INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 3)
Available online at: www.ijariit.com

Real-time facial expression recognition using CNN

Revanth Krishna
revnth@outlook.com
SRM Institute of Science and Technology, Chennai, Tamil Nadu

ABSTRACT

Enhancing modern day machines or computers to recognize various facial expressions and to understand human emotions from them in real time is an exigent research subject. Through this paper, I put forward a solution to recognize emotions by understanding different facial expressions by collecting live video through a Flask App created. I deploy a Flask App to video stream live feed captured through the local camera attached to the machine or computer system. The video captured is fed to various image extraction techniques. The facial features are identified by different operations provided by OpenCV and the region consisting of parts of the face are made to surround or enclose by a contour. This region, enclosed by the contour is used as an input to Convolutional Neural Network (CNN). The CNN model created consists of six activation layers, of which four are convolution layers and two are fully controlled layers. Each layer is designed to undergo several training techniques. The main objective of this project is to demonstrate the accuracy of Convolutional Neural Network model designed. The paper is concluded by discussing the outcomes of our project and the ways to improve the efficiency of the model. The scope of this project is also analyzed to enhance technologies developed in the near future.

Keywords— *Flask App, Convolutional Neural Network (CNN), Image extraction techniques, Training methods*

1. INTRODUCTION

According to reports from various web sources, the most widely used mode of communication used by humans is Facial expression. Facial expression recognition is an evolving technology in the field of human-computer interaction. Facial expression recognition has its branches spread across various applications such as virtual reality, webinar technologies, online surveys and many other fields.

Even though high advancements have been witnessed in this field, there are several diplomacies that exist. The traditional feature extraction methods have slower response and lack in performance. The traditional methods have high latency or delay in their response. Through these traditional methods, it is extremely difficult to extract the required features effectively and hence, is too hazardous to utilize for real time applications.

In order to provide a panacea to such issues, a facial recognition method is proposed using CNN. This model thus, can be used to solve the above stated problems or difficulties. The development of the facial expression recognition is done in Keras. A six layered CNN is developed to build and train the model. Each layer is defined with certain training techniques to enable faster and efficient feature extraction. The trained model is deployed to a web interface using Flask app. The developed model is applied for real time videos and images and its accuracy is analyzed.

2. OBJECTIVES

The objectives of the project are as follows:

- To explore the dataset FER-2013.
- To generate training and validation batches.
- To create a Convolutional Neural Network (CNN) model.
- To train and evaluate model
- To represent the model as JSON string
- To create a Flask app to serve predictions
- To design an HTML template for the Flask app
- Use the model to recognize facial expressions in real time and analyze its accuracy.

3. DATASET AND ITS FEATURES

In this project, the dataset used to train the models is FER-2013. The FER-2013 dataset consists of 35887 images, of which 28709 labelled images belong to the training set and the remaining 7178 images belong to the test set. The images in FER-2013 dataset is labeled as one of the seven universal emotions: Happy, Sad, Angry, Fear, Surprise, Disgust, and Neutral. Among these emotion classifications, the most images belong to 'happy' emotions account to 7215 images of the 28709 training images. The number of images that belong to each emotion is given by returning the length of each directory using the OS module in Python. The images in FER-2013 dataset are in grayscale and is of dimensions 48x48 pixels. This dataset was created by gathering results from Google Image search of each emotions. The number of images of each emotion is given in table 1. The number of images of each emotion type is returned by the functions of 'OS' module in python. To explore the dataset further, and to understand what kind of images lie in the dataset, we plot few example images from the dataset using the 'utils' module in python. The resultant plot of example images obtained is given in figure 1. From various research papers, we have studied that the average attainable accuracy of a training model developed using FER-2013 dataset is 66.7% and our aim is also to design a CNN model with similar or a better accuracy.

Table 1: Number of images of each emotion type

S.No.	Type of Emotion	No. of images in the dataset
1.	Angry	3995
2.	Disgust	436
3.	Fear	4097
4.	Happy	7215
5.	Neutral	4965
6.	Sad	4830
7.	Surprise	3171

Fig. 1: Plot of Example images

4. TRAINING AND VALIDATION

During training, to minimize the losses of Neural Network, an algorithm called Mini-Batch Gradient Descent has been used. Mini-Batch Descent is a type of Gradient Descent algorithm used for finding the weights or co-efficient of artificial neural networks by splitting the training dataset into small batches. This algorithm provides more efficiency whilst training data.

4.1 Generating Training and Validation Batches

To build the training model, the image size and batch size are initialized. The image size is set as per the sizes of the images of the dataset. The batch size is set a value according to the memory requirements of the CPU or GPU hardware, so as to speed up the training process. The ImageDataGenerator() class of Keras library in python is used to accept values or images to train. The images in the dataset are flipped horizontally, as the real time images obtained through camera will be a mirrored image. These images are then used to generate a training set. The parameters such as target size, color mode, batch size, class mode and shuffle are given as per the requirements. Similarly, test set is generated with same parametric values as that of training set.

4.2 CNN Model

The CNN designed is based on sequential model and is designed to have six activation layers, of which 4 are convolutional layers and the remaining 2 are fully controlled layers.

The 4 convolutional layers consists of similar training techniques such as Batch Normalization, ReLu Activation function, Maxpooling and Dropout. Batch Normalization technique is used as it trains networks faster, allows higher learning rates, simplifies deeper networks, and also makes weights easier to initialize. The ReLu activation function is used to increase non-linearity in the images and also makes evaluation quicker. Maxpooling is done to reduce the dimensionality of an image, i.e. reduce its height and width. Dropout function is added to the layer to prevent overfitting of the training data. The fourth convolutional layer has an additional training technique called Flatten. The Flatten function converts the image into a 1-Dimensional array. This then, outputs the 1-Dimensional array as input for the Fully Controlled layers.

The two fully controlled layers are designed with training techniques like Dense, Batch Normalization, ReLu activation function and Dropout. The Dense function is used to connect each neuron of the previous layer to each neuron of the next layer. The remaining functions have similar applications. The output layer has two training techniques, Dense and softmax. The softmax function outputs a vector that returns the probability distributions of a list of potential outcomes.

The volume of each successive layer becomes half of its previous layer and the number of channels doubles for each successive layer. The below given figure 2 represents the structure of our CNN model.

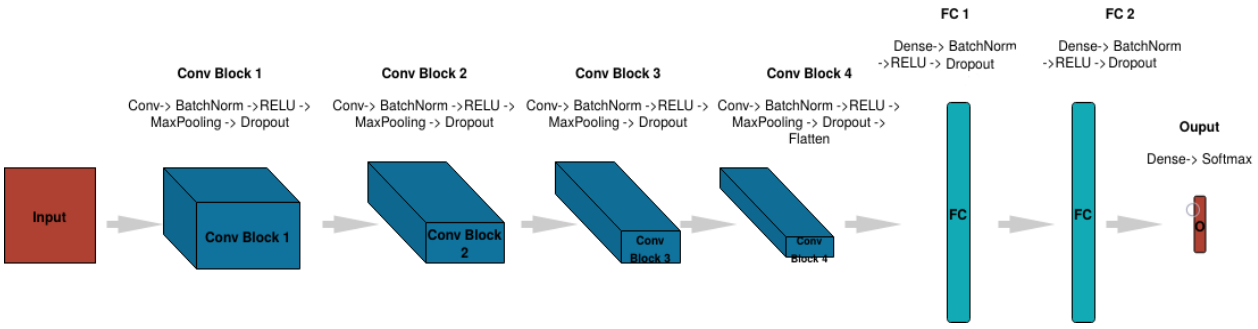

Fig. 2: CNN Architecture

Table 2: CNN model- Layer (types) and number of parameters

Layer (type)	Output Shape	Number of Parameters
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_1	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_2	(None, 12, 12, 512)	2048
activation_2 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_3	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_4	(None, 256)	1024

activation_4 (Activation)	**(None, 256)**	**0**
dropout_4 (Dropout)	**(None, 256)**	**0**
dense_1 (Dense)	**(None, 512)**	**131584**
batch_normalization_5	**(None, 512)**	**2048**
activation_5 (Activation)	**(None, 512)**	**0**
dropout_5 (Dropout)	**(None, 512)**	**0**
dense_2 (Dense)	**(None, 7)**	**3591**
Total params: 4,478,727 **Trainable params: 4,474,759** **Non-trainable params: 3,968**		

Adam optimization algorithm is used to compute the specified learning rates for the different parameters. The learning rate is taken as 0.0005, as it speeds up the training. The compile function of the module class of python is used to compile the resultant model into a training model. The compile function requires parameters such as optimizer, loss and metrics. As the interest and the agenda is to study the accuracy of the model, the metrics is set to accuracy. The loss is set to categorical_crossentropy as the data has to be classified into only the 7 defined categories and each image can belong to one classification type only. The summary of the compiled training model is given in Table 2.

The epochs value is set based on the learning rate and batch size, to make training faster. Epochs is the measure of the number of times all of the training data is used to update the weights. Similarly, parameters such as steps per epoch, validation steps, validation data and callbacks are considered to fit the training data. The ModelCheckpoint() function is used to set the file name and format, where the weights will be stored. The ReduceLROnPlateau() function is used to reduce the learning rate if no improvement is seen while monitoring a quantity. The Callbacks parameter is assigned to a list of PlotLossesTensorFlowKeras(), ModelCheckpoint(), and ReduceLROnPlateau().
PlotLossesTensorFlowKeras() is used to plot the log-losses and accuracy plot for the model.
The model.fit() function is used to fit the training data.

4.3 Represent model as JSON string.

The output of the CNN model is made to store as JSON string. A JSON or JavaScript Object Notation is used as it stores the data and allows faster exchange of data. The model.to_json() function of python is used to write the output of the trained model into JSON.

5. IMPLEMENTATION AND TESTING

5.1 Flask App and HTML template

An HTML file is created and designed, such that the body is coded to return the operations of the flask app. Therefore, an HTML template is created for the Flask app with a window of defined height and width that reads and runs the actions intended by the flask app.

Flask is a framework and also a python class datatype that is used to create instances of web application. Flask library is used to create an application that returns the webcam video feed to the HTML template by using the render.template() function. The Flask app is designed to be hosted on the computer's local IP Address (https:/localhost:5000). The structure of the flask app is shown in below given figure.

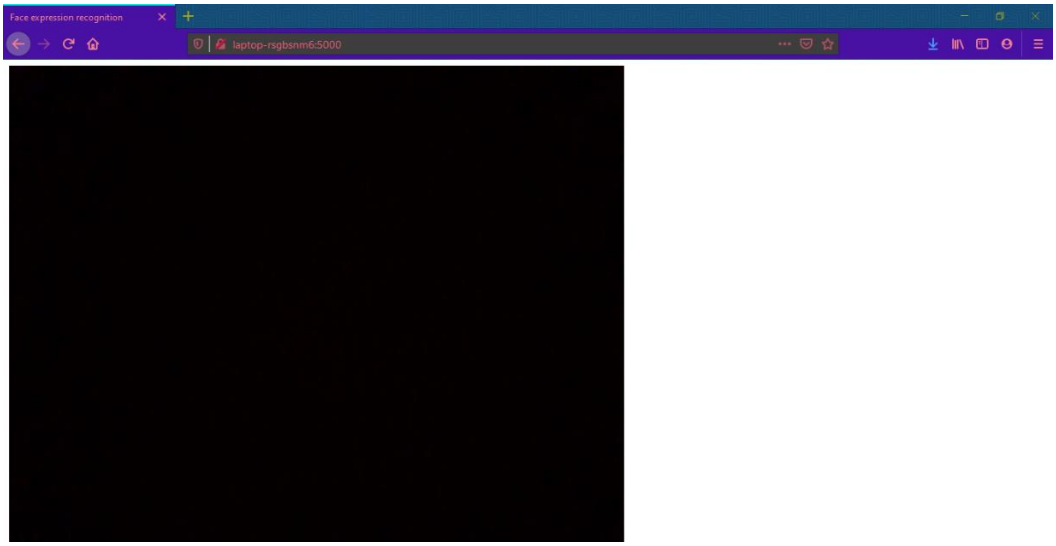

Fig. 3: An Outlook of the Flask App

5.2 Real-time Classification

In this project, OpenCV's Haar cascade is used to for the detection and extraction of the region containing the face from the video feed of webcam through the flask app. The video is converted to grayscale and the detected face is contoured or enclosed within a region defined to surround the face. This region of interest is resized to the size of the image and compared. The emotion is predicted and on top of face above contour rectangle.

5.3 Creating a Class to Output Model Predictions

A class is created to Output the predictions of the model. This class is made to connect the predictions of the model to the Flask App and is displayed on top of the contour rectangle surrounding the face region.

6. RESULTS

The CNN model designed is set to undergo 15 epochs. When trained gives an accuracy of 62.7% after the 15th epoch and the maximum efficiency achieved is also 62.7%. The plot of model accuracy over each epoch (from epoch_1 to epoch_15) is shown in the below given figure 4. The results regarding the accuracy of the training and validation set is shown in figure 5.

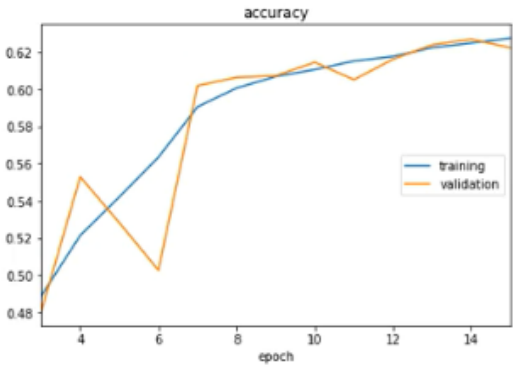

Fig. 4: Accuracy vs epoch no.

```
accuracy:
training   (min:    0.305, max:    0.627, cur:    0.627)
validation (min:    0.330, max:    0.627, cur:    0.622)
```

Fig. 5: Accuracy of the model

It has been observed that the CNN model developed can easily or flawlessly detect the following emotion types: Happy, Neutral, Sad and Surprise. The lighting conditions or the Illumination setup, play a major part in image detection and is a very important factor in the model's performance.

7. CONCLUSION AND FUTURE SCOPE

Using the FER-2013 dataset, a test accuracy of 62.7% is attained with this designed CNN model. The achieved results are satisfactory as the average accuracies on the FER-2013 dataset is 65% +/- 5% and therefore, this CNN model is nearly accurate.
For an improvement in this project and its outcomes, it is recommended to add new parameters wherever useful in the CNN model and removing unwanted and not-so useful parameters. Adjusting the learning rate and adapting with the location might help in improving the model. Accommodating the system to adapt to a low graded illumination setup and nullify noises in the image can also add onto the efforts to develop the CNN model. Increasing the layers in the CNN model might not deviate from the achieved accuracy, but the number of epochs can be set to higher number, to attain a higher accurate output. Though, increasing the number of epochs to a certain limit, will increase the accuracy, but increasing the number of epochs to a higher value will result in over-fitting.

The similar CNN model can be trained and tested for other available datasets and checked for its accuracy.

CODE: https://github.com/revanthkris/Facial-Expression-Recognition

8. REFERENCES

[1] Facial Expression Recognition with Faster R-CNN (Jiaxing Lia, Dexiang Zhanga, Jingjing Zhanga, Jun Zhanga, Teng Lia, Yi Xiaa, Qing Yana, and Lina Xuna a. The School of Electrical Engineering and Automation Anhui University, Hefei 230601, China).
[2] Real Time Facial Expression Recognition in Video using Support Vector Machines (Philipp Michel Computer Laboratory University of Cambridge CB3 0FD, United Kingdom), (Rana El Kaliouby Computer Laboratory University of Cambridge, Cambridge CB3 0FD, United Kingdom).
[3] Real-time Emotion Recognition From Facial Expressions (Minh-An Quinn, Grant Sivesind, Guilherme Reis CS 229 Stanford University)
[4] Baseline CNN structure analysis for facial expression recognition (Minchul Shin, Munsang Kim and Dong-Soo Kwon)

BIOGRAPHY

Revanth Krishna
Student
Department of Mechatronics Engineering
SRM Institute of Science and Technology, Chennai, Tamil Nadu, India

