Machine Learning Engineer Nanodegree

Capstone Report

Shvet Chakra December 18th, 2020

Emotion recognition using facial expressions

Domain Background

Facial expressions and related changes in facial patterns give us information about the emotional state of the person and help to regulate conversations with the person. Moreover, these expressions help in understanding the overall mood of the person in a better way. Facial expressions play an important role in human interactions and non-verbal communication. Classification of facial expressions could be used as an effective tool in behavioural studies and in medical rehabilitation. Facial expression analysis deals with visually recognizing and analyzing different facial motions and facial feature changes.[1]

Motivation Nowadays, when technology has penetrated into the personal life of humans with mobile devices and interaction of machines with humans is carried out on the daily basis, it is getting important for the machine to recognize the emotion of the fellow human while interating with him. Recognition of emotion can help machines powered with artificial intelligence to enhance this interaction. Possible application where this project can be helpful in wellbeing of humans. As recognition & acceptance is the first step of any problem, this solution can help the user about their mood and help them in taking appropriate step. For eg. we have facial recognition phone security feature to lock/unlock the device, if the device captures the face and determine the emotion of the users on several intervals and then give a happy mood percentage for the day. It can help them in recognizing about their mood if it is not appropriate and motivate them to take necessary action.

Problem Statement

Facial expressions play an important role in recognition of emotions and are used in the process of non-verbal communication, as well as to identify people. They are very important in daily emotional communication, just next to the tone of voice [2]. They are also an indicator of feelings, allowing a man to express an emotional state [3]. People, can immediately recognize an emotional state of a person. As a consequence, information on the facial expressions are often used in automatic systems of emotion recognition [4]. The aim of the project is to recognize seven basic emotional states: neutral, joy, surprise, anger, sadness, fear and disgust based on facial expressions.[5] Numerous investigators have used neural networks for facial expression classification. The performance of a neural network depends on several factors including the initial random weights, the training data, the activation function used, and the structure of the network including the number of hidden layer neurons, etc. Here I will try to create a CNN based neural network model to classify the images into their seven basic emotional states: neutral, joy, surprise, anger, sadness, fear and disgust.

Datasets and Inputs

The project has been chosen from a kaggle competion and hence the same datset will be used here also to train

and test the model for the proposed problem.
https://www.kaggle.com/c/emotion-detection-from-facial- expressions/data

We need to check the images in our dataset for their sizes and number of color components

	Width	Height	Components
0	350.0	350.0	3.0
1	350.0	350.0	3.0
2	350.0	350.0	3.0
3	350.0	350.0	3.0
4	350.0	350.0	3.0

Now we need to determine the minimum & maximum width and height from all the image sizes

	Width	Height	Components
max	536.000000	441.000000	3.0
min	24.000000	18.000000	3.0
mean	335.768698	333.548039	3.0
std	58.700772	63.898948	0.0

If we check few of the entries in the dataset.

	user.id	image	emotion
0	628	facial-expressions_2868588k.jpg	anger
1	628	facial-expressions_2868585k.jpg	surprise
2	628	facial-expressions_2868584k.jpg	disgust
3	628	facial-expressions_2868582k.jpg	fear
4	dwdii	Aaron_Eckhart_0001.jpg	neutral

Let us also check the different labels present in our dataset and their count to determine whether the dataset is equally distributed or not.

['anger', 'surprise', 'disgust', 'fear', 'neutral','happiness','sadness', 'contempt', 'NEUTRAL', 'SADNESS', 'DISGUST', 'FEAR', 'SURPRISE', 'ANGER', 'HAPPINESS']

neutral	6717
happiness	5309
HAPPINESS	387
surprise	356
anger	228
DISGUST	195
NEUTRAL	151
SADNESS	144
sadness	124
ANGER	24
fear	13
disgust	13
SURPRISE	12
contempt	9
FEAR	8

Now we can see the various emotion categories and their distribution in the dataset. As we can see that few of the labels are in lowercase and few are in uppercase, let's bring them into one format and observe their counts again.

neutral	6868
happiness	5696
surprise	368
sadness	268
anger	252
disgust	208
fear	21
contempt	9

The contempt and disgust seem a similar emotion so we can merge them into one.

neutral	6868
happiness	5696
surprise	368
sadness	268
anger	252
disgust	217
fear	21

It can be easily noticeable that the emotion categories are not equally distributed here. Hence, we need to drop some images from neutral and happiness category to make the distribution unbiased for any categories.

```
#shuffle the dataset , iterate over it and select 500 images from neutral and happiness category
image_info = image_info.sample(frac=1).reset_index(drop=True)
```

```
#move above selected images into a separate dataset folder
neutral = 0
happy = 0
selectedImages = pd.DataFrame(columns=["image","emotion"])
for index, row in image_info.iterrows():
    if row["emotion"] == "happiness": happy += 1

        if happy > 500:
            continue
    elif row["emotion"] == "neutral": neutral += 1

        if neutral > 500:
            continue
    selectedImages = selectedImages.append(pd.Series(row),ignore_index=True)
```

```
neutral      500
happiness    500
surprise     368
sadness      268
anger        252
disgust      217
fear         21
```

Metrics

We can use following metrics to evaluate our model

1. Classification Accuracy
2. Confusion Matrix
3. F1 Score

Classification Accuracy We can measure the classification accuracy of the proposed model and compare it with the benchmark models. The higher the accuracy percentage the better the model is performing. Classification accuracy has been captured on test data set by predicting the emotions using the trained model and matched it with the true classes.

Confusion Matrix is used to illustrate the performance of the classification model.

F1-Score the F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive.[https://en.wikipedia.org/wiki/F-score]

We will capture the F1-Score for each category of emotions and evaluate it with respect to bench mark model's F1-score.

Benchmark Model

For benchmarking we can use Alexnet or VGG16 models to perform some benchmarking. https://www.imperial.ac.uk/intelligent-digital-systems/cnn-benchmark-suite/

VGG16 Model:

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mo
de=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
)
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1
))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mo
de=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
```

```
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Alexnet Model:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2)
)
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mo
de=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2)
)
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mo
de=False)
```

```
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1
))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1
))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_m
ode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Originally both the models are pretrained on an imagenet dataset which consists of 1000 classes to categories the images into. For our purposes, we have only 7 classes to classify hence we need to change the final out_feature of the model to 7.

We have changed the classifier part of both the benchmark model as per our model.

```
    model = models.vgg16(pretrained=True)
    model.classifier = nn.Sequential(nn.Linear(25088, 4096),
                                     nn.ReLU(),
                                     nn.Dropout(0.2),
                                     nn.Linear(4096, 1024),
                                     nn.ReLU(),
                                     nn.Dropout(0.2),
                                     nn.Linear(1024,512),
                                     nn.ReLU(),
                                     nn.Dropout(0.2),
                                     nn.Linear(512, 7))

    model = models.alexnet(pretrained=True)
    model.classifier = nn.Sequential(nn.Linear(9216, 4096),
                                     nn.ReLU(),
                                     nn.Dropout(0.2),
                                     nn.Linear(4096, 1024),
                                     nn.ReLU(),
                                     nn.Dropout(0.2),
  nn.Linear(1024,512),
  nn.ReLU(),
  nn.Dropout(0.2),
nn.Linear(512, 7))
```

Solution Statement

Convolution neural networks are specialized neural networks for processing data that has a grid-like topology. Examples include time-series data and image data. Convolution leverages three important ideas that can help improve a machine learning system; sparse interactions, parameter sharing and equivariant representations.
One major advantage of using CNNs over NNs is that you do not need to flatten the input images to 1D as they are capable of working with 2D images data. This helps in retaining the “spatial” properties of images.

Project Design

Technically, deep learning is based on CNN models to train and test the each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply some activation function to classify an object with probabilistic values between 0 and 1.

The project can be mainly classified into following steps:

1) Load and preprocess the image dataset:

The dataset will be analyzed for the image attributes and the list of emotions labeled to them. Conversion of labels to lowercase and merge few labels into one has been done.

To remove the biasness because of inequalities in the number of images for each category, I have selected limited number of images from heavily filled categories.

neutral	6868
happiness	5696
surprise	368
sadness	268
anger	252
disgust	217
fear	21

neutral	**500**
happiness	**500**
surprise	**368**
sadness	**268**
anger	**252**
disgust	**217**
fear	**21**

Then we will divide the dataset into three sets, train set, validation set , and test set with 60:20:20 ratio.
For the training the model, we will apply some general image transformations such as random scaling, cropping, and flipping, along with center cropped to 224 by 224 pixels.

```
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                        transforms.RandomResizedCrop(224),
                                     transforms.RandomHorizontalFlip(),
                                     transforms.ToTensor(),
                                     transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                                     std=[0.229, 0.224, 0.225])])

   valid_transforms = transforms.Compose([transforms.Resize(255),
                                        transforms.CenterCrop(224),
                                     transforms.ToTensor(),
                                     transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                                     std=[0.229, 0.224, 0.225])])
```

Next no scaling or rotation transformations will be performed on validation set and test set but need to resize then crop the images to the appropriate size to evaluate the model's accuracy on data it hasn't seen yet.

2) Building and training the classifier

In our proposed model, we have used four convolution 2d layers where each layer is separated by a batch normalization layer, activation layer, and pooling layer.

Convolution 2D layer extracts the features from the provided input image by moving the kernel window over the image with specified moving step.

Batch normalization layer helps in stabilizing the learning process and reduces the learning epochs.

Activation function is used to transform the summed weighted input into the activation of the node. In our model we have used ReLU or **rectified linear activation function for short is an activation** function that will output the input directly if it is positive, otherwise, it will output zero.

Pooling layers are used to reduce the dimensions of the feature maps. We have used max pooling layer which takes the maximum value from the features.

```
class EmotionClassifier(nn.Module):
    """
    This is the simple CNN model we will be using to perform emotion classification for seven
emotions
    """

    def __init__(self):
        """
        Initialize the model by settingg up the various layers.
        """
        super(EmotionClassifier, self).__init__()

        self.cnn_layers = nn.Sequential(
            # Defining a 2D convolution layer
            Conv2d(3, 32, kernel_size=5, stride=2, padding=2),
            ReLU(inplace=True),
            # adding batch normalization
            BatchNorm2d(32),
            MaxPool2d(kernel_size=2, stride=2),
            # adding dropout
            Dropout(p=0.25),
            # Defining another 2D convolution layer
            Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            ReLU(inplace=True),
            # adding batch normalization
            BatchNorm2d(64),
            MaxPool2d(kernel_size=2, stride=2),
            # adding dropout
            Dropout(p=0.15),
            # Defining another 2D convolution layer
            Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            ReLU(inplace=True),
            # adding batch normalization
            BatchNorm2d(128),
            MaxPool2d(kernel_size=2, stride=2),
        # adding dropout
        Dropout(p=0.15),
        # Defining another 2D convolution layer
        Conv2d(128, 64, kernel_size=3, stride=1, padding=1),
        ReLU(inplace=True),
        # adding batch normalization
        BatchNorm2d(64),
        MaxPool2d(kernel_size=2, stride=2),
        # adding dropout
        Dropout(p=0.15),
    )

    self.linear_layers = nn.Sequential(
                            Linear(64 * 7 * 7, 1024),
                            nn.ReLU(),
                            nn.Dropout(0.2),
                            nn.Linear(1024, 512),
                            nn.ReLU(),
                            nn.Dropout(0.2),
                            nn.Linear(512,128),
                            nn.ReLU(),
                            nn.Dropout(0.2),
                            nn.Linear(128, 7)
    )
```
I have also used the dropout so that model does not overfit over the provided training dataset.

3) Refinement

- During the initial training round, I have obtained the validation accuracy of around 60 % but when the model has been evaluated on the test-set, I barely achieved 30% accuracy. On re-evaluating the architecture of the model and data augmentation, I have observed that passing the grayscale images to the model only reducing the number of input features to the model, so I have dropped the grayscale conversion transformation from the preprocessing step.
- Also, the gap between the training loss and validation loss was getting higher which made me realize, the model might be getting overfit which was resolved by adding more dropouts between the layers.
- Also, added the random jitter in the training set to make the model more robust but during the training of the model did not achieve the previous best accuracy so that has been removed from the final solution.
- Some refinements have also been done related to learning rate, number of convolution layers and training epochs.

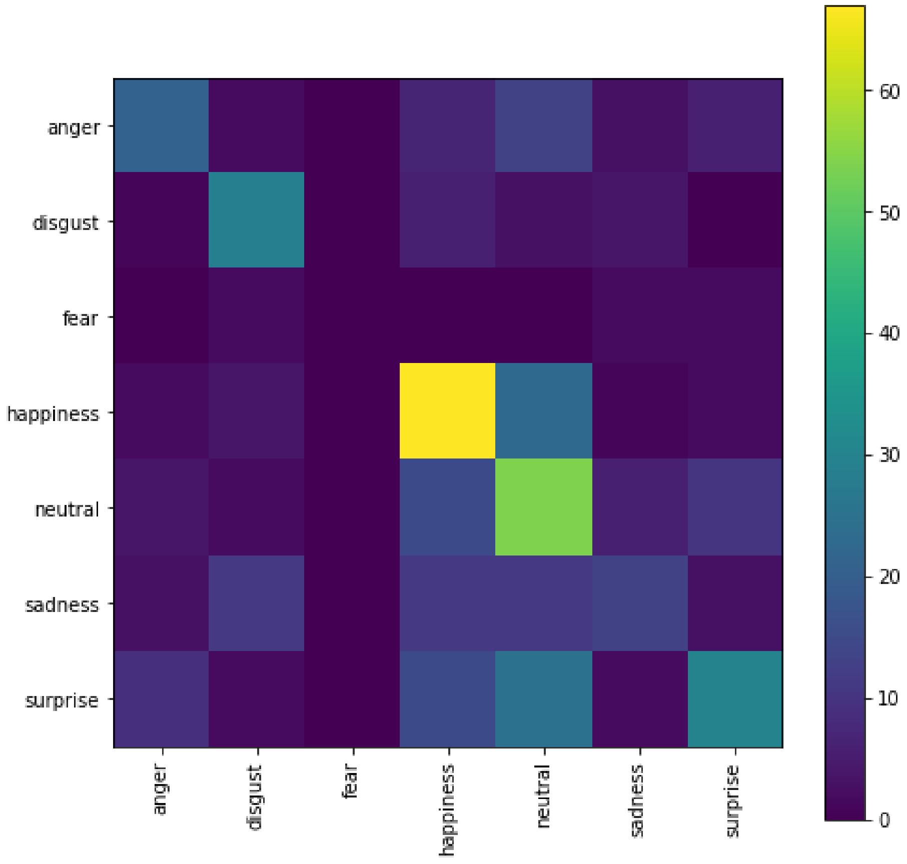
Model Evaluation and Validation

```
Classification Report of proposed model
              precision    recall  f1-score   support

       anger       0.53      0.40      0.46        52
     disgust       0.56      0.67      0.61        43
        fear       0.00      0.00      0.00         6
   happiness       0.55      0.68      0.61        99
     neutral       0.42      0.59      0.49        91
     sadness       0.42      0.25      0.31        52
    surprise       0.57      0.36      0.44        83

    accuracy                           0.50       426
   macro avg       0.43      0.42      0.42       426
weighted avg       0.50      0.50      0.49       426

Confusion Matrix
[[21  2  0  7 13  3  6]
 [ 1 29  0  6  3  4  0]
 [ 0  2  0  0  0  2  2]
 [ 2  4  0 67 23  1  2]
 [ 4  2  0 15 54  6 10]
 [ 3 11  0 11 11 13  3]
 [ 9  2  0 15 25  2 30]]
```



Classification Report of Alexnet

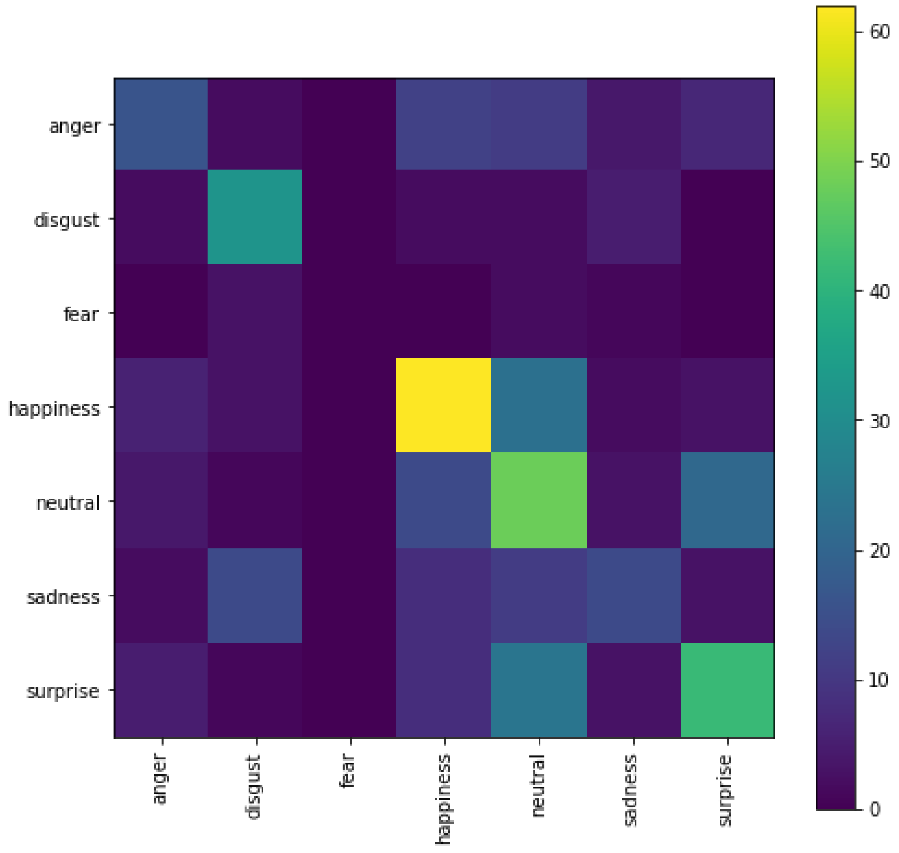
```
              precision    recall  f1-score   support

       anger       0.46      0.31      0.37        52
     disgust       0.57      0.74      0.65        43
        fear       0.00      0.00      0.00         6
   happiness       0.58      0.63      0.60        99
     neutral       0.40      0.53      0.45        91
     sadness       0.44      0.27      0.33        52
    surprise       0.55      0.51      0.53        83

    accuracy                           0.50       426
   macro avg       0.43      0.43      0.42       426
weighted avg       0.50      0.50      0.49       426
```

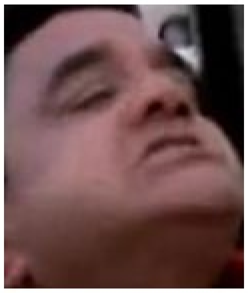
Confusion Matrix

```
[[16  2  0 12 11  4  7]
 [ 2 32  0  2  2  5  0]
 [ 0  3  0  0  2  1  0]
 [ 6  3  0 62 23  2  3]
 [ 4  1  0 14 48  3 21]
 [ 2 14  0  8 11 14  3]
 [ 5  1  0  8 24  3 42]]
```



- If we observe the classification accuracy for proposed model and alexnet benchmark model, both the models are delivering the same accuracy.
- The f1-score for happiness, anger and neutral emotion is better than alexnet model while slightly lower for disgust and surprise.
- The test accuracy of VGG16 model is quite low with only 21.4 %

Let us now visualize few of the results from the test set and see the prediction.
The below images are captured from Anger class and their predicted class has been displayed.

	Input Image	Predicted Class where true class is anger
0		anger
1		sadness
2		anger

The below images are captured from happiness class and their predicted class has been displayed.

	Input Image	Predicted Class where true class is happiness
0		happiness
1		happiness
2		sadness

Conclusion

As we have seen in the results and evaluation of the model, the proposed model is performing very close to one benchmark model.
As the dataset is limited (only around 2300 images) achieving such accuracy is quite good and the model can be easily trained on some more data and can be a useful utility in our daily lives where we are in constant demand of making the interaction of machines with human as real as possible.

References:

[1] https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-8-16

[2] Ratliff M. S., Patterson E., Emotion recognition using facial expressions with active appearance models, Proceedings of the Third IASTED International Conference on Human Computer Interaction, ACTA Press, Anaheim, CA, USA, 2008, 138–143.
[3] Tian Y. I., Kanade T., Cohn J. F., Recognizing action units for facial expression analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), no. 2, 97–115.
[4] Mao Q., Pan X., Zhan Y., Shen X., Using Kinect for real-time emotion recognition via facial expressions, Frontiers Inf Technol Electronic Eng, 16 (2015), no. 4, 272–282.
[5]International Conference on Computational Science, ICCS 2017, 12-14 June 2017,Zurich, Switzerland Emotion recognition using facial expressions, Paweł Tarnowski, Marcin Kołodziej, Andrzej Majkowski, Remigiusz J. Rak