Facial Emotion Recognition Using Deep Convolutional Neural Network

Deep Learning Model to identify emotion of people based on facial expression

Group Members:
11815050: Santosh Mahato
11805178: Alok Kumar Tiwari
11804312: Shivam Yadav

Lovely Professional University
School of Computer Science and Engineering
Advance Machine Learning

Abstract

In this project, we made a facial expression recognition using CNN (Convolutional Neural Network), one of the deep learning technologies. The proposed structure has general classification performance for any environment or subject. For this purpose, we collect a variety of databases and organize the database into six expression classes such as ‘expressionless’, ‘happy’, ‘sad’, ‘angry’, ‘surprised’ and ‘disgusted’.

Pre-processing and data augmentation techniques are applied to improve training efficiency and classification performance. In the existing CNN structure, the optimal structure that best expresses the features of six facial expressions is found by adjusting the number of feature maps of the convolutional layer and the number of nodes of fully-connected layer.

The experimental results show good classification performance compared to the state-of-the-arts in experiments of the cross validation and the cross database.

Table of Content

INTRODUCTION

Emotion is triggered by specific situations, and the recognition of human emotion is a crucial topic in the study of human-computer interfaces (HCIs) to empathize with people. When a machine communicates with people, emotion detection can give people more affinities and help to provide personalized service to people depending on their moods, which inspires confidence in people.

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face. In order to recognize facial expressions based on CNN, a much amount of well-separated training database is needed.

Deep learning can overcome the technical limitations of existing machine learning where the performance is drastically deteriorated for complex problems by using deep neural networks to extract high-level features appropriate to the given data. Convolutional Neural Networks (CNN) are developed to imitate human visual cognition processes in deep learning technology and has been widely applied to the field of image recognition and shows high performance.

The basic CNN structure consists of convolutional layers and fully-connected layers. By passing through multiple convolutional layers sequentially, it is found to extract high-level features. With the extracted high-level features, the final classification result is determined in the Fully- connected layer. The 7 expressions to be classified are ‘neutral’, ‘happy’, ‘sad’, ‘angry’, ‘fear’, ‘surprised’ and 'disgusted'.

Dataset Used

First, a database containing a large number of facial images is required for recognition of facial expressions with high accuracy. The images of the database should consist of facial images representing emotions. The database used in the 'Facial Expression Recognition Challenge' held in Kaggle in 2013(FER2013) consists of 37,000 facial images of seven facial expressions.

Dataset	Fear	Surprise	Angry	Sad	Happy	Disgust	neutral
Train	4097	3171	3995	4830	7215	436	4965
Test	1024	831	958	1247	1774	111	1233

Fig: Dataset distribution among various emotions

Fig: Images taken from Kaggle FER (2013) Dataset

Proposed Architecture

We have use convolutional neural network to train our model for facial emotion recognition as it is most commonly applied to analyze visual imagery. And also, we have used 'adam' optimizer with learning rate of 0.01% to compile this model. We have also used 'categorical_crossentropy' function as loss function.

CNN (Convolutional Neural Network)

The CNN framework is widely used for learning features and classification in various fields. It consists of a feature extraction part and a classification part. The feature extraction section is composed of successive convolution layers and pooling layers with nonlinear functions.

The first layer in a CNN network is the **CONVOLUTIONAL LAYER**, which is the core building block and does most of the computational heavy lifting. Data or imaged is convolved using filters or kernels. Filters are small units that we apply across the data through a sliding window. The depth of the image is the same as the input, for a color image that RGB value of depth is 4, a filter of depth 4 would also be applied to it. This process involves taking the element-wise product of filters in the image and then summing those specific values for every sliding action. The output of a convolution that has a 3d filter with color would be a 2d matrix.

Second is the **ACTIVATION LAYER** which applies the ReLu (Rectified Linear Unit), in this step we apply the rectifier function to increase non-linearity in the CNN. Images are made of different objects that are not linear to each other.
Third, is the POOLING LAYER, which involves downsampling of features. It is applied through every layer in the 3d volume. Typically, there are hyperparameters within this layer:

- **The dimension of spatial extent**: which is the value of n which we can take N cross and feature representation and map to a single value
- **Stride**: which is how many features the sliding window skips along the width and height

Lastly, is the **FULLY CONNECTED LAYER**, which involves Flattening. This involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the output.

My Model:

The first convolutional layer filters (48×48) size input image with 32 kernels of size 3×3 with ‘same’ padding and ‘relu’ activation function. The second convolutional layer takes as input the output of the first convolutional layer and filters it with 64 kernels of size 3× 3 with ‘same’ padding and ‘relu’ activation function. Then we have added Batch-Normalization to the model to normalize the inputs. Then we have added MaxPooling2D with ‘pool_size’ (2,2) to reduce the number of parameters and computation in the network. Then we have added Dropout with rate 25% to prevent a model from overfitting by randomly setting the outgoing edges of hidden units.

The third and fourth convolutional layers as well as first and second convolutional layers are connected to one another without any intervening pooling or normalizing layers. The third convolutional layer has 128 kernels of size 3×3 connected to the outputs of the second convolutional layer. The fourth convolutional layer has 256 kernels of size 3×3. Then Again, we are adding ‘Batch-Normalization’ to the model. Then again, we are adding MaxPooling2D with pool_size of (2,2) and also adding Dropout with rate 25%.

Then Finally, we are adding flatten to our model to convert the data into a 1-dimensional array. Then we are adding Dense layer with dimensionality of output variable as 1024 with activation function ‘relu’ to feed all outputs from the previous layer to all its neurons. Then Again, we are adding Dropout with rate 50%. Then Finally, we are adding Dense layer with output variable as number of emotions i.e., 7 with activation function as ‘softmax’.

To prevent over-fitting, the dropout technique is applied to the first two fully-connected layers. In the facial expression recognition, the number of channels in the convolutional layer and the number of nodes in the fully-connected layer is reduced in order to select an optimal structure with superior classification performance, less execution time and less training parameters.

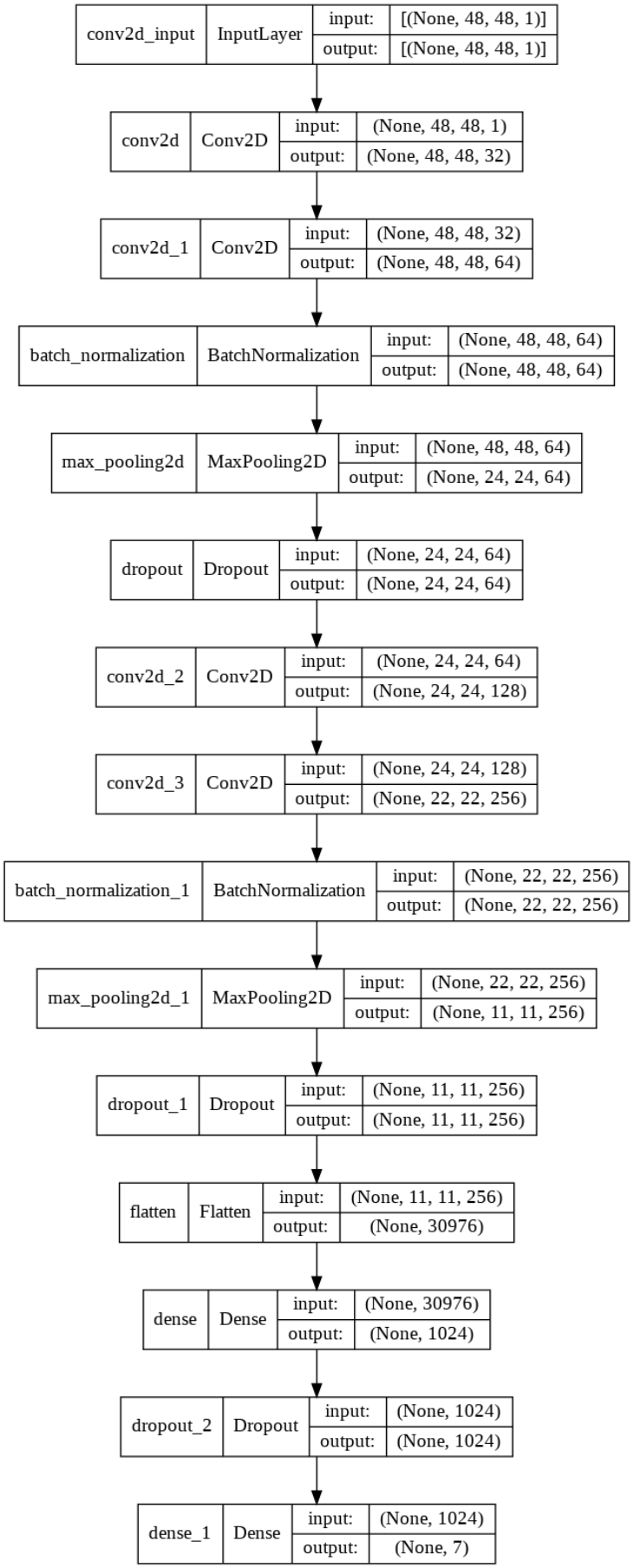

Fig: My CNN Model Flowchart

Results and Experimental analysis

Training Result final:
Batch trained: 449/449
Average train time: 26s
Execution time: 57ms/step
loss: 0.4634
accuracy: 0.8775

Testing Result final:
Batch tested: 113/113
Average train time: 4s
Execution time: 31ms/step
loss: 1.1205
accuracy: 0.6640

Final train accuracy = 87.75%,
validation accuracy = 66.40%

Classification Report:

For Training Dataset:

	Precision	Recall	F1-Score	Support
Angry	0.14	0.14	0.14	3995
Disgust	0.02	0.01	0.01	436
Fear	0.14	0.13	0.13	4097
Happy	0.25	0.25	0.25	7215
Neutral	0.18	0.19	0.18	4965
Sad	0.17	0.17	0.17	4830
Surprise	0.11	0.11	0.11	3171
Accuracy			0.17	28709
Macro avg	0.14	0.14	0.14	28709
Weighted avg	0.17	0.17	0.17	28709

For Testing Dataset:

	Precision	Recall	F1-Score	Support
Angry	0.13	0.13	0.13	958
Disgust	0.03	0.02	0.02	111
Fear	0.14	0.11	0.13	1024
Happy	0.25	0.25	0.25	1774
Neutral	0.17	0.20	0.19	1233
Sad	0.19	0.18	0.19	1247
Surprise	0.10	0.10	0.10	831
Accuracy			0.17	7178
Macro avg	0.14	0.14	0.14	7178
Weighted avg	0.17	0.17	0.17	7178

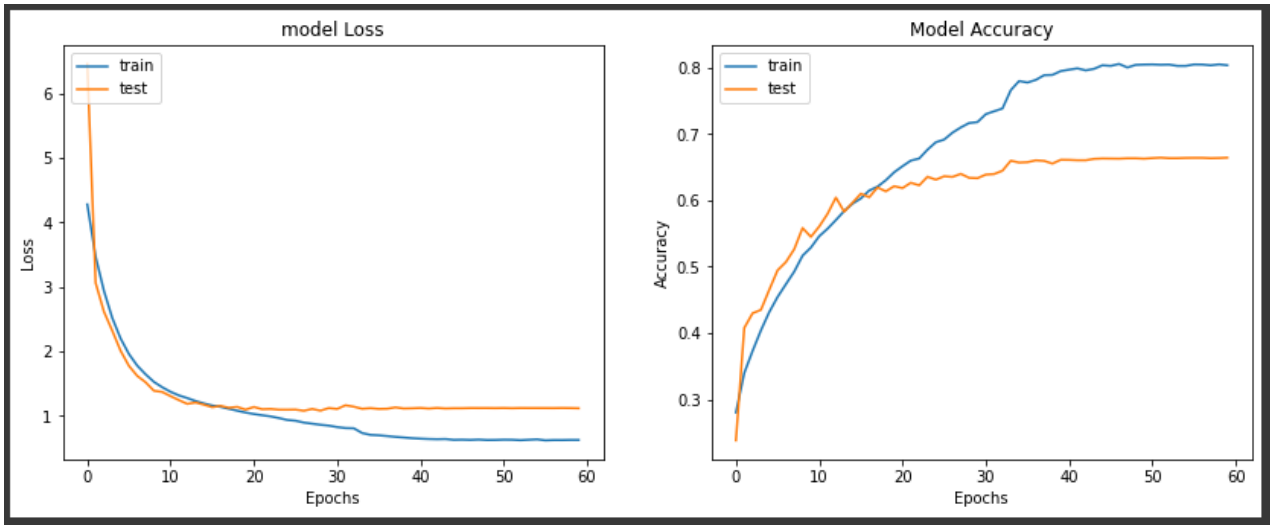

Fig: Accuracy graph for training and testing dataset

All output Screenshots

Color Bar Representation:

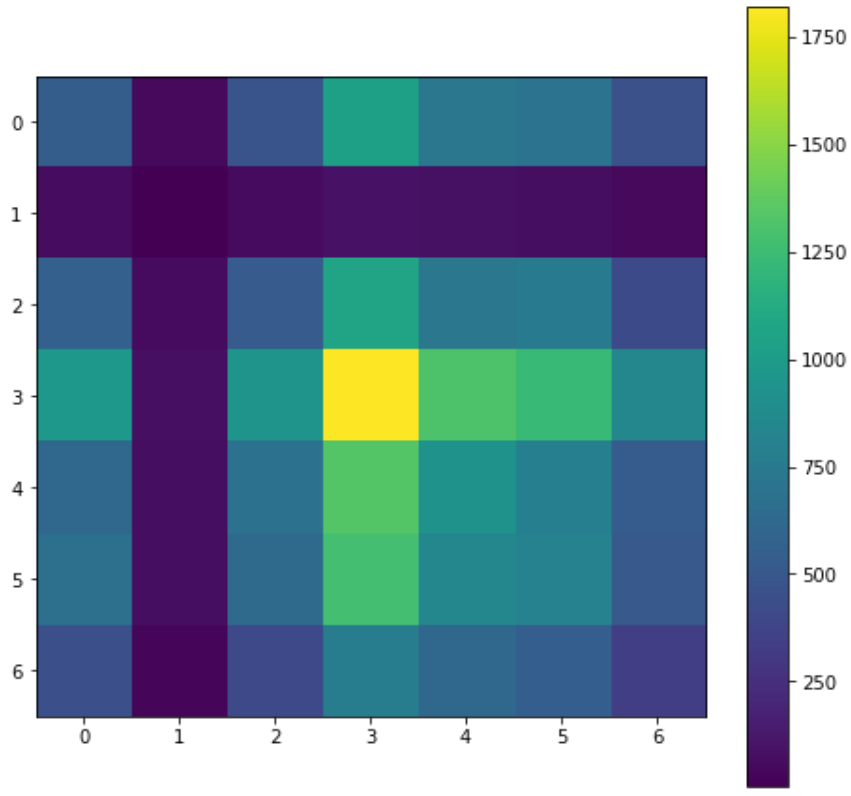

Fig: Color-bar representing Training Dataset

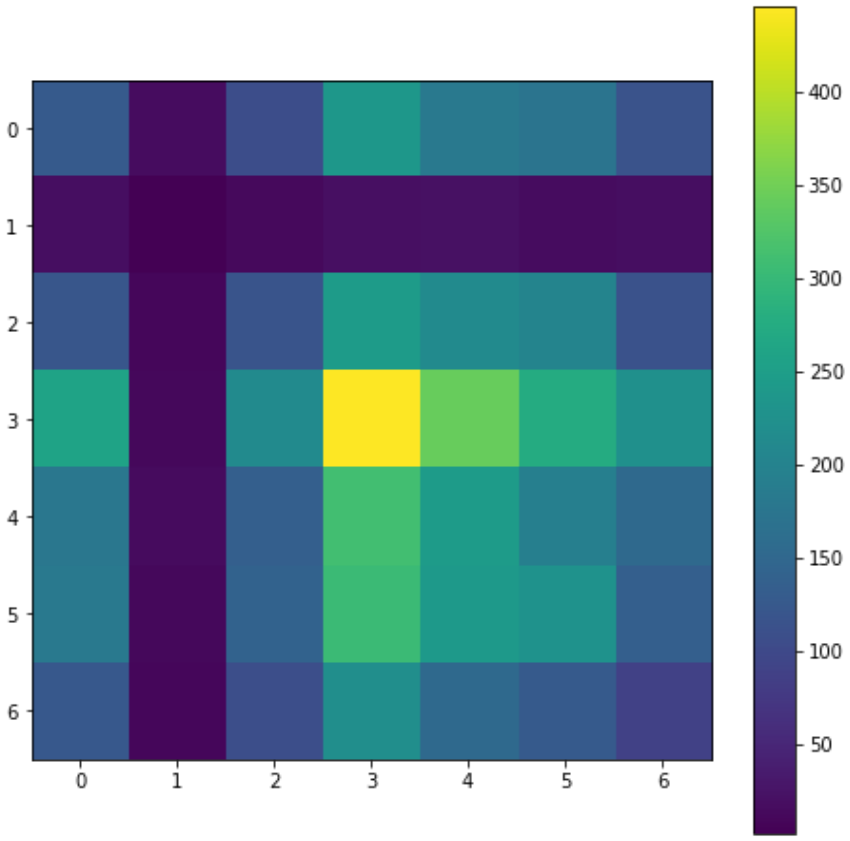

Fig: Color-bar representing Testing Dataset

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 conv2d_1 (Conv2D)           (None, 48, 48, 64)        18496

 batch_normalization (BatchN  (None, 48, 48, 64)       256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0
 )

 dropout (Dropout)           (None, 24, 24, 64)        0

 conv2d_2 (Conv2D)           (None, 24, 24, 128)       73856

 conv2d_3 (Conv2D)           (None, 22, 22, 256)       295168

 batch_normalization_1 (Batc  (None, 22, 22, 256)      1024
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 256)      0
 2D)

 dropout_1 (Dropout)         (None, 11, 11, 256)       0

 flatten (Flatten)           (None, 30976)             0

 dense (Dense)               (None, 1024)              31720448

 dropout_2 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 7)                 7175

=================================================================
Total params: 32,116,743
Trainable params: 32,116,103
Non-trainable params: 640
```

Fig: Layer-wise Model Summary

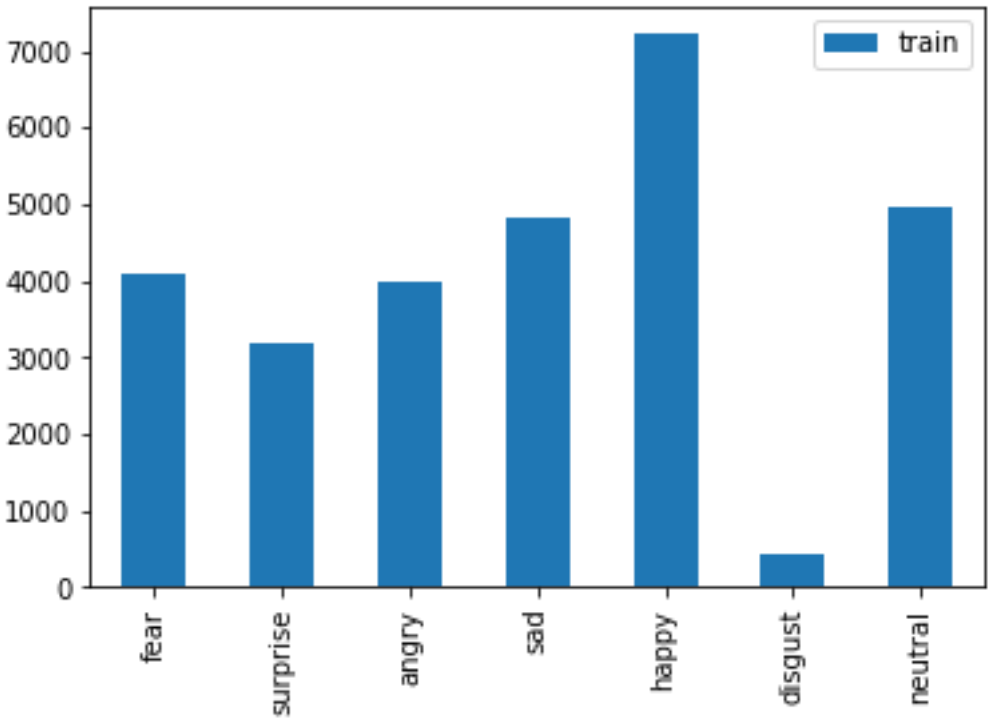

Fig: Bar-diagram of emotion classes vs Images quantity in Training dataset

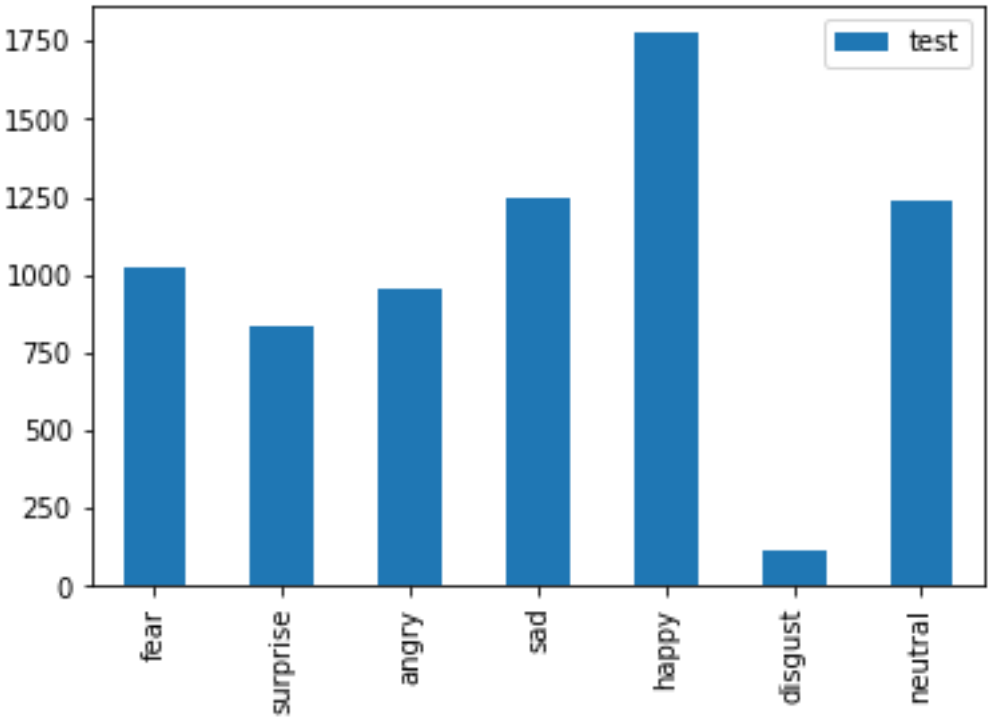

Fig: Bar-diagram of emotion classes vs Images quantity in Testing Dataset

Conclusion

In this project, we used a deep CNN for facial emotion recognition for seven expressions of ‘neutral’, ‘happy’, ‘sad’, ‘angry’, ‘fear’, ‘surprised’ and ‘disgusted’. The structure of the proposed algorithm has good generality and classification performance. In the existing CNN structure, the optimal structure for reducing the execution time and improving the classification performance was determined by adjusting the number of feature maps in the convolutional layer and the number of nodes in the fully-connected layer. Experimental results confirmed the effectiveness of data preprocessing and augmentation techniques.

However, the image resolution is very low at 48x48 pixels and contains incorrectly labeled images. If a CNN structure is designed for a low-resolution input image, the high-resolution input image must be resized to fit the structure. In this process, the classification performance is decreased because the image ratio is altered and blur occurs. Also, incorrect labeling deteriorates the classification performance.

Future Scope

As we know the facial expression recognition system are deploying in different civilian industry and defense industries are try to deploy in their security system so that they can enhance the performance of and security of their equipment. And the demand of these kind of system will going to increase in future.

References

[1] https://www.youtube.com/watch?v=avv9GQ3b6Qg

[2] https://analyticsindiamag.com/10-face-datasets-to-start-facial-recognition-projects/

[3] https://www.kaggle.com/msambare/fer2013/activity

Project Link: My GitHub Repository