Journal of Positive School Psychology http://journalppw.com
2022, Vol. 6, No. 2, 505 – 509

Real Time Facial Emotion Recognition Using CNN.

Divya Jennifer Dsouza, Aanchal Sudhir, Akshatha Shetty, Dr Uday Kumar Reddy

Abstract —

Facial emotion is the key to communication, keeping this thought in mind we demonstrated "Facial Emotion Recognition Using CNN". This plays an important role in identifying human intentions for different situations. It is highly used in social media platforms and forensics. Convolutional Neural Network popularly known as CNN is a deep learning technique that has been used to build our model. Using the FER2013 dataset, we perform preprocessing of data which is then provided as an input to the CNN, so that the resulting prediction of emotions is accurate. Additionally, we showcase a mobile application that runs our FER model on-device in real-time.

INTRODUCTION

An individual's emotional state can be accurately conveyed with the help of emotions. Automation Emotion Recognition is the new research field that has been opened up with the goal to gain information about the human emotions. The Facial Emotion Expression can be divided into three major steps that is, preprocessing the dataset, training the model and testing the model on the real world images.

The Facial Expression like anger, fear, happiness, and sadness can be identified by the facial emotion detector. Facial emotion recognition has been the most important research topic where people have tried various methods to accurately identify human emotions. Several methods consist of Naïve Bayes and Maximum entropy that has been applied to detect human facial emotions in previous researches. In our model, we have used categorical approach also termed as discrete.

The use of Deep learning methods such as CNN has helped us to get the most accurate results as compared to the previous methods[2]. To determine the facial emotion with the maximum odds, we have used CNN which analyzes the test facial emotions that classifies the images into one of the four categories - happy, sad, anger, and fear. The android app "FaceBooth" acts as a facilitator that takes an image as an input and predicts the emotions based on the image.

Connecting the android app "FaceBooth" to python backend is done with the help of Flask which is a web application framework written in python. Firstly, the flask module is imported to the project which is then started with help of run() method.

RELATED WORK

Evangelos Sariyanidi et al. [6] explained the fundamentals of automatic facial effect by uncovering the advantages and limitations of comprehensive analysis of facial emotion. Furthermore, they have explained the various limitations faced while working on the facial emotion recognition.

Stefano Berretti et al. [7] explained 3D facial expression recognition using SIFT descriptors which uses a feature descriptors with 128 floating point numbers . The facial emotion recognition is extracted from the 3D shape of the face and is addressed using 3D geometry information. In order to train the model Support Vector Machine (SVM) was used and the average recognition rate was around 78.43%.

Y.V.Venkatesh et al. [8] worked on novel approach to classify facial expression from 3D mesh dataset using modified PCA. A shape feature matrix is computed for each expression by using a matrix algebraic operations on the shape of expressive and neutral faces and is then subjected to the proposed modified PCA approach which is then used to recognize expression.

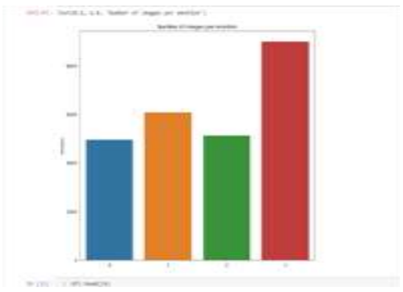
H.K.Meena et al. [9] worked on improving facial expression recognition using graph signal processing. The dimensionality of the feature vector of the facial expression is reduced using the GSP technique, based on the histogram of discrete wavelet transform and oriented gradients and then applied to a classifier.

METHODOLOGY

A. FER2013 DATASET

The FER2013 Dataset consists of seven emotions that is, happy , sad ,anger, fear, disgust , neutral and surprise. But considering the accuracy of the model, we have removed three emotions i.e, surprise, disgust and neutral.

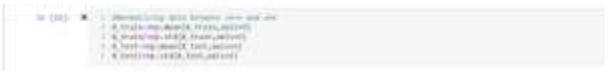
This dataset consists of three columns- emotion, pixels, usage and there are total of 38,887 rows. The emotion column consists of seven emotions numbered from 0 to 6. The pixels column consists of image which is mainly represented in the form of image pixels. The usage column tells us if the data can be used for testing or training.



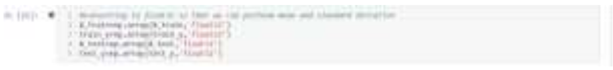
0 - Angry, 1- Sad , 2- Fear, 3-Happiness

B. Data Pre-processing

Data preprocessing is the process of refining the input dataset which is the CSV file containing the pixels of the image.This mainly includes resizing , reshaping and normalizing the data for better classification of emotions. The images are then converted into pandas data frame and numpy array.



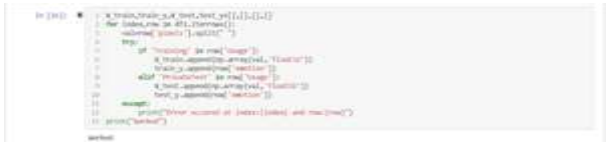
```
In [ ]:  #normalizing data between zero and one
         X_train-=np.mean(X_train,axis=0)
         X_train/=np.std(X_train,axis=0)
         X_test-=np.mean(X_test,axis=0)
         X_test/=np.std(X_test,axis=0)
```



```
In [ ]:  #converting to float32 so that we can perform mean and standard deviation
         X_train=np.array(X_train,'float32')
         train_y=np.array(train_y,'float32')
         X_test=np.array(X_test,'float32')
         test_y=np.array(test_y,'float32')
```

C. Splitting dataset

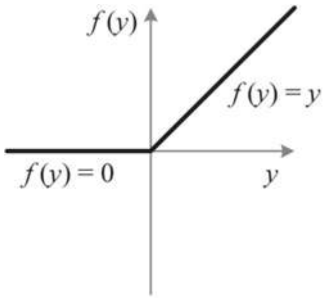
We split the data into two categories i.e training and testing. The training dataset is used to to train the model whereas the testing dataset is used to check if the model classifies the emotions accurately.



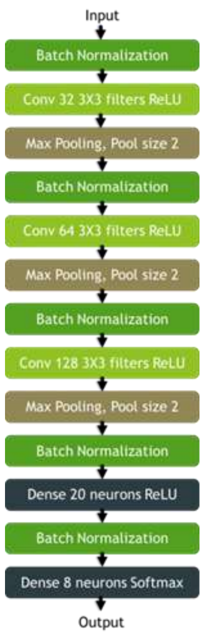
```
In [ ]:  X_train,train_y,X_test,test_y=[],[],[],[]
         for index,row in df1.iterrows():
             val=row['pixels'].split(" ")
             try:
                 if 'Training' in row['Usage']:
                     X_train.append(np.array(val,'float32'))
                     train_y.append(row['emotion'])
                 elif 'PublicTest' in row['Usage']:
                     X_test.append(np.array(val,'float32'))
                     test_y.append(row['emotion'])
             except:
                 print("Error occured at index:{index} and row:{row}")
         print("worked")

worked
```

D. Build the model using convolutional neural network (CNN)

It is a deep learning algorithm that consists of four layers-



- Convolution
- ReLU Layer
- Pooling
- Fully Connected



Convolution

The are three Convolutional Layers, the first layer has a total of 32 filters and is connected to the input.The second and the third layer consists of 64 and 128 filters respectively.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Batch Normalization

The Batch Normalization Technique is used for reducing the number of training epochs[3] and it also stabilizes the learning process. In batch normalization, a collected set of input data which is commonly known as batch is used for training a neural network.

ReLU

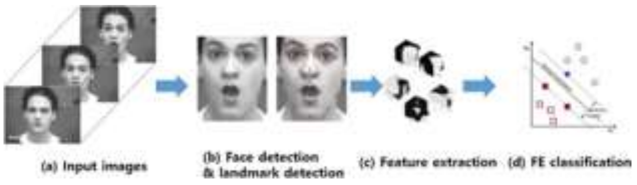
Rectified Linear Unit(ReLU) is a linear activation function which is easy to train and achieves better performance[4].It is the default activation function for many neural networks. It outputs zero, if the input is negative else it will output the input directly.

Pooling

After a nonlinearity i.e ReLU has been applied to the feature maps output, a new layer called pooling layer is added. Each feature map is operated separately by the pooling layer to create a new set of pooled feature maps. Selecting a pooling operation is a feature of pooling whose size is smaller than the size of feature map[5]. A stride of 2 pixels is always applied with 2x2 pixels.

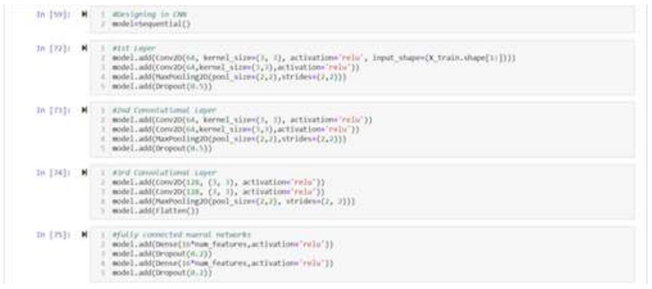
Fully Connected

The connection of neurons between two separate layers is done with the help of Fully Connected layer. Along with the neurons ,the layer also consists of biases and weights. The fully connected layer is placed before the output layer and it is one of the few last layer of the CNN. The classification of images is done by FC layer.



Dropout

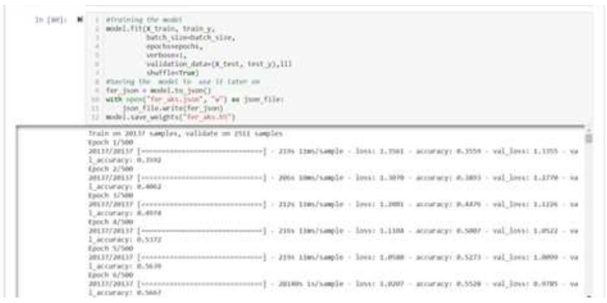
Due to the random dropping of neurons while training, the network becomes numb to the certain weights of neurons. Thus, Dropout is known as a regularization technique.



```
In [59]:  #Designing in CNN
          model=Sequential()

In [72]:  #1st layer
          model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1:])))
          model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
          model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
          model.add(Dropout(0.5))

In [73]:  #2nd Convolutional Layer
          model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
          model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
          model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
          model.add(Dropout(0.5))

In [74]:  #3rd Convolutional Layer
          model.add(Conv2D(128, (3, 3), activation='relu'))
          model.add(Conv2D(128, (3, 3), activation='relu'))
          model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
          model.add(Flatten())

In [75]:  #fully connected neural networks
          model.add(Dense(16*num_features,activation='relu'))
          model.add(Dropout(0.2))
          model.add(Dense(16*num_features,activation='relu'))
          model.add(Dropout(0.2))
```

E. *Training the model*

Training is a crucial step where it involves defining important parameters such as number of epochs, batch size and learning rate. As we train the model, the weights are updated simultaneously.



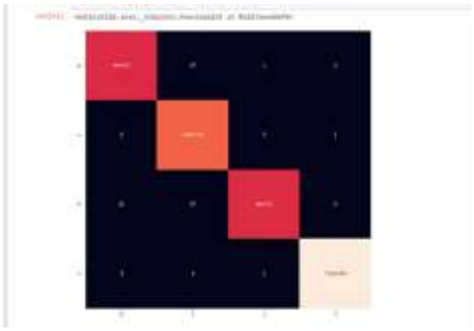
```
In [##]:
#training the model
model.fit(X_train, train_y,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_test, test_y),111
          shuffle=True)
#saving the  model to  use it later on
fer_json = model.to_json()
with open("fer_aks.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer_aks.h5")
```

```
Train on 20137 samples, validate on 2511 samples
Epoch 1/500
20137/20137 [==============================] - 219s 11ms/sample - loss: 1.3561 - accuracy: 0.3559 - val_loss: 1.1355 - va
l_accuracy: 0.3592
Epoch 2/500
20137/20137 [==============================] - 206s 10ms/sample - loss: 1.3070 - accuracy: 0.3893 - val_loss: 1.2770 - va
l_accuracy: 0.4062
Epoch 3/500
20137/20137 [==============================] - 212s 11ms/sample - loss: 1.2001 - accuracy: 0.4476 - val_loss: 1.1226 - va
l_accuracy: 0.4974
Epoch 4/500
20137/20137 [==============================] - 216s 11ms/sample - loss: 1.1108 - accuracy: 0.5007 - val_loss: 1.0522 - va
l_accuracy: 0.5372
Epoch 5/500
20137/20137 [==============================] - 219s 11ms/sample - loss: 1.0580 - accuracy: 0.5273 - val_loss: 1.0099 - va
l_accuracy: 0.5639
Epoch 6/500
20137/20137 [==============================] - 20140s 1s/sample - loss: 1.0207 - accuracy: 0.5528 - val_loss: 0.9785 - va
l_accuracy: 0.5667
```

EVALUATION

A. Test on Native Dataset

The model was tested with the help of private dataset samples from the FER2013 dataset and the model is found to be 70% accurate. The emotion which was predicted from the model is compared with the actual emotion from the dataset to draw a confusion matrix which is used to determine the overall accuracy of the classification.



Based on the above figure we can conclude that the model performs notably on the four original categories. The Fear emotion category did not perform accurately. Happy and Sad are the emotions which are predicted accurately.

D. Performance on Real-world Applications

If a system has the ability to solve real-world problems , then it can be classified as a well-built system.Keeping this in mind, our facial emotion detector was built to solve real-world problems. Here, we classify images in real- time with a help of a mobile app. The app which is supported with the help of the flask server at its backend will take a real-world image as input which is then uploaded to the flask server(backend) and the result is displayed.

The app has interactive user interface which provides an interesting interface. Our model performed well when it was tested using the computer webcam. Similar results were seen in images that were provided through the mobile app.

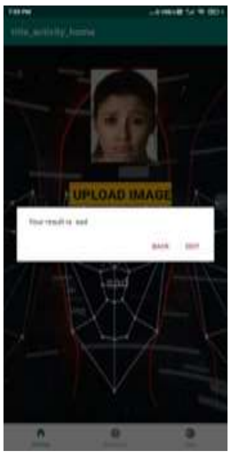


Fig 1. CNN predicting sadness

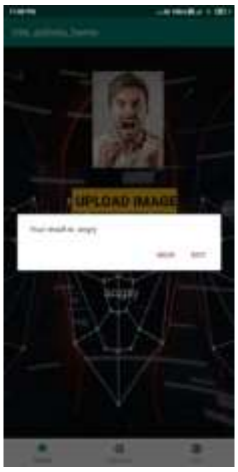


Fig 2. CNN predicting anger

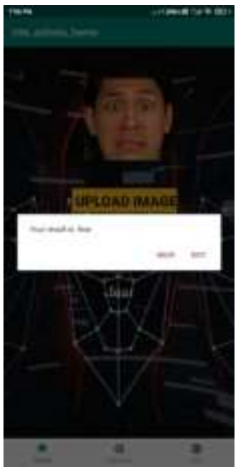


Fig 3. CNN predicting fear

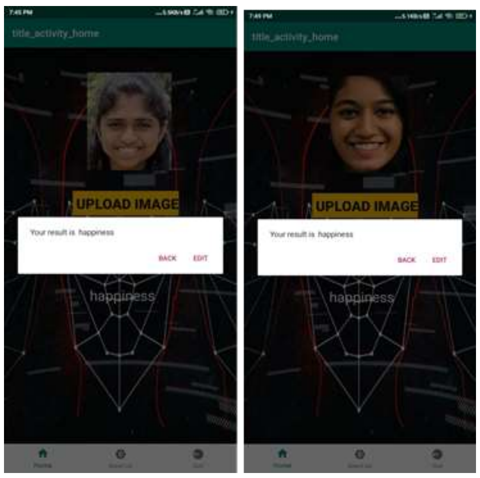


Fig 4 . CNN predicting happiness

Conclusion

In this project, CNN was built to recognize emotions and to classify them. The HAAR cascade classifier was mainly used to detect faces. In order to regularize the weights and improve stability, the concept of batch normalization was used. The trained CNN model was tested to classify images from the real time as well as native dataset. The performance on test data was analyzed. The mobile app called "FACEBOOTH" was built to classify images in realtime.

We tried our level best to improve the model and to increase the recognition rate by making it accurate to recognize the emotions even in the complex backgrounds. The results proved that our analysis was better compared to the past experimental analysis.

References:

[1] Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." *IEEE transactions on neural networks* 8.1 (1997): 98-113.

[2] Bjorck, Johan, et al. "Understanding batch normalization." *arXiv preprint arXiv:1806.02375* (2018).

[3] Lujia Chen ,(2016). "Deep Learning Models for modeling cellular transcription systems".Retrieved from http://d-scholarship.pitt.edu/30331/1/thesis12132016.pdf.

[4] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9,** 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9.

[5] E. Sariyanidi, H. Gunes and A. Cavallaro, "Automatic Analysis of Facial Affect: A Survey of Registration, Representation, and Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 6, pp. 1113-1133, 1 June 2015, doi: 10.1109/ TPAMI.2014.2366127.

[6] Berretti, S., Ben Amor, B., Daoudi, M. *et al.* 3D facial expression recognition using SIFT descriptors of automatically detected keypoints. *Vis Comput* **27,** 1021 (2011). https://doi.org/10.1007/s00371-011-0611-x.

[7] Venkatesh, Y. V., Ashraf A. Kassim, and OV Ramana Murthy. "A novel approach to classification of facial expressions from 3D-mesh datasets using modified PCA." *Pattern Recognition Letters* 30.12 (2009):1128-1137.

[8] Meena, H. K., K. K. Sharma, and Shiv Dutt Joshi. "Improved facial expression recognition using graph signal processing." *Electronics Letters* 53.11 (2017): 718-720.