# Springboard Data Science Career Track

# Capstone Project II Final Report

# Facial Expression Recognition.

Rahul Ambati

June 2018

1. Introduction.
2. Approach.
   1. Data Acquisition and Wrangling.
   2. Data Exploration.
   3. Modeling and Analysis.

3. Findings.

4. Ideas for Further Research.

5. Client Recommendations.

6. Resources.

# Introduction.

Human facial expressions are complex. We are well trained in reading different expressions, just a few years old baby can tell the difference between happy and sad. As we grow up we see more than happy or sad expressions, we see surprise, fear, anxiety and many more.

A question arises can a computer do the same, recognizing facial expressions of a person?

Intention:

If a person can understand facial expression through experience and training so can the computer. Intention is to create a computer model which can be useful in predicting or recognizing a person’s facial expression.

Client:

Any application who wants to know a person facial expression before and after an event or activity.

Example of potential clients be, **Retail** which can be used to evaluate customer interests or in **Healthcare** to see patients emotional state or in **Entertainment** to monitor audience’s expressions during an entertaining performance.

# Approach.

The problem states to recognize facial expression of a person, to solve the problem, we need to find a model which can predict a person’s facial expression.

# Data Acquisition and Wrangling.

The data has been acquired from Kaggle dataset [Facial Expression Recognition Challenge](https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge).

The dataset consists of expression, image, test category. And the dataset is clean, no cleaning was necessary.

# Data Exploration.

There are 7 facial expressions/emotions classified in the dataset.

* + 1. *Angry*
    2. *Disgust*
    3. *Fear*
    4. *Happy*
    5. *Sad*
    6. *Surprise*
    7. *Neutral*

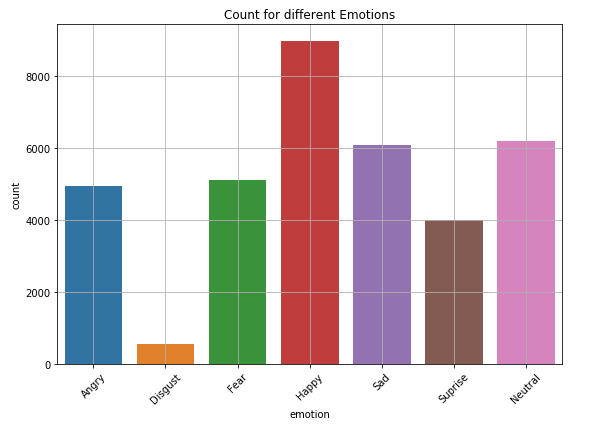


Figure 1. Number of samples for each emotion/expression.

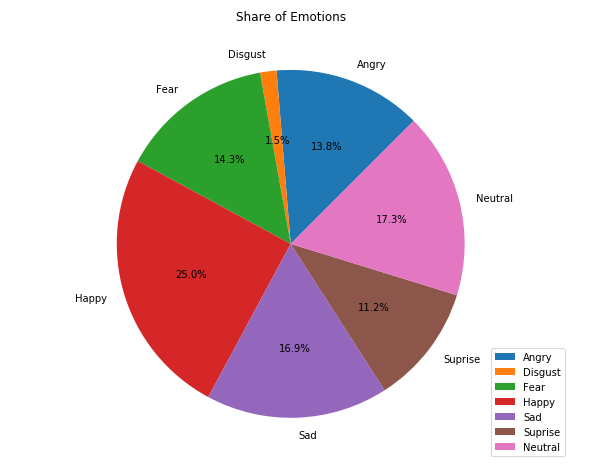


Figure 2. Share of each expression/emotion in the dataset.

From the above figures we see the *Disgust* has less number of examples.

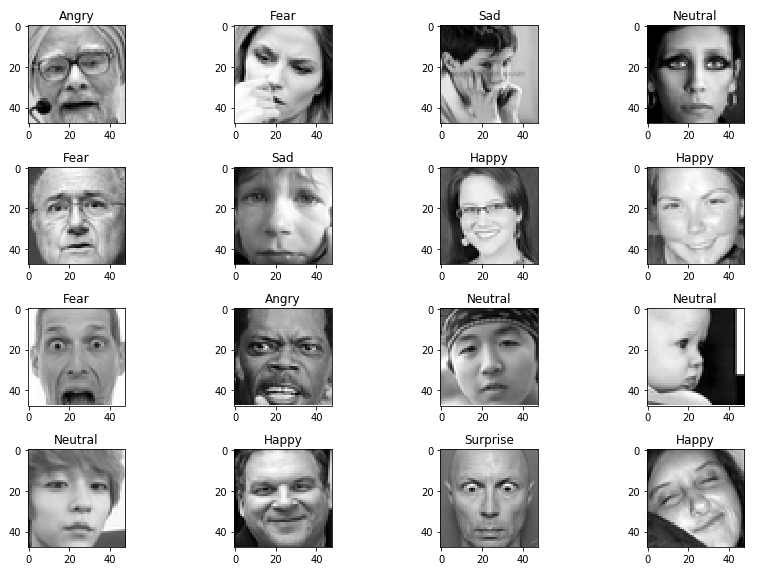


Figure 3. Sample images with expressions.

Image are 48x48 grayscale pixels.

# Modeling and Analysis.

Deep Learning is a popular technique used for image classification problems. Convolutional Neural Network (CNN) was used to create neural network (model) architecture. CNNs are known to imitate how our brains work in analyzing images.

A typical architecture of a CNN contains input layer, convolutional layers, dense layers (fully-connected layers) and output layer (Figure 4). Layers are stacked linearly in a sequence.

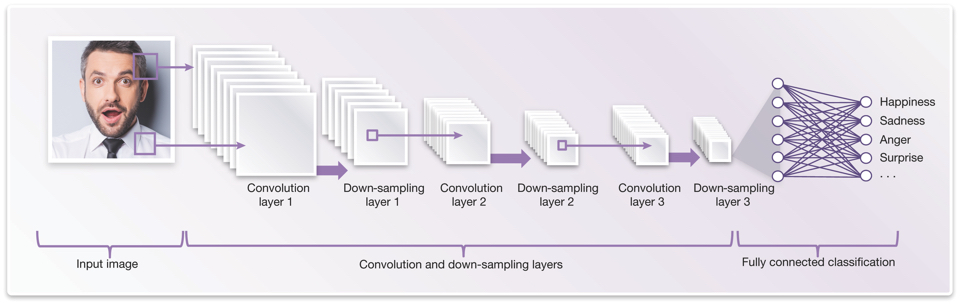


Figure 4: CNN[[1]](#footnote-1) architecture for Facial Expression Recognition.

Keras Sequential[[2]](#footnote-2) was used to build and design the model.

Input layer: It input layer has to be of fixed dimensions, must be pre-processed before fed to the layer. The images are resized to 48x48 pixels and reduced to one-dimension color image as (48,48,1) numpy array.

Convolutional layer: A convolution generates feature maps and shows features (enhanced pixels), for example an edge, a curve or some pattern. Each feature map is generated by applying filter across the image and as more convolutions happen more feature maps are generated (Figure5).

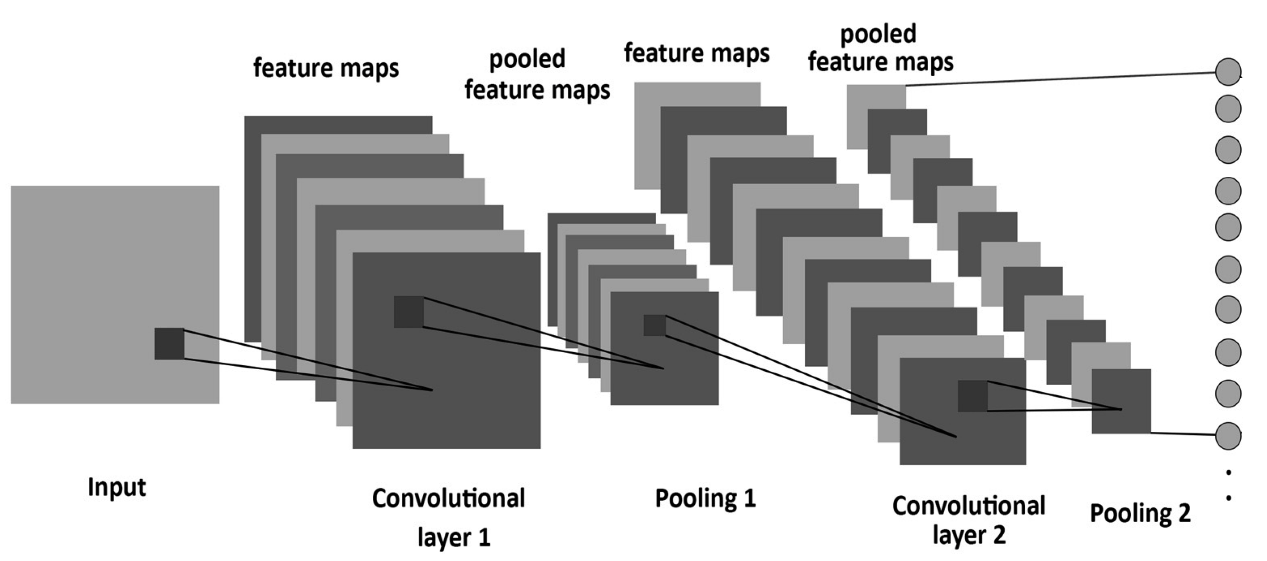


Figure 5: Convolution and pooling used in the network.

Input layer is passed to Convolution2D[[3]](#footnote-3). A set of filters (3x3) is used to generate feature maps by sliding over the image with some shared weights.

MaxPooling2D[[4]](#footnote-4) (2x2 window), is used on feature map to reduce the dimension by only keeping the maximum pixel in the window. It is applied after one or several convolutions. Once the convolutions are done, the features are Flatten[[5]](#footnote-5) and provided to dense layers as inputs.

Dense[[6]](#footnote-6) layers: Meaning fully connected layers (Figure 6), it is similar to how our neurons work. Takes input features and transforms features through layers connected and trainable weights. More layers/nodes may sound good but it might prone to over-fitting.

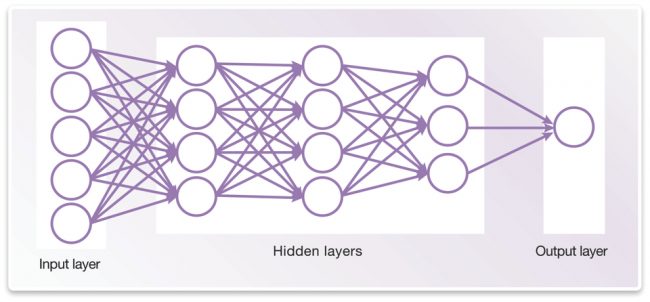


Figure 6: Neural Network with input, hidden, output layers.

Dropout[[7]](#footnote-7) is used to randomly select some portions and set their weights to zero so as to reduce over-fitting. This reduces the model sensitivity to certain noise during training.

Output layer: Contains the classification classes. The output provides probabilities of each class.

**Base Neural Network.**

I built a simple CNN with an input, 2 convolution layers with max pooling and dropout of 25%, 2 dense layers and output layer. Added Callbacks[[8]](#footnote-8) to track the history, save the model and stop the training when the validation accuracy does not increase.

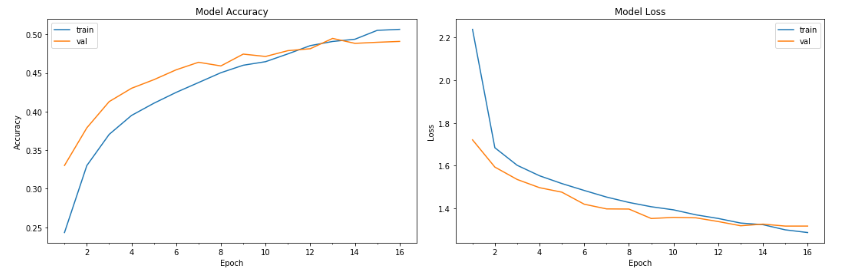
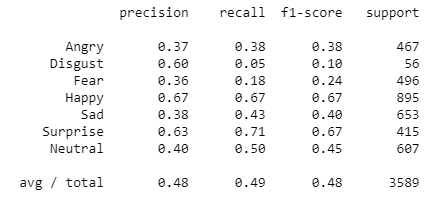


Figure 7: Base model Accuracy and Loss.

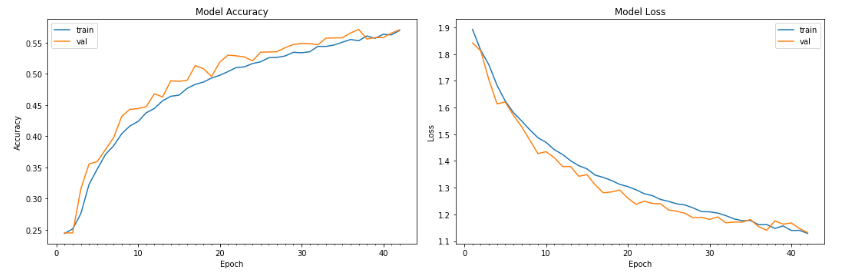
The model accuracy stopped improving pretty early. And accuracy is close to 50%, meaning it’s prediction is correct only half of the time.

Accuracy is not the only metric a classification report can provide other metrics and insights.



From the above report we see certain features has a really low recall, so we have to go **deeper** to recognize more, subtle patterns.

A **Deep Neural Network** with more convolutions and hidden layers, nodes are constructed.

Figure 8: Deep Neural Network Model Accuracy and Loss.

The model did perform well and validation accuracy is more than 55%, but we can improve the model accuracy by tuning hyper-parameters (batch size, number of epochs) and doing   
**Batch Normalization**[[9]](#footnote-9) on each layer.

[Batch Normalization](https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c) was done on each layer of the previous neural network.

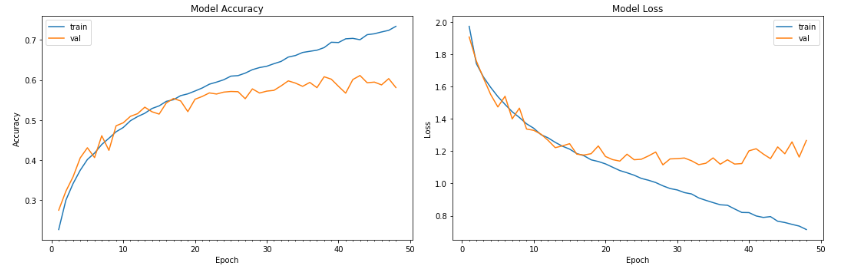
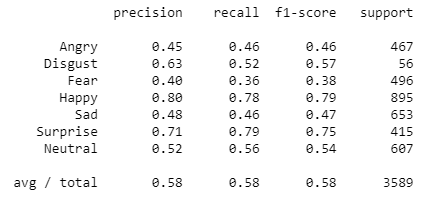


Figure 9: Deep Neural Network with Batch Normalization.

From the above figure (Figure 9) the model validation accuracy is close to 60%.   
Moreover, we see the accuracy and loss is increasing and decreasing gradually respectively in training phase but not during validation, it means it starts to over-fit or byhearting the images.   
The training is stopped if the validation accuracy does not increase.

Let’s look at other metrics.



Batch Normalization did improve a lot of things, especially the recalls for each class.

Best model was Deep Neural Network with input, 6 Convolutions with max pooling and drop out of 25% after 2 convolutions, 4 Dense layers with 256 nodes and output layer with 7 classes, with Batch Normalization on each convolution and dense layers.

**Model Validation and Analysis.**

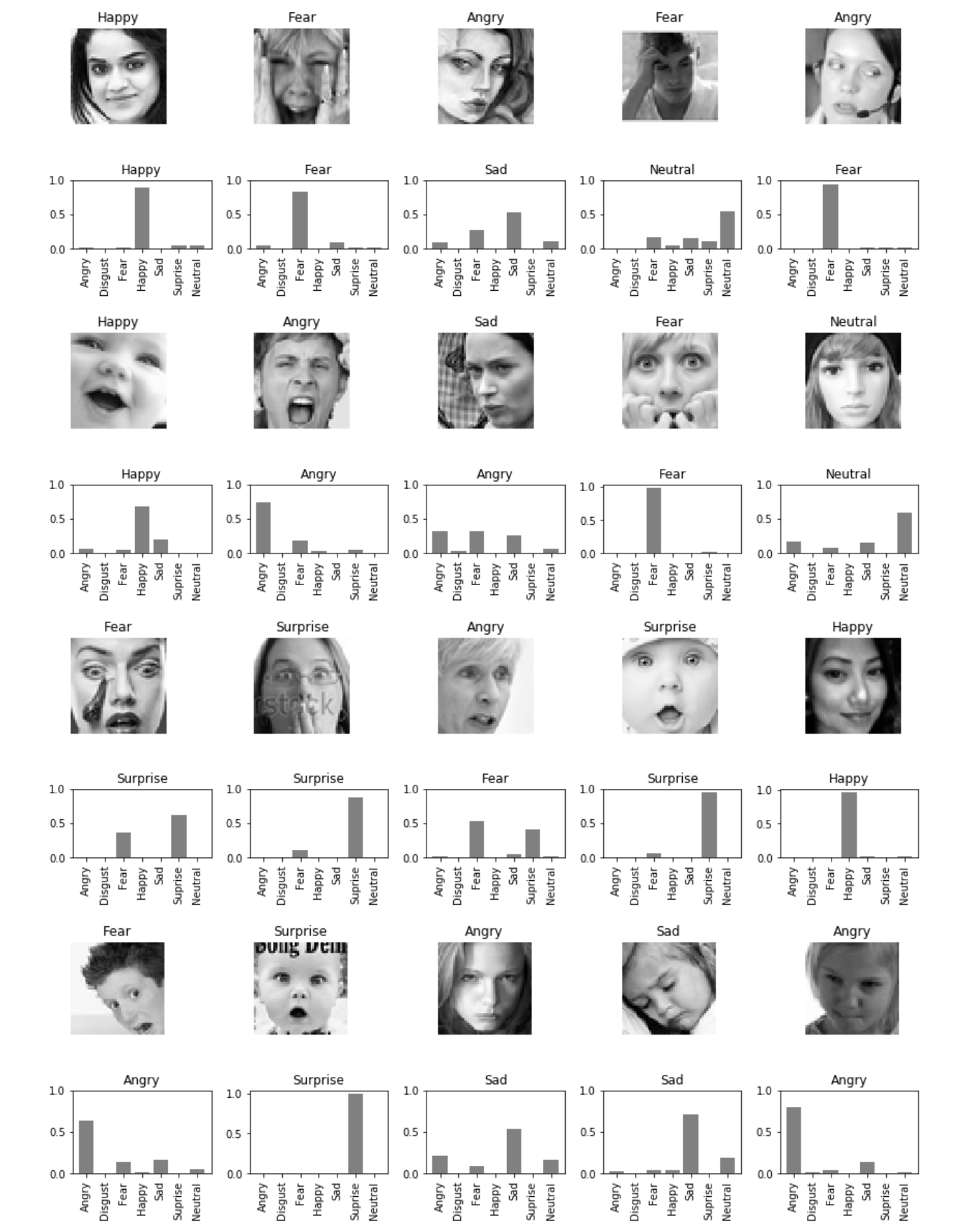
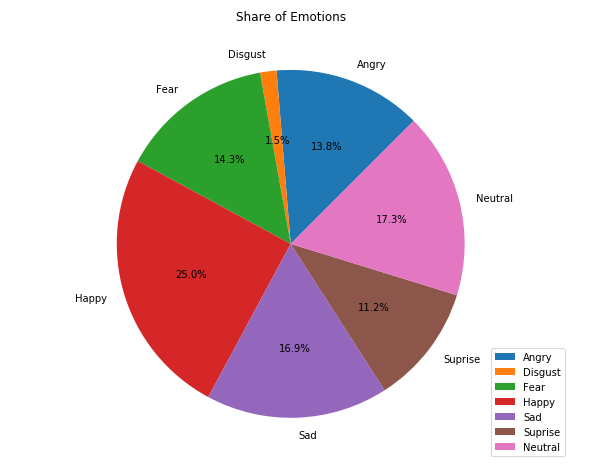
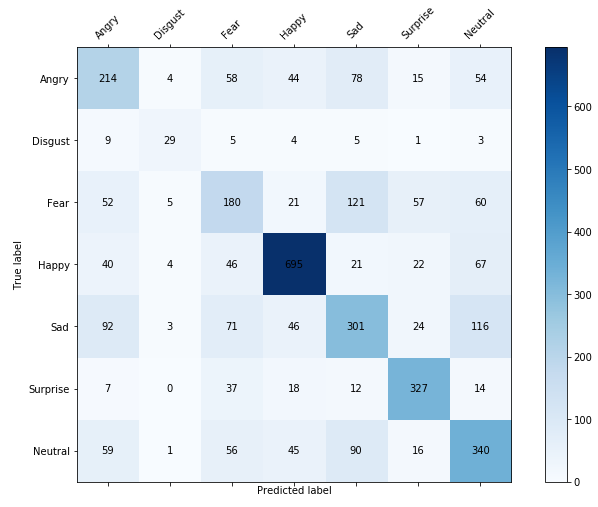
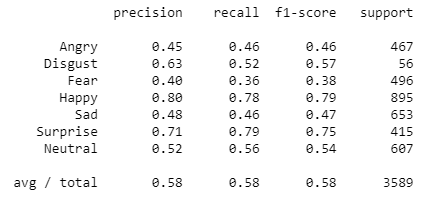


Figure 10: Prediction of facial expressions from test set.

From above figure (Figure 10) we see the model predicted 12 right out of 20 examples (60%), let’s a closer look at the confusion matrix and classification report below to get some insights.





The model performed well in classifying Happy (79% f1-score), plus we have more Happy examples, followed by Surprise (75% f1-score).

Despite Disgust being a minority class (1.5% of the sample) the model precision and recall is really good, which was unexpected.

Fear has the lowest scores of all, it misclassified itself a lot with Sad (vice-versa), then Angry and Neutral.

Neutral lies at above the 50% mark, it misclassified itself a lot with Sad.

Wondered how the inner layers look after convolutions and picking up different features.   
Below show the features it picked up after convolutions and batch normalizations.

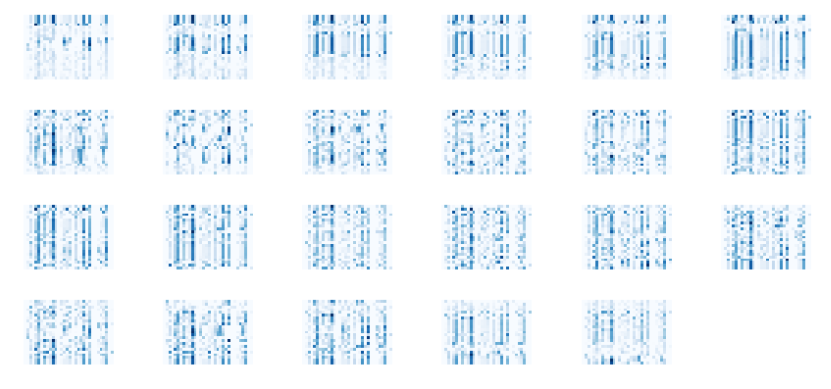


Figure 11: Feature map after 1 convolution and max pooling.



Figure 12: Feature map after 4 convolutions and max pooling.

**Fun examples** of applying the model on Mr. Bean who is known for his facial expressions.

Used [OpenCV](https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html)[[10]](#footnote-10) to detect face, crop out the image, resize the image and fed to the model to make prediction.

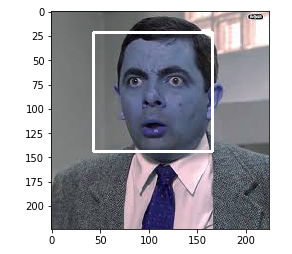
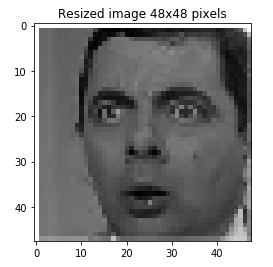
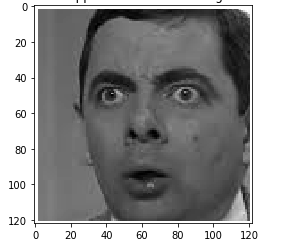
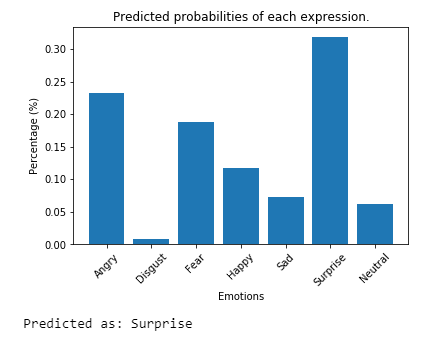
  

Figure 13: Surprised Mr. Bean.

Good, it made a good prediction. Let’s try another example.

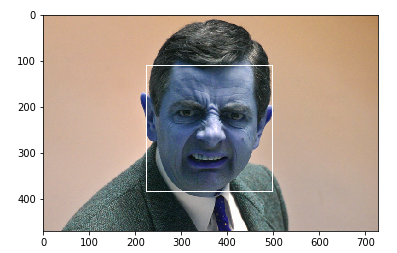
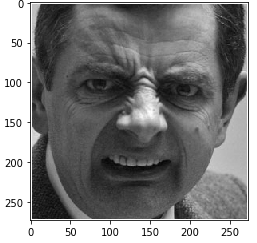
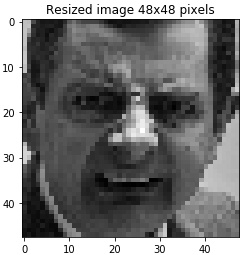
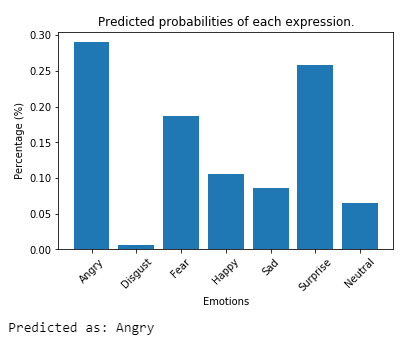
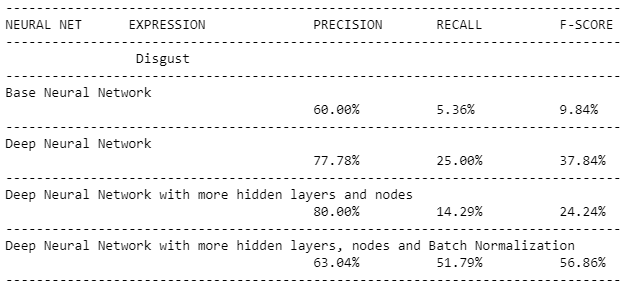
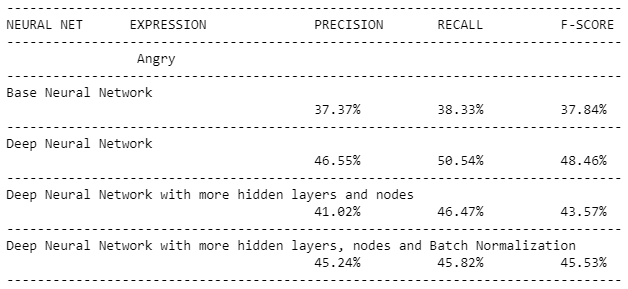
   

Figure 14: Disgust Mr. bean.

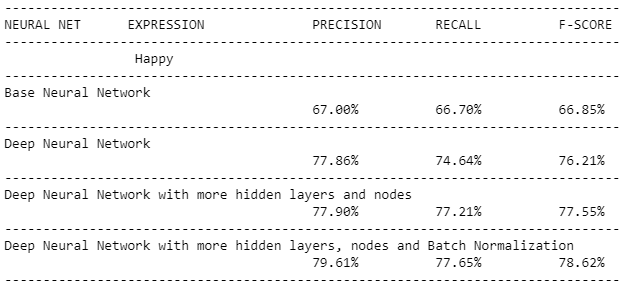
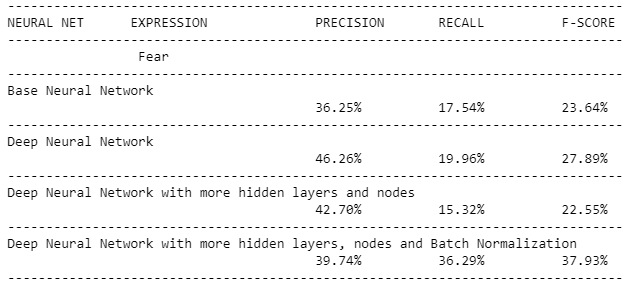
It predicted wrong and also looking at the probabilities, it was way off. Since there weren’t many examples of disgust to be trained on so it’s probabilities were more towards Angry, Surprise, Fear.

**Model Comparisons** on different expressions.

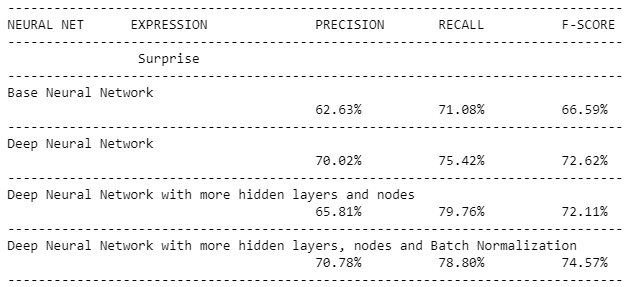
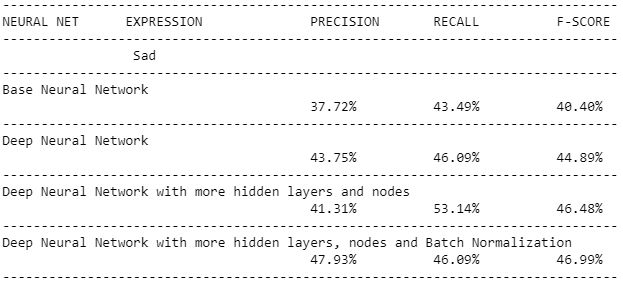
Angry Disgust



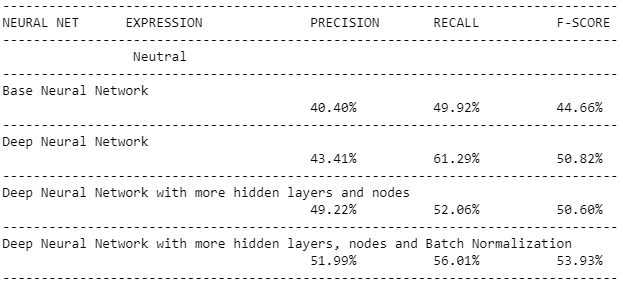
Fear Happy



Sad Surprise



Neutral



Above information shows each model metrics on each facial expression.

There were quiet some improvements with respect to base model (colored in **black**), some metrics did not improve, some models have really good metrics for certain expressions (colored in **green**) and some do not (colored in **red**).

# Findings.

Explored the data set to see facial expressions and their labels. After data exploration we see one class (disgust) having very few examples and one class (happy) with more number of examples, meaning it has imbalanced number of examples.  
  
Deep learning is good for image classification but it needs a lot of examples to be trained on and having few examples to be trained on possess a challenge.

A base neural network is created and trained on the data set, it’s accuracy was less than 50% and as expected disgust class which has less number of examples it’s precision, recall scores are very low.  
  
Going deeper, here is where deep learning comes alive which finds patters, features necessary to build a classifier. So deep neural networks are constructed and with more convolutions, hidden layers and nodes.   
More layers or nodes does not mean better results it could be over-fitting or byhearting the images. Drop-outs are added to reduce over-fitting (i.e. some nodes are pruned, so not to allow the node to have more weight and by pruning weights are distributed on other nodes). After training and validation on deep neural networks with more layers, metrics for disgust class is still low.

Batch Normalization was done on each layer (i.e. on convolution layers, dense layers) where further the weights are normalized internally. After training and validation, the metrics for disgust class improved significantly and other classes which had a low recall.

Neural Network with Batch Normalization had accuracy of 58.12% which is the best among the built neural networks. Even though it’s accuracy is not great it works 60% of the time.

I would provide this model for further training, research or to be used on any applications which wants to recognize facial expression of a person.

# Ideas for Further Research.

We explored the data set to see how many expressions are there and found few expressions having more examples and some of them not.   
This is imbalanced data set, we need to get a good number of examples on all classes so that the model learns them to distinguish different features.

Here are some suggestions that can be incorporated for further research.

* 1. Image augmentation (blur, crop, flip, rotate, translate, etc.) techniques can be used when there are less number of samples for a class.
  2. Instead of using a single model, multiple models can be ensembled and trained on the data set.
  3. Can use pre-trained models for facial expressions by training them on the data set examples. Since the pre-trained models have certain patterns, features built in all can be done is replace the classification output layer and tune or add few layers according the data set.

# Client Recommendations.

1. The model can be used in real-time facial expression rec such as on webcams or live pictures. The model can be linked up to any facial recognition software and make it to do predictions.
2. In sales, clients can use the model to detect the change in facial expression if the customers hear a deal and tweaking the deal to figure out their customers happy or surprised by the deal.
3. Using the model client can build an API (a REST API) to predict facial expressions. Using the API, a micro service can be provided in predicting facial expressions of person – FEAS (Facial Expression as a Service).

# Resources.

1. http://www.techdesignforums.com/practice/technique/facial-recognition-embedded-vision/ [↑](#footnote-ref-1)
2. https://keras.io/models/sequential/ [↑](#footnote-ref-2)
3. https://keras.io/layers/convolutional/#conv2d [↑](#footnote-ref-3)
4. https://keras.io/layers/pooling/#maxpooling2d [↑](#footnote-ref-4)
5. https://keras.io/layers/core/#flatten [↑](#footnote-ref-5)
6. https://keras.io/layers/core/#dense [↑](#footnote-ref-6)
7. https://keras.io/layers/core/#dropout [↑](#footnote-ref-7)
8. https://keras.io/callbacks/ [↑](#footnote-ref-8)
9. https://keras.io/layers/normalization/#batchnormalization [↑](#footnote-ref-9)
10. https://docs.opencv.org/3.1.0/d7/d8b/tutorial\_py\_face\_detection.html [↑](#footnote-ref-10)