Indian Institute of Technology, Kanpur

Summer Project 2020

Emotion Recognition Using Facial Expressions

DOCUMENTATION-FER4

Github Repository

Paper we implemented

We didn't implement the paper exactly as it is but followed the direction specified in the paper. There are certain variations which we did to make it our own.

Introduction

The Emotion Recognition using Facial Expressions project was offered by Brain and Cognitive Society, Science and Technology Council, IIT Kanpur in the summer of 2020. The project, as is self-explanatory from it's name, aims to to detect involuntary emotional response occurring simultaneously with conflicting voluntary emotional response and identifying true emotions by building classifiers that categorise facial expressions into 6 universal emotion categories (+1 neutral)

1. Anger
2. Happiness
3. Disgust
4. Fear
5. Sadness
6. Surprise
7. Neutral

Dataset

The Dataset used is the The Japanese Female Facial Expression, or JAFFE dataset.The database contains 213 images of 7 facial expressions (6 basic facial expressions + 1 neutral) posed by 10 Japanese female models. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects. The database was planned and assembled by Michael Lyons, Miyuki Kamachi, and Jiro Gyoba.

Sample Images from JAFFE dataset

Contents

1. Pre Processing

1. **Face detection**
   This technique is responsible for selecting the ROI of an input image that will be fed to the next steps of the FER system. The images are first converted into gray images followed by face detection using haar cascade frontalface detector.

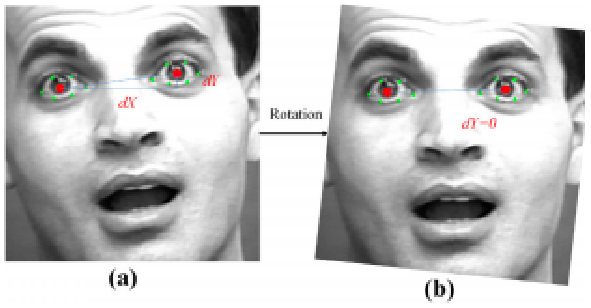
2. **Face alignment(rotation correction)**
   We used the facial landmarks outputted by "shape_predictor_5_face_landmarks". To perform face alignment of a rotated face, a rotation transformation is applied to align two facial landmarks(centre of both the eyes) with the horizontal axis, until the angle formed by them is zero.

3. **Data normalization(Histogram equilization)**
   Because of potential different lighting conditions that extracted faces may present, the recognition accuracy will likely suffer. So we used algorithms of cv2(cv2.equalizeHist(img)) for histogram equilization.

4. **Data augmentation**
   Since a small dataset can lead to overfitting, we performed random rotation, random noise and horizontal flipping on pre-processed images to increase the dataset. We have used algorithms of skimage for this.

All the images after preprocessing along with augmentated images are appended in a list and we get a final list of pre processed data which will be used further.

Libraries used for these processes were dlib,cv2,numpy,matplotlib,os,skimage and keras.preprocessing.image.



Before Face Alignment and After Face Alignment

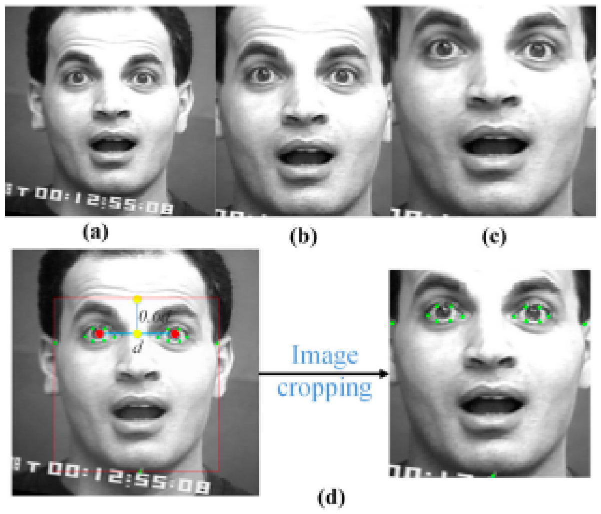
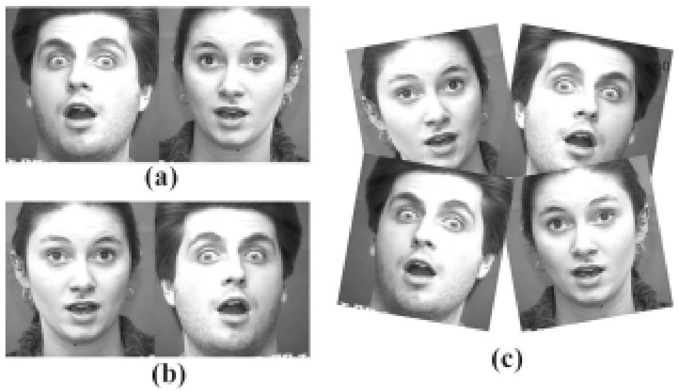


Image Cropping Methods
a)Face with background. b)Face Without background. c)Face Without Forehead(Proposed)
d)Illustrating the Proposed Face Cropping Method



Data Augmentation
a)Original images. b)Horizontally Flipped Images. c)Randomly Rotated Images

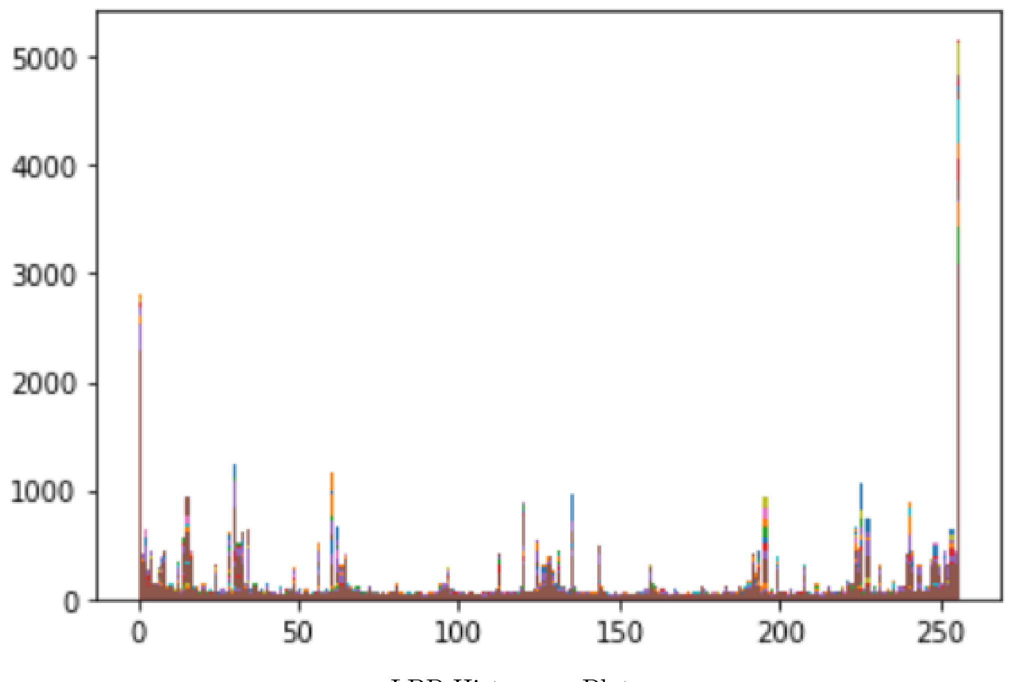
2. Feature Extraction

The list of images generated after pre-processing will be used to highlight relevant features in the images hence. In a conventional FER system, the relevant features are facial features.

For the purpose of feature extraction we will be using Local Binary Patterns (LBP).

In LBP, we focus on 9 pixels (3x3) at a time. We first compare the neighbouring pixels with the central pixel, and assign a value of '0' or '1' to the neighbouring pixel relative to the central pixel. Doing so we get a set of 8 bits which we can convert into a byte, going either clockwise or anticlockwise. Further we convert it into a decimal value which is assigned to the central pixel. The same is done for each pixel of the image, hence, highlighting facial features like eyes,nose,eyebrows and mouth.

1. After pre-processing of images, we have a list of 876 images on which we have to apply the LBP algorithm to extract their features.
2. We have created a simple function for LBP using basic functions.
3. We then run this algorithm on all images using loops, saving the ready images in a list
4. Finally we plot a histogram of each image and save them in a separate list.



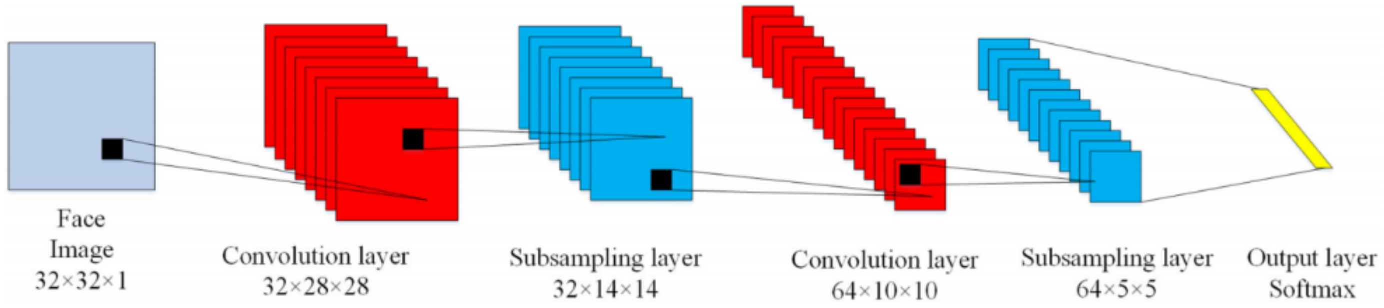
LBP Histogram Plot

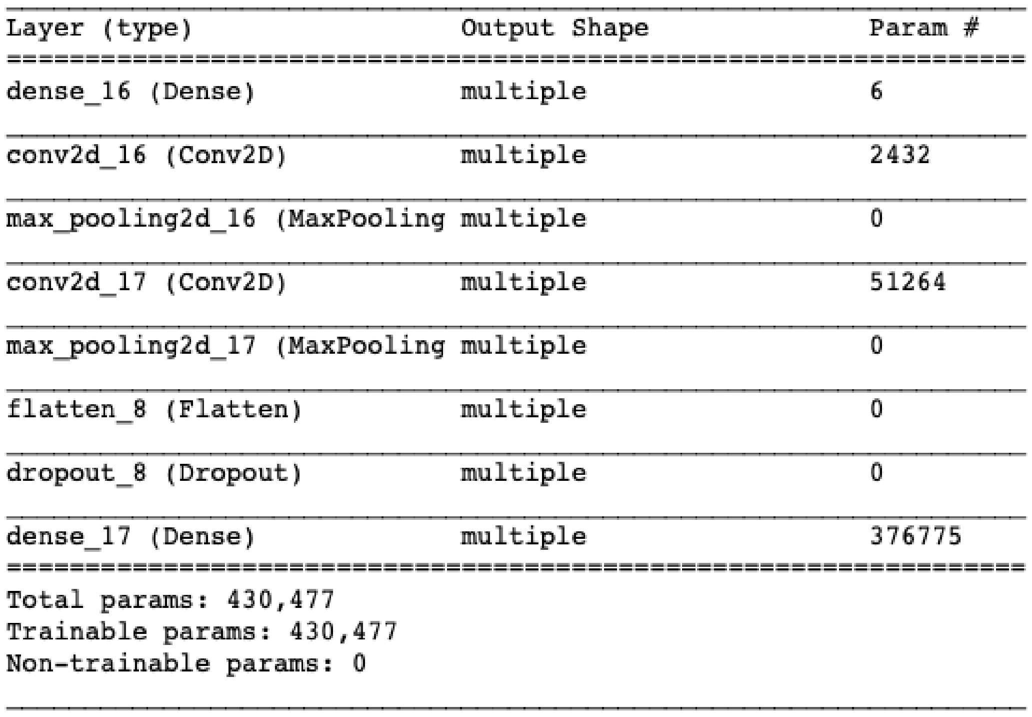
3. Classification

After Pre-processing and Feature Extraction, a CNN model has been used to classify images into one of the 6 emotions (+1 neutral) based on numeric labels assigned to each of the emotions beforehand.

Two models were predefined and the final CNN model that has been used is the one which is specified by the paper, with **a variation in Dropout value** to achieve optimum accuracy.

Xavier initializer has been used which ensures that the mean of the activations should be zero and the variance of the activations should stay the same across every layer, coupled with Relu activation function along with the Adam optimizer. Learning Rate = 0.0001


Face Image 32×32×1
Convolution layer 32×28×28
Subsampling layer 32×14×14
Convolution layer 64×10×10
Subsampling layer 64×5×5
Output layer Softmax

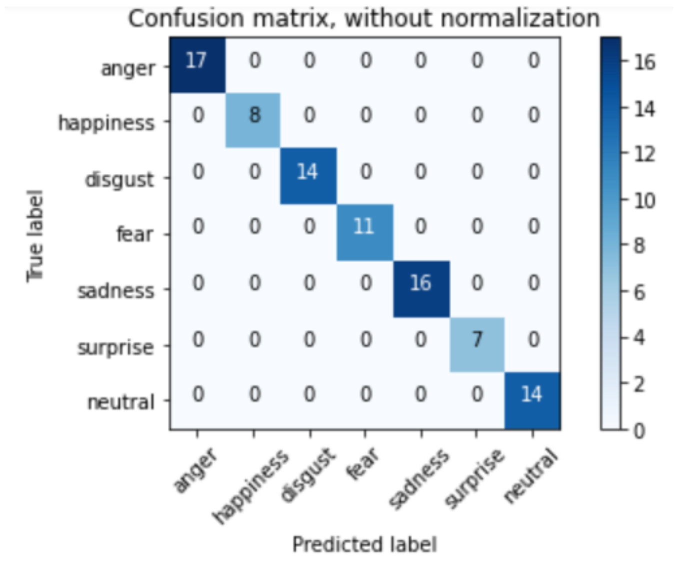


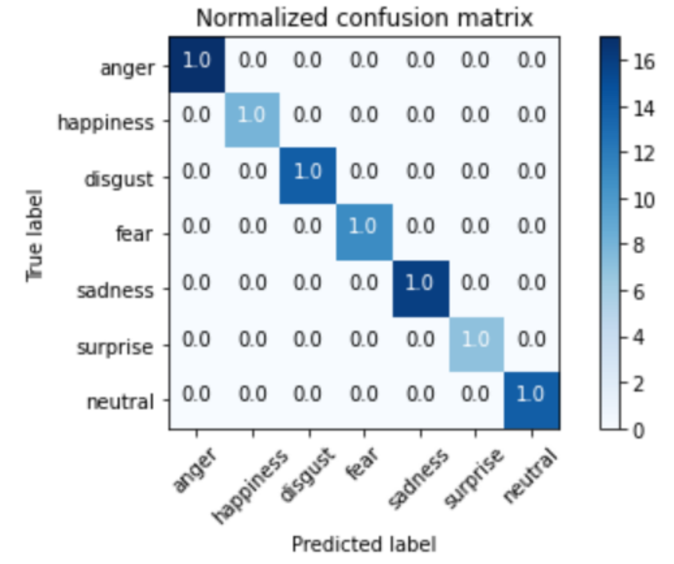
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             multiple                  6
_________________________________________________________________
conv2d_16 (Conv2D)           multiple                  2432
_________________________________________________________________
max_pooling2d_16 (MaxPooling multiple                  0
_________________________________________________________________
conv2d_17 (Conv2D)           multiple                  51264
_________________________________________________________________
max_pooling2d_17 (MaxPooling multiple                  0
_________________________________________________________________
flatten_8 (Flatten)          multiple                  0
_________________________________________________________________
dropout_8 (Dropout)          multiple                  0
_________________________________________________________________
dense_17 (Dense)             multiple                  376775
=================================================================
Total params: 430,477
Trainable params: 430,477
Non-trainable params: 0
_________________________________________________________________
```

CNN MODEL

4. Validation

After Classification is done , we run K-fold cross validation (with k=10) to measure accuracy of the neural network. As a result the computation time is reduced, the bias and variance is reduced (preventing underfitting and overfitting) because every datapoint gets to be tested once. We **expermient on the structure of the neural network by adding a hidden fully connected layer** after the second subsampling layer and comparing the accuracies for different number of neurons in this layer for the values **[0,256,512,1024]** . We find **maximum average accuracy for 0** , i.e the **neural network structure is better without the hidden fully connected layer**.





5. Results

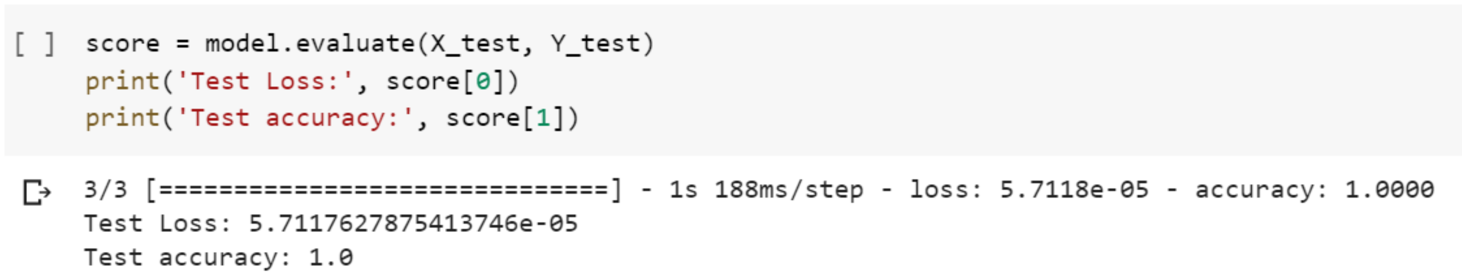
The final accuracy obtained by the model was **100% (training, validation and test accuracy) with a test loss of (5.711762e-05).** There were total 876 images in the JAFFE dataset.

The accuracy obtained using the parameters of the Paper was 76.12%

The accuracy obtained without Kfold cross validation and LBP was 65.01%, while that with just Kfold was 83.91% peak accuracy (test data)

The best accuracy obtained by the paper with the JAFFE dataset was 97.65% with an average of 97.18%

Our final model yielded the highest **accuracy of 100% which was almost consistent after the 3rd epoch.** There were differences noticed in the training times too as **our model required 51 minutes at max to train** while on the same computer, it required **approximately 9 hours to train using the paper's parameters.**



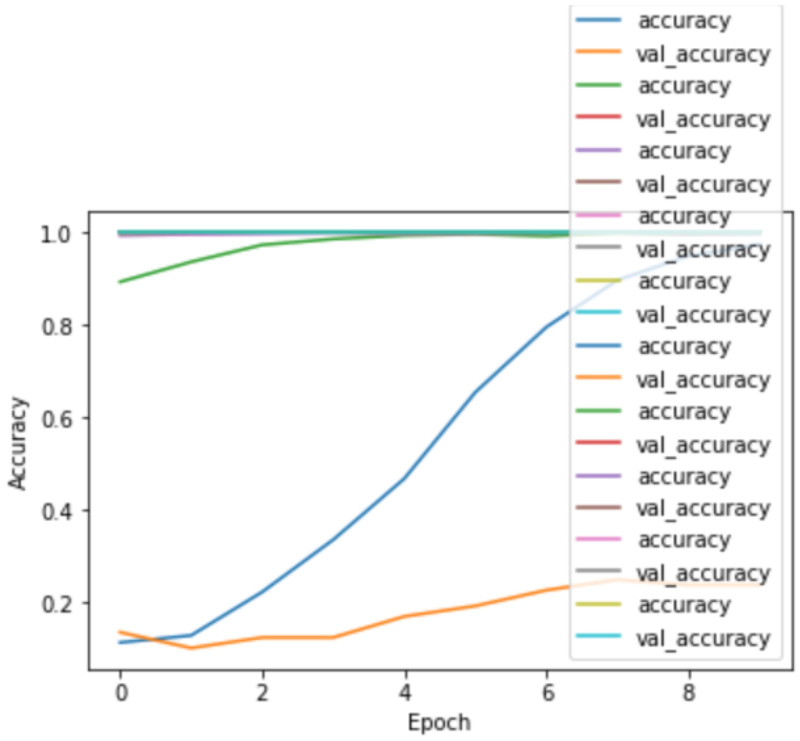
```
[ ] score = model.evaluate(X_test, Y_test)
    print('Test Loss:', score[0])
    print('Test accuracy:', score[1])
```

```
3/3 [==============================] - 1s 188ms/step - loss: 5.7118e-05 - accuracy: 1.0000
Test Loss: 5.7117627875413746e-05
Test accuracy: 1.0
```

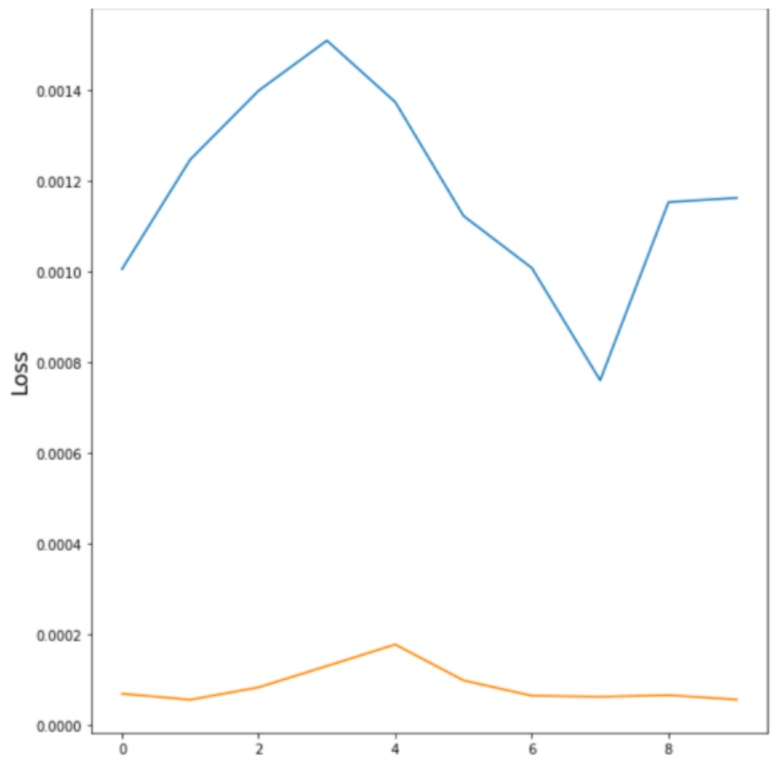
TEST LOSS AND ACCURACY OF CNN

After this , we tested our CNN Model with more testing data which were images containing expressions of Japanese Women (since our model was trained on JAFFE dataset) and obtained an **accuracy of 97.334%.**
Also, after experimenting our model by adding a fully connected layer we obtained quite low accuracies .
For neuron number 256,512 and 1024 the accuracy was in the range of 40%-50% and the trainable parameters were reduced to 50,000 from 430,477.
This happened because **due to implementation of Linear Binary Patterns, a hidden dense layer will lose lots of parameters.**

Model	**Accuracy**
CNN with parameters of the paper	76.12%
CNN with Modified Parameters	65.01%
CNN with Modified Parameters and 10-fold Cross Validation	83.91%
CNN+Modified Parameters+10 fold validation+LBP	97.18%
CNN tested on more data(Images of Japanese Women)	97.34%
CNN with a hidden dense layer with neurons(256, 512, 1024)	40%-50%



Accuracy of CNN Model without Hidden dense layer



Training loss(BLUE) and Test loss(ORANGE) vs epochs(X-axis)

6. Variations

PAPER	**OUR MODEL**
Momentum Optimiser	Adam Optimiser
Learning Rate = 0.001	Learning Rate = 0.0001
Momentum = 0.001	-
DropOut(0.5)	DropOut(0.25)
-	Local Binary Patterns
Epochs = 120	Epochs = 10
Batch Size = 16	Batch Size = Default

There are a few other variations too in the Pre-processing code **(Cropping and Alignment methods)** and in validation, which **opted out of using a hidden fully connected layer to improve accuracy**.

7. Visualization

Initial Image

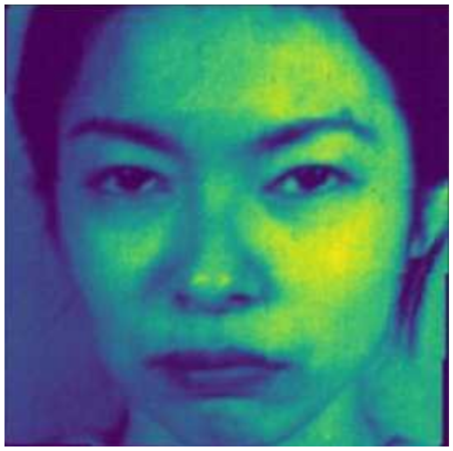


Image After Cropping

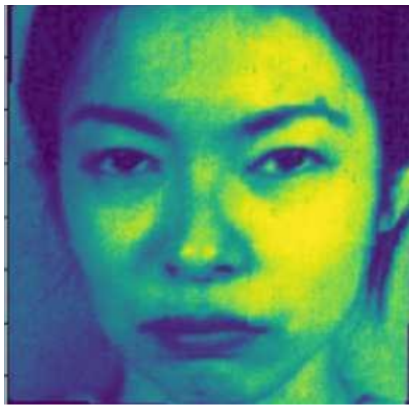
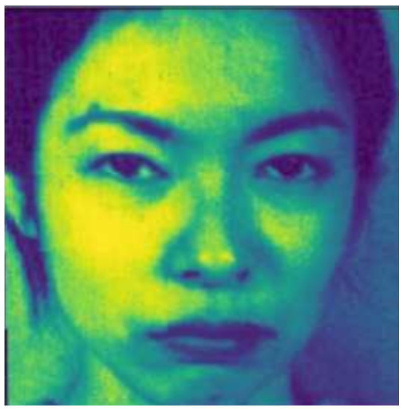


Image after histogram equalization



Flipped Image

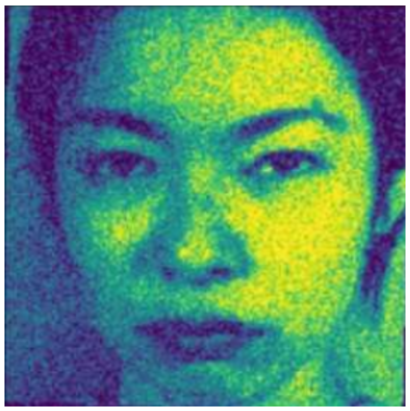


Image after noise addition

8. Impact

In this project , we present an efficient FER approach which **simplifies the CNN** and **propose new face cropping and image rotation methods** coupled with the **implementation of Linear Binary Patterns**. We envision that these **proposed data processing methods will lead to high recognition accuracies** and can be **implemented on an ordinary computer without GPU acceleration.** We hope our effort will act as catalyst for deeper research.

9. References

[1] https://keras.io/api/layers/initializers/

[2] https://keras.io/api/optimizers/

[3] https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79

[4] https://www.tensorflow.org/api_docs/python/tf/keras/Model

[5] http://www.scikit-learn.org/

[6] https://www.geeksforgeeks.org/image-processing-in-python-scaling-rotating-shifting-and-edge-detection/

[7] https://towardsdatascience.com/precise-face-alignment-with-opencv-dlib-e6c8acead262

[8] https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

[9] https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

10. Team

Team Mentor :

- Avisha Gaur

Team Members :

- Falguni Yadav
- Prachi Singh
- Pranav Kumar
- Saksham Pruthi
- Samriddhi Gupta
- Tanisha Agrawal
- Videeta Sharma
- Yatharth Gupta