Facial Emotion Recognition Using Machine Learning

Jay Chaphekar 1001763932 **Shubham Sinnarkar** 1001760386 **Apoorv Pandkar** 1001553485

Abstract

Emotion detection has been one of the topics attaining great attention. Diving into this, our design helps in mapping human emotion to some kind-of input data obtained from a human. Extending this, our paper proposes a convolutional neural network prototype for classifying and identifying the emotion on a face, real-time. we have implemented model architecture from scratch using Keras to detect emotions using Convolutional Neural Network (CNN). This research uses the FER13 dataset coupled with COCO dataset which is amassed for facial expression recognition analysis. The prototype has an average accuracy of 65%. Furthermore, the prototype has been classified with seven emotion classes namely, Anger, Disgust, Sad, Happy, Neutral, Surprise, Fear.

INTRODUCTION

Automated emotion detection is quite a task in machine learning. This challenge is an emerging topic, especially in computer vision. The real challenge lies in detecting each facial expression and classify them into their emotion classes with accuracy.

In the implementation of this model we have used sequential model of Keras which executes implementation of neurons in the sequential order. We have used various functionalities which helped us to achieve better results with regards to performance, for instance, Conv2D, Batch Normalization, Relu activation, MaxPooling.

Adding to this, we did preprocessing of the datasets in which we cropped images from COCO dataset by adding facial boundaries to the faces with gray-scaling and saved the faces in consistent format. For FER2013 dataset, we calculated number of features in the dataset and reshaped those images. Also, we have created a model to extract features from the data and from those features we have predicted the emotion from the images to build our model. We have used Keras library which allows us to create Neural Network from scratch by adding various layers and functions to tweak the functionalities of the network based on our needs.

DATASET:

We have used two datasets here, namely FER13(Facial Expression Recognition) and COCO(Common objects in context), to implement our model. FER13 is the one of the most popular dataset used to do facial recognition. This dataset has about 36000 images of faces with a resolution of 48 x 48. This dataset contains grayscale images of faces automatically registered. FER13 has labelled data classified into the seven emotions namely, Anger, Disgust, Sad, Happy, Neutral, Surprise, Fear. COCO dataset contains images of everyday scene and provides contextual information. For COCO dataset, We have gone ahead and labelled this dataset using our model. Also, we have detected the face out of the images and cropped out the faces in 48 x 48 pixels. Furthermore, we have gone ahead and grayscale them. These grayscale images have been saved and can be used as a different dataset which includes only human faces which then have been fed in the system to make predictions.

PROJECT DESCRIPTION:

The goal of the project was to develop real-time facial emotion detection using COCO dataset. The main barrier was that we could not use COCO dataset for emotion detection as COCO dataset does not have a classification for human emotions. So, we developed a model and trained it using FER2013 dataset which has a classification for the seven emotions over 36000 images. We used convolutional neural network with the help of various research papers and convolutional neural network architectures such as VGG13, VGG16 and Xception. Furthermore, by learning this architectures we created a four layer convolutional neural network of our own with additional two dense layers and used that model to create a new dataset using face extraction from COCO dataset images and labeled those images with emotions.

Main reference for our project was Facial Expression Recognition using Convolutional Neural Networks: State of the Art by Christopher Pramerdorfer, Martin Kampel. From that, we learnt that how to build an efficient Convolutional Neural Network for image classification. We used medium.com to learn pre-processing and evaluation of better result.

As we mentioned before, the above references which we used for our project either did not use the datasets which we have used or the implementation of architecture did not meet our requirement as we not only wanted to build a real-time emotion recognition classifier but also to predict emotions on the human images of the COCO dataset. For that we have used several techniques from above references which helped us in the creation of our own model. For the preprocessing part, we used the file which had facial coordinates of COCO dataset which we used to draw rectangles around them and later cropped them and pre-processed them by resizing them 48 x 48 shape and gray scaling the images. For the FER2013 dataset, we normalized the dataset by using mean and standard deviation of the dataset and resized the images to 48 x 48. After that, we split the data into training and testing dataset. Again, we have normalized both the datasets and reshaped the images.

Later, we used sequential modelling which is a plain stack of layers with exactly one input and one output tensor. After that, we added a convolution layer of 64 filters with a kernel size of 3 x 3 with the ReLu activation function, to help the computation we added batch normalization layer which normalizes the data to a mean of zero and standard deviation of one. Later, we added Max Pooling layer which extracts maximum features from the matrix with the stride of 2 x 2 which means we moved Max Pooling layer window by 2 x 2 to extract maximum features from data. After that we added dropout layer with 0.25 dropout to reduce the over-fitting. In dropout layer, we randomly dropped different layers of networks or make the layer's input 0 so that the model would not select the same layer again and again and explore other options so that the model wont learn based on learning data. In the next layer, we have used the filter of 128 with 3 x 3 kernel size. We also added batch normalization and Max Pooling with a stride of 2 x 2 and dropout of 0.25. In the third layer, we have used the similar configurations to that of second layer, that is the filter of 128 with 3 x 3 kernel size. We also added batch normalization and Max Pooling with a stride of 2 x 2 and dropout of 0.25. In the fourth layer, we have used the filter of 64 with 3 x 3 kernel size. After that, we added flatten layer which adds one extra channel dimension with the output shape of batch,1 which shapes the data in one dimensional array which makes it easier for the model to provide input as in the sparse matrix most of the values are redundant. Flatten helps in increasing the computation speed and is a good input to add to the dense layers. So, after four convolutional layers we added two dense layers which in opposition to the convolutional layers has each input node is connected to the every other input node of next layer. We have added 2 dense layers of 1024 filter with activation function ReLu. After that, for the last output layer as there are seven labeled outputs we had to give filter as the number of labels to our model. We added

SoftMax activation function which is the best activation function for the categorical data.

We compiled the model using Adam optimizer which implements Adam algorithm. The Adam optimizer is a stochastic gradient descent method that is based on adaptive estimation of first order and second order movements.

For the loss function which is used to minimize gradient values based on biased values, we need to have minimum loss possible to give accurate predictions. we have used the CategoricalCrossEntropy matric which is efficient in dealing with labels and prediction. This class is used where there are multiple label classes.

ACCURACY/PERFORMANCE:

The main objective of building the model is to get the best possible accuracy from the dataset which includes tuning hyperparameters and changing the architecture of the convolutional neural network to achieve the best possible results in order to achieve these goals. Throughout the duration of progression of project we have experimented to build an efficient network which predicts facial emotions effectively. We have read from various references about how to build a good image classification model and we implemented the model which is best of our knowledge which has significant improvement from the references. To build this model, we have tried different filters for each convolutional layers with different number of layers, activation functions, dropouts.

We have tried different activation functions such as ReLu, Sigmoid, TanH, SeLu. Although we knew that the ReLu was the best activation function , we observed that the accuracy is significantly improved by using ReLu as an activation function. We also so that changing percentage of dropout layers or removing the dropout layers also affects the performance of the model. We saw that adding dropout layers reduced overfitting that means training accuracy was going exceptionally higher as compared to the training accuracy. By adding those layers, we managed to reduce the growth of the training accuracy as compared to the testing accuracy. We also saw that adding batch normalization to the convolutional layer reduces the computation speed and gives faster results.

In addition to the network architecture, tuning hyperparameters was the important task of getting good performance. There are several hyper-parameters apart from loss and activation functions such as batch size, number of epochs. We tested our model with different batch size and epochs. Batch size is the number of input we provide to our model as in neural network we do not provide single inputs which can take ample amount of time to be executed. So, we provide inputs in batches to aid the computation. Number of epochs are the number of times we run our model to observe the results of model and to get the appropriate number of epoch which can be maximized performance as part of training our model we tried batch sizes of 16,32,64,128 and 256 which slightly affected performance and batch sizes of 16,32 took more time than rest of the batch sizes. We thought 128 batch size was optimal for model.

For the number of epochs, we tested our model on various epoch sizes such as 20,30,50,100,200 and each time we got training accuracy around 60-65%. After 160 epochs, the accuracy reached 65% which is the highest we got from our model. During training and testing of our model we saw that after 150 epochs training accuracy was going around 90% and loss was closer to 0 and there was no significant improvement in our model. From that we concluded that model was over-fitting and we had to stop. So, from these observations we can conclude that the batch size of 128 and number of epochs equal to 160 gives optimal results for our model.

In comparison to the references, we created our different model using different architecture and tuning various different hyper parameters we got 6% more accuracy from our referenced projects.

In addition to the model accuracy, we found that based on human judgement our model also predicted fairly accurate results on the images of COCO dataset.

CONTRIBUTION IN THE PROJECT:

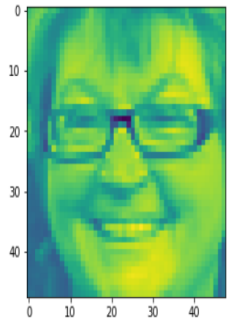
- Pre-processed COCO and FER2013 datasets.
- Extracted faces from COCO dataset images.
- Created a Convolutional Neural Network Architecture.
- Improved accuracy from 58% to 65%.
- Used the model to predict emotions on extracted faces of COCO dataset.
- Created an application to detect faces from real time video an give real time emotion prediction.

ANALYSIS:

What did I do well?

We created a model from scratch which improved accuracy of our model from 58% to 65%. Along with the accuracy, we were also able to predict emotions on COCO dataset images which gave fair results on large set of images. The application which we created to detect emotion in the real-time environment also provides efficient results.

```
plt.imshow(array_to_img(X_test[100]))
plt.show()
```




```
test_y[100]
Emotions = ["Angry", "Disgust", "Fear", "Happy", "Sad", "Surprise", "Neutral"]
```

```
op1 = model.predict(np.array(X_test[100]).reshape(-1,48,48,1))
Emotions[np.argmax(op1[0])]
```

```
'Happy'
```

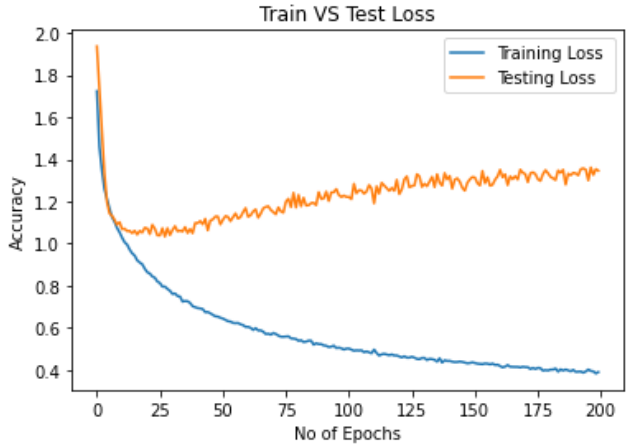
What could I have done better?

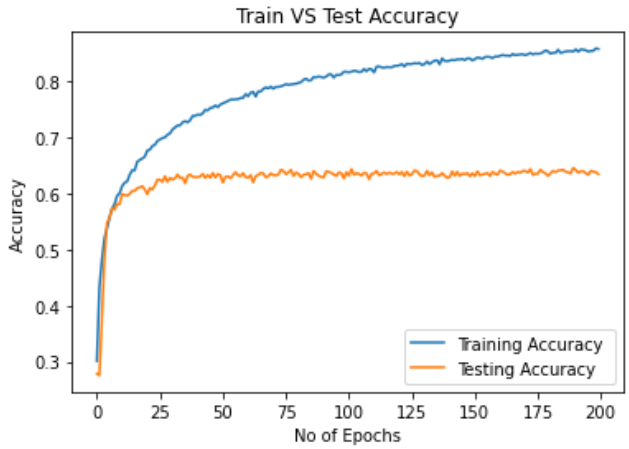
We could have improved the accuracy front by adding more images and a larger dataset. Glitches in classifying emotions where observed when the input image was blurry. Thus, we could improve the preprocessing to accommodate that limitation.

What is left for future work?

Facial emotion recognition has many applications in various fields. It can be used in Virtual reality, Human computer interactions, Gaming. This prototype can be used to develop a full-fledged application to collect emotional data and enhance the intelligence of various products. We could also pre-process COCO dataset efficiently to capture sharp

images from the dataset so we could predict those blurry images efficiently.





CONCLUSION

We have proposed a successful project producing prominent results, given an average accuracy of 65%. We have suggested a Convolutional Neural Network architecture for facial emotion recognition. We were also able to classify COCO dataset images based on their emotion and our real time video detection also works efficiently on any human face. Thus, we can say that the model is fairly accurate but far from perfect and needs some improvements.

REFERENCES:

[1] Clancey, Video-Based Emotion Recognition using CNN-RNN and C3D Hybrid Networks by Yin Fan, Xiangju Lu, Dian Li, Yuanliu Liu

[2] On Line Emotion Detection Using Retrainable Deep Neural Networks by Dimitrios Kollias, Athanasios Tagaris and Andreas Stafylopatis

[3] Real-time Convolutional Neural Networks for Emotion and Gender Classification by Octavio Arriaga Hochschule Bonn-Rhein-Sieg, Paul G. Ploger, Matias Valdenegro

[4] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

[5] Realtime-Emotion-Detection/Emotion_Anaysis.ipynb at master · neha01/Realtime-Emotion-Detection (github.com)

[6] face_classification/report.pdf at master · oarriaga/face_classification (github.com)

[7] From raw images to real-time predictions with Deep Learning | by Jonathan Oheix | Towards Data Science

[8] gitshanks/fer2013: Facial Emotion Recognition on FER2013 Dataset Using a Convolutional Neural Network (github.com)

[9] ndl.ethernet.edu.et/bitstream/123456789/60914/1/7.pdf#page=10