Project Report

Facial Expression Recognition (FER) – Using CNN

Real Time Emotion Recognition

About FER:

In this project I built a CNN which is capable of classifying seven facial emotions (Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral) with high degree of accuracy. Kaggle "Facial Expression Recognition (FER) Challenge" dataset with seven facial expression labels is used in this project. The CNN for detecting facial emotions was built using keras.

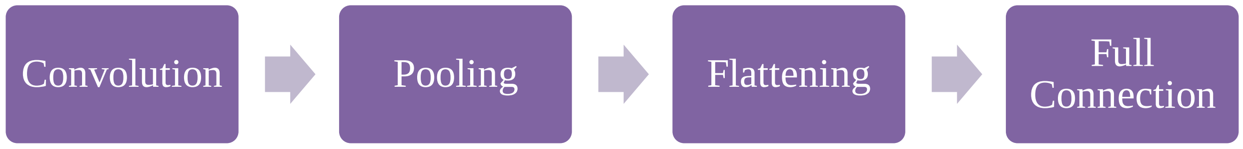
What is CNN?

Convolution Neural Network (CNN) is a category of deep learning specializes in image processing. In a CNN each neuron is affected only by a small region of the input from the previous layer. This means that the special features of the images are retained across the layers of CNN and can be used for classification. Since CNN has less number of parameters it is faster to train and less prone to over-fitting.

Why CNN for FER:

In facial expression images are in high-dimensional objects. Facial features over high-dimensional spaces can be counter-intuitive. The convolution neural network extracts larger features (dimensionality reduction) in a hierarchical set of layers. The convolution network takes large time in training and little time in practice.

Phases of CNN:



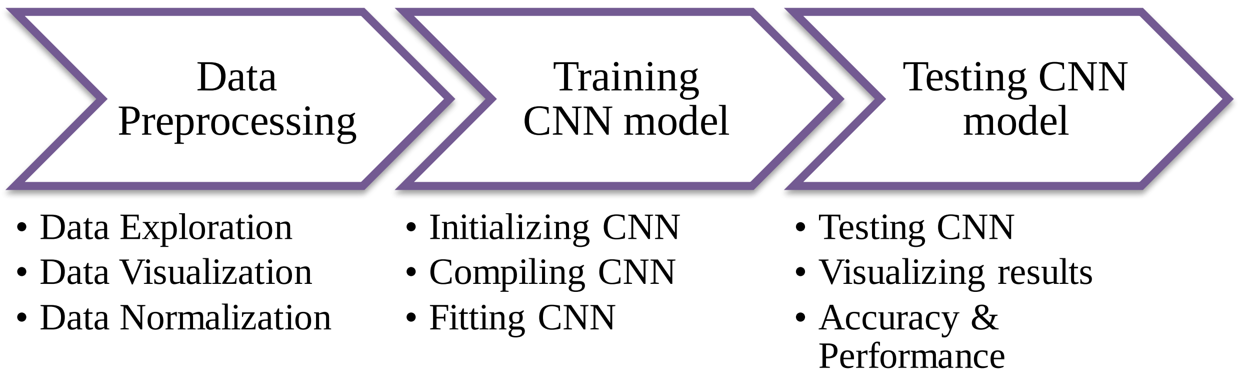
- **Convolution:** The primary purpose is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

  Input Image * Feature Detector = Feature Map

  A filter (feature detector) convolves with the input image to produce a feature map. A CNN learns the values of these filters on its own during the training process. The more number of filters, the more image features get extracted and the better network becomes at recognizing patterns in unseen images.

- **Pooling:** Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Pooling can be of different types: Max, Average, Sum etc. It also prevents over fitting.
- **Flattening:** Flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer.
- **Full Connection:** After flattening, the flattened feature map is passed through a neural network. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

My Project Flow:



Data Preprocessing:

The data preprocessing phase comprises of data exploration, visualization, and normalization. Initially I started preprocessing and gained knowledge from the data using various visualization techniques and EDA methods. From that I infer that the data is clean and have no missing values.

Then I divided the dataset into three parts: Training set, Public test set and Private test set. Training and Public test file is used during the training phase of CNN model. The unseen private test set is used to test our CNN model. The private test set is used to know how good our model is predicting correct results.

Most of the machine learning model requires normalized data as input. So I normalized the training and public test data using standardization technique. After that I reshaped the data to use it for model training.

The target variable (emotions) is already in categorical nature. Just we need to apply one hot encoding on that variable for better performance of the model.

Training CNN model:

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer.

First we have to initialize the cnn model. The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. Normalized gray scale images of size 48 X 48 pixels from Kaggle dataset are used for training, validation and testing.

Convolution and pooling is done based on batch processing. After each convolution layer downsampling / subsampling is done for dimensionality reduction. This process is called Pooling. In this project max pooling is done after convolution. Pool size of (2x2) is taken, which splits the image into grid of blocks each of size 2x2 and takes maximum of 4 pixels. After pooling only height and width are affected. Then I have added dropout layer to avoid over fitting.

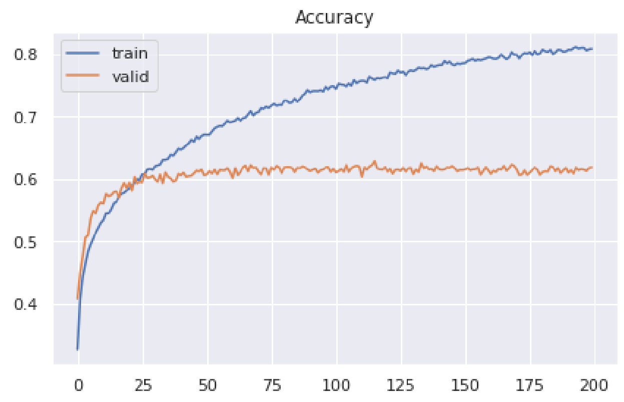
Three convolution layer and pooling layer are used in the architecture. At first convolution layer size of input image is 48x48. After first convolution, pooling is done with pool size of 2x2, which results image of size 23x23. Similarly second and third convolution & pooling layer added.

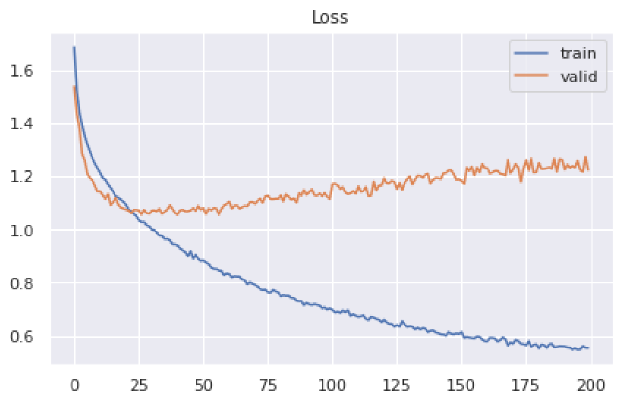
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)             (None, 46, 46, 64)        640
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)    (None, 23, 23, 64)        0
_________________________________________________________________
dropout_1 (Dropout)           (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)             (None, 21, 21, 64)        36928
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)    (None, 10, 10, 64)        0
_________________________________________________________________
dropout_2 (Dropout)           (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)             (None, 8, 8, 128)         73856
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)    (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)           (None, 4, 4, 128)         0
```

Fully connected layer is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transforms features through layers connected with trainable weights. The weights of these layers are trained by forward propagation of training data then backward propagation of its errors.

Output from the fully connected layer is connected to output layer having seven distinct classes. Using softmax activation function, output is obtained using the probabilities for each of the seven classes. The class with the highest probability is the predicted class.

During training phase after completion of 200 epochs I got training accuracy of 81.32% and validation accuracy of 61.83%.





Testing CNN model:

To test our model, I have used private test set. Since private dataset is not preprocessed we have to preprocess before predicting new results.

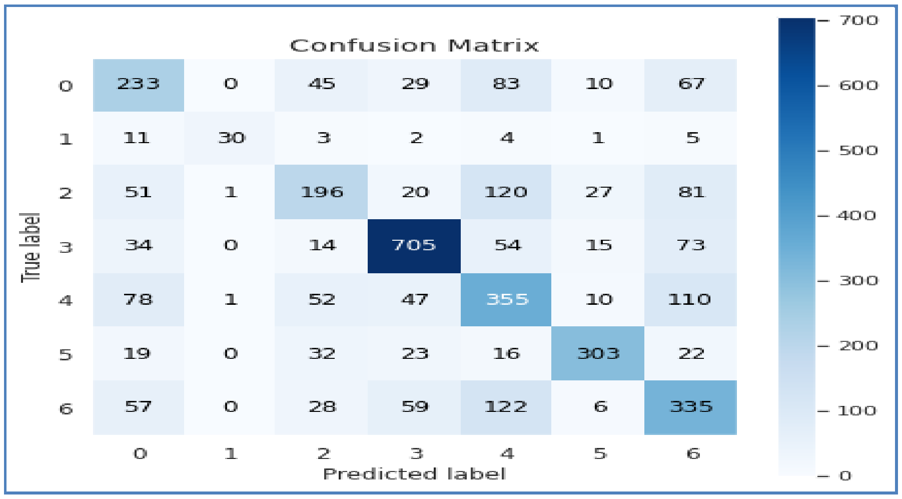
Classification report for our test result,

```
              precision    recall  f1-score   support

       Anger       0.48      0.50      0.49       467
     Disgust       0.94      0.54      0.68        56
        Fear       0.53      0.40      0.45       496
       Happy       0.80      0.79      0.79       895
         Sad       0.47      0.54      0.50       653
    Surprise       0.81      0.73      0.77       415
     Neutral       0.48      0.55      0.52       607
```

From the classification report we can infer that, the model performs really well on classifying positive emotions resulting in relatively high precision scores for happy and surprised. Model performance seems weaker across negative emotions on average. In particularly, the emotion sad has a low precision of only 0.47 and recall 0.54. The model frequently misclassified angry, fear and neutral as sad. The overall F1-score is also 0.60. Happy and surprise have higher F1-score as 0.79 and 0.77 respectively. Fear has least F1-score as 0.45 and sad, anger and neutral also have low F1-score.

Confusion matrix for our test result,



	0	1	2	3	4	5	6
0	233	0	45	29	83	10	67
1	11	30	3	2	4	1	5
2	51	1	196	20	120	27	81
3	34	0	14	705	54	15	73
4	78	1	52	47	355	10	110
5	19	0	32	23	16	303	22
6	57	0	28	59	122	6	335

How to improve model performance:

The dataset has class imbalance issue, since some classes have large number of examples while some has few. The dataset can be balanced using oversampling, by increasing numbers in minority classes to get higher accuracy.

And also by increasing the data or increasing the number of epochs we can make the model to perform better.

Conclusion:

CNN architecture for facial expression recognition as mentioned above was implemented in Python. Along with Python programming language, Numpy, Tensorflow, Keras and various visualization libraries were used.