Facial Emotion Recognition (WebUI) using python

SRH Hochschule Heidelberg

ABSTRACT

Project of a convolutional neural network (CNN) using Keras to recognize 7 universal facial expressions (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral) model integrated in web UI using flask. Model Can recognize from live video feed (from camera) and video input(static)

Submitted to

Dr. Simon Ziegler
Prof. Dr. Moeckel Gerd

Submitted by :

Rathod, Kishansinh - 11013344

Date : 16-July-2020

Index

1. Introduction

Facial Expression Recognition (FER) can be widely applied to various research areas, such as mental diseases diagnosis and human social/physiological interaction detection. With the emerging advanced technologies in hardware and sensors, FER systems have been developed to support real-world application scenes, instead of laboratory environments. Although the laboratory-controlled FER systems achieve very high accuracy, around 97%, the technical transferring from the laboratory to real-world applications faces a great barrier of very low accuracy, approximately 50% [1].

Facial expression recognition is the task of classifying the expressions on face images into various categories such as anger, fear, surprise, sadness, happiness and so on. There are total 7 universal facial expression Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral. This project is capable to identify all 7-facial expression in ideal condition.

This Project is using OpenCV to automatically detect faces in images and draw bounding boxes around them. Then our trained model can identify all expression in Web UI which is made using flask – Microweb framework. We used Convolutional Neural Network to train and save our model.

Project is divided in main three part

- **Model Training – ANN - Convolution Neural Network**
- **Image Recognition – OpenCV (haar cascase)**
- **WebUI integration – Flask**

1.1 Dataset Introduction

Here we used data from Kaggle.com which is made available for competition in 2013 named facial expression recognition (fer2013)

This is the sample image from all 7 universal facial expression from data set. Data set contains around of 20K images which we used to train our model.

We used 80% data t train model and 20% data to test model

2. Model Description

what is Convolutional Neural Network (CNN) is designed to address image recognition systems and classification problems. Convolutional Neural Networks have wide applications in image and video recognition, recommendation systems and natural language processing.

Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks.

There are **four** layered **concepts** we should understand in Convolutional Neural Networks:

1. **Convolution,**
2. **ReLu,**
3. **Pooling and**
4. **Full Connectedness (Fully Connected Layer).**

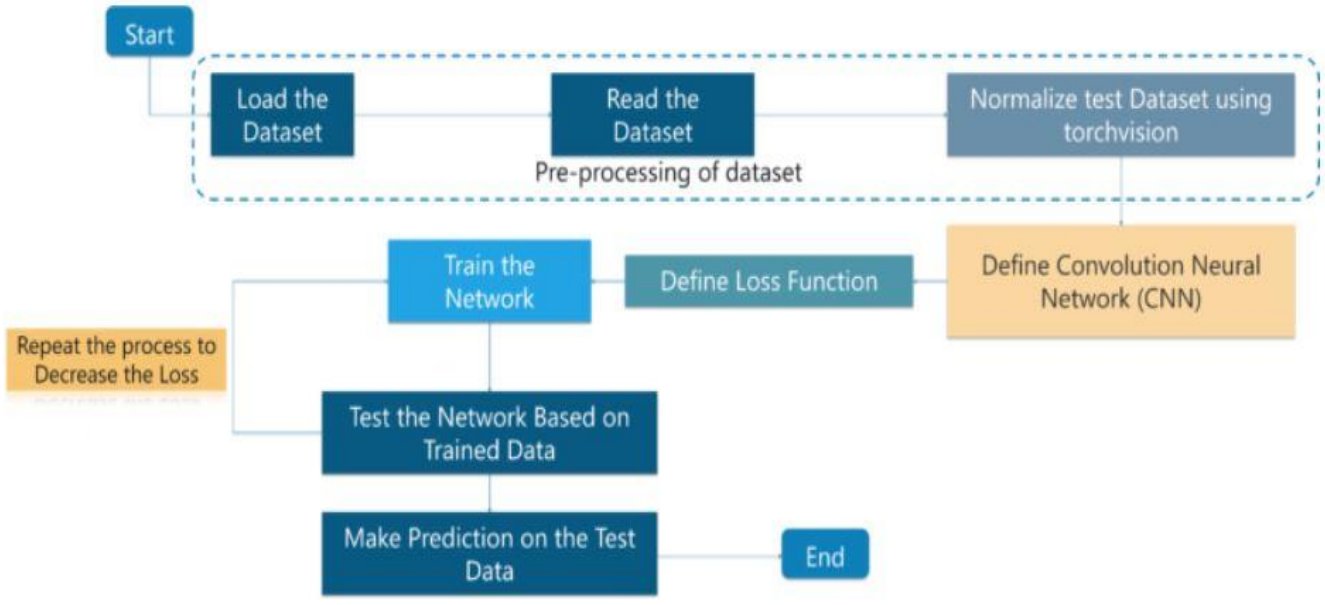

Image 2.1 : Flowchart of Convolutional Neural Network

Here we used Sequential model of **CNN**.

Our model is made of following thing

- **CNN (Sequential Model)**
  - **Convolution Layer**
    - **Conv2D**
    - **BatchNormalisation**
    - **Activation Function (relu)**
    - **Pooling (MaxPooling2D)**
    - **Dropout**
  - **Fully Connected Layer**
  - **Final Output**

Following diagram will elaborate in detail model we used here

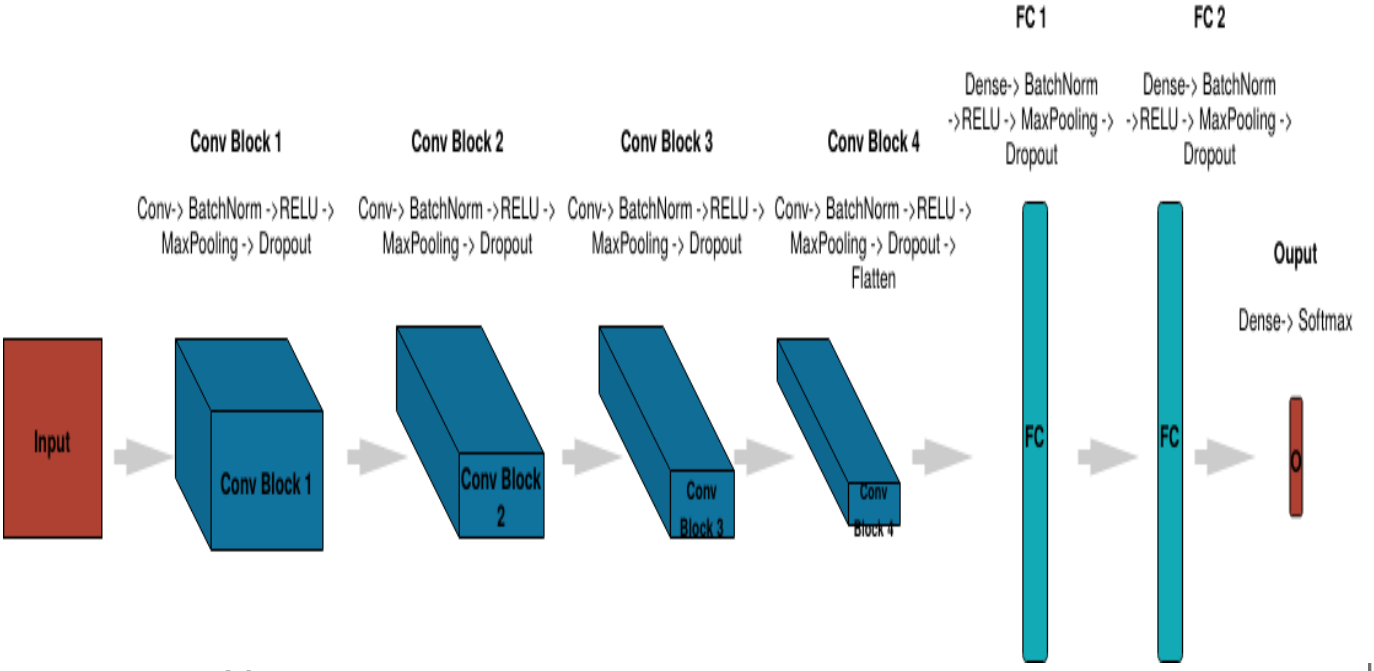

Image 2.2 Model Diagram

Here from diagram you can built model with 4 convolution layer which is followed by two fully connected layer which gives final output.

In Conv Layer we used single Conv2D then we apply batch normalisation on that output of Conv2D then we apply activation function on that then MaxPooling2D and at the end drop out to our model from overfitting.

```python
# 1 - Convolution
model.add(Conv2D(64,(3,3), padding='same', input_shape=(48, 48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

Image 2.3 : Sample Code snippet ConvLayer

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256
_________________________________________________________________
activation (Activation)      (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 512)       2048
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
```

Image 2.4 : Model Info1 Console based

```
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
_________________________________________________________________
Press any key to continue . . .
```

Image 2.5 : Model Info1 Console based

We tried model with two activation function : Relu and Elu for the better result and we finalised to go with ‘relu’ Actiovation function.

2.1Activation Function (RELU)

Rectified Linear Unit (ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable.

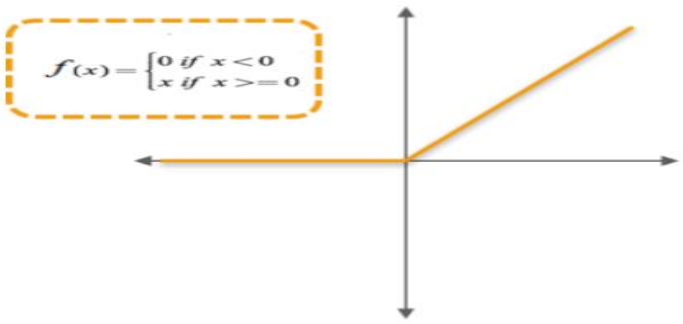

Image 2.6 : Relu Activation Function

When using Relu activation function we train our model for 15 epochs and we got given below result

```
Epoch 1/15
accuracy[==============================] - ETA: 0s - loss: 1.7934 - accuracy: 0.3130[2K
       training               (min:    0.313, max:    0.313, cur:    0.313)
       validation             (min:    0.340, max:    0.340, cur:    0.340)
Loss
       training               (min:    1.793, max:    1.793, cur:    1.793)
       validation             (min:    1.714, max:    1.714, cur:    1.714)

Epoch 00001: saving model to model_weights.h5
448/448 [==============================] - 849s 2s/step - loss: 1.7934 - accuracy: 0.3130 - val_loss: 1.7144 - val_accuracy: 0.3396 - lr: 5.0000e-04
Epoch 2/15
accuracy[==============================] - ETA: 0s - loss: 1.4763 - accuracy: 0.4351[2K
       training               (min:    0.313, max:    0.435, cur:    0.435)
       validation             (min:    0.340, max:    0.444, cur:    0.444)
Loss
       training               (min:    1.476, max:    1.793, cur:    1.476)
       validation             (min:    1.468, max:    1.714, cur:    1.468)

Epoch 00002: saving model to model_weights.h5
448/448 [==============================] - 858s 2s/step - loss: 1.4763 - accuracy: 0.4351 - val_loss: 1.4681 - val_accuracy: 0.4438 - lr: 5.0000e-04
Epoch 3/15
accuracy[==============================] - ETA: 0s - loss: 1.3380 - accuracy: 0.4870[2K
       training               (min:    0.313, max:    0.487, cur:    0.487)
       validation             (min:    0.340, max:    0.495, cur:    0.495)
Loss
       training               (min:    1.338, max:    1.793, cur:    1.338)
       validation             (min:    1.290, max:    1.714, cur:    1.290)

Epoch 00003: saving model to model_weights.h5
448/448 [==============================] - 860s 2s/step - loss: 1.3380 - accuracy: 0.4870 - val_loss: 1.2904 - val_accuracy: 0.4951 - lr: 5.0000e-04
Epoch 4/15
```

Image 2.7 : Epoch 1-3 with Relu Activation function

```
Epoch 14/15
accuracy[==============================] - ETA: 0s - loss: 0.9198 - accuracy: 0.6563[2K
       training               (min:    0.313, max:    0.656, cur:    0.656)
       validation             (min:    0.340, max:    0.638, cur:    0.638)
Loss
       training               (min:    0.920, max:    1.793, cur:    0.920)
       validation             (min:    0.982, max:    1.714, cur:    0.982)

Epoch 00014: saving model to model_weights.h5
448/448 [==============================] - 852s 2s/step - loss: 0.9198 - accuracy: 0.6563 - val_loss: 0.9824 - val_accuracy: 0.6383 - lr: 5.0000e-05
Epoch 15/15
accuracy[==============================] - ETA: 0s - loss: 0.9079 - accuracy: 0.6578[2K
       training               (min:    0.313, max:    0.658, cur:    0.658)
       validation             (min:    0.340, max:    0.641, cur:    0.641)
Loss
       training               (min:    0.908, max:    1.793, cur:    0.908)
       validation             (min:    0.977, max:    1.714, cur:    0.977)

Epoch 00015: saving model to model_weights.h5
448/448 [==============================] - 948s 2s/step - loss: 0.9079 - accuracy: 0.6578 - val_loss: 0.9772 - val_accuracy: 0.6409 - lr: 5.0000e-05
Press any key to continue . . .
```

Image 2.8 : Epoch 1-3 with Relu Activation function

With relu activation function we performed 15 epochs and at the end of 15th epoch we got accuracy on validation data about 64% which is good for our prediction model.

Given below diagram will show detail plot of accuracy vs loss during all 15 epochs

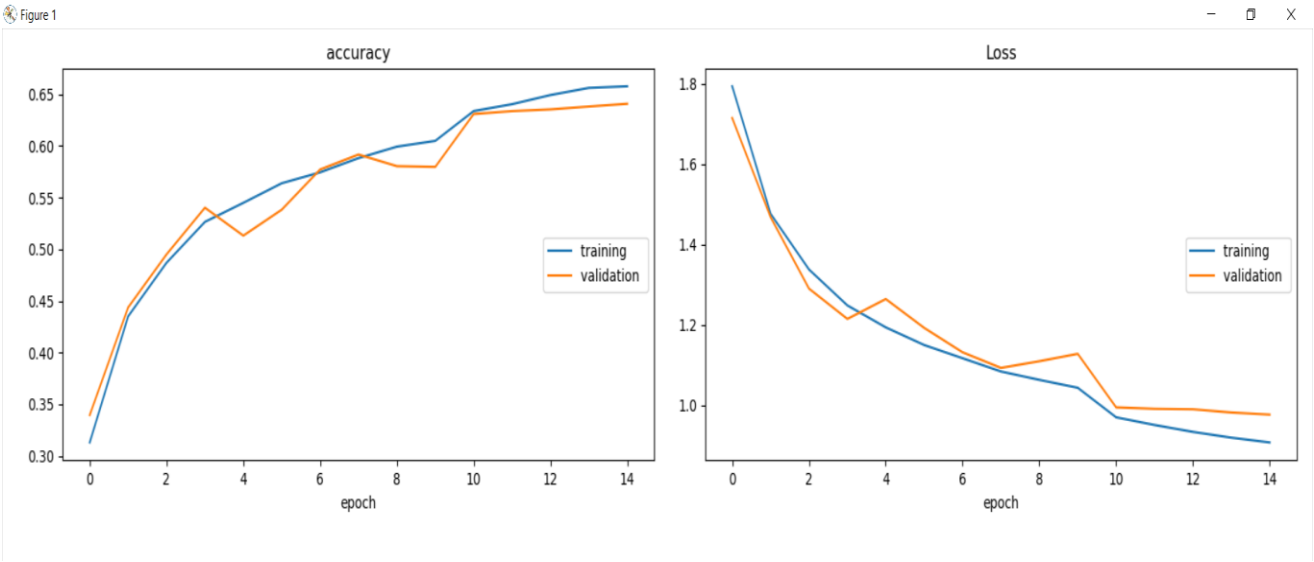

Image 2.9 : Epoch 1-3 with Relu Activation function

2.2Activation Function (elu)

Exponential Linear Unit or its widely known name ELU is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has a extra alpha constant which should be positive number.

ELU is very similiar to RELU except negative inputs. They are both in identity function form for non-negative inputs. On the other hand, ELU becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes.

$$R(z) = \begin{cases} z & z > 0 \\ \alpha.\,(e^{z}-1) & z <= 0 \end{cases}$$

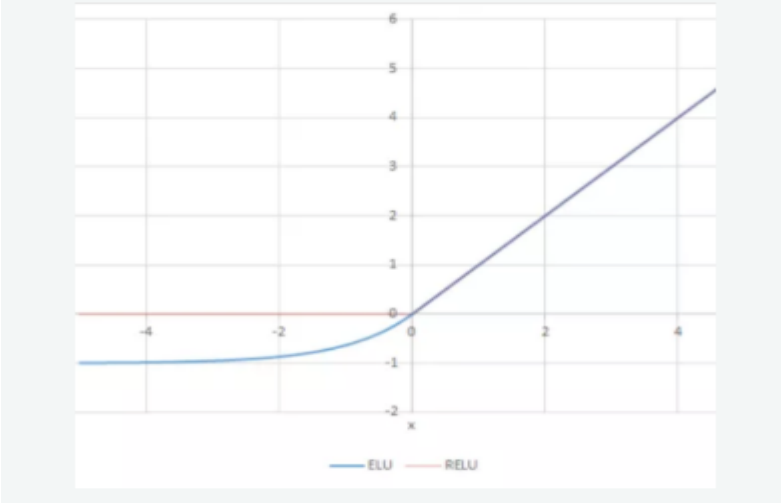

Image 2.10 : Elu Activation Function

At the beginning result of elu is slightly batter compare to relu but while going on with epochs it becomes identical to relu's result at epoch 4th and from epoch 5th Result from relu is got batter so we stopped at 4th and decided to go with Relu as an activation function

Following images will show result from eul's epoch as well as Accuracy vs. Loss graph

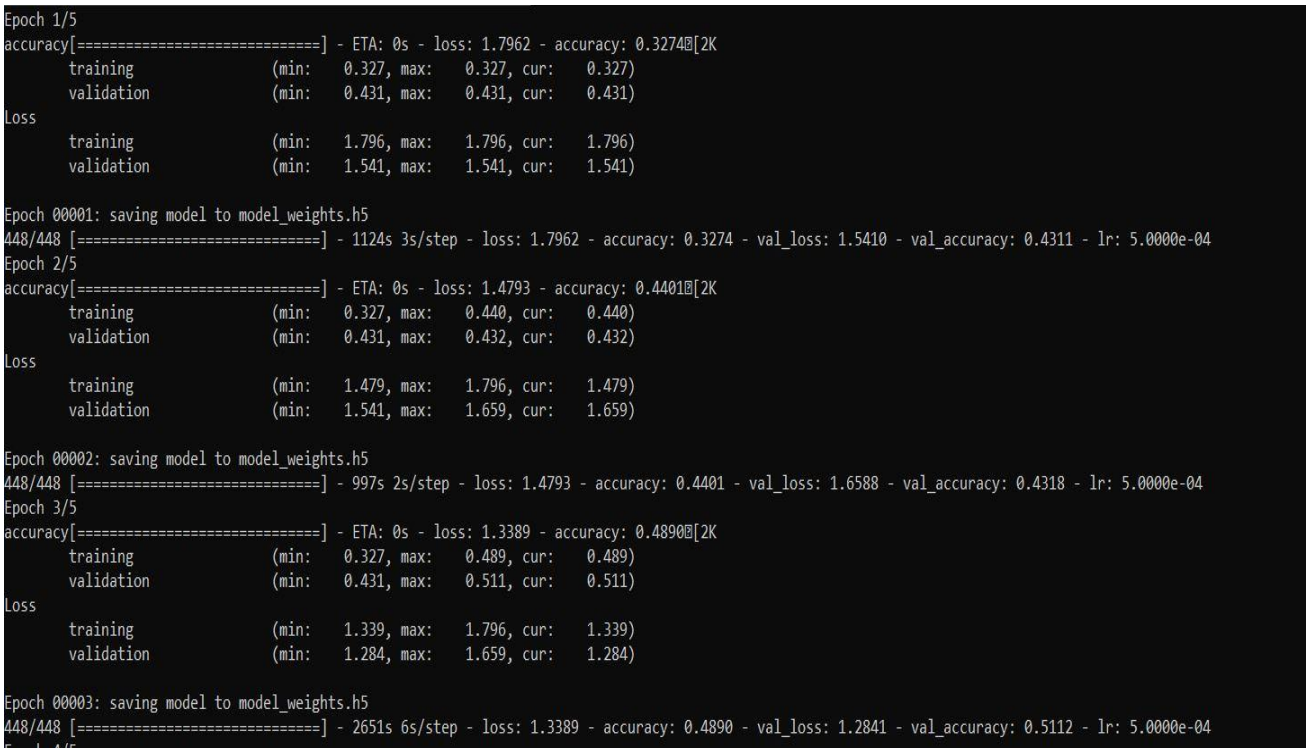

Image 2.11 : Epoch 1-3 with Elu Activation function

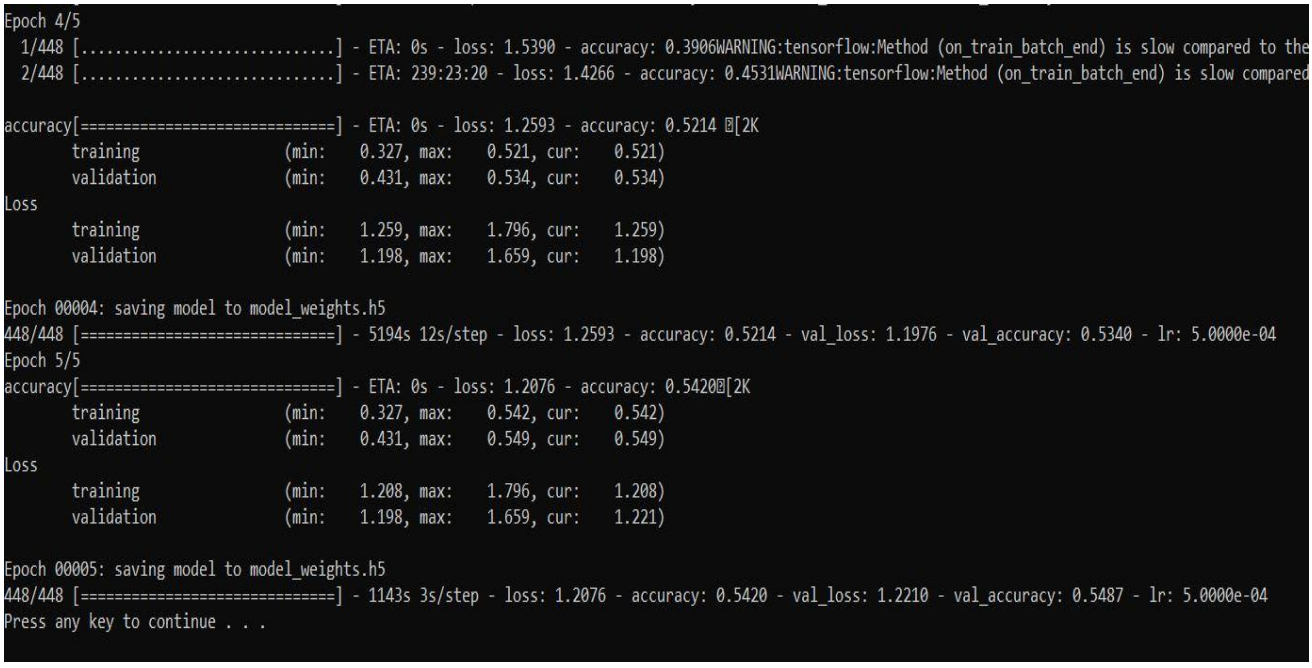

Image 2.12 : Epoch 4,5 with Elu Activation function

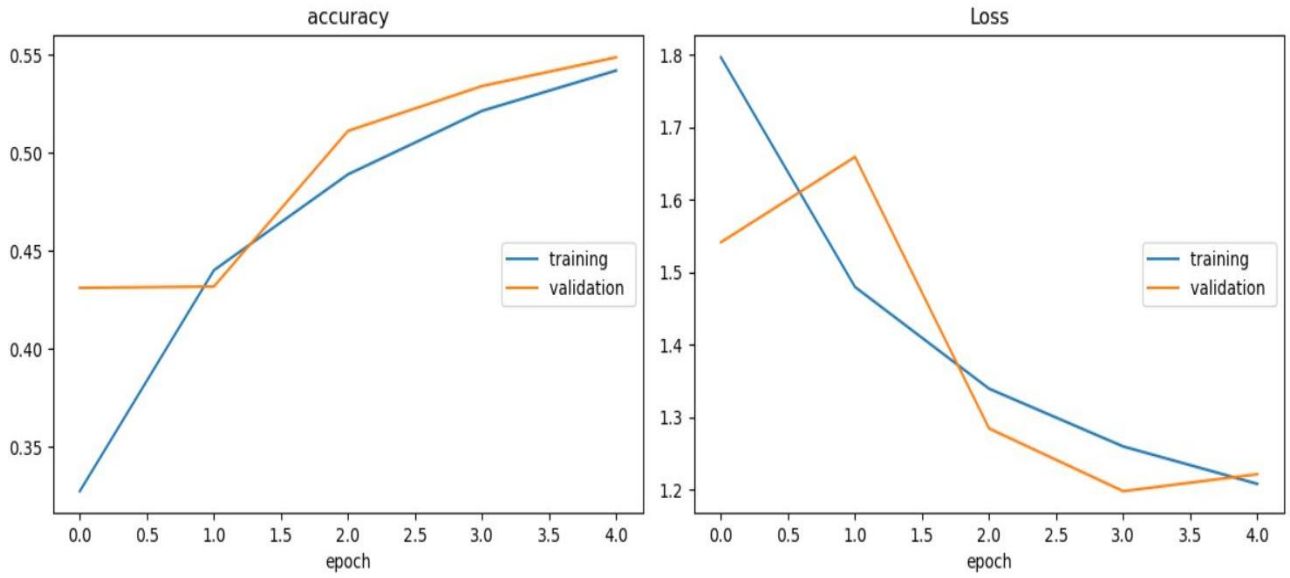

Image 2.13 : Graph Elu Accuracy vs. Loss

3. Limitations

3.1 Face detection using haar cascade

OpenCV's Haar cascade is very good predefine method for face detection but however it is still not optimum. It has its own limitations like it some times gives wrong area while there is more than two persons is present also it face difficulties in extreme side face detection. Even it cannot detect in low light

3.2 Low light prediction

Apart from problem with haar cascade low light detection, model has its own limitation in prediction in low light. It is not giving accuracy in low light if haar detects the file. This my be because of the dataset used for training are in ideal condition and it is not colour images it is grey images.

3.3 Cross angle prediction

Model also face little bit of difficulties while person suddenly start watching in other direction and their side face is towards the camera. This can be overcome using Data augmentation by populating datasets.

3.4 Problem with noisy input

However, model is not optimum with the noisy input. It not perform good when input image contain lots of noise.

4. Conclusion

Convolutional Neural Networks is a popular deep learning technique for current visual recognition tasks. Like all deep learning techniques, Convolutional Neural Networks are very dependent on the size and quality of the training data. Given a well-prepared dataset, Convolutional Neural Networks are capable of surpassing humans at visual recognition tasks. However, they are still not robust to visual artifacts such as glare and noise, which humans are able to cope. The theory of Convolutional Neural Networks is still being developed.

Talking about Model then it still has some limitation which can reduce by making some changes in model. So there absolutely scope of future enhancement in model as well as it can be integrated with eye caching web UI too in future. But considering the Model, it works well with approx. 65% of accuracy and able to predict all 7 universal facial expression in ideal condition which is satisfies the purpose.

5. References

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6514576/
2. https://www.edureka.co/blog/convolutional-neural-network/
3. https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/
4. https://forums.fast.ai/t/dense-vs-convolutional-vs-fully-connected-layers/191/3
5. https://www.kaggle.com/rkritika1508/fer-2013-emotion-recognition
6. https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#:~:text=Exponential%20Linear%20Unit%20or%20its,to%20RELU%20except%20negative%20inputs.
7. https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html
8. https://keras.io/api/layers/convolution_layers/convolution2d/
9. https://machinelearningmastery.com/machine-learning-in-python-step-by-step/