Facial Expression Recognition

Akash Reddy Gurram, Vineeth Reddy Anugu
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
{hr69338, vanugu1}@umbc.edu
December 18, 2019

***Abstract*—This document discusses the final course project report implemented for CMSC 678 Introduction to Machine Learning course. The project deals with Facial expression detection using convolutional neural networks.**

***Keywords*—Image Classification, Machine Learning, Convolutional Neural Network**

I. Introduction

The problem that we will be tackling through the implementation of our project is Facial Expression Recognition/classification, which is the task of classifying expressions of facial images into various categories. Solving this problem with the use of machine learning and computer vision is very sought after as it has many applications including and not limited to human behavior understanding, understanding mental disorders, consumer emotion recognition for retail analytics and so on. In order to achieve this, we will be using a convolutional neural network to build our model using keras.

II. Dataset

The dataset that we are using in this project is the FER2013 dataset [1] which is a well-known facial expression recognition dataset provided by Kaggle in one of their challenges. The data consists of 48 x 48 gray scale images of faces. The faces in all the images occupy almost the same size due to them being centered in order to maintain consistency among the data. The data is represented in CSV (Comma separated values) format.

The file consists of 2 main columns i.e. emotion and pixels. The emotion column consists of values from 0 to 6, each representing an emotion. The emotions included in this are

- 0 = Angry
- 1 = Disgust
- 2 = Fear
- 3 = Happy
- 4 = Sad
- 5 = Surprise
- 6 = Neutral

Whereas, the pixel column consists of strings which are space-separated pixel values in a row major order. The data consists a total of 35,887 rows of data. This data is taken and preprocessed, such as reshaping to a 48x48. The entire dataset is again split into the following:

- Training – 28709
- Public Test – 3589
- Private Test – 3589

We plan on using the Public Test data as validation data and the Private Test data as Test data. Additionally, we used another dataset called Japanese Female Facial Expressions (JAFFE)[2] to test the model, which consists of 213 images. These are the facial expressions in JAFFE dataset neutral, sadness, surprise, happiness, fear, anger, and disgust.
In the below plots we can see how the data is distributed among these classes for the entire dataset and the Training data, Public Testing data and Private Testing data parts separately.

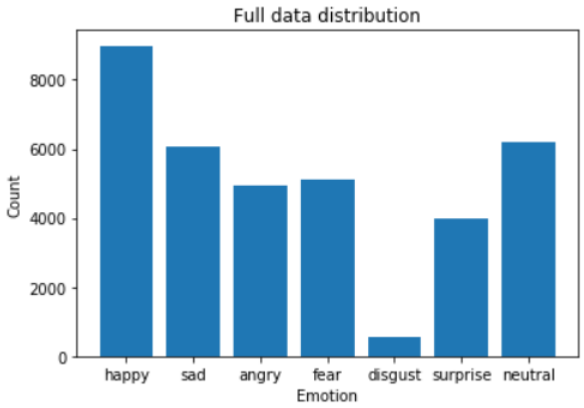

Figure 1: Full data distribution

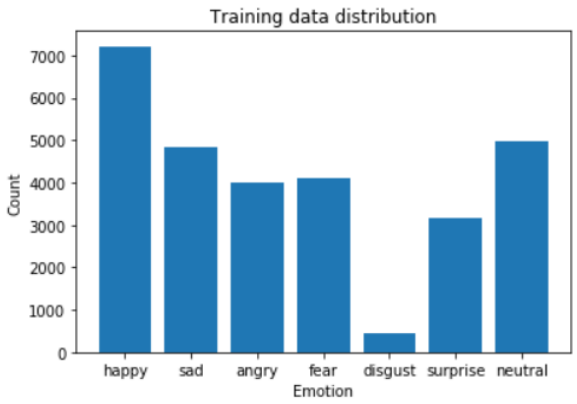

Figure 2: Training data distribution

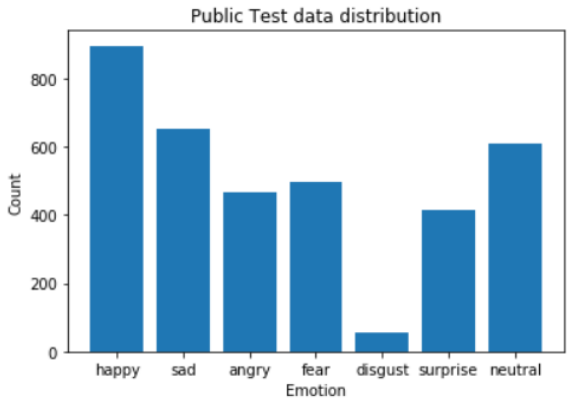

Figure 3: Public test data distribution

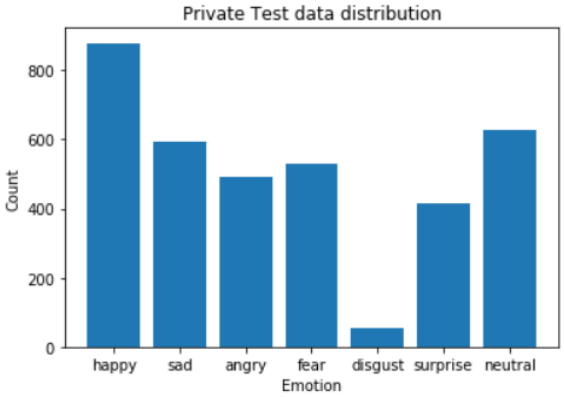

Figure 4: Private test data distribution

As seen above, we can see that the emotions we have the most data are happy, sad and neutral and the emotion with the least amount of data is disgust. So there is a good chance that our model may end up predicting happy and sad more accurately than disgust and fear.

A. *Challenges with FER2013:*

FER2013 is a challenging dataset. Below are some images from just the first 3000 samples that have caught our eye. As you can see, there are some images without any face in them, and some with animation faces. Few human faces are wrongly or incorrectly predicted.

Figure 5: Misclassifactions 1

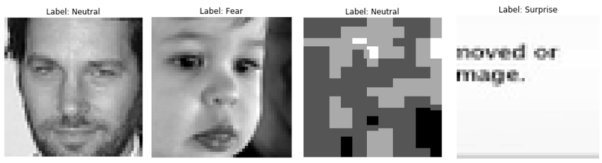

Figure 6: Misclassifactions 2

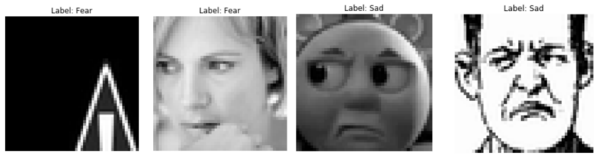

Figure 7: Misclassifactions 3

We can imagine that such number of incorrect samples will be more in the entire dataset and that is the reason that makes the classification of FER2013 hard because it will be difficult for the data to generalize correctly and be robust to incorrect data. For this dataset, the best result obtained to the best of our knowledge is 75.2% discussed in the below paper [3]. This paper has used an ensemble of modern CNN's to get that accuracy.

III. Model

We decided to implement two different models as we are a two person team and then proceed with the model that gave the best results. Let's discuss these models further below.

A. *First Model:*

For this model we will be creating a sequential convolutional neural network using keras. This model consists of 3 modules of convolutional keras layers such as, 2 Conv2D layers which are activated with a relu activation function, a Maxpooling2D layer along with a Normalization layer like BatchNormaliztion. With each module we will be increasing the number of filters for the Conv2D layers along 64, 128 and finally 256. At the end of the network we will be introducing a few core layers such as a Flatten followed by a Dense layer. This is then put though a Batch Normalization layer, a relu activation function and Dropout is then added to the output. Finally, the output is put though another Dense layer along with a softmax activation function to fulfill probability density.

The model is compiled using an Adam optimizer, categorical cross entropy loss and accuracy, loss metrics. Then the resulting model is used to fit our given input, which are collections of 48x48 images in order to train it. We achieved a training accuracy of 97% and validation accuracy of 60% after 20 epochs. The accuracy on the test data was approximately 60%.

In order to test this model further, we used the JAFFE (Japanese Female Facial Expression) dataset. We cleaned the data and made some transformations to create a dataset that we can use it with our model. We trained the model on this data and achieved a training accuracy of 97% and then a validation accuracy of 87.5% .

B. Second Model:

We also implemented a model with separable convolutions using keras. This model consists of one module with 2 Conv2D layer each followed by Batch Normalization and a relu activation function. For the next 4 modules we first have a residual layer containing a Conv2D layer followed by Batch Normalization. The residual layer is followed by 2 SeperableConv2D layers each again followed by Batch Normalization and a relu activation function and finally put through a MaxPooling2D layer. At the end we end with another layer of Conv2D followed by a GlobalAveragePooling2D layer which is then put through a softmax activation function. The Conv2D layers contain filters following the progression as follows: 8, 16, 32, 64, 128. The Final Conv2D containing the same number of filters as the number of possible classifications i.e. 7.

The model also used a l2 regularizer with a 0.01 value in order to prevent the model from over-fitting.The model is compiled using an Adam optimizer, categorical cross entropy loss and accuracy, loss metrics. Then the resulting model is used to fit our given input, which are collections of 48x48 images using cv2 from OpenCV in order to train it. With this model we were able to achieve a training accuracy of 61% and a validation accuracy of 59.7% for 20 epochs.

Since we obtained better results using the first model, we will be focusing more on the results and experiments done on that model for this report going forward.

IV. Experiments & Results

The complete code was written on Google Colab [3] platform because this platform provides GPU for faster execution.The datasets used i.e., FER2013 and JAFFE are both uploaded to Google Drive [4] and accessed from there.

In order to evaluate the performance of our model, we recorded some metrics when we compiled our model. The metrics are: Accuracy and Loss . We recorded them for both our training and our validation phases.

A. First Model:

Let's look at some representations of the performance of our model based on these metrics.

The data was already in random order. So we trained the model with Shuffle False and True, kept the learning rate as 0.001 and changed the batch size from 32 to 64. Of all these options we went with the last row in Figure 8 because that was giving us the best accuracy.

Once these hyperparameters are set, we trained the model, and then tested it. From the plots below, we can see the results.

Shuffle	Epochs	Learning Rate	Batch Size	Accuracy
False	20	0.001	32	56%
False	20	0.001	64	58%
True	20	0.001	64	60%

Figure 8: Choosing the right hyperparameters

Accuracy: From the below Figures 9 and 10, we can observe that the training accuracy steadily increases through 20 epochs and reaches a high of 97% and the validation accuracy fluctuates a bit. It reaches the highest of 60% and at the end of 20 epochs stays close to 60%.

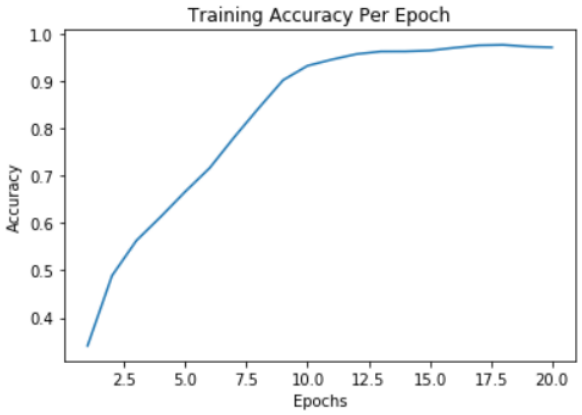

Figure 9: Training Accuracy

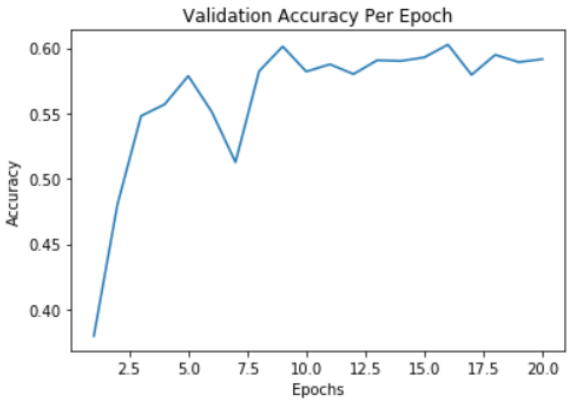

Figure 10: Validation Accuracy

Loss: From the below figures 11 and 12, we observe that the training loss decreases gradually and reaches close to zero at the end of the 20th epoch. However, the same cannot be said for validation loss. It decreases first but then increases gradually at the end of 20th epoch.

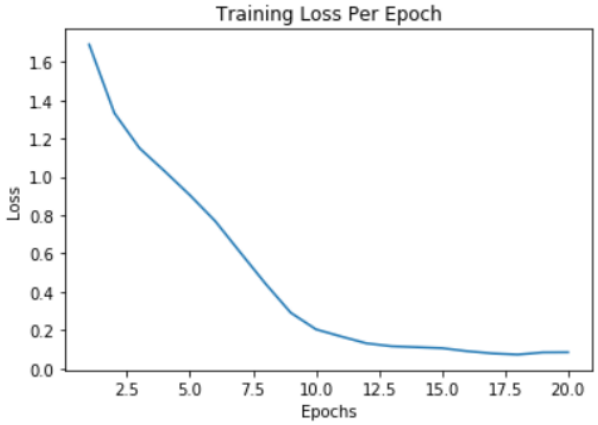

Figure 11: Training Loss

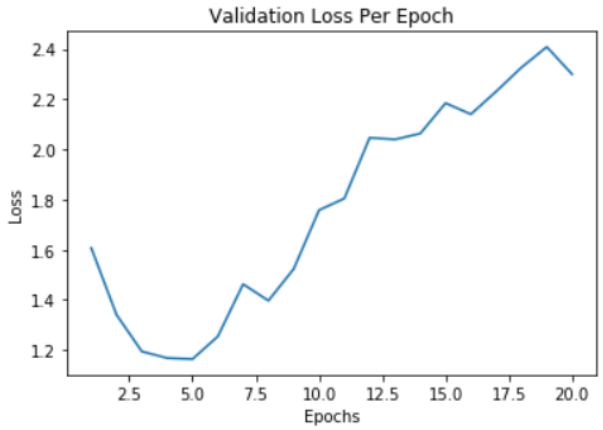

Figure 12: Validation Loss

B. *Second Model:*

Let's look at some representations of the performance of our model based on these metrics.

Accuracy: As we can see, the accuracy of our model steadily increases per each epoch as seen in the plot below which shows both training and validation accuracy of the model.

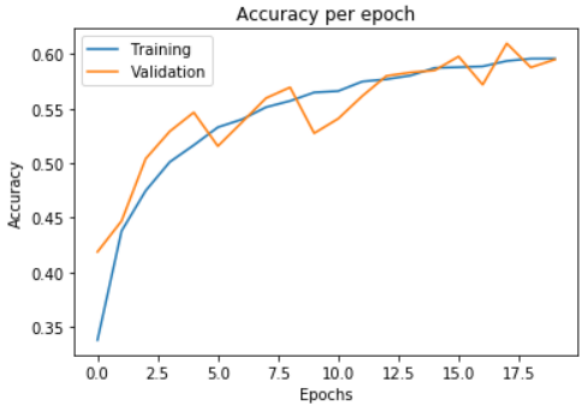

Figure 13: Training Accuracy

Loss: Like accuracy, but vice versa, we can see our model steadily decreasing the loss per epoch as seen in the plot below showing training and validation losses.

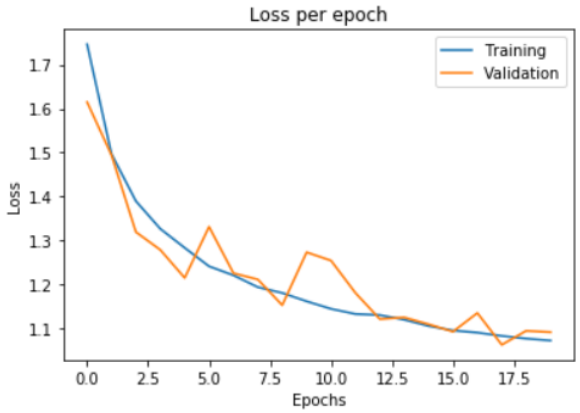

Figure 14: Training Loss

C. *Other Results:*

We tested the model, with some random images to check if it was predicting at least few of them correctly. We made the following observations:

- The model was predicting surprise almost perfectly. Usually, if there is an image with mouth open, then it is mostly predicting it as surprise, sometimes fear. Figure 16 is a correct surprise prediction. However, if we look at Figure 17, it is supposed to be Fear, but probably with the mouth being open, the model predicted it as Surprise.
- Another observation is that the model predicts happy correctly, whenever the teeth of the person are visible.
- Also, angry and sad are predicted correctly at least for the below images

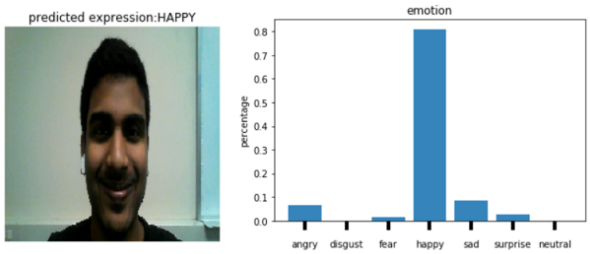

Figure 15: Happy Prediction (That's me)

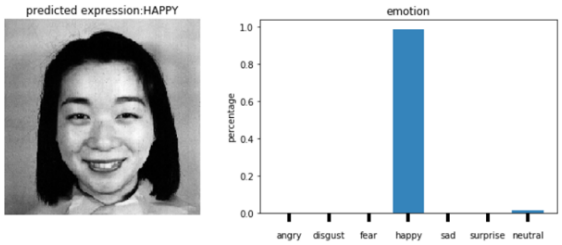

Figure 16: Happy Prediction

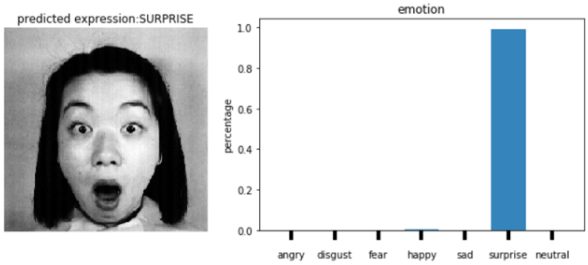

Figure 17: Surprise Prediction

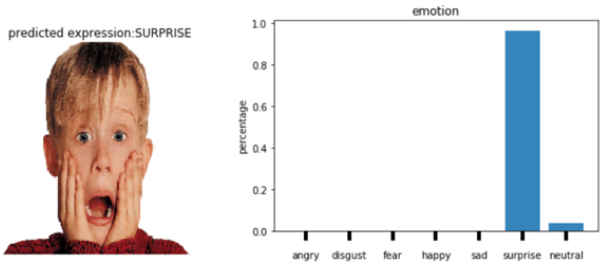

Figure 18: Fear Prediction

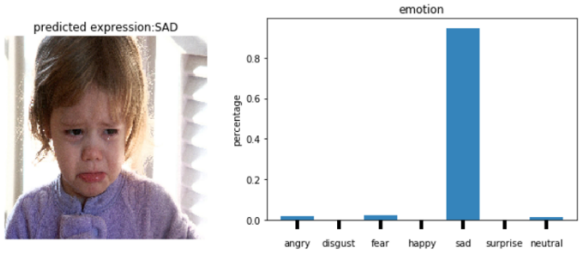

Figure 19: Sad Prediction

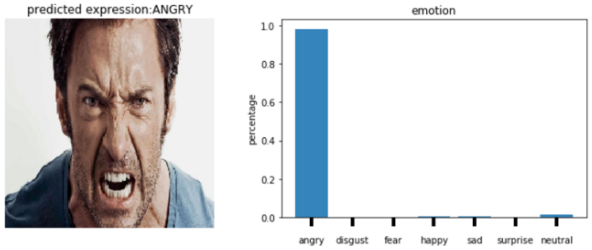

Figure 20: Angry Prediction

D. JAFFE Results:

We trained the First model with JAFFE dataset. This is a small dataset with only 213 images. The accuracies increase gradually, with the training accuracy reaching 97% at the end of 30 epochs. The validation accuracy reaches 87.5%. It is evident from the plots below.

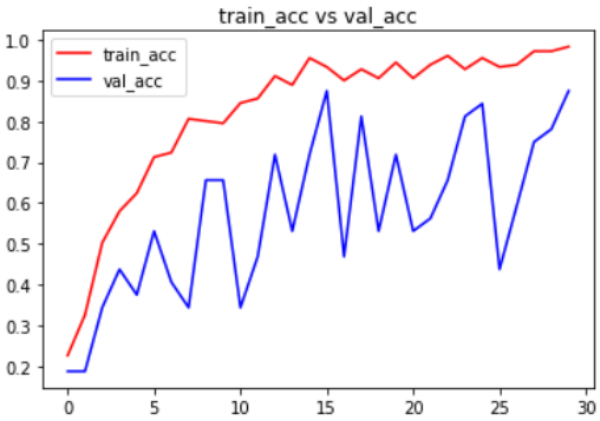

Figure 21: Training and validation Accuracy

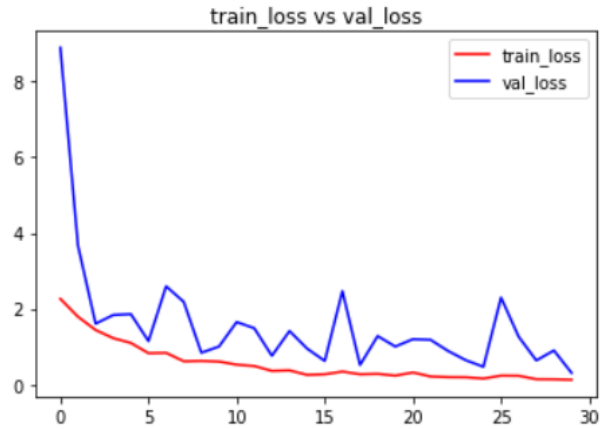

Figure 22: Training and validation Loss

V. Related Work

The dataset that we have used is FER 2013 dataset from Kaggle. They have hosted an online competition that is named "Challenges in Representation Learning: Facial Expression Recognition Challenge". There are many users currently on the leader board who have implemented the classification task using different model architectures.

So far, the state of the art result was achieved by Pramerdorfer et al. [5] with an accuracy of 75.2% outperforming other works. This approach gained this high accuracy by creating an ensemble of modern deep CNNs. Using an ensemble, as expected gave them good results.

A. Baseline

The baseline for our model is the VGG pretrained neural network available in keras deep learning library. VGG16 [6] is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 33 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

VI. Future Work

In the future, we would try to improve the accuracy of the model. Ensemble method would surely work. We were taught ensembles on the last day of the class. Given more time, maybe we can implement it to improve the accuracy.

Additionally, as another method to improve accuracy, we will try to feed the CNN with Face landmarks and HOG features.

Acknowledgment

We would like to thank Professor Dr. Hamed Pirsiavash for providing us with proper guidance and helping us getting through the project guidelines. Also, we would like to thank the TA Erfan Noury for his timely respone to queries regarding the project on Piazza.

REFERENCES

[1] FER2013 [Online] available:
https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/

[2] JAFFE [Online] available:
https://zenodo.org/record/3451524.Xfmj8ehKhPY/

[3] Google CoLab [Online] available:
http://colab.research.google.com/

[4] Google Drive [Online] available:
http://drive.google.com/

[5] State of the art FER2013:
https://arxiv.org/abs/1612.02903/

[6] VGG16:
https://neurohive.io/en/popular-networks/vgg16/