Springboard Data Science Career Track

Capstone Project II Final Report

Facial Expression Recognition.

Rahul Ambati

June 2018

1. Introduction.
2. Approach.
    a. Data Acquisition and Wrangling.
    b. Data Exploration.
    c. Modeling and Analysis.
3. Findings.
4. Ideas for Further Research.
5. Client Recommendations.
6. Resources and References.

1. 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. The intention is to create a supervised machine learning model, which can be useful in predicting a person's facial expression, according to a set of predefined labels.

Client:
Any application that needs to classify a person's facial expression according to a set of predefined labels, before and after an event or activity.
Examples of potential clients, are: **Retail** used to evaluate customer interests, or in **Healthcare** used to assess patient's emotional state, or in **Entertainment** used to monitor audience's expressions during an entertaining performance.

2. Approach.

We will build classification models using various neural network architectures, including deep ones.

a. Data Acquisition and Wrangling.

The data has been acquired from Kaggle dataset Facial Expression Recognition Challenge.

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

b. Data Exploration.

The dataset has been prelabeled according to the following classes (also, see Figures 1 and 2 below):

i. *Angry*
ii. *Disgust*
iii. *Fear*
iv. *Happy*
v. *Sad*
vi. *Surprise*
vii. *Neutral*

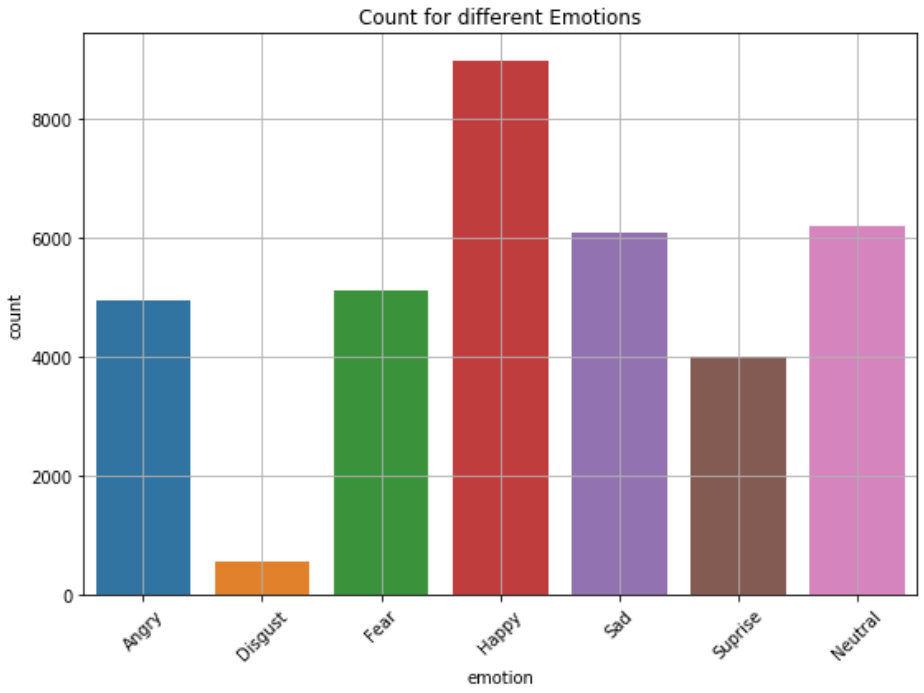


Figure 1. Number of samples for each label.

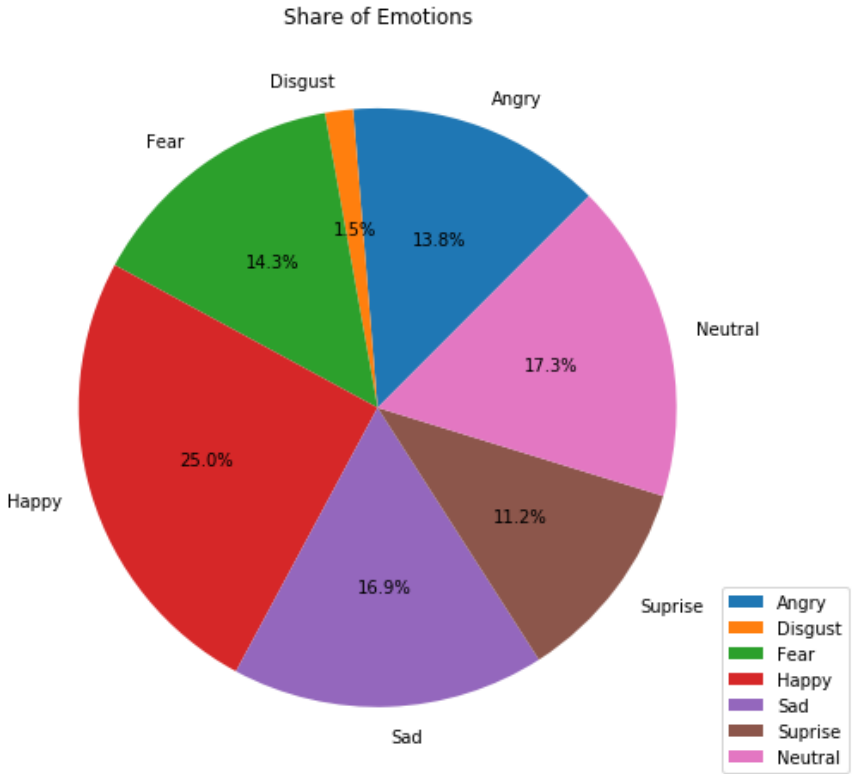


Figure 2. Share of each label in the dataset.

From the above figures we see the *Disgust* has the smallest number of examples. Figure 3 below shows a small sample of the dataset with the corresponding labels. Each image is

48x48 pixels in grayscale.

Figure 3. Sample images with labels.

c. 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 intend 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.

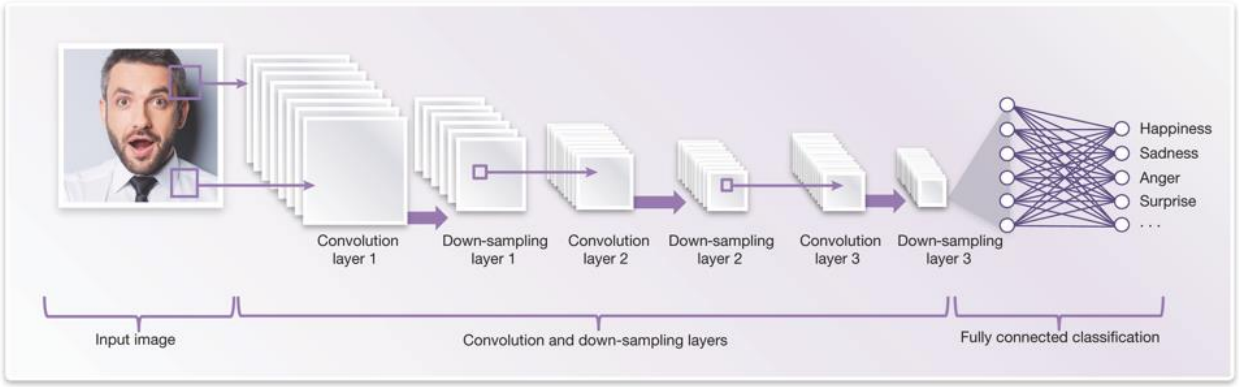


Figure 4: CNN[1] architecture for Facial Expression Recognition.

Keras Sequential[2] was used to implement this architecture.

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 (see Figure5).

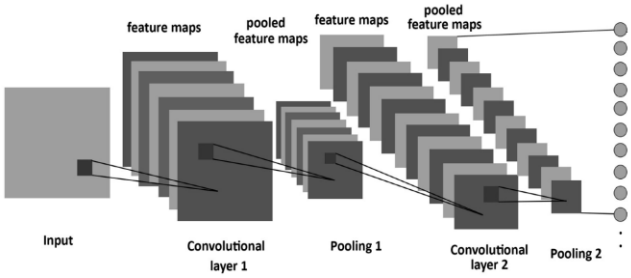


Figure 5: Convolution and pooling used in the network.

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

[1] http://www.techdesignforums.com/practice/technique/facial-recognition-embedded-vision
[2] https://keras.io/models/sequential
[3] https://keras.io/layers/convolutional/#conv2d

MaxPooling2D[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 flattened[5] and provided to dense layers as inputs.

Dense[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 be prone to over-fitting.

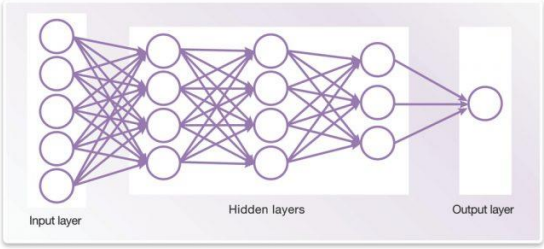


Figure 6: Neural Network with input, hidden, output layers.

Dropout[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] to track the history, save the model and stop the training when the validation accuracy does not increase according to a certain threshold.

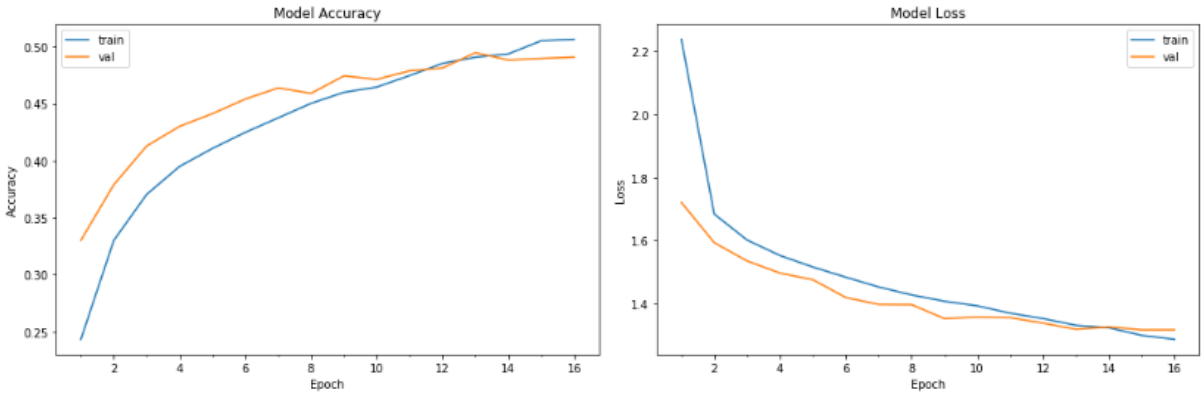


Figure 7: Base model Accuracy and Loss.

[4] https://keras.io/layers/pooling/#maxpooling2d
[5] https://keras.io/layers/core/#flatten
[6] https://keras.io/layers/core/#dense
[7] https://keras.io/layers/core/#dropout
[8] https://keras.io/callbacks/

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 per class.

	precision	recall	f1-score	support
Angry	0.37	0.38	0.38	467
Disgust	0.60	0.05	0.10	56
Fear	0.36	0.18	0.24	496
Happy	0.67	0.67	0.67	895
Sad	0.38	0.43	0.40	653
Surprise	0.63	0.71	0.67	415
Neutral	0.40	0.50	0.45	607
avg / total	0.48	0.49	0.48	3589

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

A **Deep Neural Network** is constructed with 6 convolutions, max pooling and drop out of 25% after 2 convolutions, 4 dense layers with 256 nodes and output layer of 7 classes.

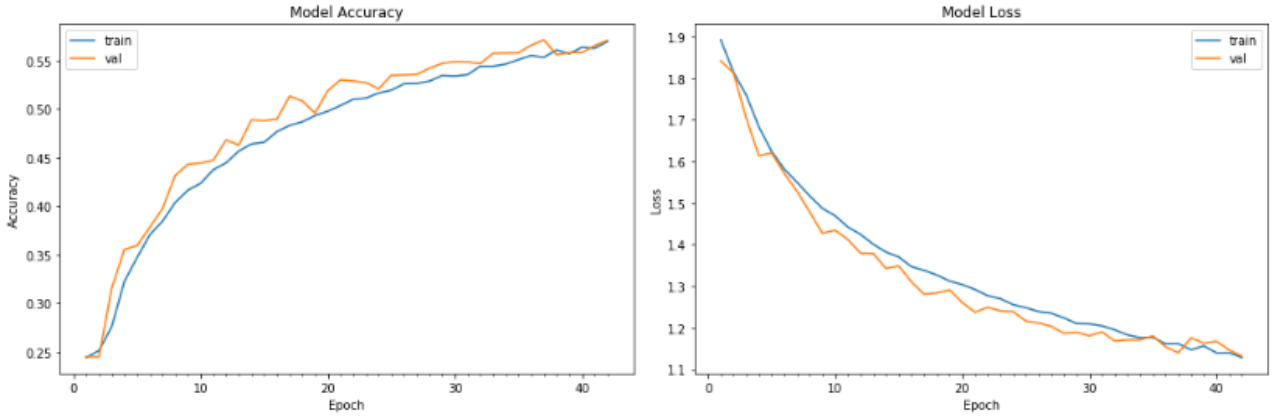


Figure 8: Deep Neural Network Model Accuracy and Loss.

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] on each layer.

[9] https://keras.io/layers/normalization/#batchnormalization

Batch Normalization was done on each layer of the previous neural network.

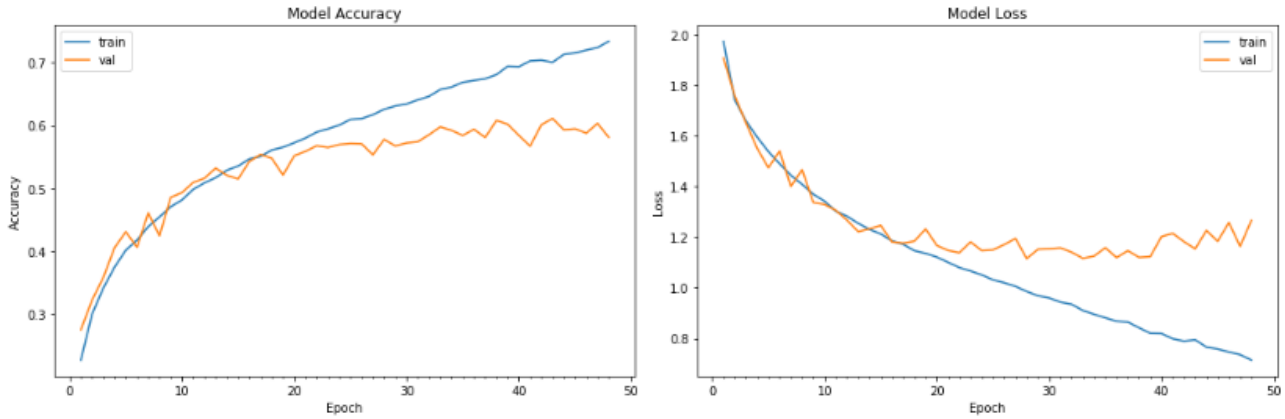


Figure 9: Deep Neural Network with Batch Normalization.

From Figure 9, we see that 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 according to a certain threshold.

Let's look at other metrics.

	precision	recall	f1-score	support
Angry	0.45	0.46	0.46	467
Disgust	0.63	0.52	0.57	56
Fear	0.40	0.36	0.38	496
Happy	0.80	0.78	0.79	895
Sad	0.48	0.46	0.47	653
Surprise	0.71	0.79	0.75	415
Neutral	0.52	0.56	0.54	607
avg / total	0.58	0.58	0.58	3589

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.

Using the model, predict it on the public test set.

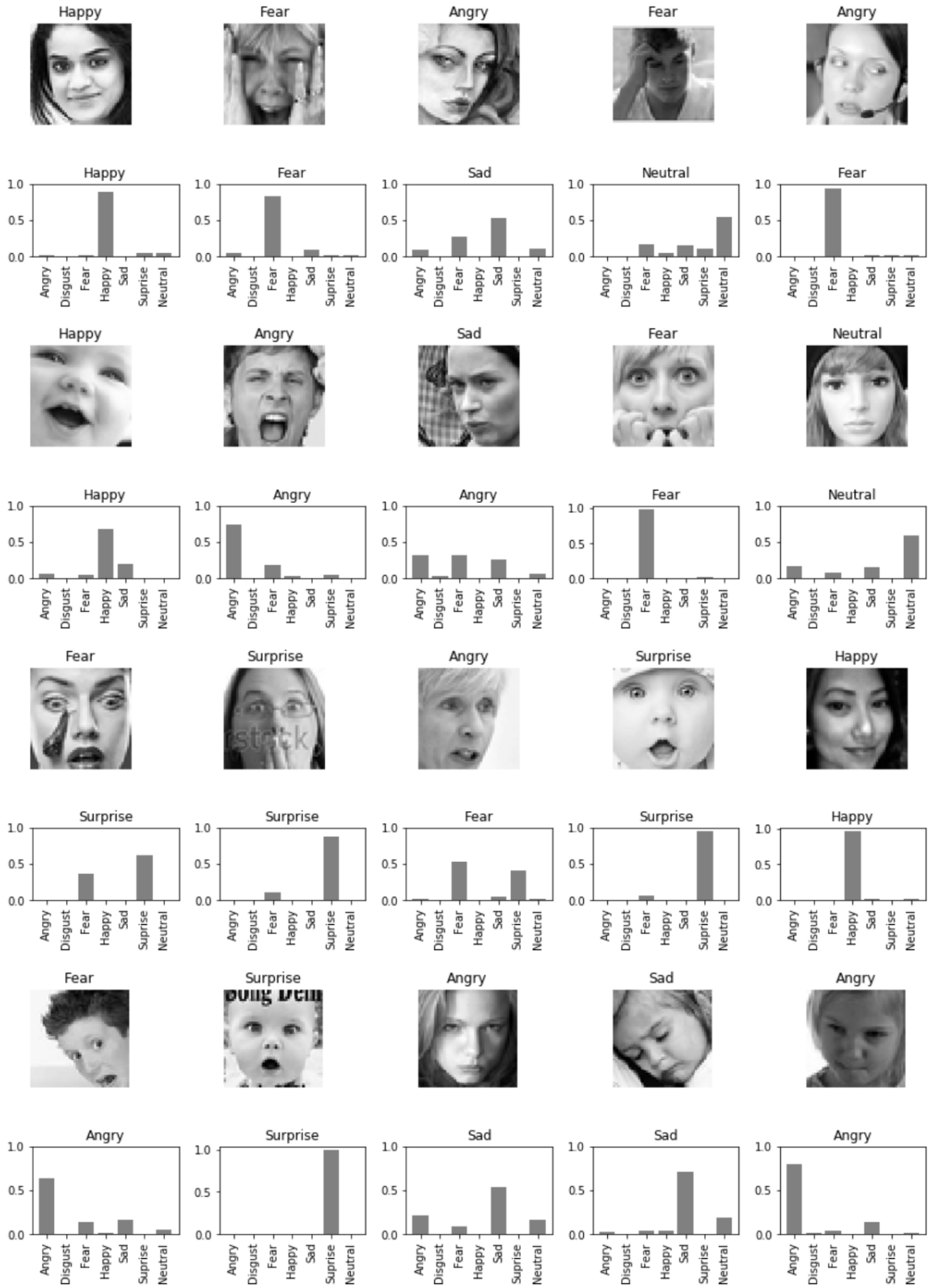
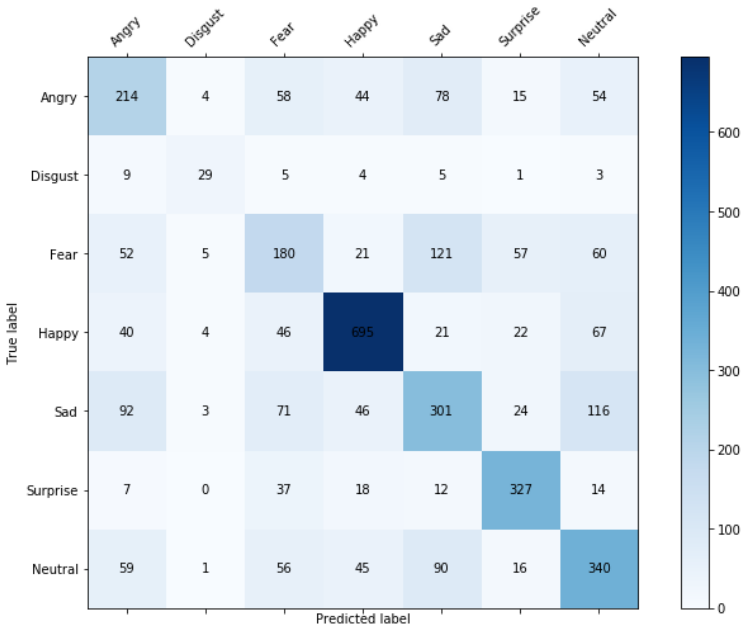
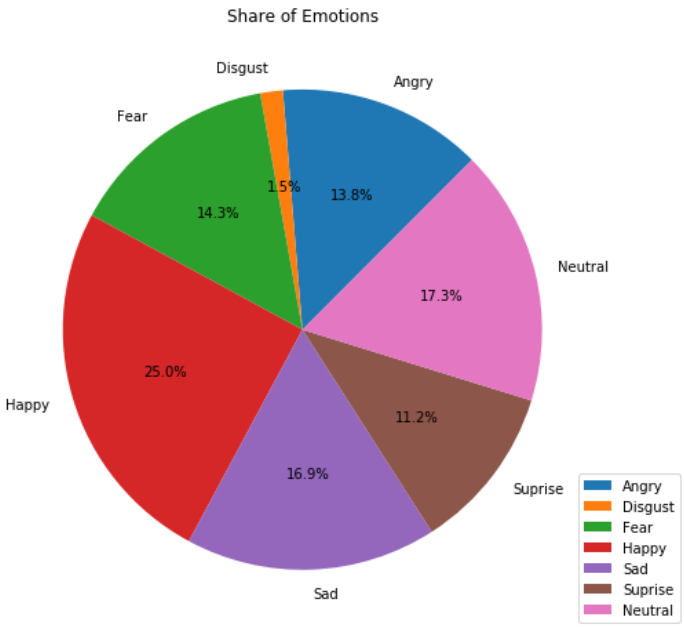


Figure 10: Prediction of facial expressions from test set.

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





	precision	recall	f1-score	support
Angry	0.45	0.46	0.46	467
Disgust	0.63	0.52	0.57	56
Fear	0.40	0.36	0.38	496
Happy	0.80	0.78	0.79	895
Sad	0.48	0.46	0.47	653
Surprise	0.71	0.79	0.75	415
Neutral	0.52	0.56	0.54	607
avg / total	0.58	0.58	0.58	3589

The model performed well in classifying Happy (79% f1-score) plus “Happy” is the largest class 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 was misclassified as Sad (and vice-versa), then misclassified as Angry and Neutral.

Neutral lies at above the 50% mark, it was misclassified as Sad.

It is a common practice to visualize how the inner layers look after convolutions and batch normalizations as they pick up different features. See Figures 11-13.

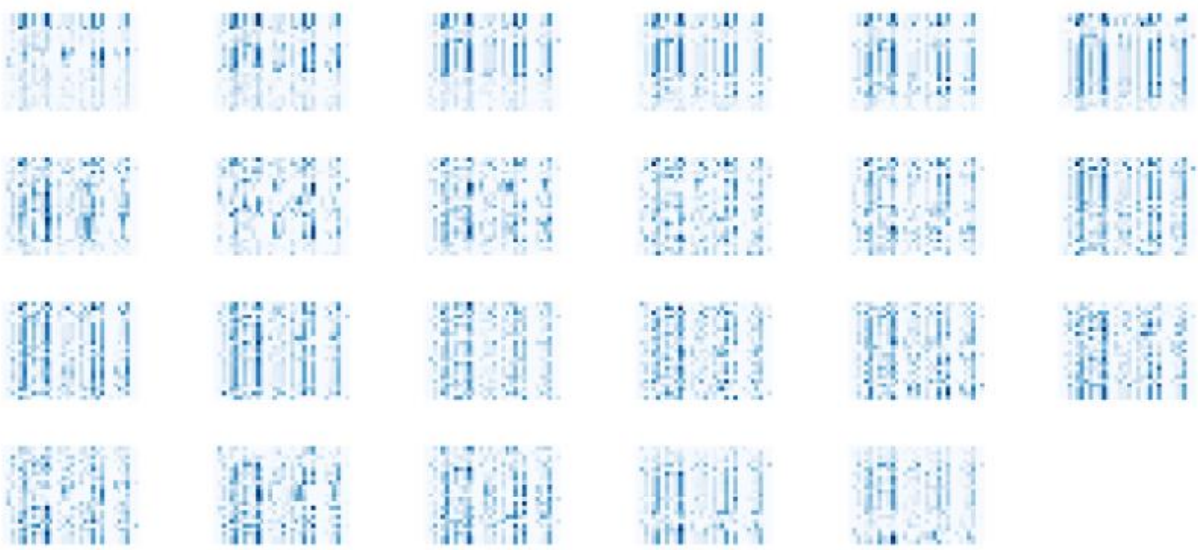


Figure 11: Features after first convolution, batch normalization

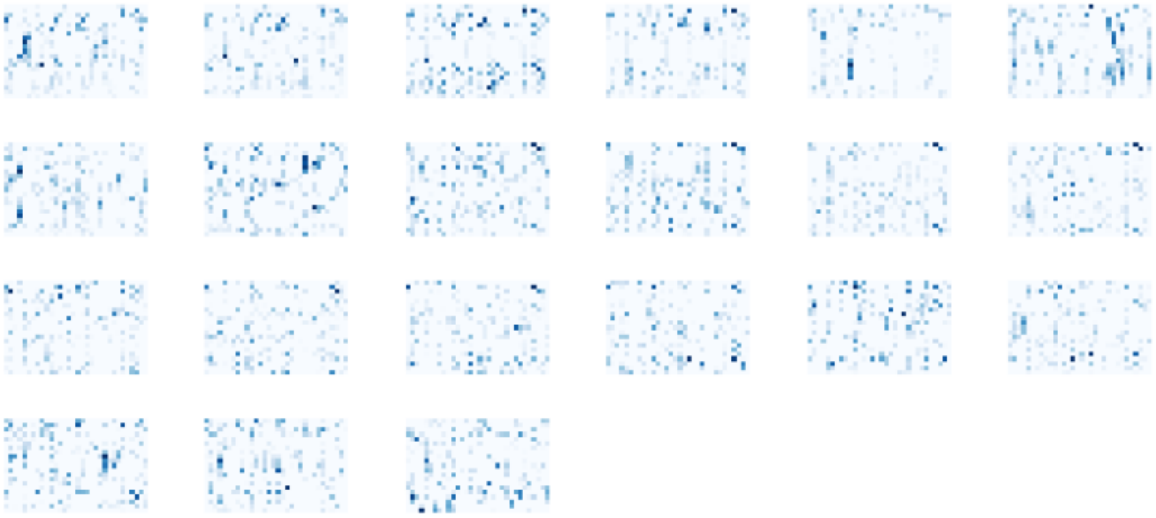


Figure 12: Features after 2 convolution, batch normalization and max pooling.

Figure 13: Feature map after 4 convolutions, batch normalizations and max pooling.

We see that after the first and second convolutions with batch normalizations after each layer, low-level features are found. Figure 12 gives some high-level features it picked up after 2 convolutions.

After 4 convolutions and batch normalizations (Figure 13) we see a reduced number of pixels (due to Max-pooling[10], which reduces the number of features) and only high-level features are in existence. The further the convolution, the more it picks the high-level features from the previous high-level features, there by further reducing the number of pixels.

Towards Production Mode

In order to simulate putting the models into production mode, I used OpenCV[11] to detect face, crop out the image, resize the image and feed it into the model to make prediction (see Figure 14.)

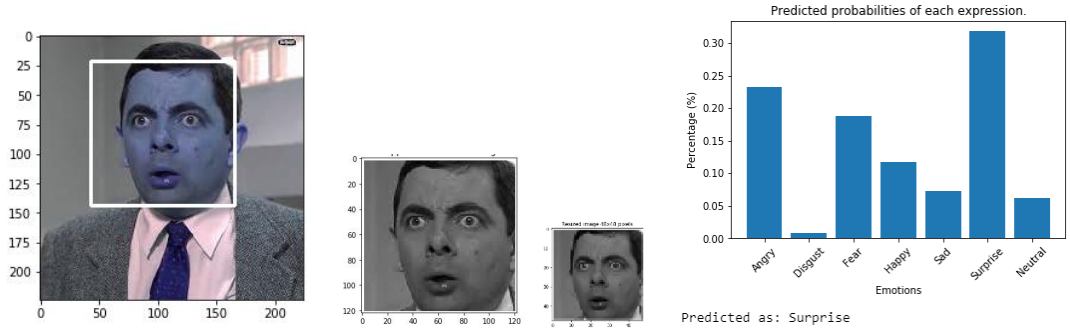


Figure 14: Surprised Mr. Bean.

This first example was correctly predicted. Let's try another example (see Figure 15.)

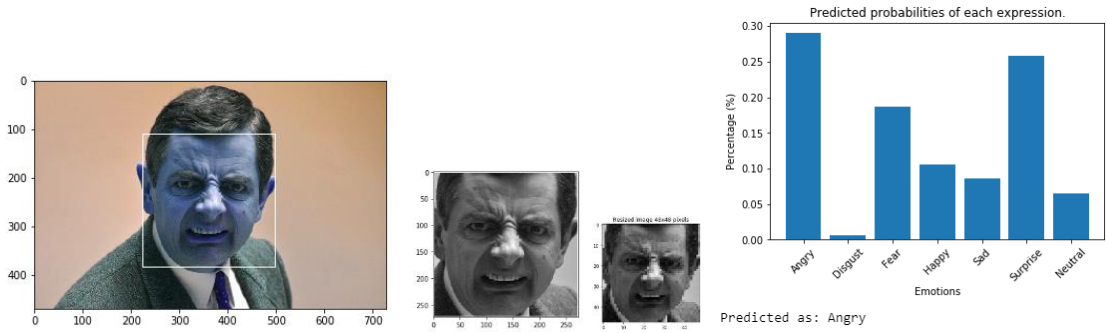


Figure 15: Disgusted Mr. Bean.

In this case the prediction was incorrect, and by looking at the probabilities, we can see that they are way off. Since there weren't many examples of disgust to be trained on so it's probabilities were more towards Angry, Surprise, and Fear.

[10] https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks
[11] https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html

Model Comparison

There were quite a few 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**).

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Angry			
Base Neural Network		37.37%	38.33%	37.84%
Deep Neural Network		46.55%	50.54%	48.46%
Deep Neural Network with more hidden layers and nodes		41.02%	46.47%	43.57%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		45.24%	45.82%	45.53%

Figure 16: Metrics for Angry.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Disgust			
Base Neural Network		60.00%	5.36%	9.84%
Deep Neural Network		77.78%	25.00%	37.84%
Deep Neural Network with more hidden layers and nodes		80.00%	14.29%	24.24%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		63.04%	51.79%	56.86%

Figure 17: Metrics for Disgust.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Fear			
Base Neural Network		36.25%	17.54%	23.64%
Deep Neural Network		46.26%	19.96%	27.89%
Deep Neural Network with more hidden layers and nodes		42.70%	15.32%	22.55%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		39.74%	36.29%	37.93%

Figure 18: Metrics for Fear.

Fear expression (Figure 18) has low metrics in general for all neural networks.

Disgust and Fear expressions had a low recall (Figures 17,18), and after batch normalization it's recall improved significantly. And despite Disgust having few training samples the batch normalization did good in recognizing the expression.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Happy			
Base Neural Network		67.00%	66.70%	66.85%
Deep Neural Network		77.86%	74.64%	76.21%
Deep Neural Network with more hidden layers and nodes		77.90%	77.21%	77.55%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		79.61%	77.65%	78.62%

Figure 19: Metrics for Happy.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Sad			
Base Neural Network		37.72%	43.49%	40.40%
Deep Neural Network		43.75%	46.09%	44.89%
Deep Neural Network with more hidden layers and nodes		41.31%	53.14%	46.48%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		47.93%	46.09%	46.99%

Figure 20: Metrics for Sad.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Surprise			
Base Neural Network		62.63%	71.08%	66.59%
Deep Neural Network		70.02%	75.42%	72.62%
Deep Neural Network with more hidden layers and nodes		65.81%	79.76%	72.11%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		70.78%	78.80%	74.57%

Figure 21: Metrics for Surprise.

Happy and Surprise (Figures 19,21) have really good metrics for all neural networks. Also, most of the neural networks built have good metrics in predicting Happy and Surprise expressions, but not for Disgust and Fear.

NEURAL NET	EXPRESSION	PRECISION	RECALL	F-SCORE
	Neutral			
Base Neural Network		40.40%	49.92%	44.66%
Deep Neural Network		43.41%	61.29%	50.82%
Deep Neural Network with more hidden layers and nodes		49.22%	52.06%	50.60%
Deep Neural Network with more hidden layers, nodes and Batch Normalization		51.99%	56.01%	53.93%

Figure 22: Metrics for Neutral.

After batch normalization when compared to other neural networks overall metrics improved, and except for Sad and Neutral expressions (Figures 20,22) the metrics remained the same i.e. did not improve significantly.

3. Findings.

The dataset is clearly imbalanced.

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, which can be overcome by adding more examples for the minority classes.

A baseline neural network was created and trained on the data set, it's accuracy was less than 50% and as expected the class with the smallest number of samples has very low precision and recall scores.

Going deeper, here is where deep learning comes alive which finds patters, features necessary to build a classifier. So deep neural networks were constructed and with more convolutions, hidden layers and nodes.

More layers or nodes does not mean better results, since there could be overfitting or byhearting of the images. Drop-outs are added to reduce overfitting (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 the Disgust class is still low.

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

A neural network architecture with Batch Normalization had accuracy of 58.12% which is the best among the implemented architectures. 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 requires to recognize the facial expression of a person.

4. Ideas for Further Research.

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

a. Add samples to the minority classes in the dataset.

b. Image augmentation (blur, crop, flip, rotate, translate, etc.) techniques can be used when there are less number of samples for a class.

c. Instead of using a single model, an ensemble of multiple models can be used.

d. 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 one can replace the classification output layer and tune, or add few layers, according the data set.

5. Client Recommendations.

a. The model can be used in real-time facial expression recognition such as on webcams or live pictures. The model can be linked up to any facial recognition software and make it to do predictions.

b. In sales, clients can use the model to detect the change in facial to drive flash-deals in real time.

c. 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).

6. Resources and References.

Tools:

Anaconda
Jupyter Notebook
GitHub

Language:

Python 3.6

Packages:

Numpy
Pandas
Skimage
Keras
OpenCV[12]

Packages for visuals:

Matplotlib
Seaborn

Blogs, Videos and References:

Machine Learning & Deep learning Fundamentals[13]
Understanding Convolutions[14]
Improving Deep Learning Performance[15]
CNN simplified, demystified[16]
Neural Networks in Keras[17]

[12] https://github.com/opencv/opencv
[13] https://www.youtube.com/playlist?list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU
[14] https://stackoverflow.com/questions/42883547/what-do-you-mean-by-1d-2d-and-3d-convolutions-in-cnn
[15] https://machinelearningmastery.com/improve-deep-learning-performance
[16] https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified
[17] https://parneetk.github.io/blog/neural-networks-in-keras