Machine Learning Advanced Nanodegree

Capstone Project

Veluvolu Akhil

(akhilchowdary78@gmail.com)

October 13th, 2018

Abstract: -

In this project I tried to develop a Convolutional Neural Network (CNN) for recognition of facial expression. The goal is to classify each facial image into one of the seven facial emotions: anger, disgust, fear, happiness, sadness, surprise and neutrality. The CNN models are trained with different depth using pixels of gray-scale images from the Kaggle FER dataset. To reduce the over fitting of the models, techniques like dropout, regularization and batch normalization are used. I experimented with different architectures and methods, ultimately achieved an accuracy of 0.51 in a seven class classification task.

1 INTRODUCTION

Facial behavior is one of the most important aspect for sensing human emotion and intentions among people. Automatic facial expression recognition is an interesting and challenging problem which has important applications in areas like human-computer interaction. It helps to build intelligent robots which have the ability to understand human emotions. Many other real world examples include**,** world retail-stores are investing more to satisfy the customer with their product offerings, the reactions of customers could be used as feedback to improve the product offerings or to remove certain products disliked by many. Similarly, analysis of facial expressions of drivers would help in determining their stress level and such information could be used to alert drivers if they are stressed and, in a state, unsafe for driving (or) should transfer control from manual to self-drive mode. In this paper, I presented an approach based on Convolutional Neural Networks (CNN) for facial expression recognition. The input into our system is pixels of an image; then, we use CNN to predict the facial expression label which should be one these labels: anger, happiness, fear, sadness, disgust and neutral**.**

2 Dataset and Features

I trained and tested our models on the data set from the Kaggle Facial Expression Recognition Challenge, which comprises of 48-by-48-pixel grayscale images of human faces, each labeled with one of 7 emotion categories: anger, disgust, fear, happiness, sadness, surprise, and neutral. We used a training set of 22,967 examples, a validation set of 5,742 examples, and a test set of 7,178 examples. After reading the raw pixel data, they are normalized by subtracting the mean value of all the pixels and further divided by its variance. Below are the few example images from dataset.

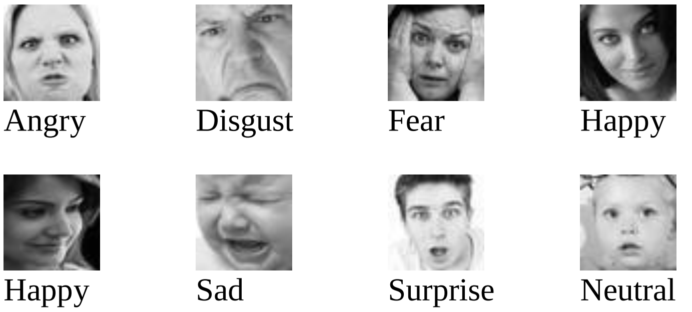


Figure 1 Sample images from the dataset and labeled with their corresponding class

2.1 Data Exploration

The no of occurrences of each class in the dataset are shown in Figure 2.

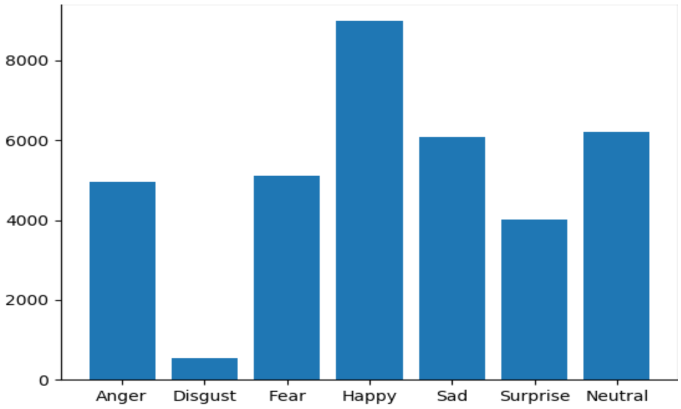


Figure 2 No of occurrence for each category in the dataset

2.2 Evaluation metrics

Accuracy is a common metric for binary classifiers; it takes both true positives & true negatives with equal weight

	Predicted: Yes	Predicted: NO
Actual: Yes	TP	FN
Actual: NO	FP	TN

$$\text{Accuracy} = (TP+TN) / (TP + FP + TN + FN)$$

Precision is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP).

$$\text{Precision} = TP / (TP + FP)$$

Recall is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

$$\text{Recall} = TP / (TP + FN)$$

Sensitivity refers to the rate of correctly classified positive and is equal to TP divided by the sum of TP and FN. Sensitivity may be referred as a True Positive Rate.

$$\text{Sensitivity} = TP/\,(TP + FN)$$

Specificity refers to the rate of correctly classified negative and is equal to the ratio of TN to the sum of TN and FP

$$\text{Specificity} = TN\,/\,(FP + TN)$$

3 Experiments

Feed Forward Neural Network: -

First, I implemented this with a standard six layered neural network using the Keras library (with tensor flow as backend) in Python. The five fully connected hidden layers have 640 neurons and an output layer containing 7 neurons, one corresponding to each class of labels. The hidden layers implement a ReLU activation function and output layer implements SoftMax activation function. The model has been trained on training data for 10 epochs with 200 steps per epoch and achieved train accuracy of 49.39%& test accuracy of 35.52%.

Convolutional Neural Network

Deeper CNN: -

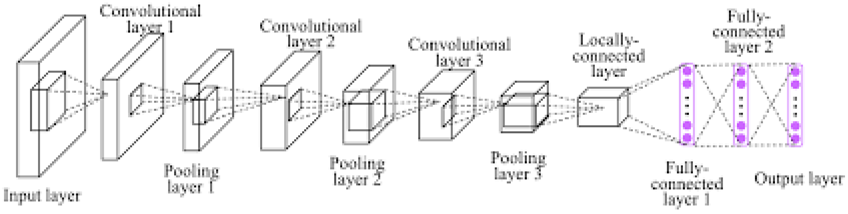


Figure 5 Architecture of Deep CNN

Now I trained a deeper CNN with three convolutional layers and two FC layers. The above figure shows the architecture of this deep network. The first convolutional layer had 32 5×5 filters, the second one had 64 5×5 filters and the last one had 128 5×5 filters. In all the convolutional layers, we have L2 Regularization, batch normalization, dropout, max-pooling and ReLU as the activation function. The hidden layer in the first FC layers had 128 neurons and the second FC layer had 256 neurons. In the FC layers, batch normalization, dropout, L2 Regularization and ReLU are used as an activation function. SoftMax is applied as an activation function for output layer. Trained the model for n epochs with a batch size of 32, and the network with all the images in the training set. Moreover, cross-validation has been done for the hyper parameters to achieve the highest accuracy with the model. An accuracy of 50.10% on the validation set and 50.27% on the test set has been obtained. The table below depicts the values for each hyper-parameter in this model that had the highest accuracy.

Parameter	Value
Learning Rate	0.001
Decay	1e-4
Regularization	1e-7

Hyper parameters set for the model

Implementation Details

All the classifiers are implemented in Keras (with Tensor flow as backend) and trained on Udacity GPU. GPU allows us to experiment with many different model configurations because of its computation speed. With each of model, I performed feature-wise mean subtraction and normalization to the input data. The model has been trained for N (step that validation loss is stopped updating + 4 steps) epochs with a batch size of 32, saving full run history and saving the model's best weights in HDF5 format based on validation loss, Training process will be terminated if the validation loss didn't improve for 4 continuous epochs. An accuracy of 71.93%, validation accuracy of 48.75% has been achieved. The model has been evaluated on test data set and achieved an accuracy of 47.36%. A better accuracy can be achieved by tweaking the hyper parameters like learning rate, decay rate and regularization and retrain the model on training dataset. After finding best hyper parameters a training accuracy of 69.85%, validation accuracy of 50.75% has been achieved. Evaluated the model on test data set achieved an accuracy of 51.05%. Data augmentation has also been done in the form of random horizontal shifts, random vertical shifts, and random horizontal flips to reduce over fitting and improve accuracy of models. Data augmentation has been applied to our training data and retrained the model and found that validation & test accuracy are dropping off when training accuracy are improving. But in this case, applying it doesn't improve accuracy may because it is a Gray Scale image with small shape (48*48).

Results

To compare the performance of the Feed Forward Neural Network model with the deep model, a plot has been made between the loss history and the obtained accuracy in these models. Figures 6 and 7 exhibit the results. As seen in Figure 7, the deep network enabled to increase the validation accuracy by 16.21%. Furthermore, one can observe that the deep network has reduced the over fitting behavior of the learning model by adding more non-linearity and hierarchical usage of anti-over fitting techniques such as dropout and batch normalization in addition to L2 regularization. From Figure 6, we can also see that the Feed Forward Neural Network converged faster and the training accuracy quickly reached its highest value.

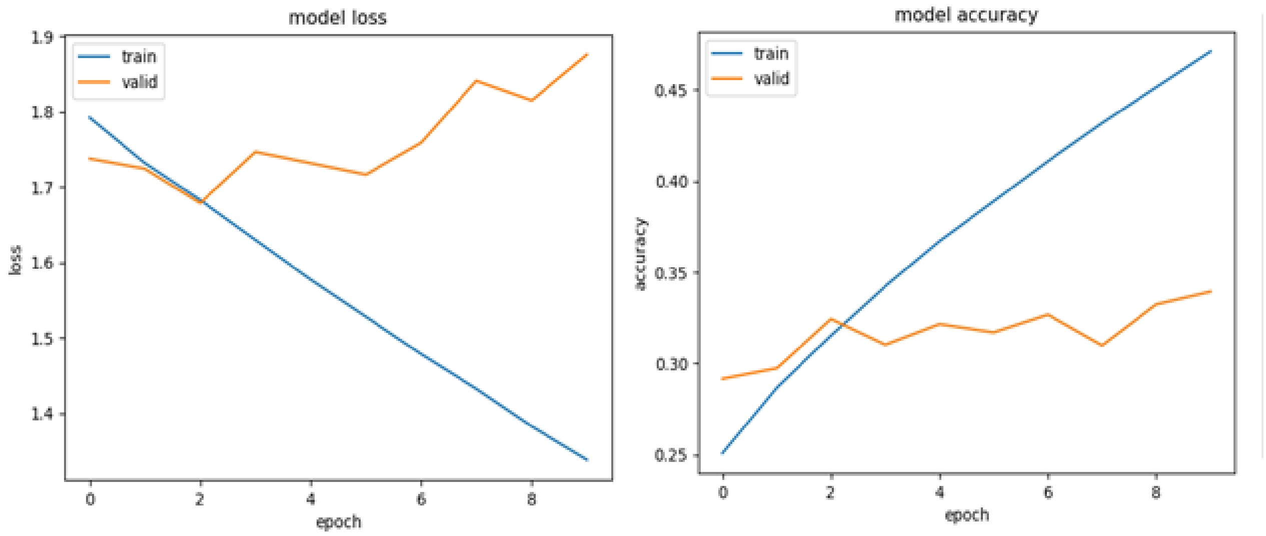


Figure 6 Loss history & Accuracy history of Feed Forward Neural Network

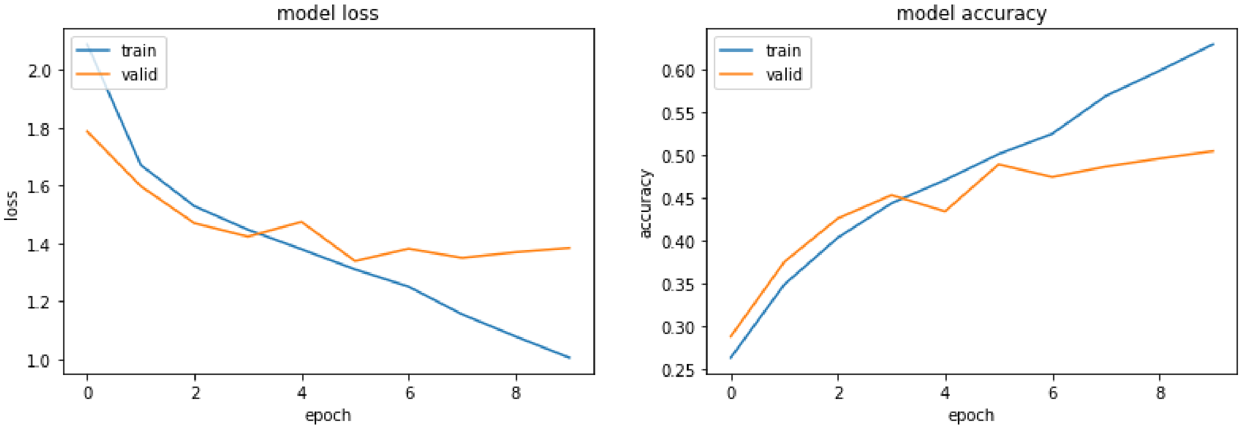


Figure 7 Loss history & Accuracy history of Deep CNN

Moreover, the confusion matrices for the feed forward neural network and deep CNN network are computed. Figures 8 and 9 present the visualization of the confusion matrices. As demonstrated in these figures, the deep network results in higher true predictions for most of the labels. It is interesting to see that all models performed well in predicting the happy label, which implies that learning the features of a happy face is easier than any other expressions. Additionally, these metrics reveal which labels are likely to be confused by the trained networks. For example, Learning features of disgust are difficult to find compared to the other expressions and we can see the correlation of sad label with the fear and neutral labels. There are lots of instances that their true label is sad but the classifier has misclassified them as fear or neutral. These mistakes are consistent because of images in the dataset, even as a human, sometimes it is difficult to recognize whether a sad expression is actually sad or neutral. This is due to the fact that all people do not express the emotions in the same way.

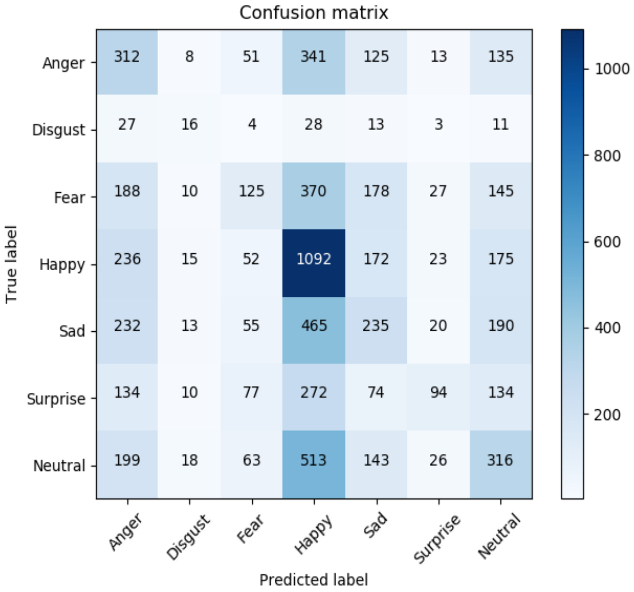
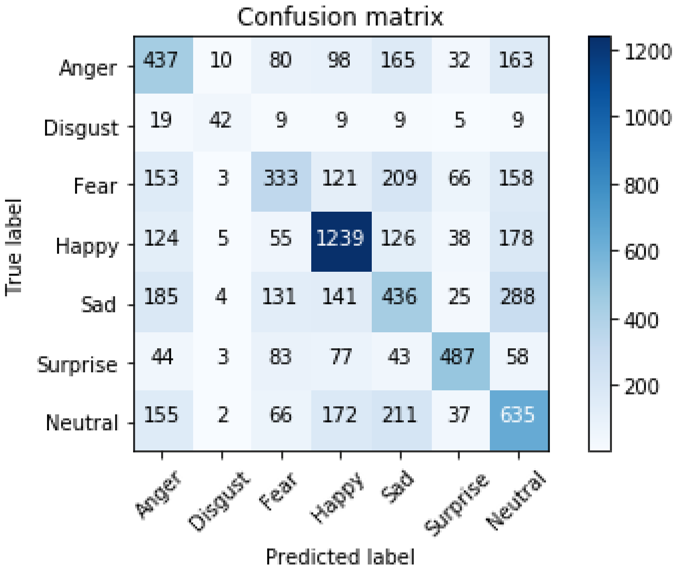


Figure 8 Confusion matrix of Feed Forward Neural Network.



	Anger	Disgust	Fear	Happy	Sad	Surprise	Neutral
Anger	437	10	80	98	165	32	163
Disgust	19	42	9	9	9	5	9
Fear	153	3	333	121	209	66	158
Happy	124	5	55	1239	126	38	178
Sad	185	4	131	141	436	25	288
Surprise	44	3	83	77	43	487	58
Neutral	155	2	66	172	211	37	635

Figure 9 Confusion Matrix for Deep CNN

Conclusion

In this project, I built Feed Forward Neural Network and Deep Convolutional Neural Network to recognize facial expression from 48*48 gray-scale pictures downloaded from Kaggle facial expression recognition challenge. We used different depth of Conv(ReLU activation) - Max-Pooling architecture and SoftMax activation for output layer. For regularization, we experimented with L2 regularization and dropout in our deeper CNN. To optimize the net, we used Adam with learning rate and decay. The final testing accuracy obtained using this architecture was 51.19%. Some of the difficulties while improvising this are that the images are very small and in some cases it is very hard to distinguish which expression is on each image, even for humans. An attempt has been made by adding more layers and more filters to improve accuracy which couldn't be possible for which the reason might be the dataset itself.

References

https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

https://arxiv.org/ftp/arxiv/papers/1608/1608.02833.pdf

https://arxiv.org/pdf/1409.1556.pdf

http://cs229.stanford.edu/proj2014/Jithin%20Thomas,%20Bhrugurajsinh%20Chudasama,%20Chinmay%20Duvedi,%20Learning%20Facial%20Expressions%20From%20an%20Image.pdf