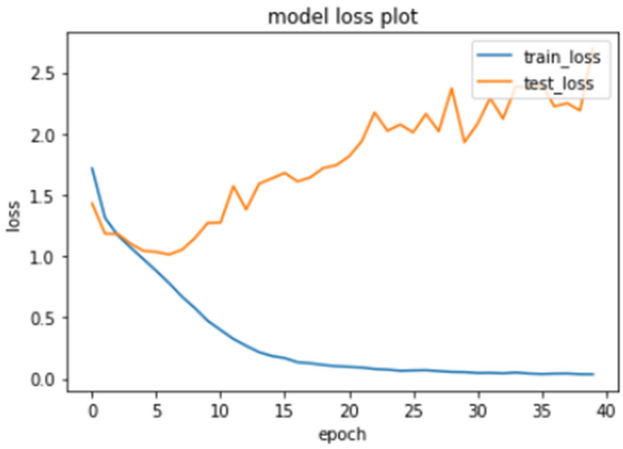
- Structure of this model obtained using plot is as follows:



- Results I got for this model are as following:

```
===> Results: VGG16: Fine Tunned <===

Training Accuracy: 98.88%
Validation Accuracy: 66.29%
```

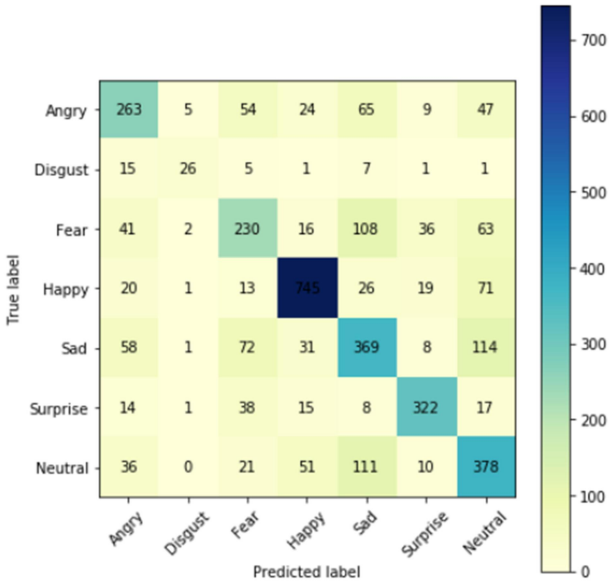




- When evaluated on test data I got following results:
- **Accuracy : 65.00%**

```
Evaluating model on test data
3589/3589 [==============================] - 1s 262us/step

Test Accuracy: 65.00%

Loss: 2.85
```

Confusion matrix for above results:



These were all results of my project.

- **In Progress:**

- All things are implemented. Only thing I am working on is some visualization of results for comparison and to include in paper.