Emotion Detection Using Facial Expression

Project Report

Abstract

The task is to detect and categorize different facial expressions of humans into various categories such as anger, fear, surprise, sadness, happiness, and so on. By detecting facial expressions during regular intervals of time while doing work we try to understand human behavior, mood as well as any mental strain or disorder. Using this most of the companies can improve the mental well-being of their employees and understand their mental state while working on different tasks. We can also use the same architecture on other video-based platforms to gain important insight into a person's mental state which often gets overlooked. This can also help improve Human Machine Interaction.

1 Introduction

Facial expression recognition can play an important role for large scale as well as the small companies and industries to collect the behavioral data of their employees. This will help in analyzing the efficiency of task performance of a particular employee by detecting different expressions while working on different tasks. It is not feasible to give this task to a human hence there is a need for a more sophisticated method to automate the process with great efficiency.

2 Importance of Project

Facial expression is one of the most effective forms of nonverbal communication. The facial expressions basically tell the state of mind of the person. The Facial Expression Recognition (FER) model can be used by different companies in order to keep track of the mental health of their employees. The FER model can be used in Online Learning Environment by the teachers so that the teachers can find out whether their students are taking interest in the lecture or not. As technology is advancing at a very fast rate, we know there will be more and more Human-Machine Interaction (HMI) in the future, we also know that facial expressions give a significant amount of information that could be useful in the HMI environment.

3 Literature Survey

Various models have been developed in the past for Facial Expression Recognition (FER). Various classification algorithms like K-Nearest Neighbors which classifies the data based on the neighbors [1] have been used in the past for FER. But the KNN works well when we have the low number of features since for FER we require a lot of features like spatial features, textures, etc. so to overcome this high dimensionality of the input data, SVM can be used as it is good for classifying the data having high dimensions. [2] Vasanth uses SVM for predicting facial expressions. In the first step they did face localization for getting the face image from the whole picture, in the second step they found out the exact location of the eye and mouth and after that, they did the feature extraction at these positions. In the final classification step, they use SVM for facial expression recognition of images.

In paper [3],the FER2013 dataset is introduced. It explains all about the dataset and states the human error for it. It also compares different models on the same dataset to find whether or not feature learning algorithms are ahead of other methods for this task. It concludes that convolutional neural networks are capable of outperforming hand-designed features but the difference in accuracy is not very extreme.

A simple yet efficient convolution neural network called LeNet-5 is described which was created to classify handwritten digits of size 32x32 pixels. This CNN is compared with various other

benchmark methods for handwritten digit recognition at the time and this paper concludes that LeNet-5 outperforms all other models for this task of multi-label classification of small images.[4]

Deep and shallow neural networks are compared for the task of facial expression detection. They have used various post-processing and visualization techniques for both and have concluded that deep neural networks perform better for the task of facial expression detection. In addition to that, they also used hybrid feature sets but that did not improve the accuracy, hence it is concluded that different feature extraction techniques don't help CNNs as they intrinsically learn key facial features by using raw pixel data.[5]

There is a problem of internal covariate shift which tells that the training of neural networks is complicated as the distribution of each layer's inputs changes during training as the parameters of previous layer changes and this requires careful parameter initialization which slow down the training and makes it blatantly hard to train models with saturating nonlinearities.By using Batch normalization we can use much higher learning rates and be less careful about initialization. Batch Normalization also work as a regularizer and in some cases it removes the need for Dropout.Batch Normalization is capable of achieving the same accuracy with 14% less training steps and beats the original model by significant margin.[7]

4 Data Description

4.1 Data Source

The dataset we are using for this task is called the FER 2013 dataset. It is created by Pierre Luc Carrier and Aaron Courville using Google Image search API with some keywords like 'blissful', 'anger' etc. along with the variations of words related to gender,age, and ethnicity. We got the dataset from the kaggle website.

4.2 Data Description

The dataset contains three columns namely "pixels, "emotion" and "usage". The data given is labeled, "pixels" column comprises 48x48 pixel grayscale images of faces. Each entry in the emotion column is a label with a number (0 to 6) which represents the emotion of the corresponding face in the image (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The "pixels" column contains a string surrounded in quotes for each image. The contents of this string are space-separated pixel values in row-major order. The dataset contains 35887 images, with 4953 "Anger" images, 547 "Disgust" im-ages, 5121 "Fear" images, 8989 "Happiness" images, 6077 "Sadness" images, 4002 "Surprise" images, and 6198 "Neutral" images. This dataset contains no label noise per se and the human accuracy of this dataset is $68 \pm 5\%$.

5 Data Analysis Processing

5.1 Analysis

The data is already preprocessed into the correct format i.e the 'pixels' column contains the pixel array of grayscale images of the faces. But the distribution of data is highly imbalanced among different classes of 'emotion'. The distribution of the data points of different classes among the 'emotion' column of the dataset is as follows:

Figure 1: Distribution of classes

So we need to either balance the training data by explicitly generating more face images of the class having few data points so that each class will be considered equally probable. Also, a string is given as the value for the 'pixels' column in the dataset. So before proceeding further we need to convert these 'pixels' values into a list of numbers.

5.2 Normalization

We have done the normalization of the pixels by dividing each pixel value by 255(Using min max normalization since minimum value of pixel is 0 and maximum is 255). Now the value of each pixel is in the range of 0 to 1, both inclusive.

5.3 Data Augmentation

We have done Data Augmentation to increase the image data by randomly rotating, randomly zooming, horizontal flipping, horizontal and vertical shifting the images.After image augmentation, the distribution of classes are as follows:

Figure 2: Distribution of classes after augmentation

6 Baselines

In this project, we are predicting the 'emotion' of the given 'pixels' which represents a 48x48 grayscale image. There are a lot of features like the position of eyes, eyebrows, mouth, and other features like the texture of face, etc which we need to consider in order to predict its expression. So at first we used SVM for facial expression classification since it is good at handling high dimension data.

Since we have to classify images into multiple labels we also consider using CNN as CNN works great with images because of properties like parameter sharing and sparsity of connection.

6.1 Support Vector Machine

We have used SVM for predicting the facial expression of the given image, for this we simply pass the pixel array in the svm model as input for its training. We got 39.2% accuracy for testing data and 61.3% accuracy for training data. We have used only 8000 data instances for training model because it becomes computationally inefficient for large number of data instances.

6.2 Convolutional Neural Networks

The architecture of the model is inspired by LeNet-5 discussed in [4]. The model has 8 layers. First layer is a convolution layer with 6 filters, each of size 5*5. This layer uses the relu activation function. Second layer is a max pooling layer of size 2*2. Third layer is a convolution layer with 16 filters, each of size 5*5. This layer uses the relu activation function. Fourth layer is a max pooling layer of size 3*3. Fifth layer is a flatten layer which converts the 3D tensor to 1D to put it into a Dense layer. Sixth layer is a Dense layer with 128 nodes which uses relu activation function. Seventh layer is a Dense layer with 32 nodes which uses relu activation function. Finally the last layer is a Dense layer with 7 nodes that uses the softmax activation function to predict the label. The model is trained using categorical cross entropy loss function and using mini batch gradient descent as the optimizer. After fitting the model with 50 epochs and 60% training and 20% validation data it is observed that the training accuracy reaches 95% and validation accuracy remains constant after a certain number of epochs. This behavior shows overfitting of the model on the training data. The final testing accuracy for the model is 49.2% which is not good as compared to the benchmark accuracy we want to reach for this dataset, i.e 68% which is the human accuracy for this dataset.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 44, 44, 6)         156
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 18, 18, 16)        2416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 128)               73856
_________________________________________________________________
dense_1 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 231
=================================================================
Total params: 80,787
Trainable params: 80,787
Non-trainable params: 0
```

Figure 3: Architecture of model

7 Optimizations

Choosing the CNN model some optimization techniques are used to reduce the overfitting and also to increase the validation accuracy as well as the F1 Score the CNN model.

7.1 Transfer Learning with ResNet:

We as humans learn things faster if we have already learned something similar. The same concept applies to machine learning also, to classify the data we do not need to train the model from scratch instead of this we could use the same pre-trained

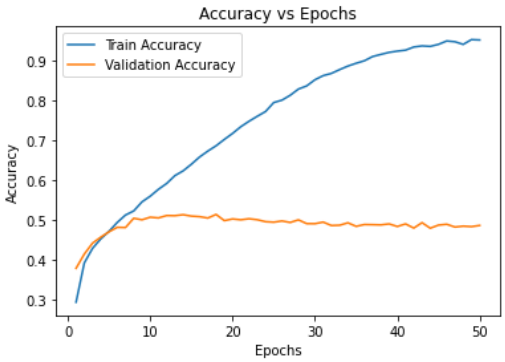

Figure 4: Accuracy vs Epochs

model to solve our classifying problem. This concept is called transfer learning. We have use ResNet

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 2, 2, 2048)	23587712
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 128)	1048704
dense_1 (Dense)	(None, 32)	4128
dense_2 (Dense)	(None, 7)	231

Total params: 24,640,775
Trainable params: 24,587,655
Non-trainable params: 53,120

Figure 5: Architecture of Transfer Learning with ResNet

as the base model for Transfer learning adn we have added two densely connected layers in the end to process the ouput from ResNet with newly trained weights. The results are as follows: From these graphs we can see that the model is not able to predict the validation set correctly and is still overfitting a lot on the training set. The evaluation results of the model on the dataset is as follows: Accuarcy and F1 are similary to what we got in the baseline, this is because ResNet works good on Face detection but here we need to extract the features from face to detect the expressions which is not a good job for ResNet.

7.2 Batch Normalization with Deep Neural Network:

It is a technique to improve the performance of the Neural Network model. In this method, we normalize the inputs of each layer in such a way that, they have a mean activation output zero and a unit standard deviation (i.e =0 and =1). So along with the normalization of the input, we also do the

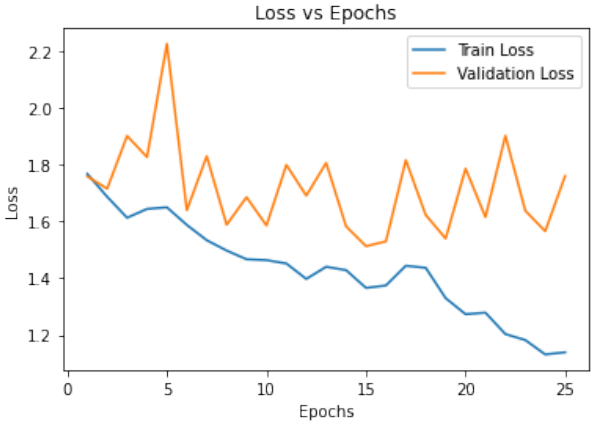

Figure 6: Accuracy vs Epochs

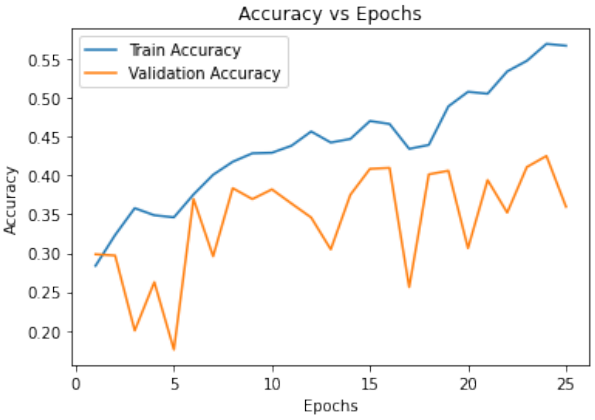

Figure 7: Loss vs Epochs

F1 Score : 0.422800902197653
Classification Report

	precision	recall	f1-score	support
0	0.270753	0.348651	0.304803	1001.0
1	0.600000	0.159292	0.251748	113.0
2	0.241379	0.160199	0.192584	1005.0
3	0.567970	0.684624	0.620865	1782.0
4	0.352201	0.274510	0.308540	1224.0
5	0.861314	0.149178	0.254310	791.0
6	0.355351	0.549921	0.431726	1262.0
macro avg	0.464138	0.332339	0.337797	7178.0
weighted avg	0.439451	0.403455	0.384109	7178.0

Figure 8: Classification Report

normalization of the outputs of activation functions of each neuron of a layer before passing it to the next layer.

$$h_{ij}^{norm} = (h_{ij} - \mu_j)/\sigma_j \quad \text{and} \quad h_{ij}^{final} = \gamma_j * h_{ij}^{norm} + \beta_j$$

Here

$$\gamma_j \quad \text{and} \quad \beta_j$$

are trainable parameters that will get trained along with the weights.

7.3 Reducing overfitting with DropOut with Regularization

Deep Learning Neural Networks overfit very quickly on the datasets especially when the dataset is small having fewer variations. One way to overcome this problem is to use the ensemble technique. We can make different base models and can find the average of the results of the base models. But this ensemble technique is computationally very expensive as we need to store the results of the base models which is very large. So for reducing overfitting we can use the dropout method. In this method, we drop (or disable) some number of nodes of the layers (except the output layer). Thus using this method we will be able to simulate a large number of different models by randomly dropping out nodes during training. This method is relatively cheaper than using actual base models and also gives good output results. The architecture of the model is as shown below: Here every

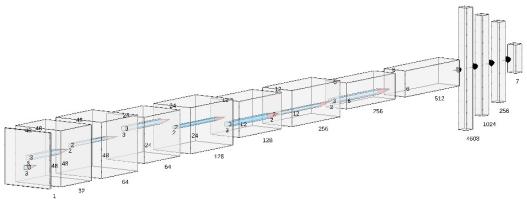

Figure 9: Architecture after dropout

convolution layer is followed by a max pooling and a dropout layer.Since a dropout layer is disabling some nodes of the convolution layers during every training epoch, the result of our model can not be highly dependent on one set of fixed nodes hence reducing overfitting by a lot.We can see that the model is not overfitting by the following graphs.

We checked how the model is working with the test set, it was observed that the accuracy on the test is **64.4%** which is very close to human accuracy for this dataset. Since accuracy is not a good measure when it comes to understanding how a well a

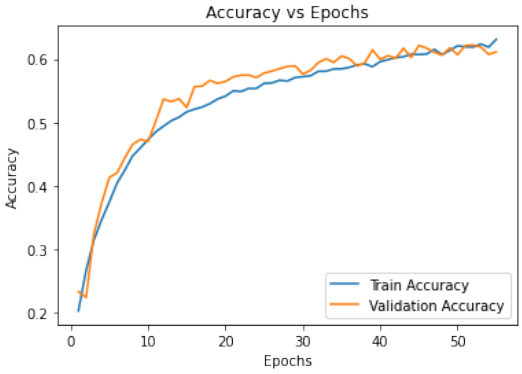

Figure 10: Accuracy Plot after dropout

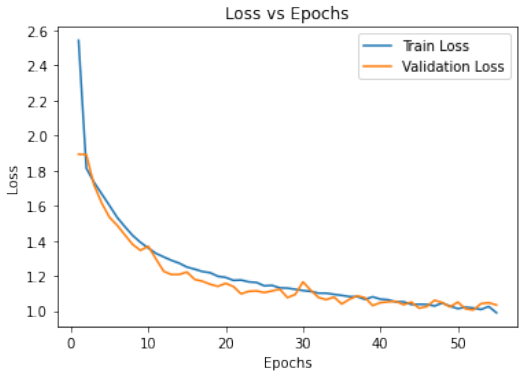

Figure 11: Loss Plot after dropout

model is working on 7 class classification problem, hence we have calculated Precision, Recall and F1 score with their micro and macro averages for all the classes.Below is the classification report and confusion metrics

	precision	recall	f1-score	support
0	0.594171	0.542221	0.567009	3008.0
1	0.825788	0.828161	0.826973	1740.0
2	0.502268	0.535116	0.518172	3104.0
3	0.792358	0.872164	0.830348	3614.0
4	0.620164	0.429907	0.507799	2461.0
5	0.841761	0.608864	0.706617	2324.0
6	0.504520	0.728682	0.596228	2451.0
macro avg	0.668719	0.649302	0.650449	18702.0
weighted avg	0.661203	0.649342	0.647364	18702.0

Figure 12: Classification Report after dropout

8 Final Model

Out of all the models explained above the CNN along with dropout layers and max-pooling works best for this dataset.We have used our own CNN

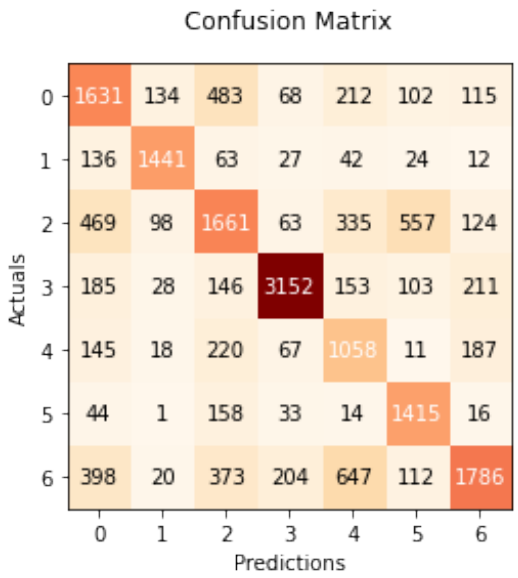

Figure 13: Confusion Matrix after dropout

architecture which consists of 5 convolution layers.The 1st convolution layer is directly connected to the 2nd convolution layer.There onward each convolution layer is followed by a max-pooling layer and a dropout layer this helps in feature extraction and reducing overfitting.
We can see from Figure 11 that there is no overfitting as the training and testing loss are decreasing together. In addition to that our accuracy is very close to the benchmark accuracy for this project that is the Human accuracy on this dataset. We observed a good F1 score for each of the classes as well as for the overall problem.

9 Result/ Analysis

9.1 Comparison of final model with Baseline:

In the baseline CNN architecture we have observed very poor validation accuracy of 49% and the model is overfitting on training dataset.
After applying dropout and data Augmentation on the dataset to make the dataset balanced the CNN with dropout layers works well on the dataset as shown by the above plots (Figure 10 and Figure 11). The accuracy for this model is then increased to 64% and also the overfitting is reduced drastically on the training set. F1 score for the same is 0.67 which is a good score for this dataset as compared to the benchmark. The outcomes of the best model is shown below:

	precision	recall	f1-score	support
0	0.594171	0.542221	0.567009	3008.0
1	0.825788	0.828161	0.826973	1740.0
2	0.502268	0.535116	0.518172	3104.0
3	0.792358	0.872164	0.830348	3614.0
4	0.620164	0.429907	0.507799	2461.0
5	0.841761	0.608864	0.706617	2324.0
6	0.504520	0.728682	0.596228	2451.0
macro avg	0.668719	0.649302	0.650449	18702.0
weighted avg	0.661203	0.649342	0.647364	18702.0

Figure 14: Classification Report after dropout

10 Conclusion

Facial Expression from the input image is detected and classified into one of the 7 classes (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).It is observed that the Normal CNN model with some optimizations outperform the baseline models and a pre-trained **ResNet** Model with an accuracy of 63% and a F1 Score of 0.67 which is good enough to classify facial expressions as the human accuracy for detecting facial expression is $68 \pm 5\%$ It is further observed that pretrained models are not always gives the best results unless used for the same purpose for which they are trained, the ResNet model failed to give a good score because face detection and facial expression recognition are different tasks the later needs to extract the features from the face and hence is a more complicated Task.

11 Work Distribution

At first the data is analysed and preprocessed by Anurag to make it feasible to feed it into a ML/DL model,then a SVM model is trained. It was than modified and tuned by Jayesh to improve the accuracy on the both training and testing set. The Deep Learning Model(CNN) was then compiled and trained by Risabh to make it more robust.This model is then evaluated and tested by Jayesh using the accuracy curve. Short comings of CNN model is then analyzed and future improvements are suggested by Risabh. We come up with some optimization techniques to reduce the overfitting, Data Augmentation is done by Anurag , different optimization techniques are used by Jayesh and Risabh to come with the final results.

All the members contributed in preparing Literature Review, report and PPT. All the members

contibruted equally to give better results.

References

[1] Nazia Perveen, Nazir Ahmad, M. Abdul Qadoos Bilal Khan, Rizwan Khalid, Salman Qadri, "Facial Expression Recognition Through Machine Learning".

[2] Vasanth P.C*, Nataraj K.R, "Facial Expression Recognition Using SVM Classifier".

[3]Ian J. Goodfellow 1 , Dumitru Erhan 2 , Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio, "Challenges in Representation Learning: A report on three machine learning contests"

[4] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[5] Shima Alizadeh, Azar Fazel, "Convolutional Neural Networks for Facial Expression Recognition".

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

[7] https://arxiv.org/abs/1502.03167