OPERATING SYSTEM PROJECT FILE

FACIAL EXPRESSIONS RECOGNITION

NAME OF THE STUDENT	UNIVERSITY ROLL NO.
SHIVAM THAKUR	2020UEA6589
PRAGYADITYA JHA	2020UEA6578

ABSTRACT

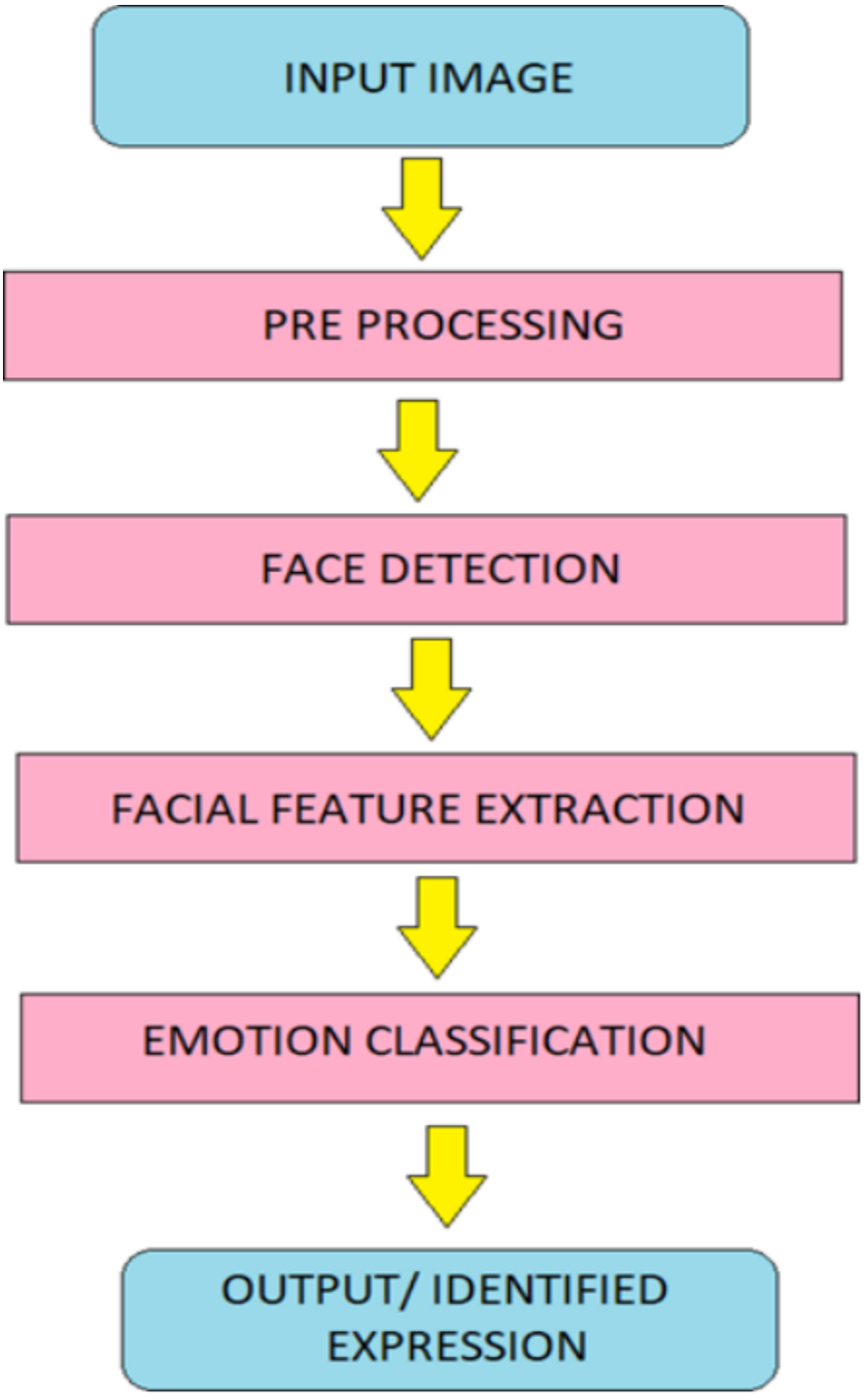
Emojis are small images that are commonly included in social media text messages. The combination of visual and textual content in the same message builds up a modern way of communication. Emojis or avatars are ways to indicate nonverbal cues. These cues have become an essential part of online chatting, product review, brand emotion, and many more. It also led to increasing data science research dedicated to emoji-driven storytelling. With advancements in computer vision and deep learning, it is now possible to detect human emotions from images. In this deep learning project, we will classify human facial expressions to filter and map corresponding emojis or avatars. This project is not intended to solve a real-world problem, instead it allows us to see things more colorful in the chatting world. Emojify is a software which deals with the creation of Emoji's or Avatars.

So, in this project I tried to use facial expressions recognition to implement the emojify facial expressions using Convolutional Neural Network (CNN) and Deep Learning.

Tech Stack Used

- Deep Learning
- Convolutional Neural Network (CNN)
- OpenCV

METHODOLOGY



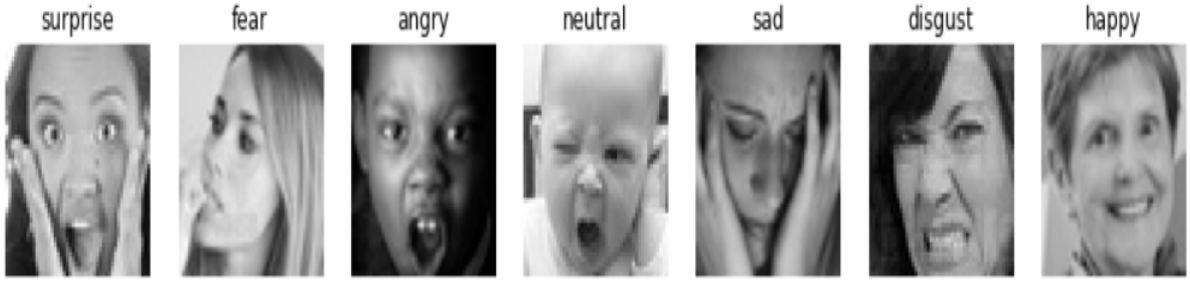
THE WORKFLOW OF OUR PROJECT

DATASET & ITS FEATURES

In this project, the dataset used to train the models is FER-2013. The FER-2013 dataset consists of 35887 images, of which 28709 labelled images belong to the training set and the remaining 7178 images belong to the test set. The images in FER-2013 dataset is labeled as one of the seven universal emotions: Happy, Sad, Angry, Surprise, Disgust, Fear and Neutral. Among these emotion classifications, the most images belong to 'happy' emotions account to 7215 images of the 28709 training images. The number of images that belong to each emotion is given by returning the length of each directory using the OS module in Python. The images in FER-2013 dataset are in grayscale and is of dimensions 48x48 pixels. This dataset was created by gathering results from Google Image search of each emotions. The number of images of each emotion is given in table 1. The number of images of each emotion type is returned by the functions of 'OS' module in python. To explore the dataset further, and to understand what kind of images lie in the dataset, we plot few example images from the dataset using the 'utils' module in python. The resultant plot of example images obtained is given in figure 1. From various research papers, we have studied that the average attainable accuracy of a training model developed using FER-2013 dataset is 66.7% and our aim is also to design a CNN model with similar or a better accuracy.

The number of images each emotion have in the Train and Test dataset are:

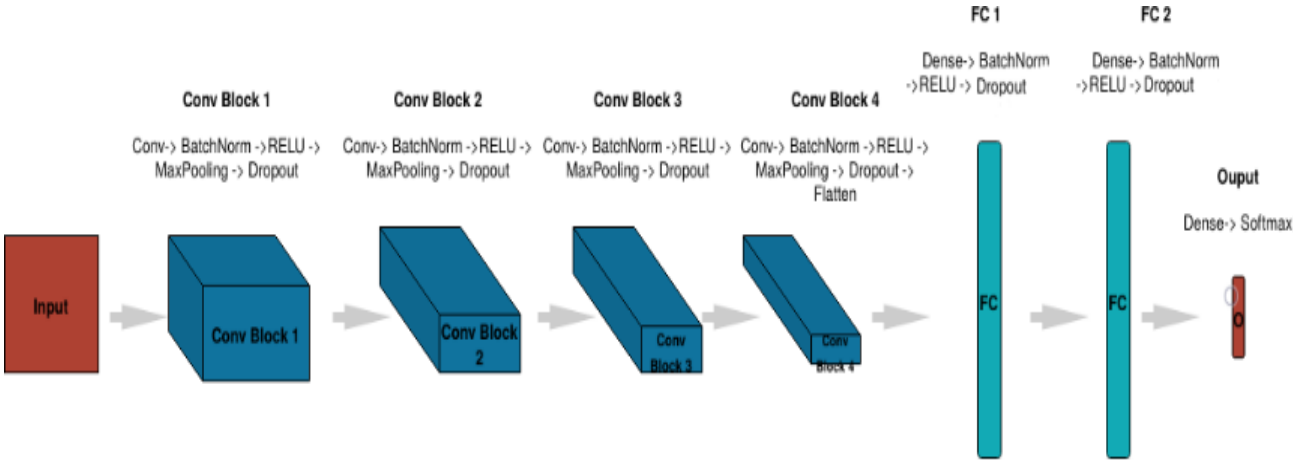
	surprise	fear	angry	neutral	sad	disgust	happy
train	3171	4097	3995	4965	4830	436	7215
	surprise	fear	angry	neutral	sad	disgust	happy
test	831	1024	958	1233	1247	111	1774



IMPLEMENTATION

CNN Model

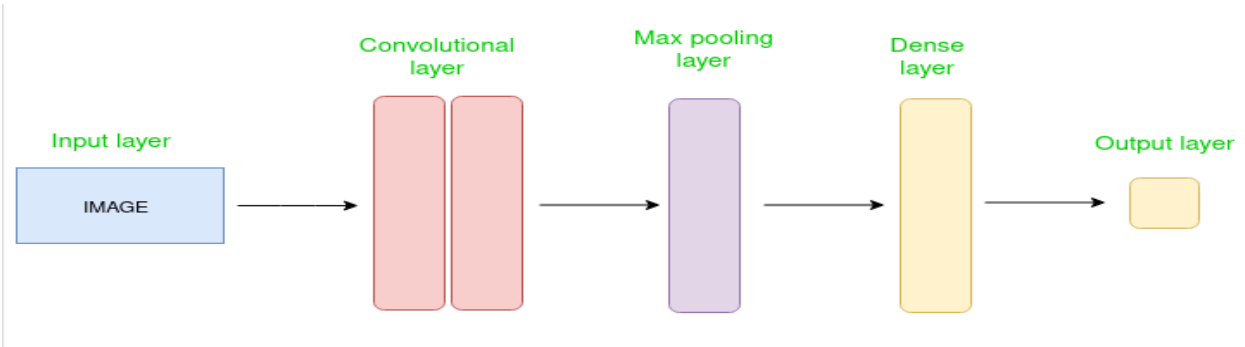
The CNN designed is based on sequential model and is designed to have six activation layers, of which 4 are convolutional layers and the remaining 2 are fully controlled layers.



Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.



MODEL SUMMARY

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	1088
activation (Activation)	(None, 48, 48, 64)	0
batch_normalization (BatchNo	(None, 48, 48, 64)	256
conv2d_1 (Conv2D)	(None, 48, 48, 64)	65600
activation_1 (Activation)	(None, 48, 48, 64)	0
batch_normalization_1 (Batch	(None, 48, 48, 64)	256
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 128)	131200
activation_2 (Activation)	(None, 24, 24, 128)	0
batch_normalization_2 (Batch	(None, 24, 24, 128)	512
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	262272
activation_3 (Activation)	(None, 12, 12, 128)	0
batch_normalization_3 (Batch	(None, 12, 12, 128)	512

```
conv2d_4 (Conv2D)            (None, 12, 12, 128)       262272
_________________________________________________________________
activation_4 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
activation_5 (Activation)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 1,315,335
Trainable params: 1,314,311
Non-trainable params: 1,024
_________________________________________________________________
```

Model Architecture Image:

https://drive.google.com/file/d/1R2mMVC5wa_CpnWIh8_Uurw-Ns0Nq0pZs/view?usp=sharing

Model Hyper Parameters

Number of epochs: 200

Activation function used: elu

Padding type: ‘same’ means padding is of same size as of input size.

Max pooling size: 2x2

Loss: categorical_crossentropy

Optimizer used: Adam

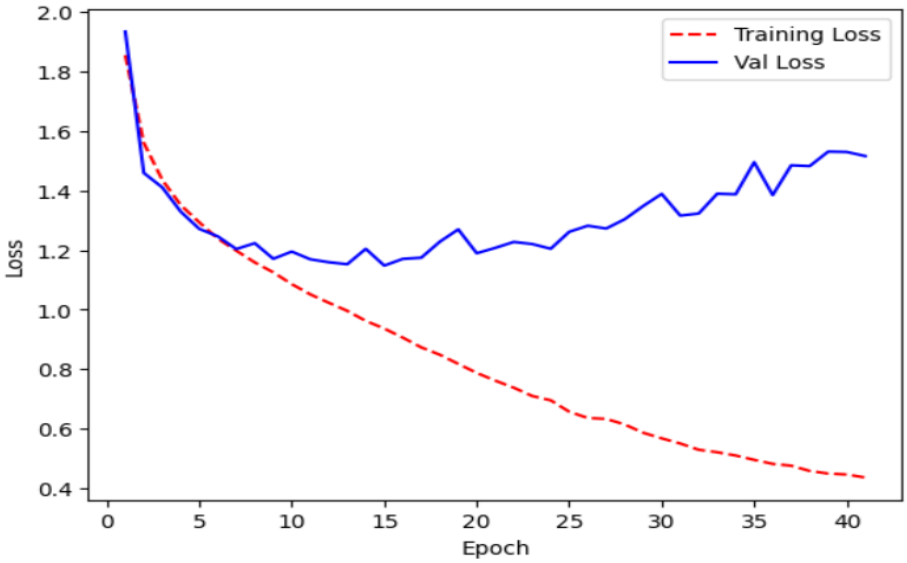
Model Training Result

Model early stopping during training: 41st epochs

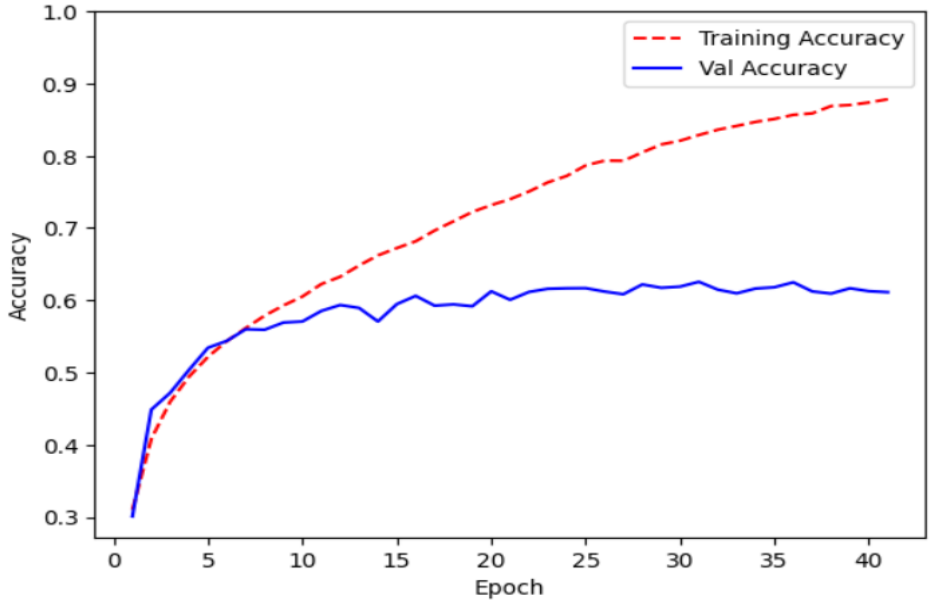
Validation accuracy: 62.55

Test accuracy: 62.48

Model Performance Curve

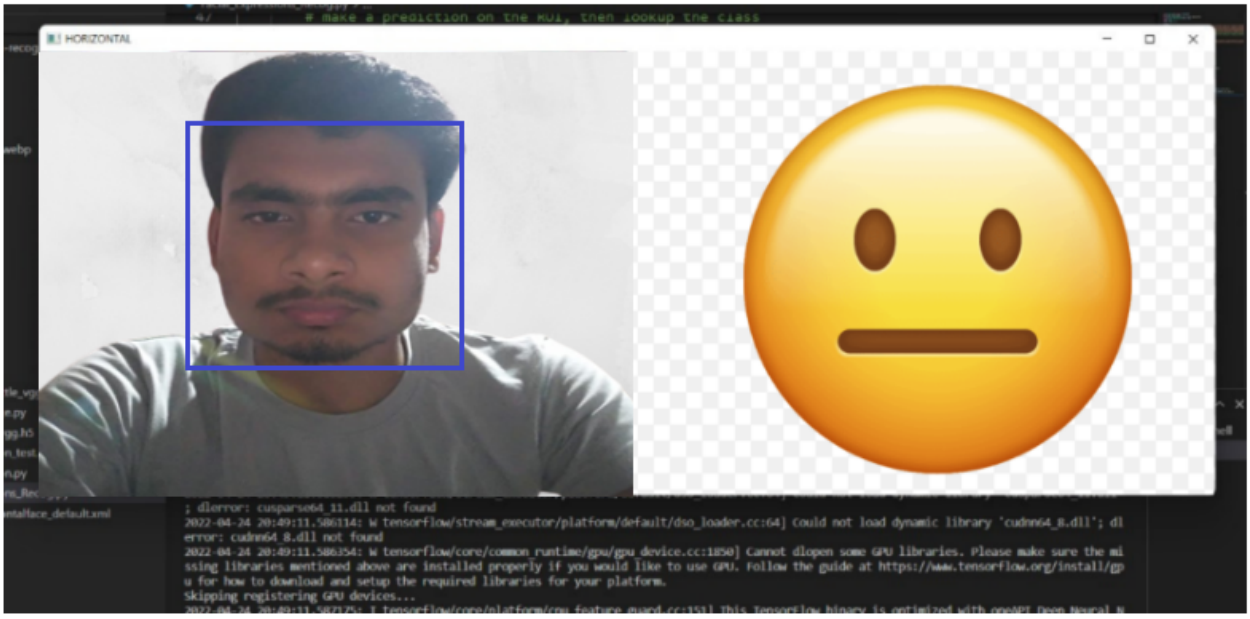


Loss Curve

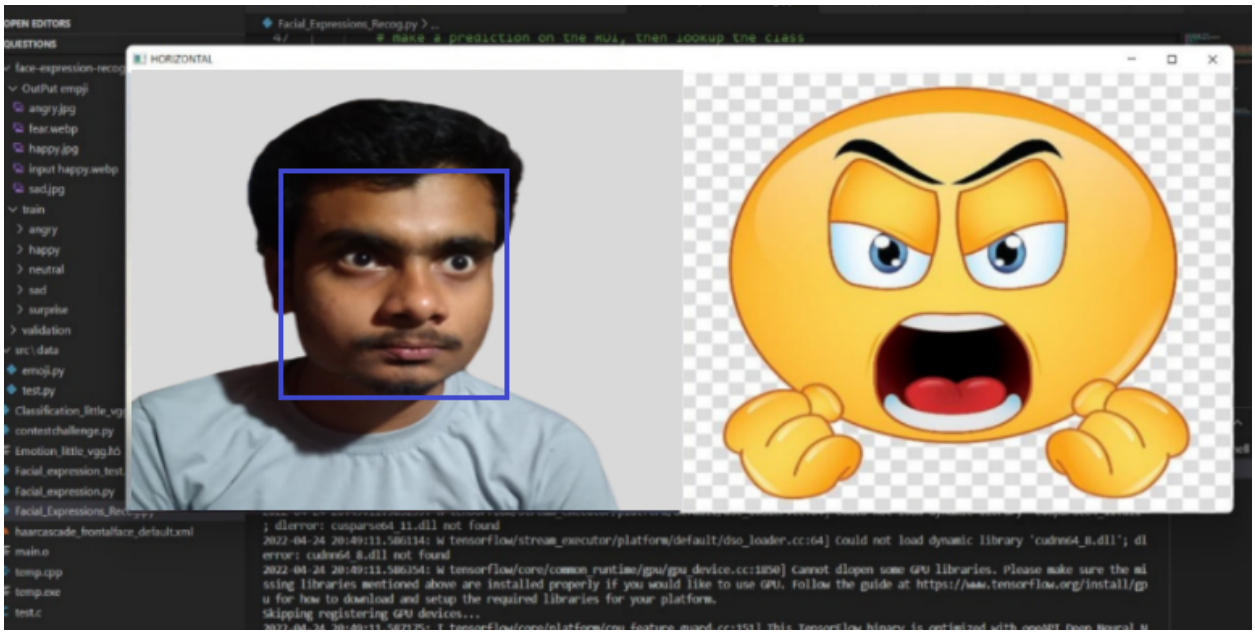


Accuracy Curve

OUTPUT



Neutral



Anger

CONCLUSION

Using the FER-2013 dataset, a test accuracy of 56% and train accuracy is 62.939% is attained with this designed CNN model. The achieved results are satisfactory as the average accuracies on the FER-2013 dataset is 65% +/- 5% and therefore, this CNN model is nearly accurate. For an improvement in this project and its outcomes, it is recommended to add new parameters wherever useful in the CNN model and remove unwanted and not-so useful parameters. Adjusting the learning rate and adapting with the location might help in improving the model. Accommodating the system to adapt to a low graded illumination setup and nullify noises in the image can also add onto the efforts to develop the CNN model. Increasing the layers in the CNN model might not deviate from the achieved accuracy, but the number of epochs can be set to higher number, to attain a higher accurate output. Though, increasing the number of epochs to a certain limit, will increase the accuracy, but increasing the number of epochs to a higher value will result in over- fitting.

FUTURE WORK

1. **At this level, we have included only 5 emojis to mask our facial expressions, but our future work would include far more emojis so that we can map emojis as per the expressions accurately.**
2. **To improve the accuracy of our model. In order to match it to the average accuracy of FER dataset.**
3. **To integrate this model with the website and provide option to the user to click or upload the photo as well as video to generate the emoji based on the facial expressions.**