DOCUMENTATION-FER 4

https://github.com/Consilium5128/Emotion-Recognition-FER4

The project, 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

PAPER

Li2020_Article_FacialExpressionRecognitionWit.pdf

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.

DATASET

The Dataset used is the The Japanese Female Facial Expression, or JAFFE dataset.

VARIATIONS FROM PAPER

PAPER	OUR MODEL
Momentum optimizer	Adam optimizer
Learning rate=0.001	Learning rate=0.0001
momentum = 0.001	-
DropOut(0.5)	DropOut(0.25)
-	Local Binary Patterns(LBP)
Epochs=120	Epochs=10
Batch Size=16	Batch Size=Default

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

TRAINING AND TESTING DATA

We used Kfold Cross Validation on JAFFE dataset (with augmentation) to train the model and simultaneously test it too. No external dataset was used for testing (No cross-database testing)

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. The link for the codes of all the pre processing steps is here-

LIBRARIES USED-dlib,cv2,numpy,matplotlib,os,skimage, keras.preprocessing.image

FEATURE EXTRACTION

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.

CLASSIFICATION

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

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 test the structure of the neural network by adding a hidden fully connected layer and comparing the accuracies for different number of neurons in this layer [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.

RESULTS

The final accuracy obtained by the model was 100% (both training and validation accuracy) with a loss of (5.711762e-05) on a total of 876 images of the JAFFE dataset with a 10 split K-fold Cross Validation

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

The accuracy obtained without K-fold cross validation and LBP was 65.01%, while that with just Kfold was 83.91% peak accuracy (validation 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.

VISUALIZATION

INITIAL IMAGE

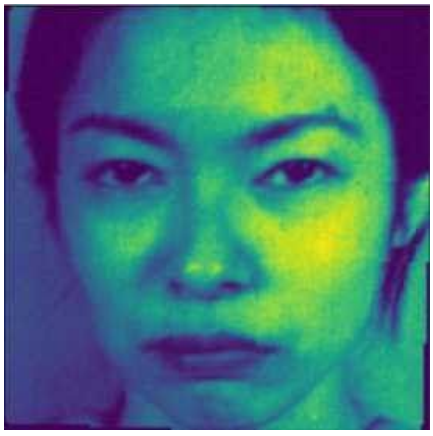


IMAGE AFTER CROPPING AND FACE ALIGNMENT

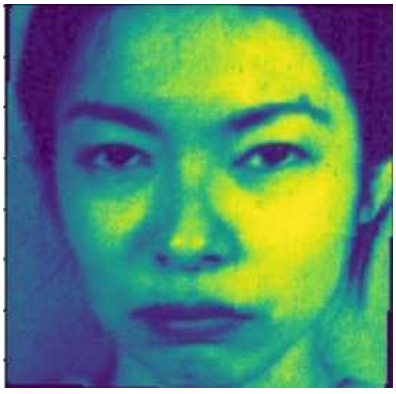
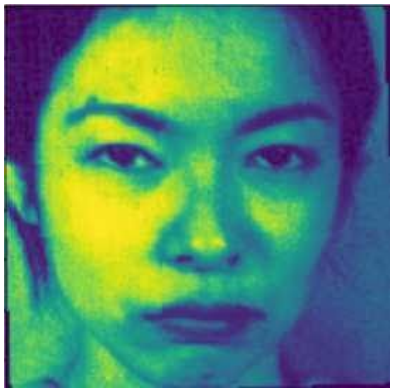
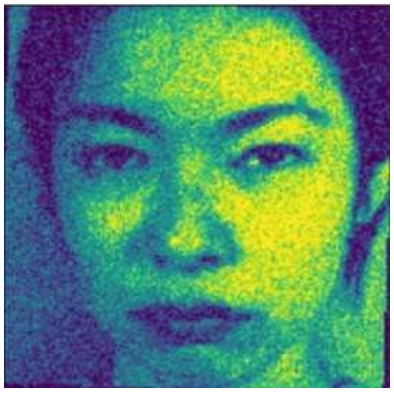


IMAGE AFTER APPLYING HISTOGRAM

NORMALIZATION

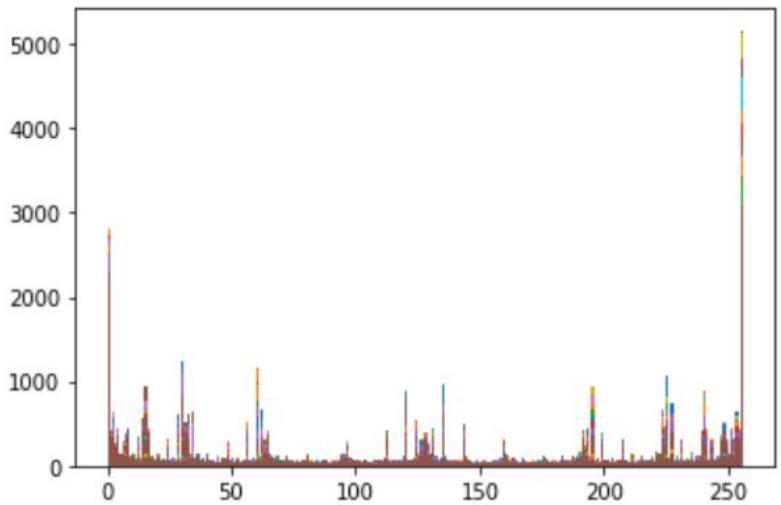


FLIPPED IMAGE

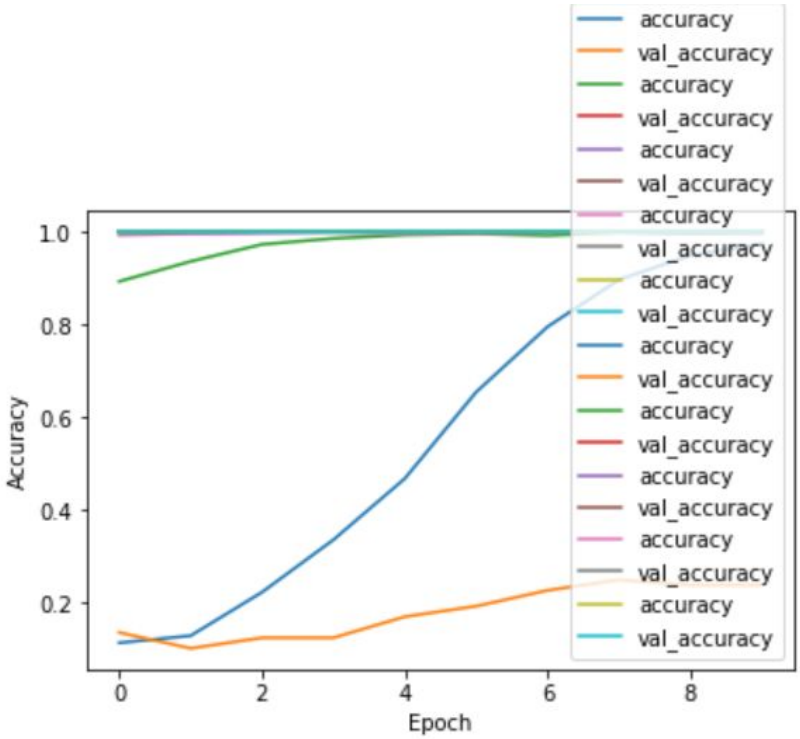


NOISE ADDITION

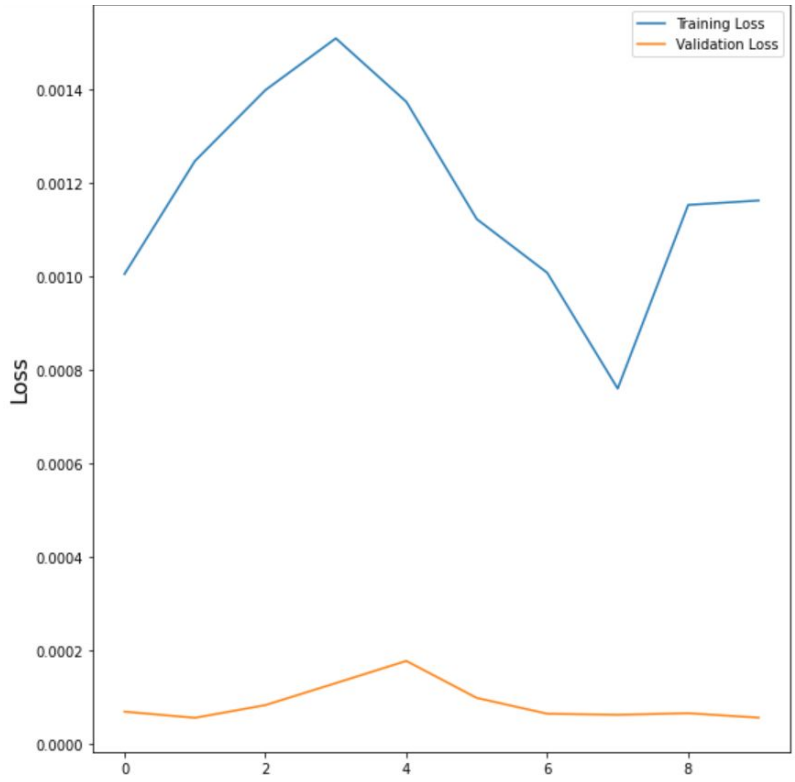
LBP HISTOGRAM



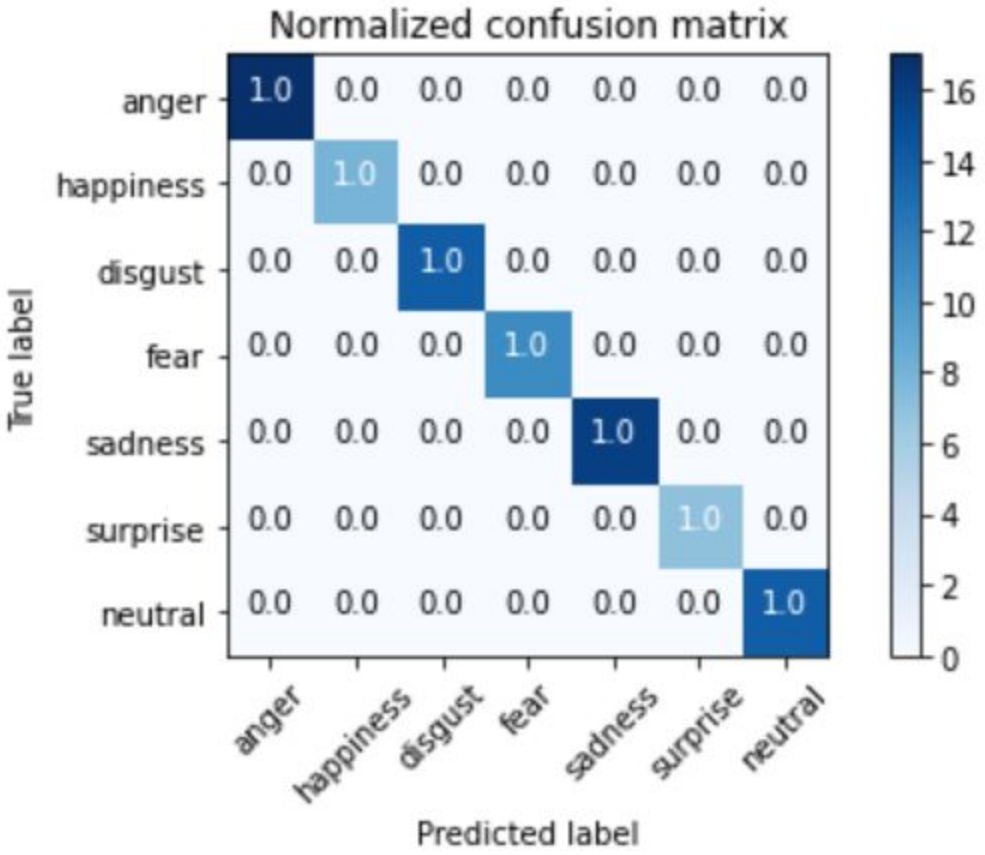
ACCURACY PLOT



LOSS PLOT



NORMALIZED CONFUSION MATRIX



TEAM

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

REFERENCES

1. **www.geeksforgeeks.org**
2. **www.stackoverflow.com**
3. **https://keras.io/api/layers/initializers/**
4. **https://keras.io/api/optimizers/**

5. https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79
6. https://www.tensorflow.org/api_docs/python/tf/keras/Model
7. www.scikit-learn.org
8. www.machinelearningmastery.com
9. https://towardsdatascience.com/precise-face-alignment-with-opencv-dlib-e6c8acead262
10. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/
11. https://www.geeksforgeeks.org/image-processing-in-python-scaling-rotating-shifting-and-edge-detection/
12. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/