COL774

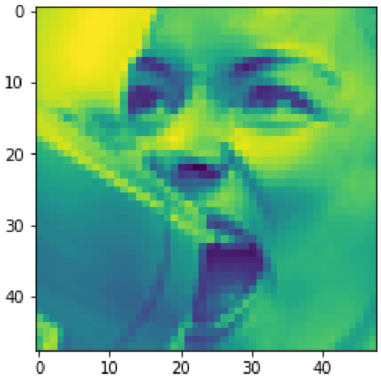
Assignment 4

Kartikeya Gupta(2018CS10349)
Seshank Achyutuni(2018CS10387)

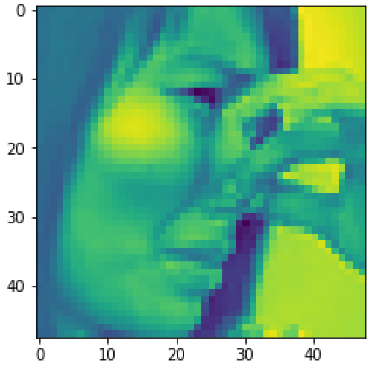
Dataset:

Training dataset included 19376 images of size 48x48 flattened to 2304 pixels. Range of pixels was from -3.43 to 2.44 with an average of 0.00013 so we can assume that data is already normalized. Output labels are from 0 to 6 with each representing an emotional state.

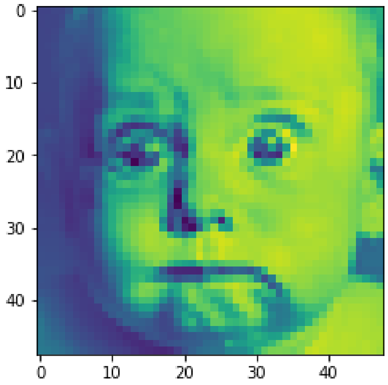
0:



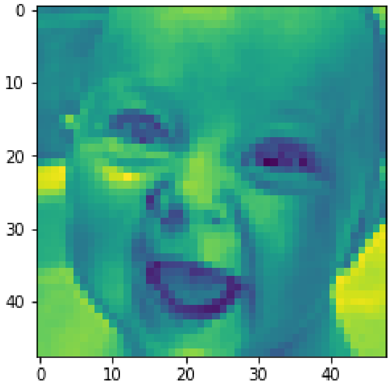
1:



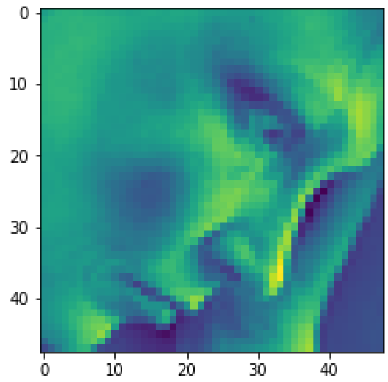
2:



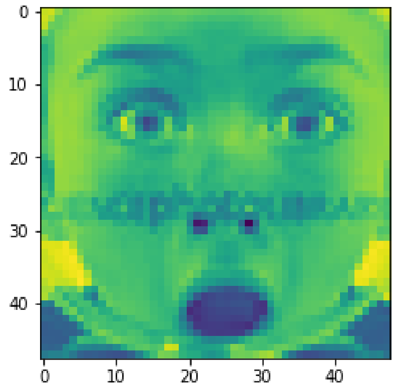
3:



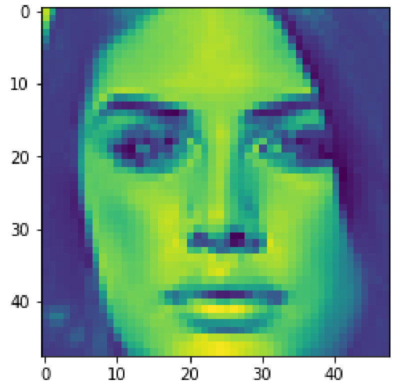
4:



5:



6:



Q1. Non Competitive part

a) Vanilla Neural Network

A fully connected neural network with one hidden layer with 100 perceptrons. Input is an image of size 2304 pixels. Optimizer used is SGD with a learning rate = 0.01. Cross entropy is used as the loss function. Mini batch with batch size = 100 is used. This model tend to overfit the training data

Model:

```
Sequential(
  (0): Linear(in_features=2304, out_features=100, bias=True)
  (1): ReLU()
  (2): Linear(in_features=100, out_features=7, bias=True)
)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 100]         230,500
              ReLU-2                  [-1, 100]               0
            Linear-3                    [-1, 7]             707
================================================================
```

Convergence Criteria: Maximum of 100 epochs or difference of consecutive cost < 1e-5 for at least 10 mini batches(we have stopped early because larger the number of epochs of training more the model overfits)

Without early stopping

Training Accuracy: 98.02%

Test Accuracy: 39.15%

With Early Stopping

Training Accuracy: 82.86%

Test Accuracy: 40.84%

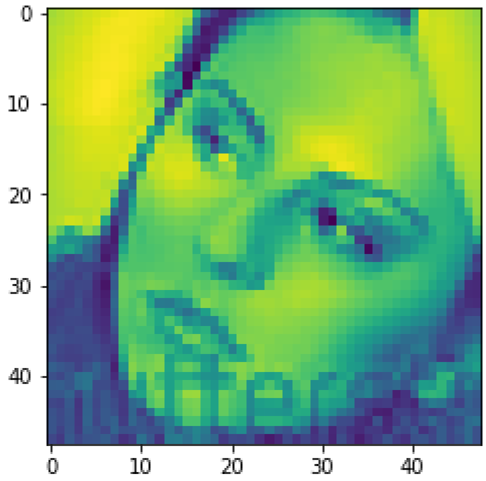
Macro F1 score: 0.3860

Confusion Matrix:

```
[271,   3, 134, 169, 179,  27, 109]
[ 16,  13,  12,  22,  15,   7,  13]
[126,   4, 276, 129, 189,  67, 128]
[131,   2, 182, 902, 182,  52, 171]
[182,   2, 152, 181, 383,  39, 158]
[ 61,   3, 140,  78,  65, 304,  66]
[164,   3, 141, 217, 202,  43, 344]
```

b) Feature Engineering

Gabor Filter:



Original Image

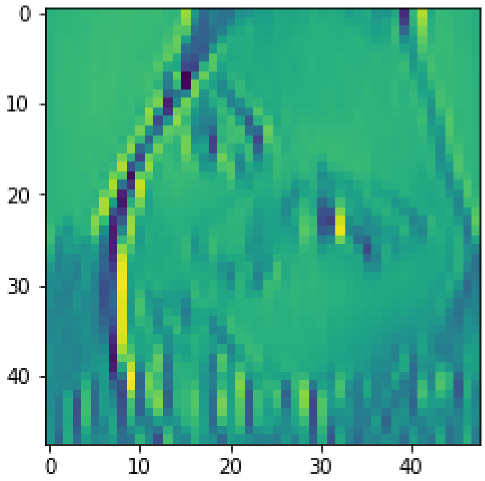


Image After applying filter

Training Accuracy: 92.43%

Test Accuracy:38.31%

Macro F1 score: 0.3431

Confusion Matrix:

```
[230,   1, 123, 183, 164,  40, 151]
[ 18,  10,  12,  24,  11,   9,  14]
[116,   3, 208, 158, 190,  93, 151]
[130,   1, 131, 905, 191,  80, 184]
[169,   3, 129, 194, 343,  46, 213]
[ 50,   1,  76,  84,  60, 371,  75]
[127,   1, 110, 208, 191,  69, 408]
```

Histogram of Oriented Gradients:

Training Accuracy: 99.96 %

Test Accuracy :44.64%

Macro F1 score: 0.418

Confusion Matrix:

```
[ 325,    4,  112,  117,  153,   35,  146]
[  21,   29,   11,   11,   14,    2,   10]
[ 129,    7,  273,  117,  164,   97,  132]
[ 128,    4,  102, 1021,  154,   57,  156]
[ 158,   15,  149,  149,  371,   51,  204]
[  64,    4,   93,   74,   46,  372,   64]
[ 146,    4,  134,  146,  166,   45,  473]
```

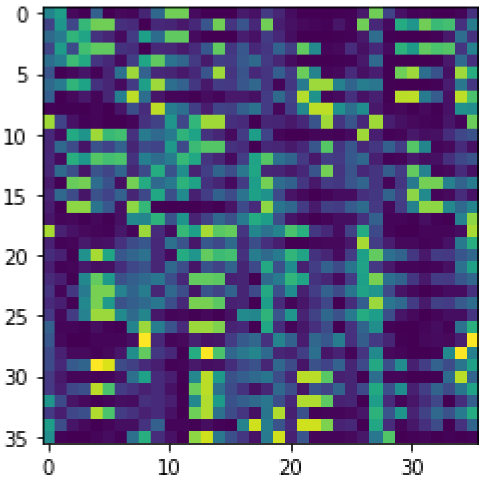


Image after applying filter

c) Convolutional Neural Network

Model:

```
Sequential(
  (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(3, 3))
  (1): BatchNorm2d(64)
  (2): ReLU()
  (3): MaxPool2d(kernel_size=2, stride=2)
  (4): Conv2d(64, 128, kernel_size=(2, 2), stride=(2, 2))
  (5): BatchNorm2d(128)
  (6): ReLU()
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (8): Flatten(start_dim=1, end_dim=-1)
  (9): Linear(in_features=512, out_features=256, bias=True)
  (10): BatchNorm1d(256)
  (11): ReLU()
  (12): Linear(in_features=256, out_features=7, bias=True)
)
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 16, 16]             640
       BatchNorm2d-2           [-1, 64, 16, 16]             128
              ReLU-3           [-1, 64, 16, 16]               0
         MaxPool2d-4             [-1, 64, 8, 8]               0
            Conv2d-5            [-1, 128, 4, 4]          32,896
       BatchNorm2d-6            [-1, 128, 4, 4]             256
              ReLU-7            [-1, 128, 4, 4]               0
         MaxPool2d-8            [-1, 128, 2, 2]               0
           Flatten-9                  [-1, 512]               0
           Linear-10                  [-1, 256]         131,328
      BatchNorm1d-11                  [-1, 256]             512
             ReLU-12                  [-1, 256]               0
           Linear-13                    [-1, 7]           1,799
================================================================
```

Without early stopping

Training Accuracy: 96.34%

Test Accuracy: 40.84%

With Early Stopping

Training Accuracy: 60.29%

Test Accuracy: 41.01%

Macro F1 score: 0.4084

Confusion Matrix:

```
[270,   7, 114, 147, 165,  51, 138]
[ 14,  15,  15,  21,  11,   6,  16]
[104,   2, 264, 125, 194, 114, 116]
[158,   7, 131, 934, 181,  60, 151]
[166,   4, 151, 155, 358,  54, 209]
[ 42,   1,  87,  71,  49, 401,  66]
[143,   3, 129, 207, 187,  49, 396]
```

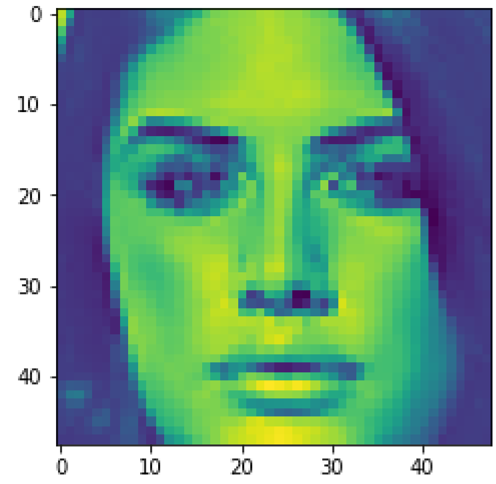
Q2. Competitive part

Data Augmentation:

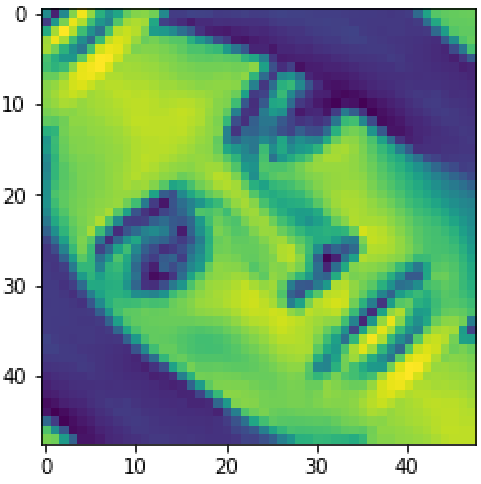
Data given is skewed with following distribution:

```
[2677  295 2755 4866 3289 2151 3343]
```

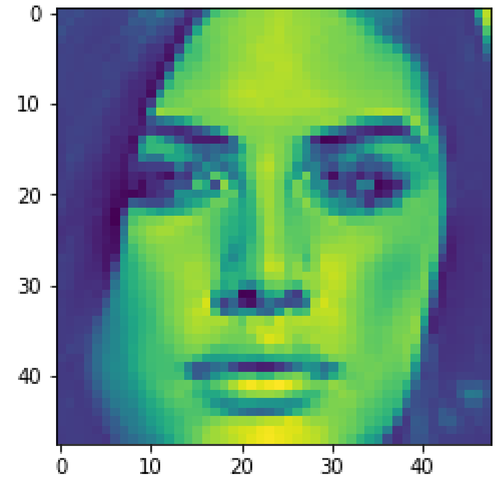
Data Normalization was tried to improve Macro F1 score but resulted in no change in score. Image transformation was also applied as orientation, contrast or may have noise in it so to capture all this variation in image transformation was applied using skimage library.



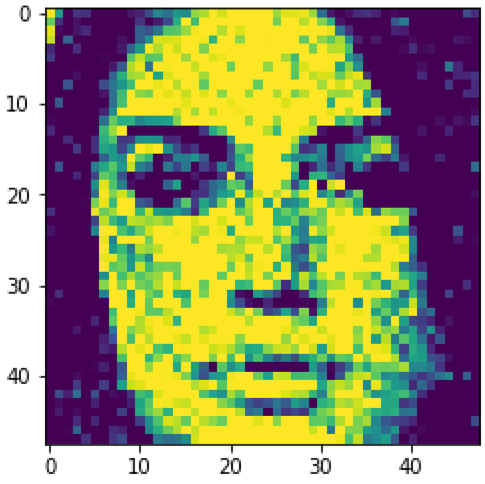
Original Image



Rotated Image



Flipped Image



Added Random Noise

Adam and SGD optimizer was tried changing parameters. Both provided similar result however SGD converged at a faster rate given same criteria for convergence learning rate 0.01, momentum=0.9 and weight_decay=0.0001.

Loss function used is CrossEntroyLoss().

Model:

From the part above we deduced that adding a convolution layer and applying filters on the training set helps in increases. So, the following model was implemented with Histogram of Oriented Gradients filter. Data augmentation was also applied to check its effect on macro F1 score and test accuracy.

Model 1

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 34, 34]             640
              ReLU-2           [-1, 64, 34, 34]               0
            Conv2d-3           [-1, 64, 34, 34]          36,928
              ReLU-4           [-1, 64, 34, 34]               0
       BatchNorm2d-5           [-1, 64, 34, 34]             128
         MaxPool2d-6           [-1, 64, 17, 17]               0
            Conv2d-7          [-1, 128, 15, 15]          73,856
              ReLU-8          [-1, 128, 15, 15]               0
       BatchNorm2d-9          [-1, 128, 15, 15]             256
           Conv2d-10          [-1, 128, 15, 15]         147,584
             ReLU-11          [-1, 128, 15, 15]               0
      BatchNorm2d-12          [-1, 128, 15, 15]             256
        MaxPool2d-13            [-1, 128, 7, 7]               0
           Conv2d-14            [-1, 256, 5, 5]         295,168
             ReLU-15            [-1, 256, 5, 5]               0
      BatchNorm2d-16            [-1, 256, 5, 5]             512
           Conv2d-17            [-1, 256, 5, 5]         590,080
             ReLU-18            [-1, 256, 5, 5]               0
      BatchNorm2d-19            [-1, 256, 5, 5]             512
           Conv2d-20            [-1, 256, 5, 5]         590,080
             ReLU-21            [-1, 256, 5, 5]               0
      BatchNorm2d-22            [-1, 256, 5, 5]             512
           Conv2d-23            [-1, 256, 5, 5]         590,080
             ReLU-24            [-1, 256, 5, 5]               0
      BatchNorm2d-25            [-1, 256, 5, 5]             512
        MaxPool2d-26            [-1, 256, 2, 2]               0
          Flatten-27                 [-1, 1024]               0
          Dropout-28                 [-1, 1024]               0
           Linear-29                  [-1, 256]         262,400
             ReLU-30                  [-1, 256]               0
          Dropout-31                  [-1, 256]               0
           Linear-32                  [-1, 128]          32,896
             ReLU-33                  [-1, 128]               0
          Dropout-34                  [-1, 128]               0
           Linear-35                   [-1, 64]           8,256
```

```
             ReLU-36                  [-1, 64]               0
          Dropout-37                  [-1, 64]               0
           Linear-38                   [-1, 7]             455
================================================================
Total params: 2,631,111
Trainable params: 2,631,111
Non-trainable params: 0
----------------------------------------------------------------
```

Test Accuracy: 43.005%

Train Accuracy: 91.31%

Model 2

After using the model above, due to the low test accuracy percentage, we started looking for more robust models. After looking through a few research papers, we found a few models such as ResNet, AlexNet and VGG Face-net. After looking at these models, we created our own structure by combining the parts of each of these models we felt would benefit our model. Then we tuned the hyper-parameters, and trained a better model than the previous one.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 46, 46]             640
              ReLU-2           [-1, 64, 46, 46]               0
            Conv2d-3           [-1, 64, 46, 46]          36,928
              ReLU-4           [-1, 64, 46, 46]               0
       BatchNorm2d-5           [-1, 64, 46, 46]             128
         MaxPool2d-6           [-1, 64, 23, 23]               0
            Conv2d-7          [-1, 128, 21, 21]          73,856
              ReLU-8          [-1, 128, 21, 21]               0
       BatchNorm2d-9          [-1, 128, 21, 21]             256
           Conv2d-10          [-1, 128, 21, 21]         147,584
             ReLU-11          [-1, 128, 21, 21]               0
      BatchNorm2d-12          [-1, 128, 21, 21]             256
        MaxPool2d-13          [-1, 128, 10, 10]               0
           Conv2d-14            [-1, 256, 8, 8]         295,168
             ReLU-15            [-1, 256, 8, 8]               0
      BatchNorm2d-16            [-1, 256, 8, 8]             512
           Conv2d-17            [-1, 256, 8, 8]         590,080
             ReLU-18            [-1, 256, 8, 8]               0
      BatchNorm2d-19            [-1, 256, 8, 8]             512
           Conv2d-20            [-1, 256, 8, 8]         590,080
             ReLU-21            [-1, 256, 8, 8]               0
      BatchNorm2d-22            [-1, 256, 8, 8]             512
        MaxPool2d-23            [-1, 256, 4, 4]               0
           Conv2d-24            [-1, 512, 2, 2]       1,180,160
             ReLU-25            [-1, 512, 2, 2]               0
      BatchNorm2d-26            [-1, 512, 2, 2]           1,024
```

```
           Conv2d-27            [-1, 512, 2, 2]       2,359,808
             ReLU-28            [-1, 512, 2, 2]               0
      BatchNorm2d-29            [-1, 512, 2, 2]           1,024
           Conv2d-30            [-1, 512, 2, 2]       2,359,808
             ReLU-31            [-1, 512, 2, 2]               0
      BatchNorm2d-32            [-1, 512, 2, 2]           1,024
        MaxPool2d-33            [-1, 512, 1, 1]               0
           Conv2d-34            [-1, 512, 1, 1]       2,359,808
             ReLU-35            [-1, 512, 1, 1]               0
      BatchNorm2d-36            [-1, 512, 1, 1]           1,024
           Conv2d-37            [-1, 512, 1, 1]       2,359,808
             ReLU-38            [-1, 512, 1, 1]               0
      BatchNorm2d-39            [-1, 512, 1, 1]           1,024
           Conv2d-40            [-1, 512, 3, 3]       2,359,808
             ReLU-41            [-1, 512, 3, 3]               0
      BatchNorm2d-42            [-1, 512, 3, 3]           1,024
        MaxPool2d-43            [-1, 512, 1, 1]               0
           Conv2d-44           [-1, 1024, 5, 5]       4,719,616
             ReLU-45           [-1, 1024, 5, 5]               0
      BatchNorm2d-46           [-1, 1024, 5, 5]           2,048
           Conv2d-47           [-1, 1024, 5, 5]       1,049,600
             ReLU-48           [-1, 1024, 5, 5]               0
      BatchNorm2d-49           [-1, 1024, 5, 5]           2,048
           Conv2d-50            [-1, 256, 5, 5]         262,400
             ReLU-51            [-1, 256, 5, 5]               0
      BatchNorm2d-52            [-1, 256, 5, 5]             512
           Conv2d-53            [-1, 512, 3, 3]       1,180,160
             ReLU-54            [-1, 512, 3, 3]               0
      BatchNorm2d-55            [-1, 512, 3, 3]           1,024
           Conv2d-56            [-1, 128, 3, 3]          65,664
             ReLU-57            [-1, 128, 3, 3]               0
      BatchNorm2d-58            [-1, 128, 3, 3]             256
           Conv2d-59            [-1, 256, 2, 2]         295,168
             ReLU-60            [-1, 256, 2, 2]               0
      BatchNorm2d-61            [-1, 256, 2, 2]             512
          Flatten-62                 [-1, 1024]               0
           Linear-63                  [-1, 256]         262,400
             ReLU-64                  [-1, 256]               0
           Linear-65                  [-1, 128]          32,896
             ReLU-66                  [-1, 128]               0
           Linear-67                   [-1, 64]           8,256
             ReLU-68                   [-1, 64]               0
           Linear-69                    [-1, 7]             455
================================================================
Total params: 22,604,871
Trainable params: 22,604,871
Non-trainable params: 0
```

Test Accuracy: 57.%