Table Of Contents:

Topics Page No.

Problem Definition

- The aim of facial emotion recognition is to help identify the state of human emotion based on facial images.
- The challenge on facial emotion recognition is to automatically recognize facial emotion state.
- We will identify 7 emotions: {anger .disgust ,fear ,happy ,sad ,surprise ,neutral}.
- We will observe accuracy with respect to changing hyperparameters like learning rate, epochs.

Analysis and Design of problem

Data Collection

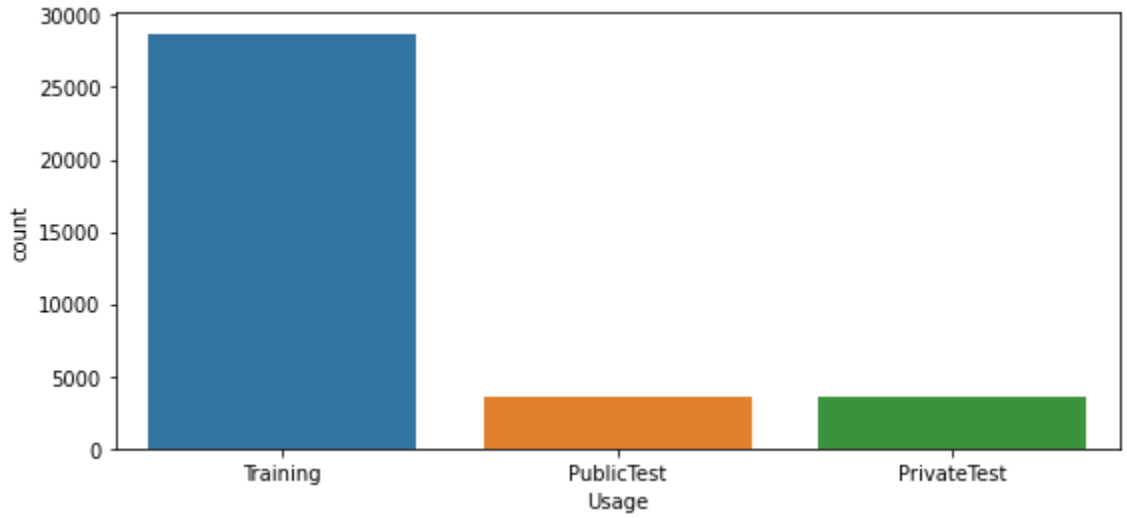
- The name of the data set is fer2013 which is an open-source data set that was made publicly available for a Kaggle competition. It contains 48 X 48-pixel grayscale images of the face.
- There are seven categories (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral) present in the data.
- The CSV file contains two columns that are emotion that contains numeric code from 0-6 and a pixel column that includes a string surrounded in quotes for each image.

- The “pixels” column contains a string surrounded in quotes for each image. The contents of this string are space-separated pixel values in row-major order. test.csv contains only the “pixels” column and our task is to predict the emotion column.

# emotion	A pixels	A Usage
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 110 126 143 153 158 171...	Training
0	151 150 147 155 148 133 111 140 170 174 182 154 153 164 173 178 185 185 189 187 186 193 194 185 183 ...	Training
2	231 212 156 164 174 138 161 173 182 200 106 38 39 74 138 161 164 179 190 201 210 216 220 224	Training

- In this dataset we have 3 labels i.e. Training , Public test and private test and total data is 35887 images.

```
Training       28709
PublicTest      3589
PrivateTest     3589
```



```
{Angry   : 14%,
 Disgust : 1%,
 Fear    : 15%,
 Happy   : 25%,
 Sad     : 16%,
 Surprise: 12%,
 Neural  : 17%}
```

DATA Preprocessing:

Prepare data for modeling

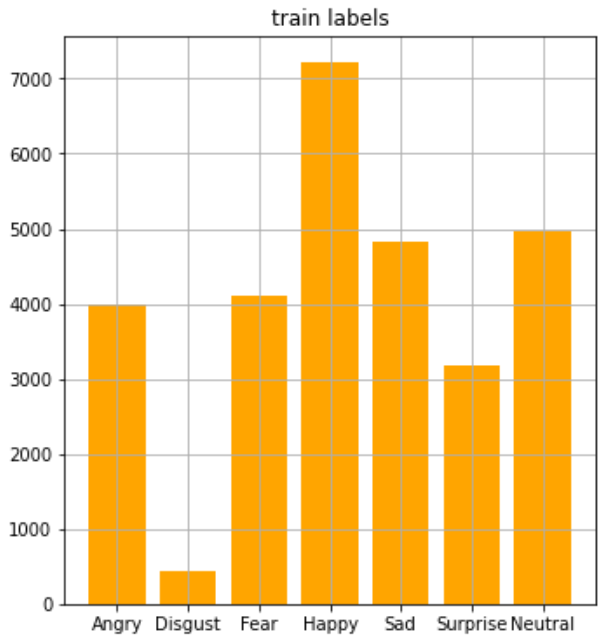
- Apply image preprocessing techniques such as resize, reshape, convert into grayscale, and normalization.
- Use the power of vectorization by converting images into NumPy arrays and pandas data frame whenever it's necessary.
- Convert the images into NumPy arrays using OpenCV and make the output as categorical using pandas.

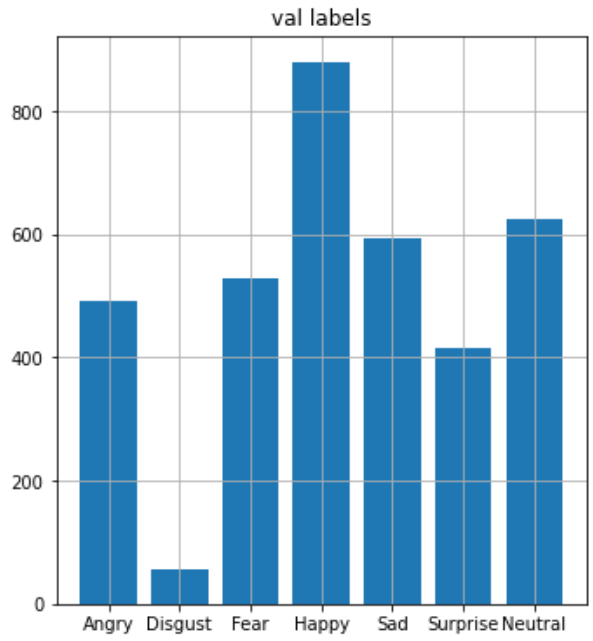
We have our data frame with labels and pixel data. We will convert this into an image and label array.

Splitting of Dataset:

Split the data set into the train and validation set. So that we can check whether the model is overfitted to the training dataset or not using the validation dataset.

- We have divided our dataset into 3 parts: train labels, val labels and public testing labels.
- We have trained our model on train labels and val labels and we are testing it on public testing labeled images.





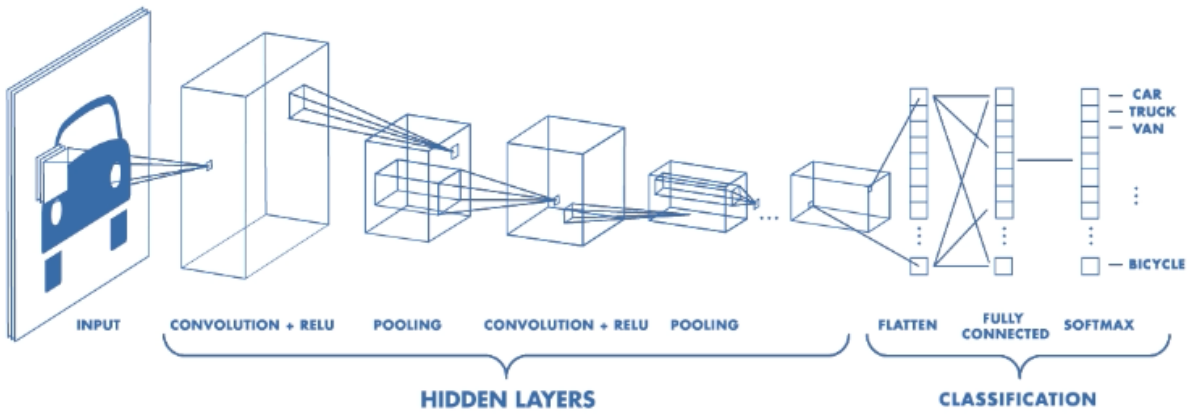
Understanding of CNN

Architecture:

We use three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer** (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet **architecture**.

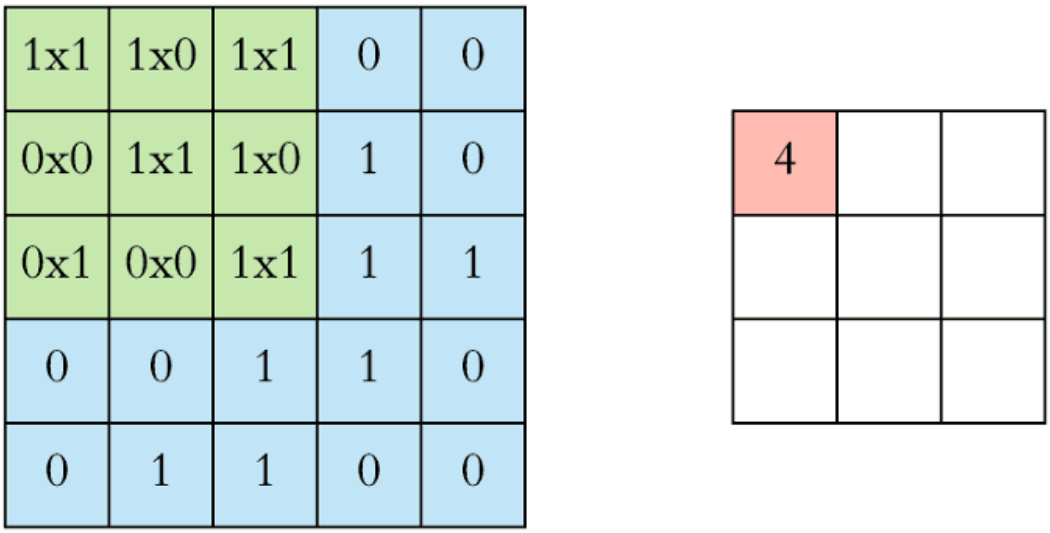
- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)

- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)



Convolutional Layer：**Convolutional layers** are the **layers** where filters are applied to the original image, or to other feature maps in a deep **CNN**. This is where most of the user-specified parameters are in the network.

Feature extraction： **Feature extraction** is a type of dimensionality reduction where a large number of pixels of the image are efficiently represented in such a way that interesting parts of the image are captured effectively



Pooling layer : Pooling layer progressively reduces the spatial size of the representation to reduce the amount of parameters and computation in the network. **Pooling layer** operates on each feature map independently.

Flattening layer : **Flattening** is converting the data into a 1-dimensional array for inputting it to the next layer. We **flatten** the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer

Dense layer: Each neuron in a layer receives an input from all the neurons present in the previous layer.The **dense** layer is a fully connected layer that is all the neurons in a layer are connected to those in the next layer

CNN MODEL

So we have taken 2 models:

- MODEL 1

Model with 5 layers

```
"sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 16928)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2166912
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 2,168,135
Trainable params: 2,168,135
Non-trainable params: 0
```

- MODEL 2

  Model with 7 layers

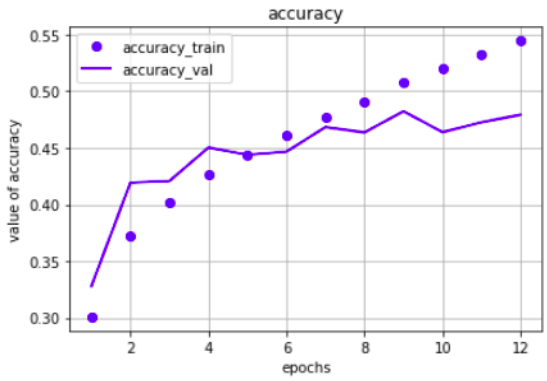
  "sequential"

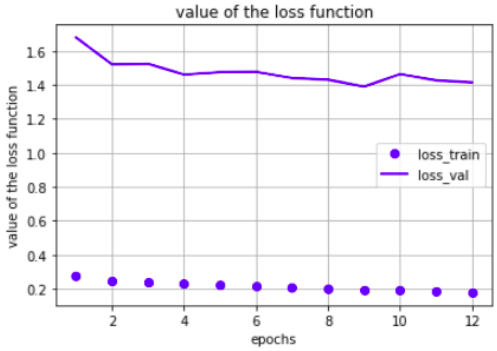
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 64)                262208
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 455
=================================================================
Total params: 318,407
Trainable params: 318,407
Non-trainable params: 0
```

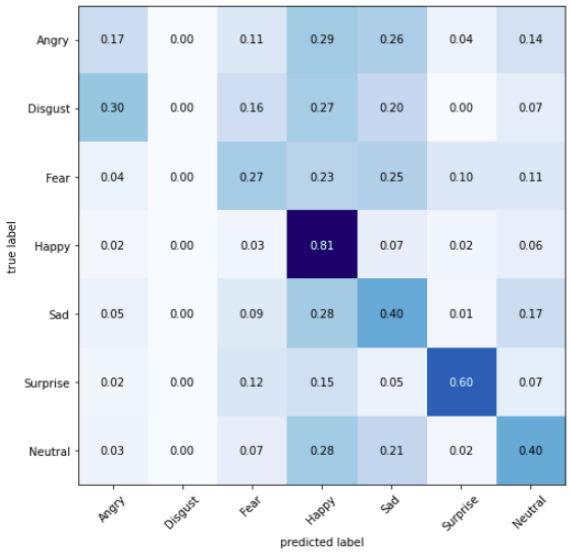
Performance measurement Criteria

- For CNN models we will change different parameters like epoch , learning rate,etc and we choose a model based on the values of Loss function , accuracy of model and most intelligent guess of model. Accuracy is the First Priority
- The minimum the value of loss function the better the model
- The maximum accuracy implies a better model.
- Most intelligent guess is for which emotion model is most accurate.

For Example



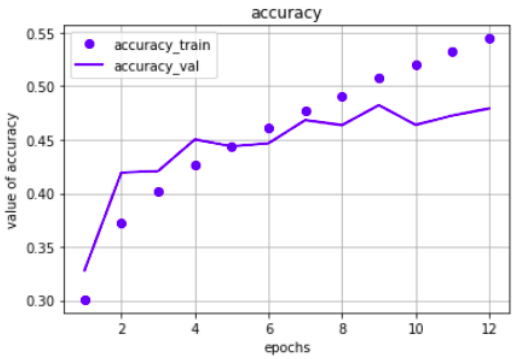




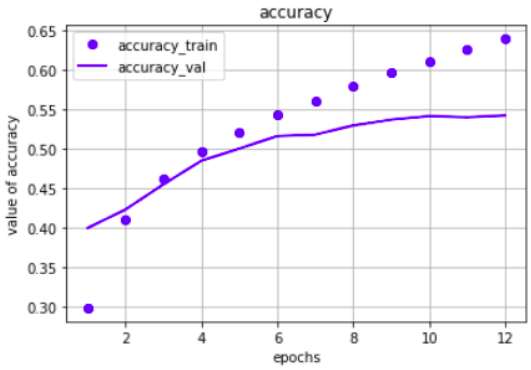
Experimentation And Results:

We first run our two models on learning rate 1e-3 and epoch 12:

CNN MODEL 1



CNN MODEL 2



We choose CNN Model 2 because of higher Accuracy.

Then We run our Model 2 on 2 different learning rates and 3 different Epochs.

Following are the results and graphs for the same

Learning rate:1e-3

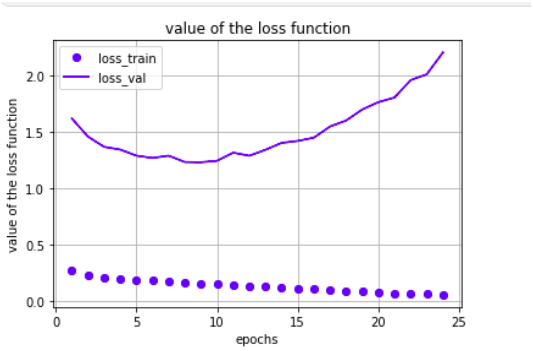
Epochs: 24 ,15 ,12

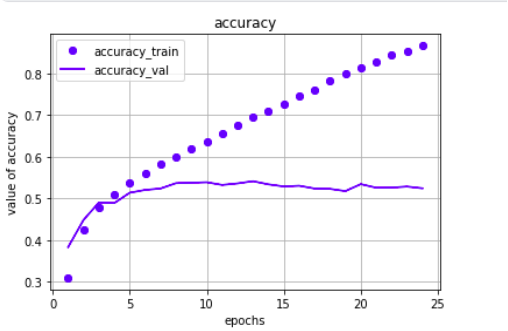
Learning rate:1e-2

Epochs: 24 ,15 ,12

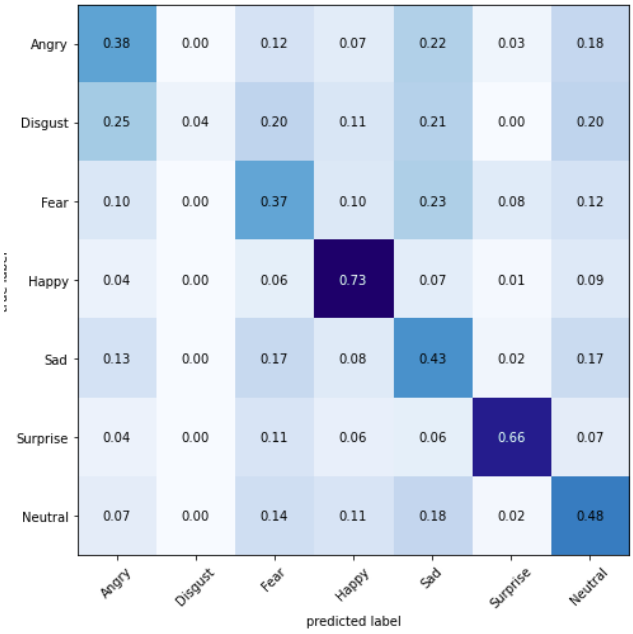
Learning rate =1e-3

- Epoch=24

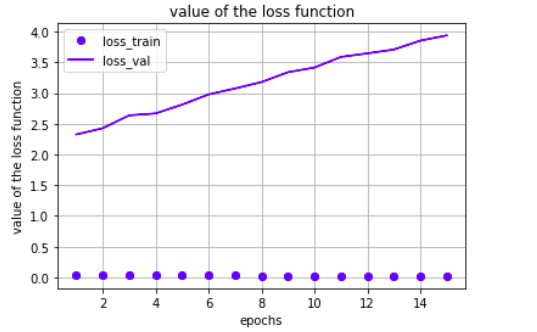


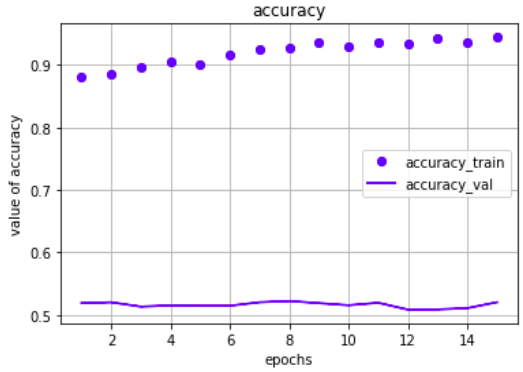


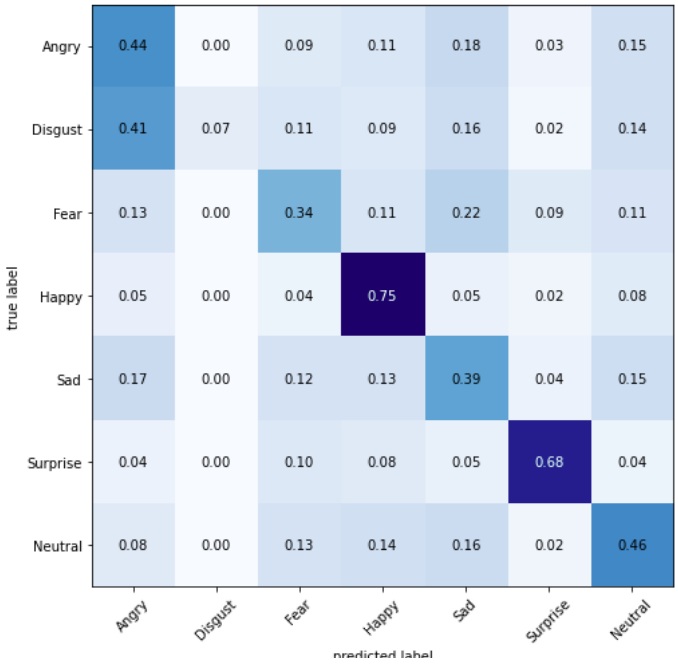
+ Code + Markdown



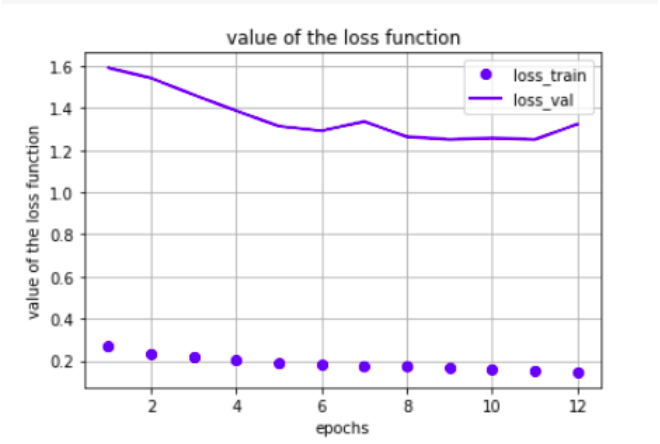
- Epoch 15

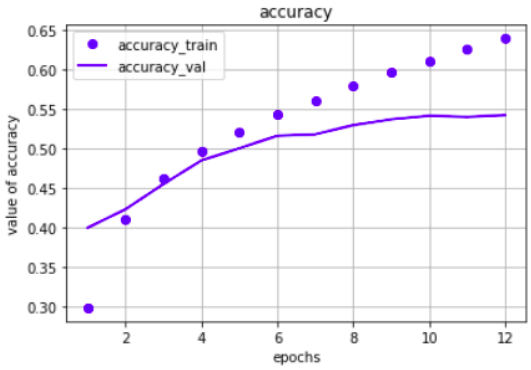

value of the loss function
loss_train
loss_val
value of the loss function
epochs

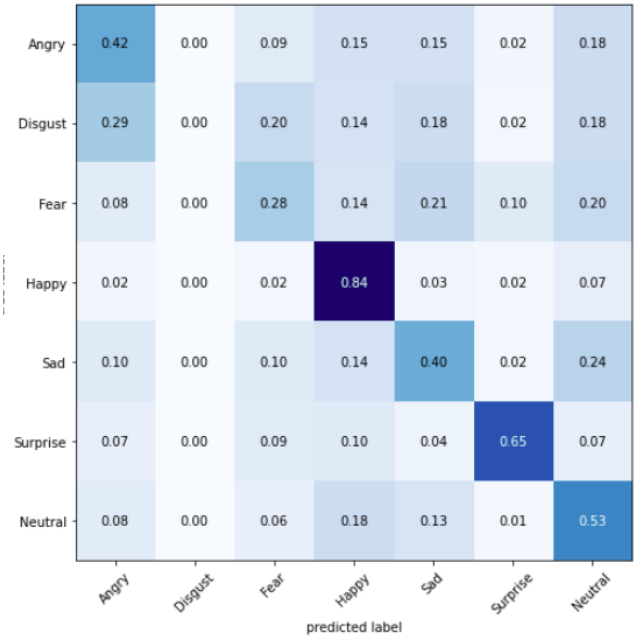

accuracy
accuracy_train
accuracy_val
value of accuracy
epochs


Angry 0.44 0.00 0.09 0.11 0.18 0.03 0.15
Disgust 0.41 0.07 0.11 0.09 0.16 0.02 0.14
Fear 0.13 0.00 0.34 0.11 0.22 0.09 0.11
Happy 0.05 0.00 0.04 0.75 0.05 0.02 0.08
Sad 0.17 0.00 0.12 0.13 0.39 0.04 0.15
Surprise 0.04 0.00 0.10 0.08 0.05 0.68 0.04
Neutral 0.08 0.00 0.13 0.14 0.16 0.02 0.46
true label
Angry Disgust Fear Happy Sad Surprise Neutral
predicted label

- Epoch=12

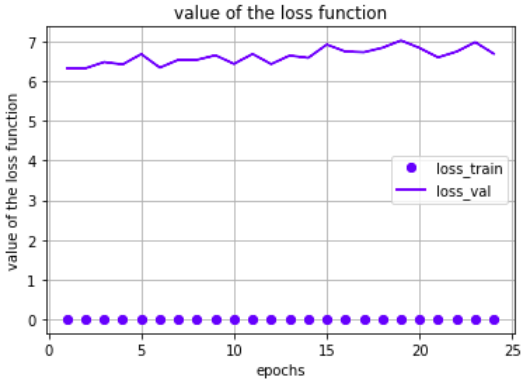

value of the loss function
loss_train
loss_val
value of the loss function
epochs

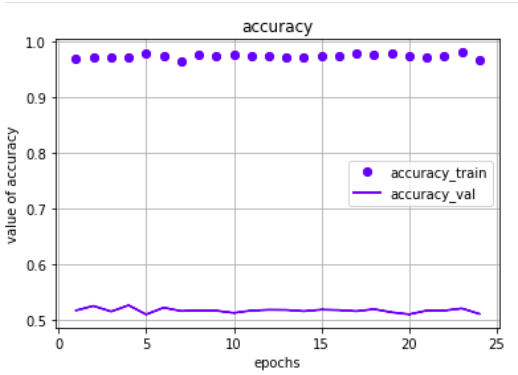

accuracy
accuracy_train
accuracy_val
value of accuracy
epochs

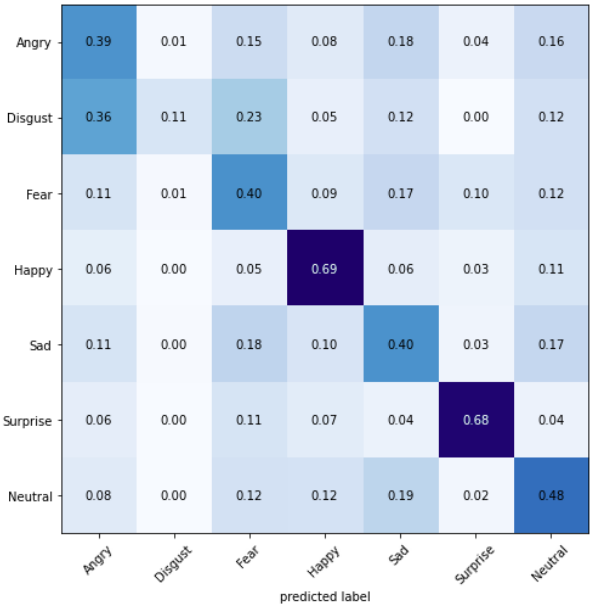

Angry 0.42 0.00 0.09 0.15 0.15 0.02 0.18
Disgust 0.29 0.00 0.20 0.14 0.18 0.02 0.18
Fear 0.08 0.00 0.28 0.14 0.21 0.10 0.20
Happy 0.02 0.00 0.02 0.84 0.03 0.02 0.07
Sad 0.10 0.00 0.10 0.14 0.40 0.02 0.24
Surprise 0.07 0.00 0.09 0.10 0.04 0.65 0.07
Neutral 0.08 0.00 0.06 0.18 0.13 0.01 0.53
Angry Disgust Fear Happy Sad Surprise Neutral
predicted label

Learning rate=1e-2

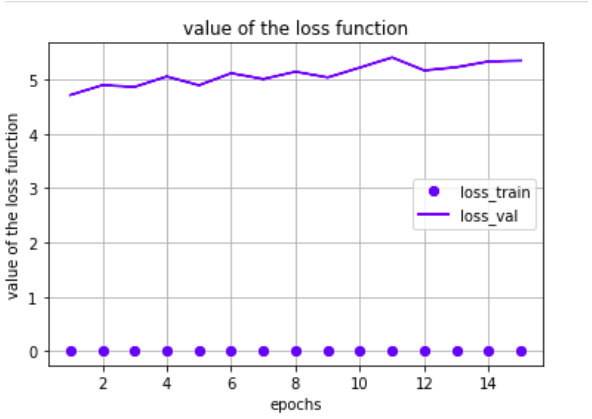
- Epoch=24

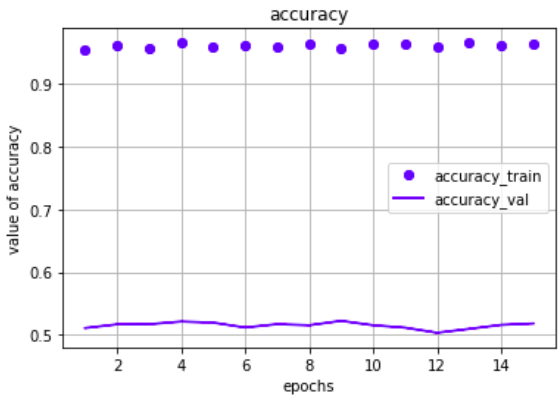


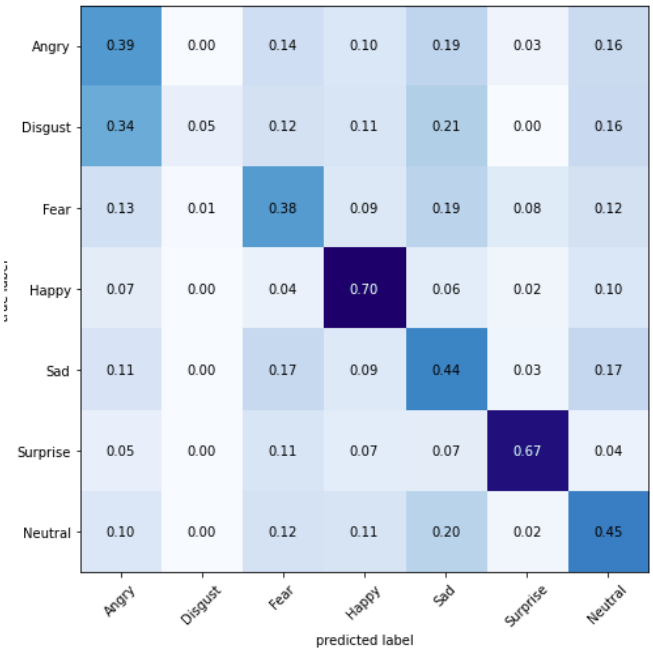




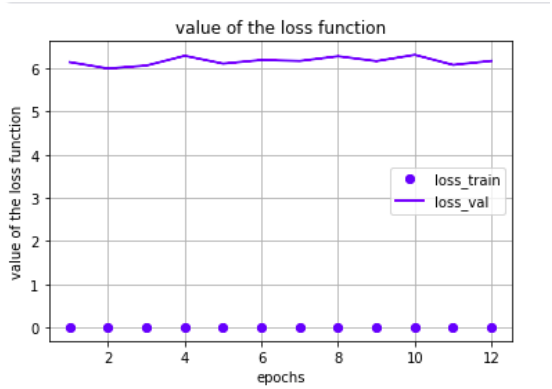
- Epoch=15

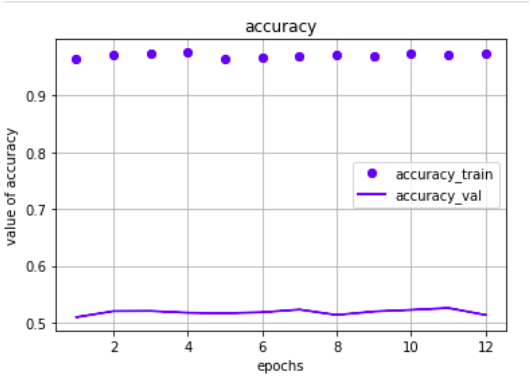

value of the loss function
value of the loss function
epochs
loss_train
loss_val

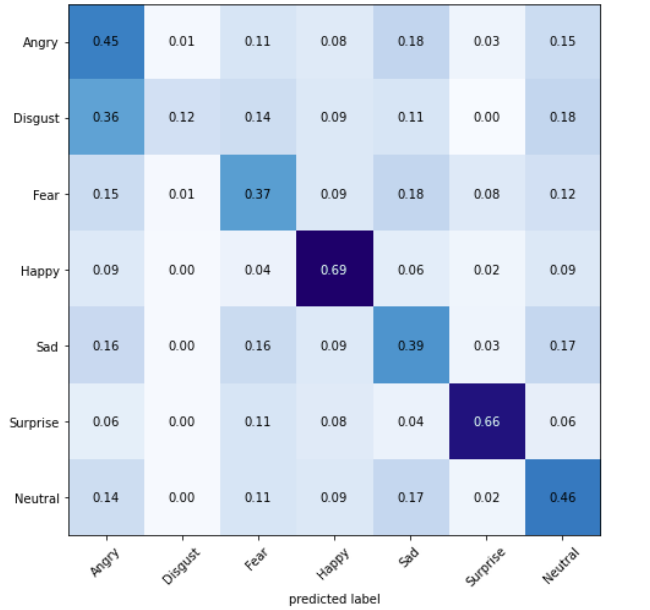

accuracy
value of accuracy
epochs
accuracy_train
accuracy_val


Angry 0.39 0.00 0.14 0.10 0.19 0.03 0.16
Disgust 0.34 0.05 0.12 0.11 0.21 0.00 0.16
Fear 0.13 0.01 0.38 0.09 0.19 0.08 0.12
Happy 0.07 0.00 0.04 0.70 0.06 0.02 0.10
Sad 0.11 0.00 0.17 0.09 0.44 0.03 0.17
Surprise 0.05 0.00 0.11 0.07 0.07 0.67 0.04
Neutral 0.10 0.00 0.12 0.11 0.20 0.02 0.45
Angry Disgust Fear Happy Sad Surprise Neutral
predicted label

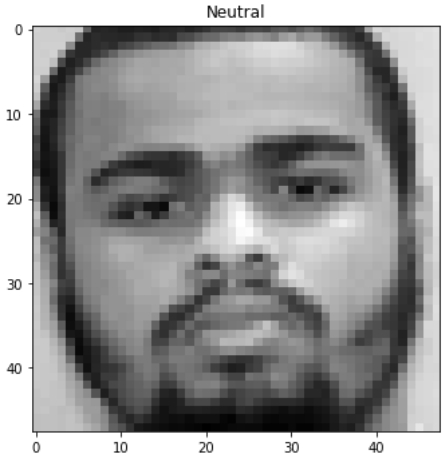
- Epoch=12

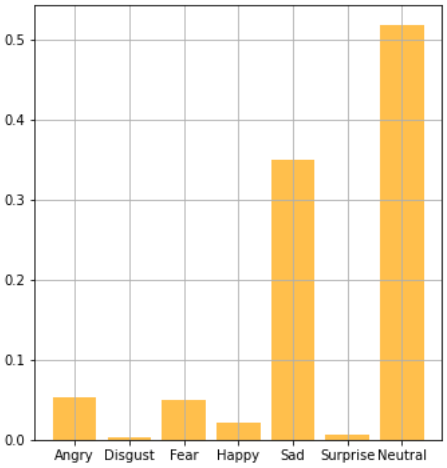

value of the loss function
value of the loss function
epochs
loss_train
loss_val

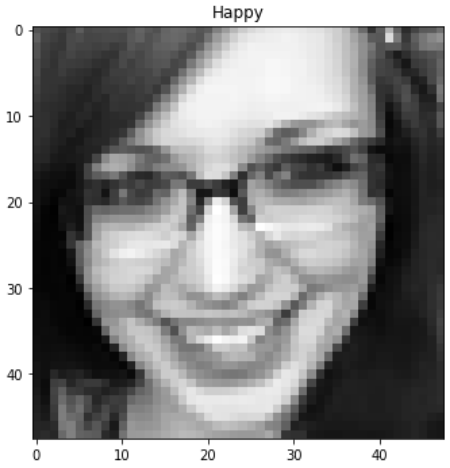

accuracy
value of accuracy
epochs
accuracy_train
accuracy_val

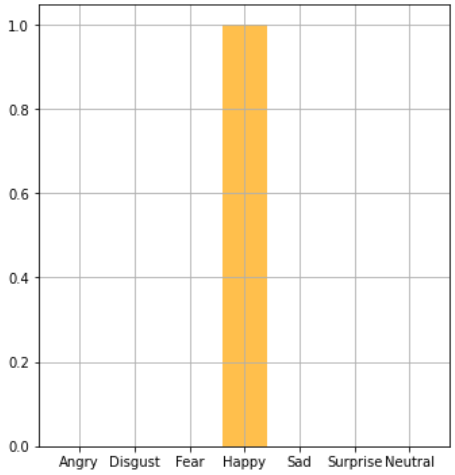

Angry 0.45 0.01 0.11 0.08 0.18 0.03 0.15
Disgust 0.36 0.12 0.14 0.09 0.11 0.00 0.18
Fear 0.15 0.01 0.37 0.09 0.18 0.08 0.12
Happy 0.09 0.00 0.04 0.69 0.06 0.02 0.09
Sad 0.16 0.00 0.16 0.09 0.39 0.03 0.17
Surprise 0.06 0.00 0.11 0.08 0.04 0.66 0.06
Neutral 0.14 0.00 0.11 0.09 0.17 0.02 0.46
Angry Disgust Fear Happy Sad Surprise Neutral
predicted label

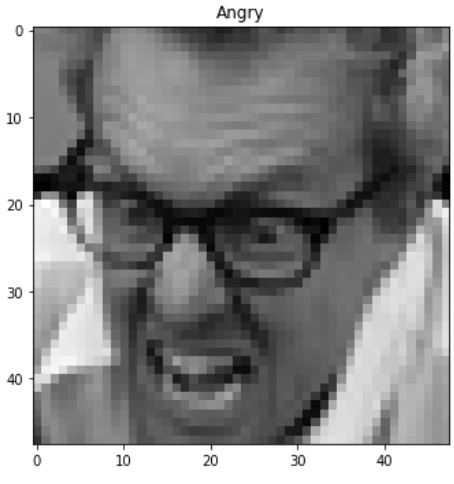
Demonstration on Model 2 with Epoch =12 and Learning rate =1e-3

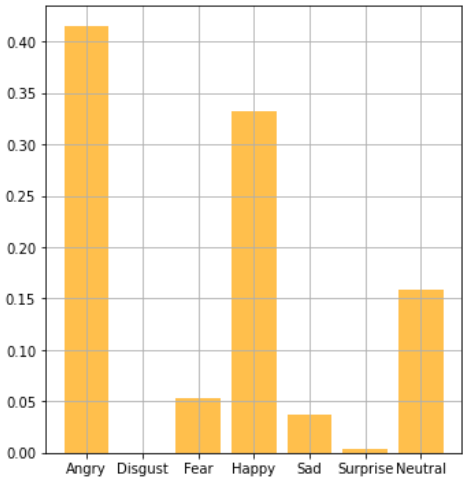






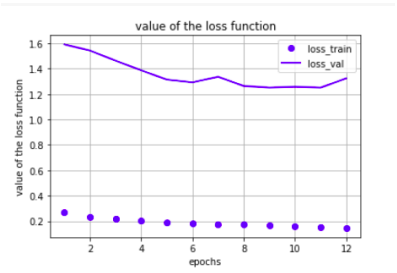


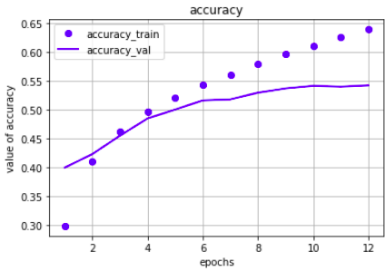


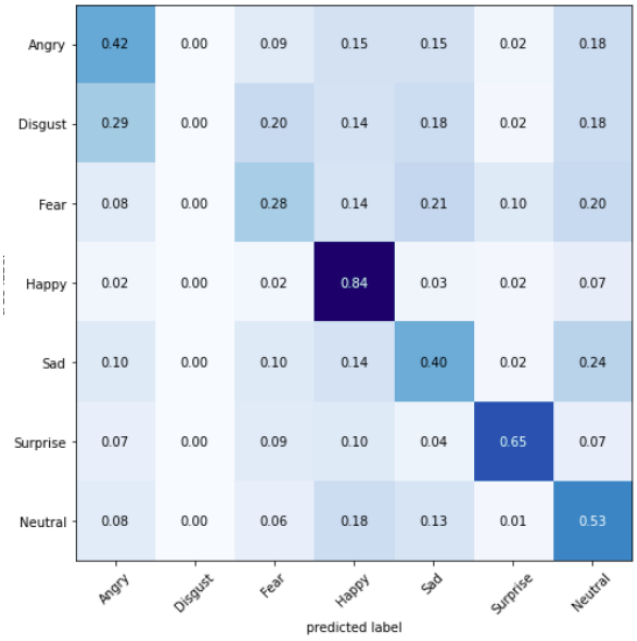


Conclusion:

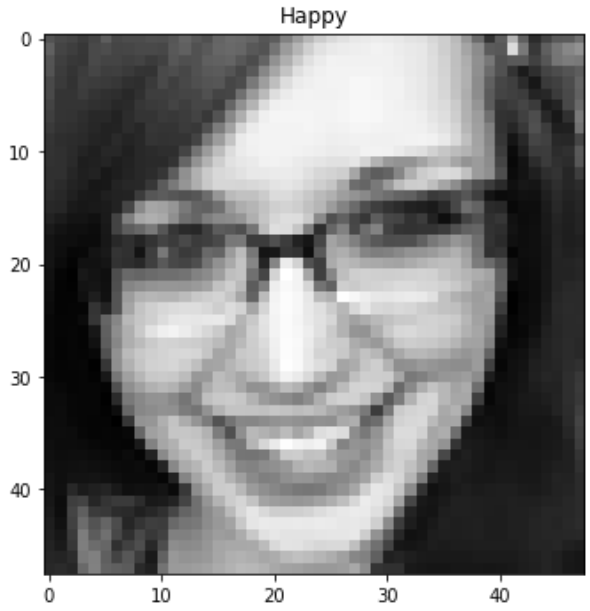
- CNN Model 2 has better accuracy than CNN Model 1. So we choose CNN model 2.
- For different learning rates we have seen Accuracy and intelligence are almost the same but values of loss function are much greater in learning rate 1e-2 than 1e-3 .So 1e-3 is more preferable.
- Now for Learning rate 1e-3 we choose the number of epochs =12 cause we have seen that for this epoch, the value of loss function is much lesser than values of loss function in epochs 24 and 15.
- We Conclude that CNN Model 2 with learning Rate 1e-3 and Epochs 12 is the best among our Experimentation.

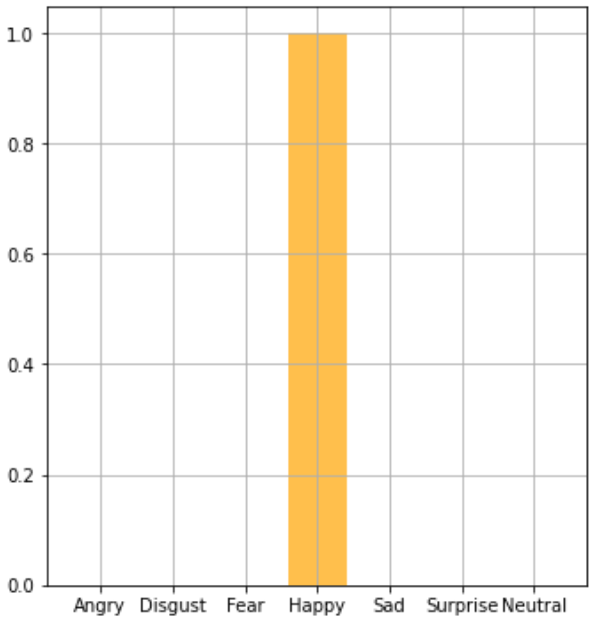






- Our model is most intelligent on Happy images.





References:

CNN Understanding:
https://www.youtube.com/watch?v=m0fWjP3yIEo&list=WL&index=7&t=2903s

DATASET: https://www.kaggle.com/deadskull7/fer2013

Code:

https://colab.research.google.com/drive/1Kwg9u7mF1QvxVG_HS_DKITa78tnd_kAj?usp=sharing