Machine Learning Engineer Nanodegree

Capstone Project: Facial Expression Recognition

Arun Rajora

June 1st,2017

I. Definition

Project Overview

It has been easy for humans to recognize faces and expressions on a face. It helps us in communication as it adds more context to a conversation or might convey a quick response to a situation. Adding such capabilities to machines is of great interest as it gives more understanding of the user. In this project, we will attempt to create a model which can recognize expressions on a person's face.

According to a paper by Hatice Gunes and Hayley Hung (paper can be found on http://www.sciencedirect.com/science/article/pii/S026288561630049X#bbb0005), "Automatic facial expression recognition is a multidisciplinary research field that spans across computer vision, machine learning, neuroscience, psychology and cognitive science. In automatic facial expression recognition research, the most common approach is to classify continuous expressive facial displays according to specific labels, categories or dimensions. Ekman's theory of basic emotions is the most commonly used scheme when creating vision-based systems that attempt to recognize facial expressions of emotions and analyze human affective behavior. The main assumption is that emotions that are felt inside the body are displayed externally via the face, and these in turn can be universally mapped into the six categories of happiness, sadness, surprise, fear, anger and disgust."

For this problem, the data consists of 48x48 pixel grayscale images of faces where facial expression on each face has been categorized in to one of the following categories- Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral.

Problem Statement

In this project, the objective is to classify the expression on a human face in an image into the following seven classes- Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral.

We have a dataset consisting of images of people's face which have been labelled with the appropriate expression the face depicts. Our goal is to train a classifier which can classify faces in an image into the expression expressed by that face.

The dataset has been obtained from a contest on Kaggle. The link for the dataset is https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data .

> *(Citation for the dataset: "Challenges in Representation Learning: A report on three machine learning contests." I Goodfellow, D Erhan, PL Carrier, A Courville, M Mirza, B Hamner, W Cukierski, Y Tang, DH Lee, Y Zhou, C Ramaiah, F Feng, R Li, X Wang, D Athanasakis, J Shawe-Taylor, M Milakov, J Park, R Ionescu, M Popescu, C Grozea, J Bergstra, J Xie, L Romaszko, B Xu, Z Chuang, and Y. Bengio. arXiv 2013.)*

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

According to the contest page, "This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest."

For this project, convolutional neural network will be used to classify the face in an image into the expression expressed by the face. Convolutional neural network has been very effective in such kind of problems due to- translation invariance (the model learns the features irrespective of its position in the image) and the model is effective in learning and representing complex abstract concepts.

In the leaderboard of the contest page (https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard), the top 15 teams had a score of at least 0.57899 (private leaderboard). We will attempt to achieve similar results and if possible get better results.

Metrics

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value and $n_{samples}$ is the number of samples, then the accuracy_score is-

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Since it is a classification problem and our objective is to predict each label as accurately as possible, the accuracy score gives a justified score to measure it. If more samples get predicted accurately, we will get a higher score.

It would be better to know if the model failed to recognize a particular label or falsely recognizes a particular label. Hence, confusion Matrix will also be calculated so as to analyze the results of the model more in depth.

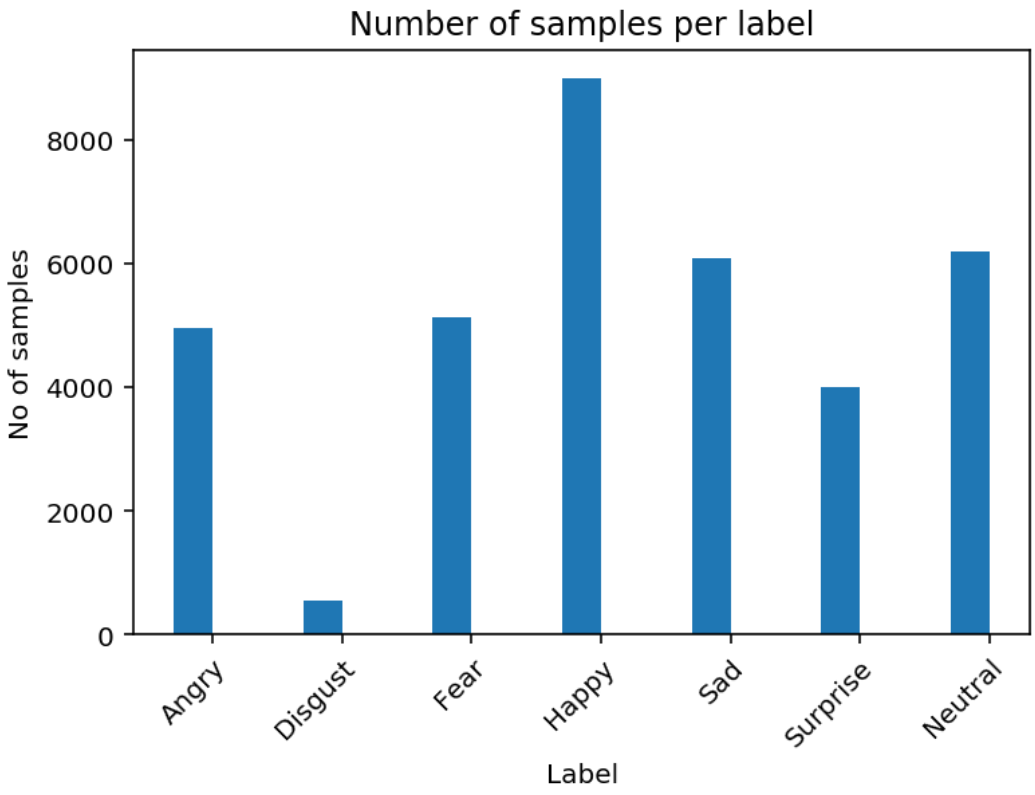
II. Analysis

Data Exploration and Exploratory Visualization

The dataset can be obtained on the following url https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data . The data consists of 48x48 pixel grayscale images of faces where facial expression on each face has been categorized in to one of the following categories- Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral.

The file fer2013.csv contains three columns, "emotion", "pixels" and “Usage”. The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string is a space-separated pixel values in row major order. The “Usage” column contains one of the following- “Training ”, “PublicTest ”, “PrivateTest”. It depicts whether the data belongs to training, public or private test set. The training set, public test set and private test set consists of 28,709, 3,589 and 3,589 examples respectively.

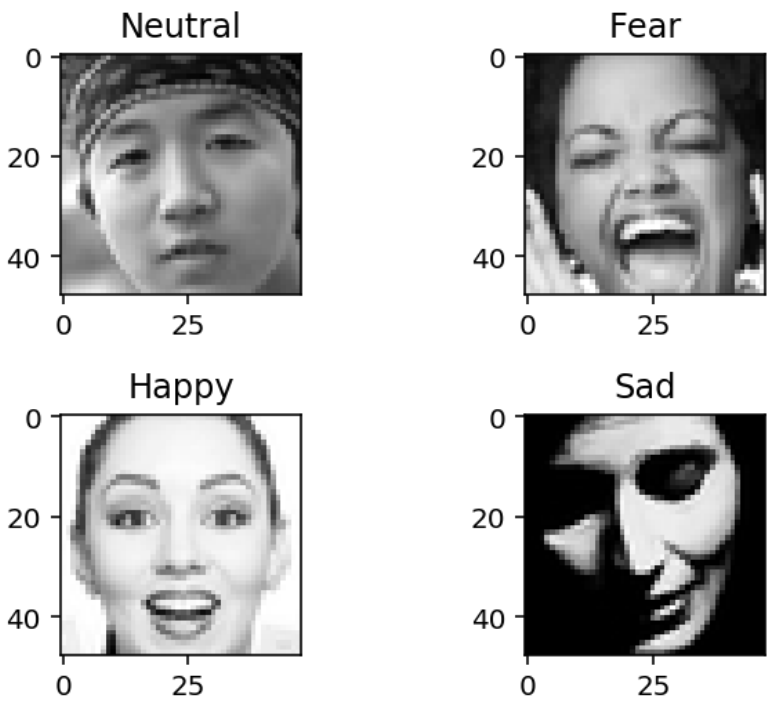
On looking at the number of samples found for each label in the training set, the number of samples for each labels were- 4953 for ‘Angry', 547 for 'Disgust', 5121 for 'Fear', 8989 for 'Happy', 6077 for 'Sad', 4002 for 'Surprise' and 6198 for 'Neutral'.



The number of samples for each category is unevenly distributed. The number of samples for disgust is very low (547). This will definitely affect our model in a bad way. In order to deal with it, we will add more sample points by transforming already present images in various ways and adding them to the dataset.

Of 35887 samples, there are 28709 samples in training set (80% of the data), 3589 in validation set (10% of the data) and 3589 in test set (10% of the data).

Some of the sample images are below-



Algorithms and Techniques

For training a model for this problem, CNN (Convolutional neural network) will be used. CNN seems to be a good solution for this problem due to two main properties. Firstly, CNN does a good job at learning abstract concepts. Example, the model might learn different kinds of lines and color patches on the first layer and different shapes on the next layer. The model then might learn in further layers to recognize faces and then smiles and frowns and different expressions. Secondly, the model has an interesting property of translation invariance. Translation invariance means that a feature might be present in any part of the image and the model will treat them equally. For example: a face with a smile might be present on top corner of an image or in the center, it does not matter. The only thing that matters is that a face with a smile is present.

Convolutional neural network consists of multiple convolutional layers followed by pooling operations followed by fully connected layer. To prevent the model to overfit the data, dropping layers will also be used. For activation function, ReLU (rectified linear units) will be used. The layers have been explained below:

- Convolutional layer: It is used to generate 2D filters on the data to generate various features. In the first layer, there might be filters which recognize lines. As we move deep into the network, there might be filters which recognize smiles, frown and other facial expressions.
- Pooling Layer: It is used to reduce (down-sample) the spatial size of the representation to reduce the number of parameters in the network.

- Flatten Layer: It is used to flatten the feature map to lower dimension to feed it to the fully connected network.
- Fully Connected Layer: It is used to perform the classification on the features extracted by the convolutional layers.
- There are few hyper parameters like: keep probability- to stop the Fully connected layers from overfitting, we use dropout layer which based on the keep_probability, might ignore nodes. Batch size: Small batches of training data is created specified by batch size. Epoch is the number of times the whole training data (all of the training batches) are trained.

Benchmark

In order to benchmark our results, we will first compare out results with a model which uniformly and randomly classifies the images. The result of this random classifier is 0.14210086375.
To further compare our results, we will compare our results with the scores on the private leaderboard of the contest page. The top 15 scorers had a score of at least 0.57899 on the private leaderboard.

III. Methodology

Data Preprocessing

The following steps have been taken to preprocess the data:

1. ***Train/Validate/Test split:*** The 35887 samples have been split into 28709 training samples (80% of the data), 3589 validation samples (10% of the data) and 3589 test samples (10% of the data).
2. ***Reshape and Normalization:*** The image was originally as a string in the input data containing space separated pixel values between 0 to 255. The string is converted into 2d matrix and all the pixel values have been normalized to values between 0 to 1.
3. ***Adding more data in training samples:*** The images are flipped horizontally, rotated left and right by random values in range 10 and 30 degrees and blurred a little. These values are then added to the training samples. Since the label for an image does not change on doing these transformations, the label values remain the same.
4. ***Randomization:*** The training dataset is randomly shuffled.
5. ***One Hot Encoding:*** The labels have been one hot encoded for the dataset.
6. ***Batching:*** The training dataset is divided into 15 batches each consisting of 10,000 images for easier processing.

Implementation

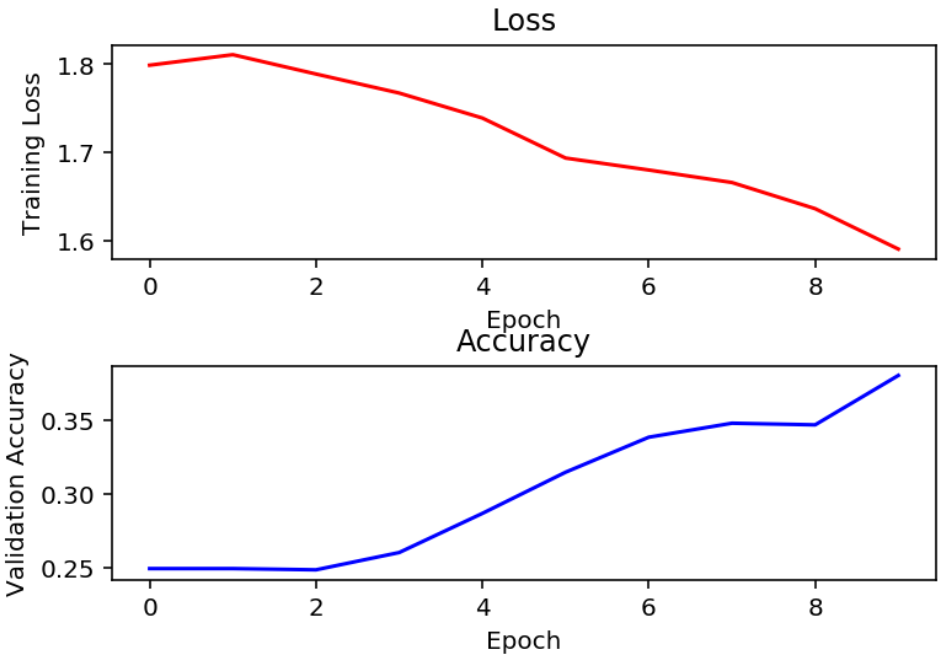
At this stage, we have 1,43,545 images for training (due to various transformations on the original dataset). Now a convolutional neural network is trained. The architecture of the network is as follows:

1. There is an input layer of dimension 48x48x1.
2. Then there is 4 convolutional layers of size 64, 128, 256 and 512 outputs respectively. Each convolutional layer is followed by max pooling operation.
3. Then there is a dropout followed by a flatten layer.
4. Then there is a dropout followed by two fully connected layers of size 512 and 256 respectively. Then there is again a dropout followed by two fully connected layers of size 128 and 64 respectively. This is followed by a dropout.
5. Then there is a final output layer.
6. The activation function used is ReLU.
7. The cost function used is SoftMax cross entropy.
8. The optimizer used is Adam Optimizer.
9. The initial bias is set to zero.
10. The initial weights used are random values from a normal distribution of mean 0 and standard deviation of Sqrt (2/ (N)) where N is the number of input of the previous layer.
11. Training loss and validation accuracy is calculated for each epoch.
12. First, the data is trained for one single batch of data.
13. Then the data is trained for the whole dataset.
14. Finally, the model is saved and results analyzed.

Refinement

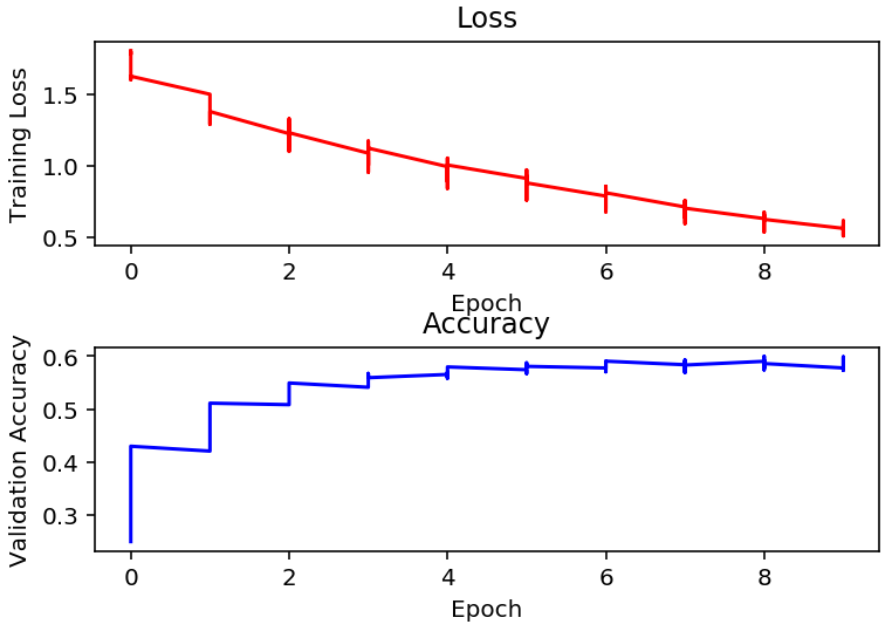
Different values for hyperparameters were used and the best training accuracy was found on 10 epochs with batch size 256 and keep probability of 0.9. The various hyperparameters that were used were the combination of 5 and 10 epochs, 256, 1024 and 2048 batch sizes and 0.5,0.6,0.7,0.8,0.9 keep probability. Due to large time taken by training time, higher number of epochs and lower keep probabilities were not used.

After training the model on just a single batch of the data for 10 epochs with batch size 256 and keep probability 0.9, we get an accuracy score of 0.380050182343. The training loss decreases and the accuracy score increases. This can be verified from the graph below.



On training on a single batch for other various hyperparameters, the training score varied from 0.24 to 0.35. Hence, the complete dataset was trained on just the hyperparameters which gave the best results.

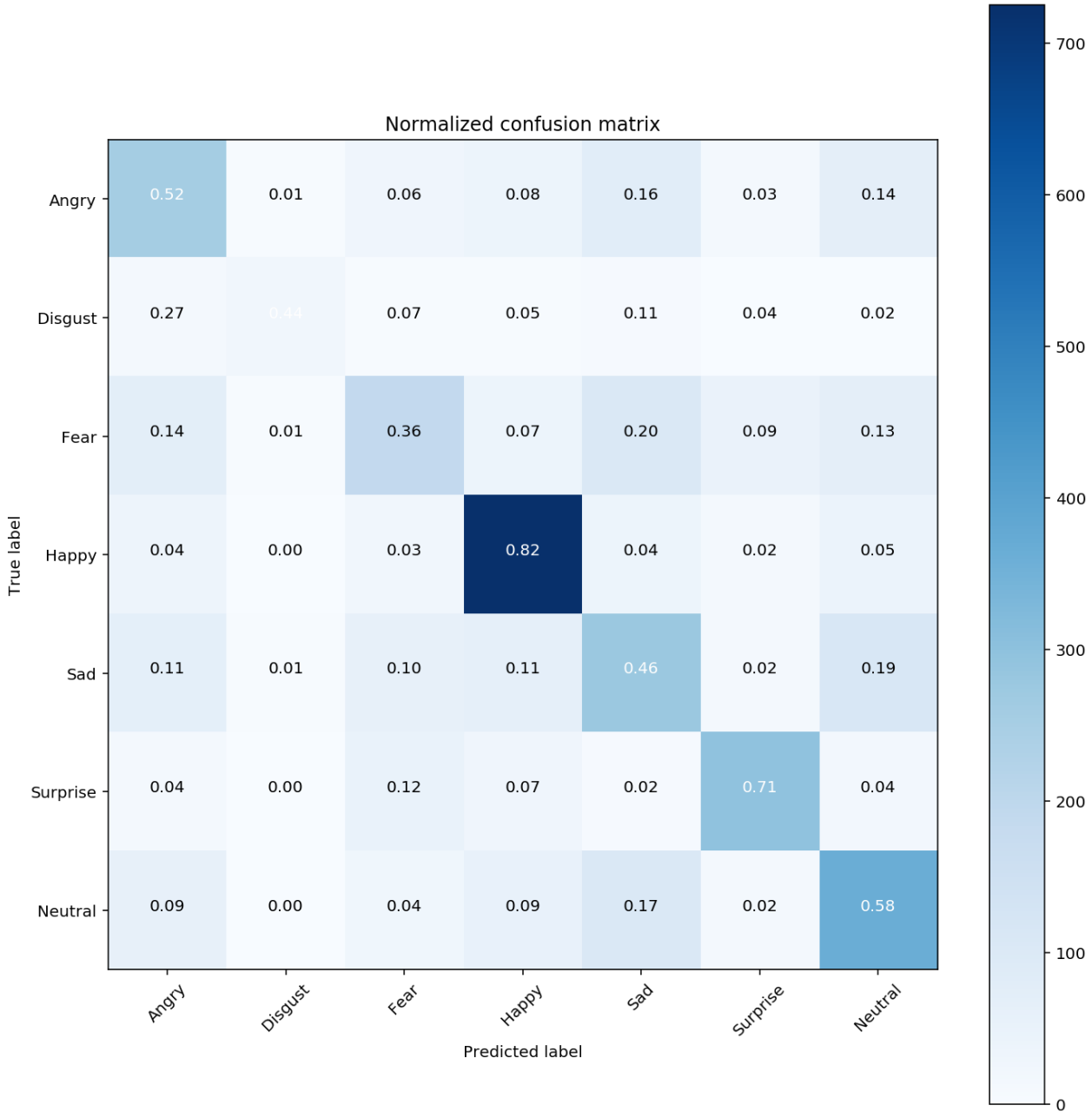
After training the model on complete dataset for 10 epochs, the validation accuracy turned out to be 0.594037353992. The training loss was still decreasing while the validation accuracy almost got flat at around 0.59.



IV. Results

Model Evaluation and Validation

The model is now evaluated on test dataset and it gets an Accuracy score of 0.593480050564. The confusion matrix is also calculated and plotted. The confusion matrix shows that the model performs best for happy expression and worst for disgust expression. This can be correlated to the amount of data present for these labels. The plotted confusion matrix is:



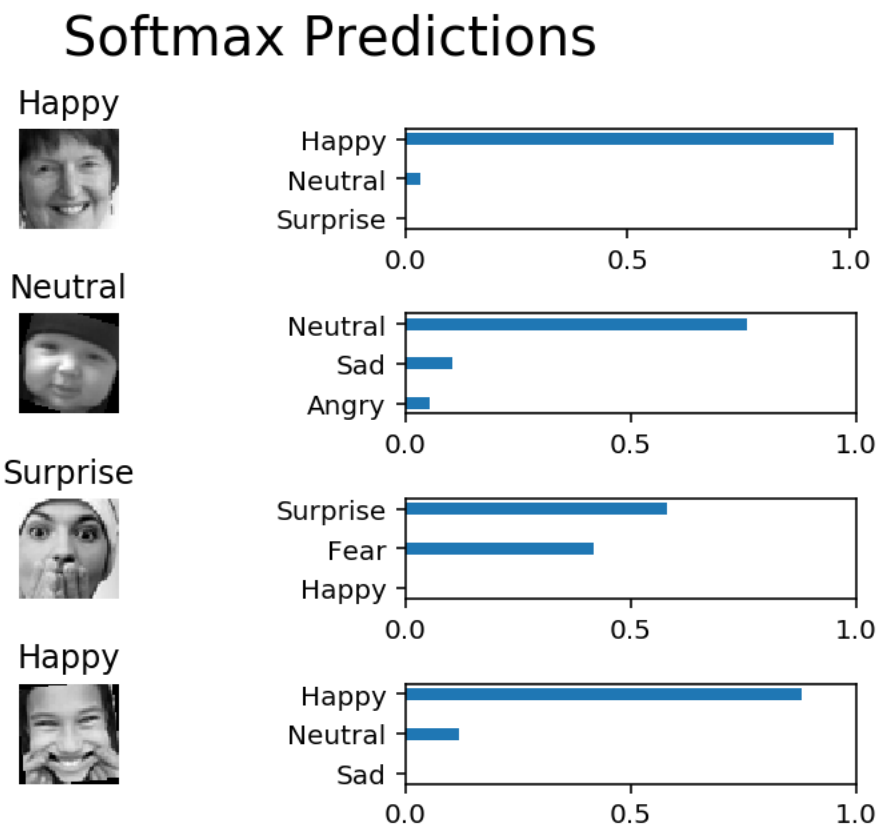
Justification

The random classifier used for benchmarking had an accuracy score of 0.14210086375. Our model exceeds that score by a huge difference with an accuracy score of 0.593480050564.
To further compare our results, we will compare our results with the scores on the private leaderboard of the contest page. The top 15 scorers had a score of at least 0.57899 on the private leaderboard. Our score is 0.593480050564 which is better than that. But our score is still less than those of winners. This shows that there is still a lot of scope of improvement.

V. Conclusion

Free-Form Visualization

The confusion matrix shown above tells a lot about the model. Model did relatively well for labels which had more samples than labels with relatively less samples. The recall and precision of each label can also be visualized. Further, some random images have been plotted and their probability scores for being a label is also visualized as shown below.



Reflection

It can be said that convolutional neural network is an appropriate model for image recognition. In this project, the input images are preprocessed, then the data is sent to a convolutional neural network whose architecture is described above. Further, the model is evaluated using the stated metric.

The interesting aspect of this project was tuning the hyperparameters. On using higher probabilities of the dropout, the model trains faster but very soon, it begins to overfit and there is no significant improvement. On setting those probability to a lower value, the model learns relatively slow, but does not overfit the data that fast.

The model meets the expectation for being a possible solution for the problem. But, the model has a lot of scope of improvement in different areas as discussed in improvement section below.

Improvement

Some of the areas that I think can bring improvement to the model are-

1. The dataset had different number of samples for various labels. For 'disgust', it had very low number of samples. This affected the model in a bad way which can be seen in the confusion matrix. The data can be made uniformly distributed by adding more data from other sources like scrapping some website and labelling them. Although it is a tedious task, but data is a key component for machine learning.
2. The model was overfitting after 5 epochs. The overfitting can be reduced by using regularization techniques like L2 regularization.
3. The learning rate can be made low and drop probability can be decreased. This will possibly improve the accuracy score as it will reduce the amount of overfitting done by the model. But this will significantly increase the training time.
4. The research paper found on this link- https://arxiv.org/abs/1307.0414 , talked about the winner using primal objective of an SVM as the loss function for training. Such methods can also be explored.
5. The model could be trained for more epochs as long as it does not start overfitting.
6. The architecture of the network (number of layers and inputs) could also be changed to look at the effect on the performance.