# Real-Time Facial Emotion Recognition

*Jay Choksi, Prathamesh Tari, Dhruv Patel*

## ABSTRACT

***In this paper real-time facial emotion recognition by implementing a Convolution Neural Network has been proposed. The fact that deep learning can be used to recognize the facial emotions of humans unlocks new wide-impact possibilities to the field of human-computer interaction. The main task here is to recognize constantly changing facial emotions of a person and segregation them into 1 of the 7 categories namely: happy, sad, neutral, surprised, anger, fear and disgust. The conditions of the image or video are not necessarily ideal with low illumination, scale, lesser pixels based on the camera quality, etc. A six convolution-layered CNN is implemented to reach at an optimum conclusion. Experiments and trials based on different models related comparison show shat the best combination is achieved using the Adam optimizer along with batch processing and data augmentation techniques obtaining a test accuracy around 65% on the FER2013 dataset.***

## INTRODUCTION

It is easy to classify a person's facial expressions by looking at them through the eyes. It is however difficult to implement the model used by the computer to identify the facial expression with the same accuracy and precision as humans. One of the major non-verbal ways of communication between humans. Hence, recognizing facial emotions is an integral aspect in human computer interaction also having endless applications. The range of applications include customer specific or targeted advertisements, surveillance, robot-human interaction, virtual reality (VR), etc. There are a plethora of applications for real-time facial emotion recognition: a real time analysis of customer satisfaction in a spa. A smart home changing music and lighting based on owner's emotions.

### *1. DATASET DEMOGRAPHICS*

The facial emotions have been ubiquitous and constant throughout the years and one such dataset which captures the humongous variety and quantity of emotions is the FER2013 dataset used in a Kaggle competition.

*Figure 1 Example images from the FER2013 dataset, illustrating variations in lighting, age, pose, intensity of expressions and occlusions that occur under realistic conditions. These images depict expressions, namely anger, disgust, fear, happiness, sadness, surprise, neutral.*

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial emotions in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

The Training csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. Test csv contains only the "pixels" column and your task is to predict the emotion column. The training set consists of 28,709 examples.

## 2. *CONVOLUTION NEURAL NETWORK MECHANICS*

Convolution Neural Network is a class of deep, feed-forward artificial neural networks that has been applied to analyzing visual imagery. They are made up of neurons that have learnable weights and biases. Each neuron receives some input, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other.

Layers in CNN:

• INPUT layer will hold the raw pixel values of the image, of width 32, height 32, and with three color channels Red, Green, Blue.

• CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

• RELU layer will apply an elementwise activation function, such as the max (0, x) thresholding at zero.

• POOL layer will perform a down sampling operation along the spatial dimensions (width, height)

• FC (i.e. fully-connected) layer will compute the class scores.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 44, 44, 64) | 1664 |
| conv2d_2 (Conv2D) | (None, 40, 40, 64) | 102464 |
| batch_normalization_1 (Batch | (None, 40, 40, 64) | 256 |
| max_pooling2d_1 (MaxPooling2 | (None, 20, 20, 64) | 0 |
| conv2d_3 (Conv2D) | (None, 18, 18, 128) | 73856 |
| conv2d_4 (Conv2D) | (None, 16, 16, 128) | 147584 |
| batch_normalization_2 (Batch | (None, 16, 16, 128) | 512 |
| max_pooling2d_2 (MaxPooling2 | (None, 8, 8, 128) | 0 |
| conv2d_5 (Conv2D) | (None, 6, 6, 256) | 295168 |
| conv2d_6 (Conv2D) | (None, 4, 4, 256) | 590080 |
| batch_normalization_3 (Batch | (None, 4, 4, 256) | 1024 |
| max_pooling2d_3 (MaxPooling2 | (None, 2, 2, 256) | 0 |
| flatten_1 (Flatten) | (None, 1024) | 0 |
| dense_1 (Dense) | (None, 1024) | 1049600 |
| batch_normalization_4 (Batch | (None, 1024) | 4096 |
| activation_1 (Activation) | (None, 1024) | 0 |
| dropout_1 (Dropout) | (None, 1024) | 0 |
| dense_2 (Dense) | (None, 1024) | 1049600 |
| batch_normalization_5 (Batch | (None, 1024) | 4096 |
| activation_2 (Activation) | (None, 1024) | 0 |
| dropout_2 (Dropout) | (None, 1024) | 0 |
| dense_3 (Dense) | (None, 7) | 7175 |

Total params: 3,327,175
Trainable params: 3,322,183
Non-trainable params: 4,992

We implemented a 6 convolution layered neural network for visualizing the internal working from layer-to-layer.

Visualizing intermediate activations consists in displaying the feature maps that are output by various convolution and pooling layers in a network, given a certain input. This gives a view into how an input is decomposed unto the different filters learned by the network. Each channel encodes relatively independent features, so the proper way to visualize these feature maps is by independently plotting the contents of every channel, as a 2D image. 2 The 1st layer acts as collection of various edge detectors. At that stage, the activations are still retaining almost all of the information present in the initial picture.

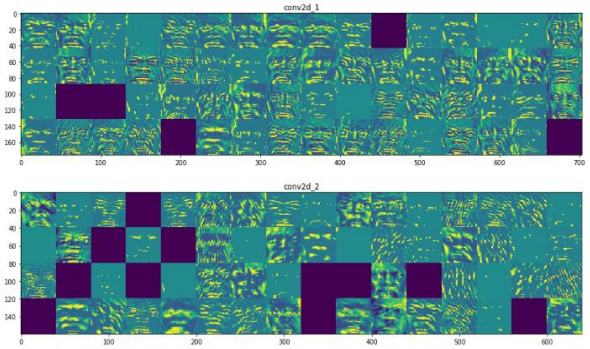


*Figure 2 Showing the first two layers in our CNN model*

*As we go higher-up, the activations become increasingly abstract and less visually interpretable. Higher-up presentations carry increasingly less info about the visual contents of the image, and increasingly more info related to the class of the image.*

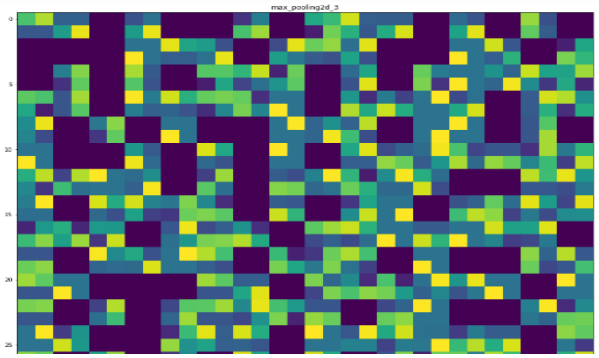


*Figure 3 Showing the 12th layer in our CNN model*

The activations of layers higher-up carry less and less information about the specific input being seen, and more and more information about the target. A deep neural network effectively acts as an information distillation pipeline, with raw data going in, and getting repeatedly transformed so that irrelevant information gets filtered out while useful information get magnified and refined.

Thus, In this paper, we consider the task of predicting basic emotions from single images and live video stream using CNNs. We note that it is straight-forward to adapt image-based methods to support image sequences by integrating per-frame results using graphical models. The conclusions drawn in this paper are thus relevant for sequence based facial emotion recognition as well.

## EXPERIMENTS

We are implementing CNN model with three convolutional layers. The input image size has been fixed as 48x48. We are using RELU as activation function. Tests were done using SELU, but the model was underfitting and thus choosing SELU over RELU was not an optimal choice. We trained initially with 50 epochs and gradually increased the epochs as the training accuracy was also increasing with the number of epochs. We trained the model on 200 epochs and we were getting training accuracy of 86%. But the model was overfitting as the test accuracy was around 40%. So, in order to decrease the overfitting, we decrease the number of epochs to 100. This solved the overfitting problem. To increase the testing accuracy, we added batch normalization in every convolution layer. Batch normalization reduces the amount by what the hidden unit values shift around. It normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. Adding batch normalization further improved the test accuracy to up to 65%. We fine-tuned hyper-parameters of the model and finally got training and validation accuracy of 67%.

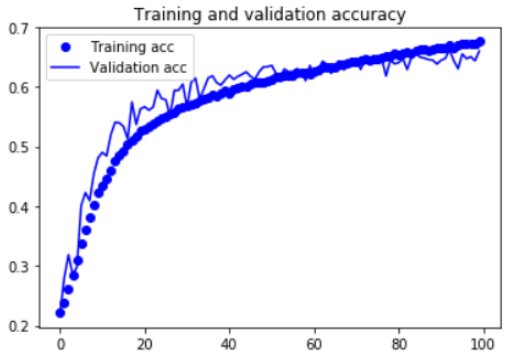


*Figure 4 Training and validation accuracy*


*Figure 5 Training and validation loss*

We checked the model by making prediction of a custom image out of test set. We chose the Figure 7 for the test.

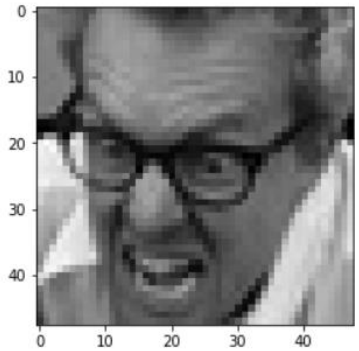


*Figure 6 Sample image for testing*

And we can see what the model predicted in the Figure 8.

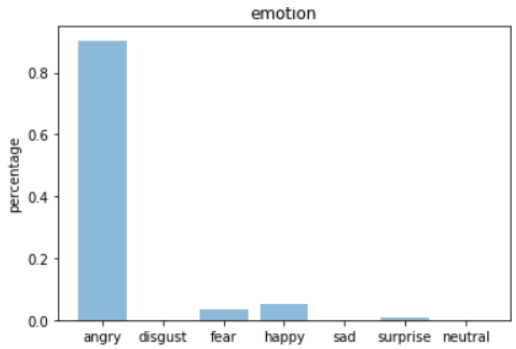


*Figure 7 Prediction of model on sample image*

Out of the many test we conducted it seems that model is able to predict well for expressions such as happy, neutral, angry, surprise. The model however struggles to predict disgust. It is because the training set didn't have enough image for the model to learn. We can see from the confusion matrix in Figure 9 the model performance.

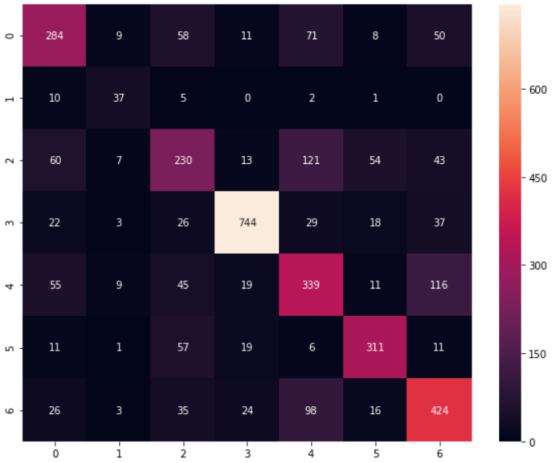


*Figure 8 Confusion Matrix for prediction of various expressions [Index: 0 – Angry 1 – Disgust 2 – Fear 3 – Happy 4 – Sad 5 – Surprise 6 - Neutral]*

For live streaming we are using OpenCV model to capture live frames from the webcam. The individual frames are pre-processed by first cropping it. Next, the cropped image is converted to grayscale and an adaptive histogram equalization technique CLAHE is applied. Contrast Limited Adaptive Histogram Equalization in OpenCV divides the image into tiles of specified pixels and applies histogram equalization of tiles. The resulting single or multiple bounding boxes are fed into the trained CNN model for expression recognition. A face bounding rectangle is drawn on the frame and the classified emotion is plotted. OpenCV captures flipped frames in BGR color space, the frames are flipped as well as converted to RGB before preprocessing. The process of real time expression classification is shown in Figure 10.

## CONCLUSION

We successfully implemented a real time facial expression recognition system. After various test of face tilts, illumination variation and fine tuning various hyper parameters of the model we got overall test accuracy of 65%. The model was trained on Kaggle competition dataset and the winner of the competition had got 35% accuracy, so our model has twice the better accuracy. The model was trained on

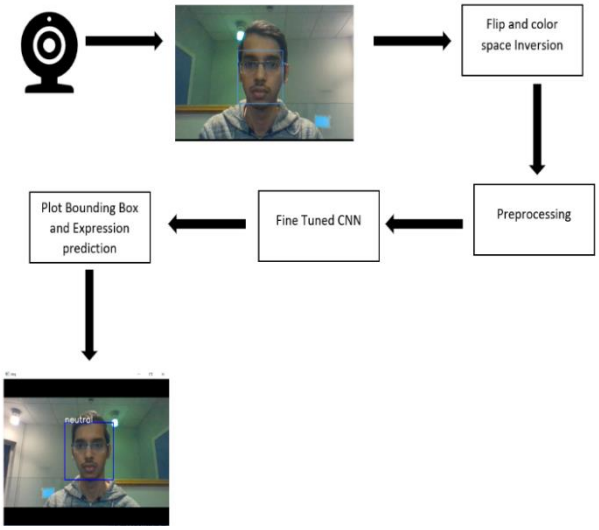


*Figure 9 Process of Real Time Expression Detection and Classification*

grayscale image and it can work on colored image by converting the image to grayscale during preprocessing and resizing it to the predefined values. Although the model was trained to recognize the facial expression the personal test images were with exaggerated expressions and may not generalize a subtle expression. Larger number of dataset would possibly train a more robust model.

Future scope would be to create a model which could differentiate between subtle facial expression change and to display the model prediction percentage in real time.

## REFERENCES


[1] https://en.wikipedia.org/wiki/Convolutional_neural_network

[2] https://realpython.com/face-recognition-with-python/

[3] https://github.com/fchollet/deep-learning-with-python-notebooks

[4] https://www.youtube.com/watch?v=_fRp408faul

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

[6] https://towardsdatascience.com/tagged/opencv