VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : **B.Tech CSE with Spec. in AI & ML**

Course Title : **Machine Learning Essentials**

Course Code : **CSE1015**

Slot : **A1**

Title: **Facial Emotion Detection, Depression Analysis using CNN and ResNet**

Team Members:

Siddharth Shankar Das | 20BAI1125

Rakesh Kumar KS | 20BAI1055

Prithvi MR | 20BAI1146

Faculty: Dr. R. Rajalakshmi

Sign:

Date: 29.04.2022

CSE1015 – Machine Learning Essentials

J Component Report

Facial Emotion Detection, Depression Analysis using CNN ResNet.

J Title

By

20BAI1125	Siddharth Shankar Das
20BAI1055	Rakesh Kumar K S
20BAI1146	Prithvi M R

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGENEERING w/ SPEC. IN AI& ML

Submitted to

Dr. R. Rajalakshmi

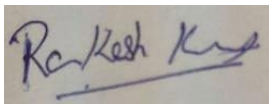
School of Computer Science and Engineering

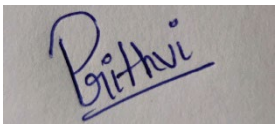
VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION BY THE CANDIDATE

I hereby declare that the report titled "**Facial Emotion Detection using CNN, KNN and detecting Depression"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr. R. Rajalakshmi, Associate Professor, SCOPE, Vellore Institute of Technology, Chennai.**





Signature of the Candidate(s)

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Rajalakshmi,** School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Ganesan, Dean,** School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Head of the Department** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Facial Emotion Detection using CNN"** is a bona-fide work of **Rakesh Kumar K S (20BAI1055), Siddharth Shankar Das (20BAI1125) , Prithvi M R (20BAI1146)** carried out the "J"-Project work under my supervision and guidance for CSE1015 – Machine Learning Essentials.

Dr. R. Rajalakshmi

SCOPE

TABLE OF CONTENTS

ABSTRACT

Facial emotion recognition (FER) is an important topic in the fields of computer vision and artificial intelligence owing to its significant academic and commercial potential. This project aims to build a meaningful Machine Learning model that can classify a given image into a fixed range of emotions such as neutral, angry, happy, sad, etc. The Machine Learning algorithm that'll be used here is Convolutional Neural Network or CNN for short. It mimics the human brain(neurons to be specific) and has multiple layers from input to output that can learn different aspects of a given input. The goal is to maximize the accuracy achieved at the end of training. Such a model can be used for various real life problems such as assisting blind people during social interactions, during therapy sessions, etc.

This documentation gives a brief about the project and our work on it. This report also assumes that you're familiar with the fundamentals of CNN, perceptron/artificial neurons.

Using Added data, and ResNet we were able to get 76%+ accuracy which was 105 more than the research paper result.

INTRODUCTION

Facial expressions can be used as a vital identifier for what a person is feeling at the moment. Most of the times, the emotion/expression that people show on their face is a direct non-verbal way of expressing the emotion they feel. There are various algorithms that can be used to arrive at a solution for this problem, like ANNs and its variations, CNN etc. We're not gonna work with ANN as the computational complexity of ANN is higher than CNN so it won't be feasible to train them easily when the dimension of the dataset is larger. To add more, CNNs are better with pattern recognition within images compared to ANNs and our input is an image in which we have to detect patterns. CNNs also have comparatively less number of parameters to train(like weights) and lesser number of parameters means lesser will be the overfitting. So it is important to take such precautions to make sure our model performs good in a real world scenario where it would be getting never-before-seen inputs. The first layer would take the input and convolve it using matrix dot product. Such a matrix is called a filter. The next layer pools the output of this layer and so on, till the last layer, which is the output layer, where the loss function associated with each class/category of output is calculated. We usually select the class with the highest probability as the predicted output class

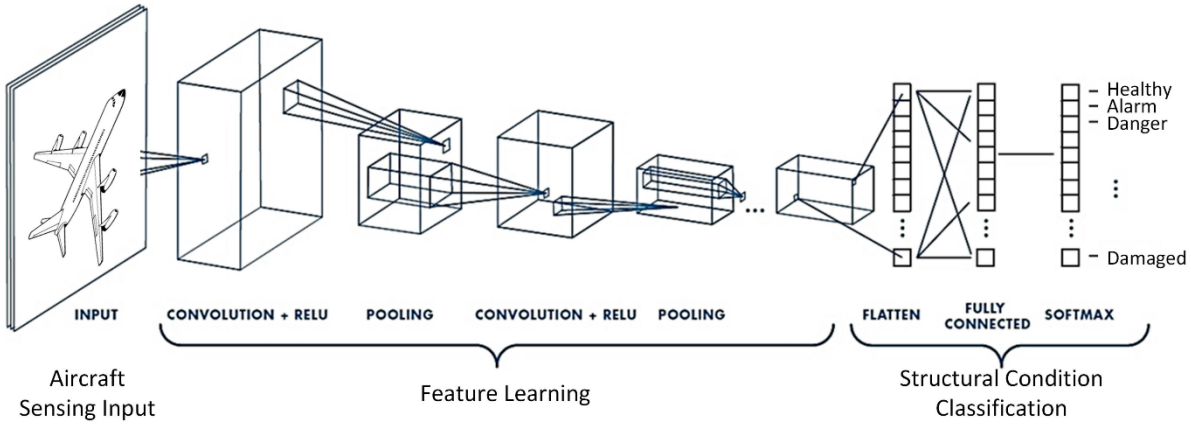


Fig. 1: An example Convolutional Neural Network

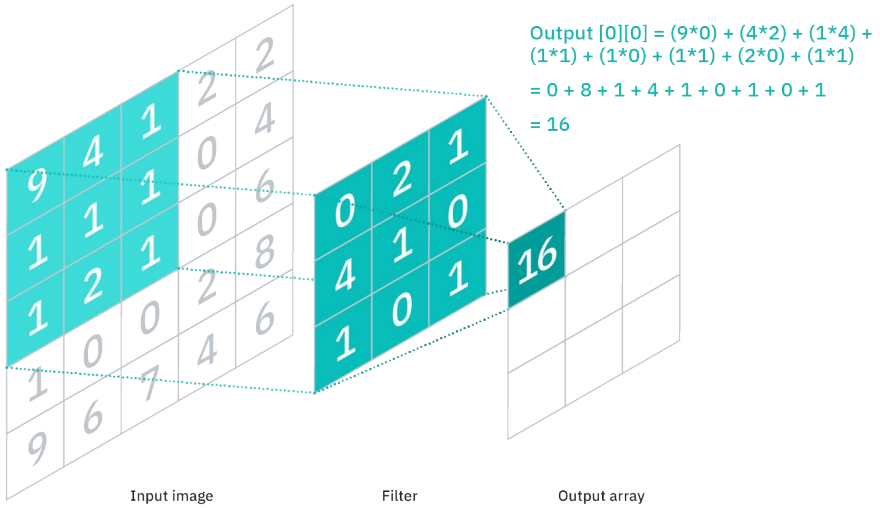


Fig. 2: How a filter/kernel is applied to the input matrix to get the output matrix

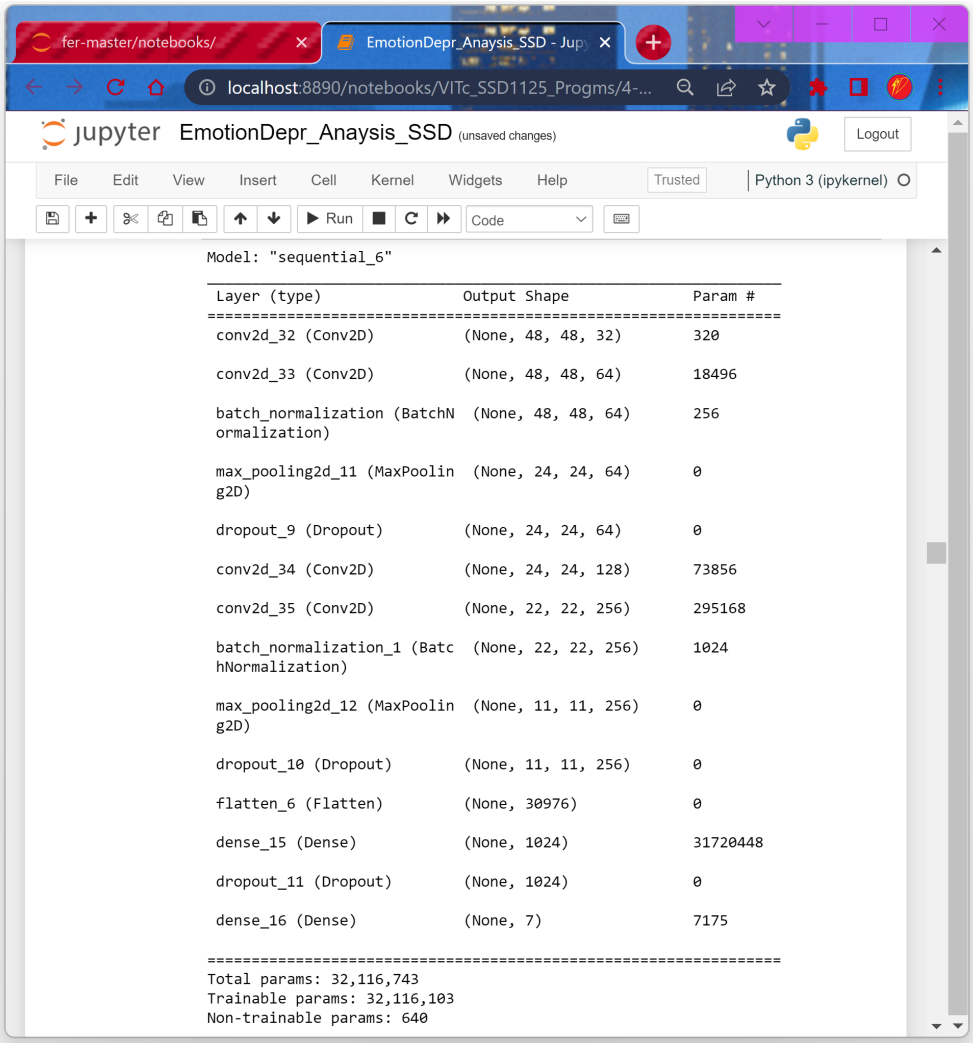


Fig. 3: Summary of One of Our initial Model

LITERATURE REVIEW

We have reviewed 3 research papers each. Here's the summary of it:

Papers

Rakesh Kumar K S:

Source: https://bit.ly/3s5Uhgy

The paper talks about how using the eyebrows and mouth as an anchor point in grey scaled images of faces is helpful for detecting emotions such as Happy, Sad, Surprised, Angry with 80% accuracy. This technique of the extracting intransient facial feature does not require any manual intervention like manual assignment of feature points to detect the various features of face and use that to read emotions

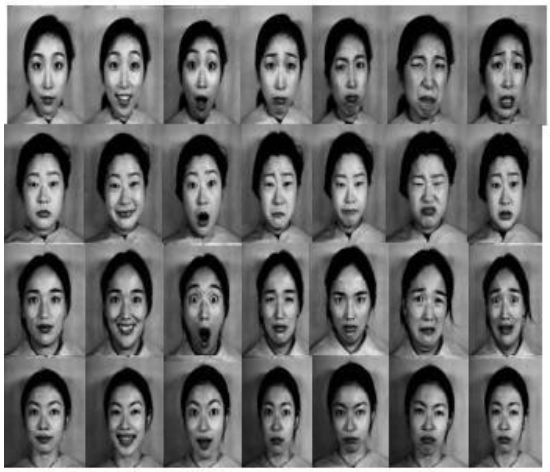


Figure 7: Some of the images from JAFFE database

Source: https://bit.ly/3H7WYlM

The paper talks about how the perceived waiting time of a customer in a fast food restaurant. The survey was on Malaysian restaurants but it's generally the same over everywhere. Perceived waiting time depends upon how long the customer waits for his/her food to arrive after ordering, is he alone or has some company, whether he is free or is pre-occupied with other works such as chatting or speaking over a call, etc. The research arrived at a conclusion that surprisingly the perceived waiting time has a positive correlation with customer satisfaction as many people are willing to wait for good food, though it is inconsistent. But waiting time above a point has led to dissatisfaction. We can predict this dissatisfaction and prevent the customer from getting angry or leaving the restaurant without getting the food they ordered by using our proposed model. We can prioritize the orders of such customers so that the identified problem can be addressed using the proposed model

Table 4.14: Correlation between waiting time and customer's satisfaction

		Waiting Times	Customer Satisfaction
Waiting Time	Pearson Correlation	1	.292**
	Sig. (2-tailed)		.000
	N	205	205
Customer Satisfaction	Pearson Correlation	.292**	1
	Sig. (2-tailed)	.000	
	N	205	205
**. Correlation is significant at the 0.01 level (2-tailed).			

Source: https://bit.ly/3s5RSCx

The paper has proposed a gloves, called VibroGlove to help blind or visually impaired people understand the emotion of other people. A blind or visually impaired person cannot judge the emotion or facial expression of another person that is in front of him. This might make it harder for them to have conversation with the other person or to work with them. The gloves can detect the other person's expression and use haptic interface(by making use of Vibrotactors) to convey that emotion to the blind or visually impaired person. This research paper is relevant to our topic as our proposed model can help detect other's emotion by seeing their face. We can make use of haptic or use speakers to convey the emotions of other person to the blind or visually impaired person.

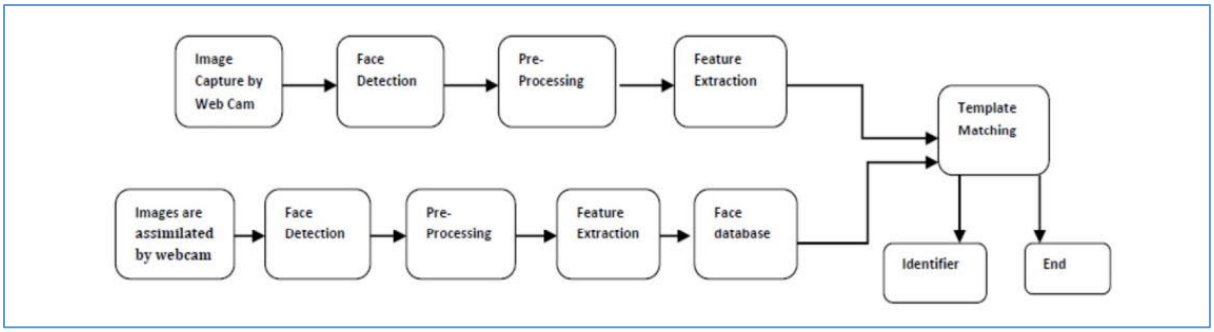
Siddharth Shankar Das:

Source: https://bit.ly/3LQoaJE

This thesis describes in depth how the identification of a real time face can be in order to create a safety alert framework for the workplace. The machine learning algorithm Haarcascade classifier was used to build the given four distinct classes for identification of security equipment and eventually identify the faces in both images and video using python open CV.

It also tells about the Face recognition history + relation with ML, as Face detection is the initial and important of face recognition. In face recognition process, after detection of face or image, the basic question that arises is to "whom this face or image belong". The facial recognition process solves this question by evaluating through four stages like 1.detection 2.feature extraction 3.tracking and 4.recognition.
They have used the Given Face Recognition process, to successfully recognize faces.
Many experiments have been wiped out of the field of face recognition, but the accuracy rate is smaller compared to other person biometric details such as fingerprints, expression, eyes, palm geometry, retina etc.

Many experiments have been wiped out of the field of face recognition, but the accuracy rate is smaller compared to other person biometric details such as fingerprints, expression, eyes, palm geometry, retina etc



This Research paper has used and Described the KNN algorithm, stating, One of the basic classification algorithms in machine learning is known to be the k-NN algorithm. In machine learning, the k-NN algorithm is considered a well monitored type of learning. It is commonly used in the sorting of related elements in searching apps. From this we also came to know the importance of this Classification algorithm.

We also came to know about the Pre-Processing techniques used. Elbow for classifier: The Elbow curve is created to train the KNN module for different N-neighbour values and finally pick a value that gives us an error. It also stated how they have used OpenCV with Face recognition, Along with KNN, OpenCV, we also came to know about the Regression, KNN Result, also about How they have used, HaarCascade Classifier, stating: This is a technique focused on machine learning, during which a course work is prepared from a broad measure of positive and negative images. It's normal to identify issues in various images. It's a pretrained facial data model and it's popular to identify faces.

Using these Machine Learning techniques, the Research Papers showed how they are able to detect faces, with great accuracy.

Facial recognition has numerous advantages in the public arena, including expanding well-being and security, obviation wrongdoings, and diminishing human cooperation. Recognition has been an ordinary piece of Airport security evaluating for a long time, recognizing felons and possible dangers to carriers and travellers. Facial acknowledgment can likewise tag photographs in your apportioned storage through Apple, Tesla, or Google. And with the help of this Research Paper, we'll(we're) be able to implement it using Machine Learning techniques.

Source: https://bit.ly/3s3u9CJ

This paper explores the ways, including text and emotion history, to detect Depression with the help of Artificial Intelligence, and Machine Learning.

Depression is a leading cause of mental ill health, which has been found to increase risk of early death. Moreover, it is a prima cause of suicidal thought process and leads to significant scathe in daily life. Emotion artificial intelligence is a field of ongoing research in emotion detection, specifically in the field of text mining. With the help of this Research Paper, we can be able to prevent.

At first, they have given us the symptoms and social problem to be solved, which is Depression and its further impact, including-> Depressive Mood, Loss on interest in activities,

Suicidal thoughts, feeling of worthlessness or hopelessness, worsened ability to think and concentrate etc.

The authors of this article have really given a great solution for this current social issue, and due to Covid- 19+Quarantine/Isolations, it's increasing at a very fast rate, and we all need an ML solution for this, which will also be one of our Project problems to be solved.

The structure of the Research Paper is: Description of the classifiers used in the implementation, methodology of the research paper, with explanation and discussion, which states how well this research paper has been done.

They have used dataset of about 10,000 Tweets and they have used the ratio of 80:20 split of training and test dataset.

Source: https://bit.ly/3GYqjiA

This Research Paper helps tackle a current social problem, which can be used to tackle many futuristic issues.

Images and videos have become omnipresent on the internet, which has encouraged the development of algorithms that can analyse their semantic content for various applications, including search and summarization.

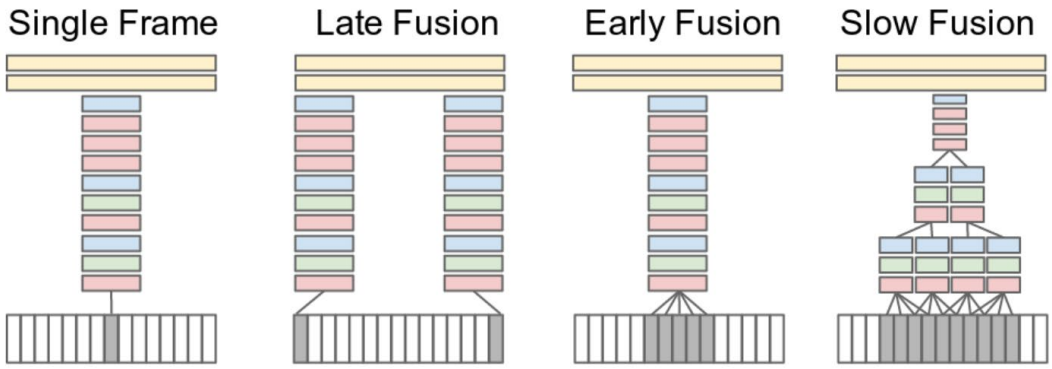
The Research Paper also teaches us the Importance of CNN(Convolutional Neural Networks, which has been established as a powerful class of models for image recognition problems.

They have provided empirical evaluation of CNNs on large-scale video classification using a new dataset of approx. 1 million YouTube videos belonging to 487 classed. They have studied multiple approaches for extending the connectivity of a CNN in time domain to take advantage of local spatio-temporal information and suggested a multiresolution, foveated architecture as a promised way of speeding up the training.

From a practical point of view, there were no video classification benchmarks that matches the scale and variety of existing image datasets because videos are significantly more difficult to collect, annotate and store. Thanks to this research paper, we are able to classify it.

They have used multiple approaches for extending CNNs into video classification on a large-scale dataset of 1 million videos with 487 categories (which we release as Sports-1M dataset) and reported significant gains in performance over strong feature-based baselines, and highlighted it.

Unlike Images, video classification is far more complex as videos vary widely in temporal extent and cannot be easily processed with a fixed-sized architecture., and they have successfully done it. Using this below approach



They have also infused Time information using CNNs giving priority to each frames along with Early, Late and slow Fusions which combined to become a Multi-resolution CNNs. The Sports-1M dataset consists of 1 million YouTube videos annotated with 487 classes. And the research paper studied the performance of all of it, with the created CNN model

Studying this Research paper was also really helpful for our Project, as Large Video classification is somewhat a prat of Real time analysis, and which can be used to classify according to our conditions(Emotion)

Prithvi M R:

Source: https://b.gatech.edu/3rAxb0P

This paper deals with emotion detection and sentiment analysis of images. Recognizing that analysing content from social media websites and/or photo-sharing websites like Flickr, Twitter, Tumblr, etc., can give insights into the general sentiment of people about any topic we wish like an upcoming election or a concert or just about anything. Also, it would be useful to understand the emotion an image depicts to automatically predict emotional tags on them - like happiness, fear, etc. They aim to predict the emotional category of an image into 5 distinct categories which are Love, Happiness, Violence, Fear, and Sadness. They try to achieve this by fine-tuning 3 different convolutional neural networks for the tasks of emotion prediction and sentiment analysis. They have collected data from Flickr and have experimented with various techniques and classification methods like SVM on high level features of VGG-ImageNet, fine-tuning on pretrained models like RESNET, Places205-VGG16 and VGGImageNet.

Here are some of the experiments performed - using a pretrained neural network to get the feature representation of their dataset and then use this representation as an input to SVM and classify the data using one vs all SVM classifiers, fine tuning the SVM classifiers, performing sentiment analysis on their data to understand their results better, fine tuning a model which was pretrained on a scene-based database called Places205, fine tuning with a pretrained ResNet50 etc.

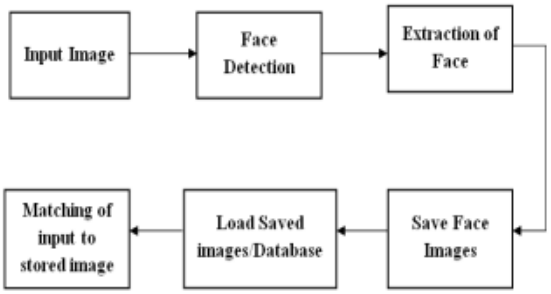
In all of the fine-tuning experiments, it has been observed that standard data augmentation techniques like mirroring and random cropping of images increase the accuracy. Having a higher learning rate for the later layers also yields a better accuracy. Finally, their results show that deep learning does provide promising results with a performance comparable

to some methods using handcrafted features on emotion classification task, and also a few methods using deep learning for sentiment analysis.

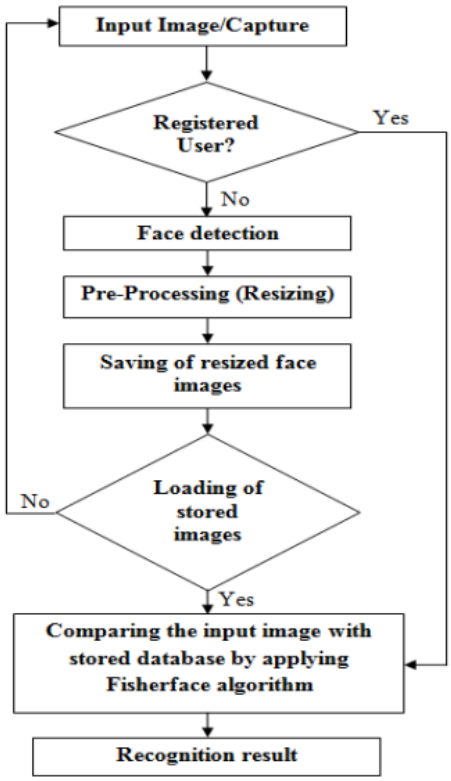
Source: https://bit.ly/3uYY40k

This paper deals with a method for facial recognition and emotion detection using support vector machines. Here, in order to deal with the dimensionality of the images/ frames which play an important role in ML problems, they have used principal component analysis (PCA) in order to reduce the dimensionality of the images (converting high dimensional space to low dimensional space) and linear discriminant analysis (LDA) which computes the group of characteristic features that normalizes the different classes if image data for classification.

The flowchart for facial recognition as used in this research is given below:



The input image is subjected for face detection to detect the face. The detected faces are then extracted from the image and these images are saved as a database. Saved images are used to compare with the input image. The matching of input image is performed to identify the user's identity. The recognition result gives identification of the person. The step-by-step architecture used is also given below:



Facial features such as eyes, nose, lips and face contour are considered as the action units of face and are responsible for creation of expressions on face, are extracted using open-source software called dlib. SVM classifier compares the features of training data and testing data to predict any emotion of the face. Multi-SVM classifier is used for classification of different emotions. Finally, it is also observed that accuracy of both face recognition and emotion detection can be increased by increasing the number of images during training. The detection time is significantly less and hence the system yields less run-time along with high accuracy

Source: https://bit.ly/3EERJus

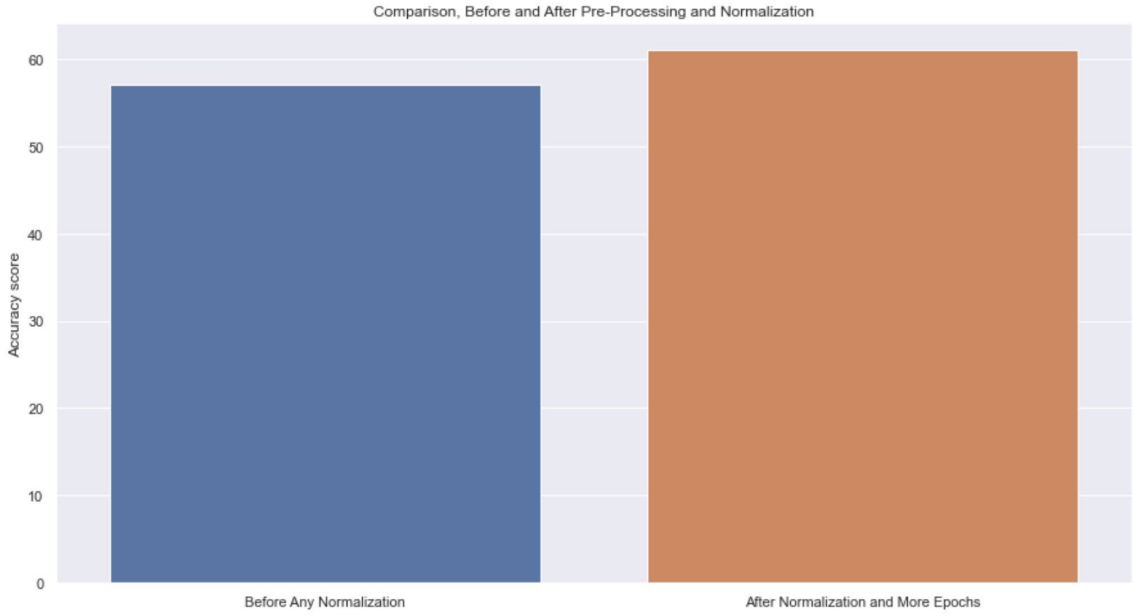
This paper deals with face detection and recognition of natural human emotions using Markov random fields. In the first part, they implement skin detection using Markov random fields models for image segmentation and skin detection where a set of several colored images with human faces have been considered as the training set. It detects the skin color using MRFs estimation in YCbCr space. The second part includes eye and mouth detection and extraction. It uses the HLV color space of the specified eye and mouth region. The third part detects the emotions pictured in the eyes and mouth, using edge detection and measuring the gradient of eyes' and mouth's region figure.

The proposed face detection algorithm contains two major modules: firstly, face localization for finding face candidates and secondly facial feature detection for verifying detected face candidates. Here, the algorithm first detects skin regions that possibly contain a human face. The skin detection algorithm is a segmentation technique that first transforms the image from the RGB to YCbCr color space and the skin detection algorithm is based on statistical image processing model using Bayesian estimation.

The proposed algorithm was evaluated on several image databases, the color images used for assessment have been taken under varying lighting conditions and with complex backgrounds, images contain multiple faces with variations in position, scale, orientation, and facial expression and the average size of each image is 400 X 500 pixels. The final experimental findings s proved that the algorithm can detect multiple faces of different sizes with a wide range of facial variations in an image. The proposed system does not depend on the orientation of the face, and half profile face views can be detected as long as one eye and mouth are visible, or at least part of them and faces can also be detected in the presence of facial hair.

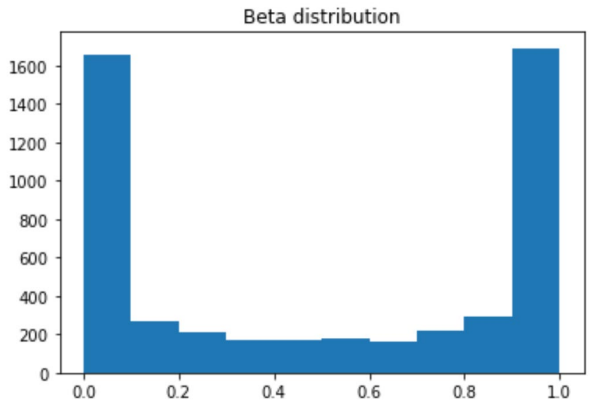
Proposed METHODOLOGY

Initially just like the research articles, we also Applied the basic Deep learning techniques and CNN to get an initial comparable model. Then after comparing it with the previous one, it came out to be same.



Then we did few Pre-Processing techniques, along with the Batch Normalization. And were able to increase the accuracy upto 5%. i.e. from 57 to approx. 62%.

As the dataset originally is in Image format, So for CNN it required a lot of computational power, so we used the pre-saved csv formal. But since we were instructed to also use it in the original Format, so using MixUp argument we used the original Dataset.



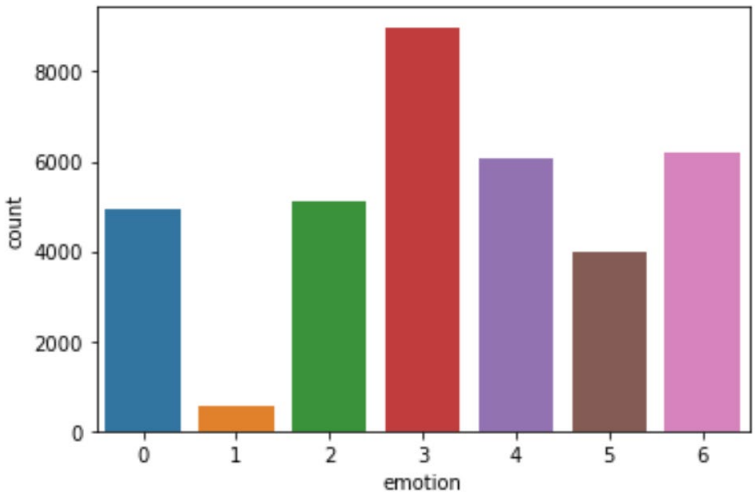

```
Epoch 1/5
718/718 [==============================] - 621s 865ms/step - loss: 0.3584 -
acc: 0.3703 - val_loss: 0.3436 - val_acc: 0.3932
Epoch 2/5
718/718 [==============================] - 639s 891ms/step - loss: 0.3381 -
acc: 0.4389 - val_loss: 0.3466 - val_acc: 0.4076
Epoch 3/5
718/718 [==============================] - 625s 871ms/step - loss: 0.3323 -
acc: 0.4517 - val_loss: 0.3372 - val_acc: 0.3895
Epoch 4/5
718/718 [==============================] - 624s 869ms/step - loss: 0.3269 -
acc: 0.4634 - val_loss: 0.3150 - val_acc: 0.4700
Epoch 5/5
718/718 [==============================] - 637s 888ms/step - loss: 0.3247 -
acc: 0.4721 - val_loss: 0.3133 - val_acc: 0.4593
```

As the Image dataset is taking a lot of Computation power and time to do only 5 epocs, with an Accuracy of less than 50%, we'll now again use the .csv dataset

As the Image's epochs were taking a lot of time, so we again tried the csv file.

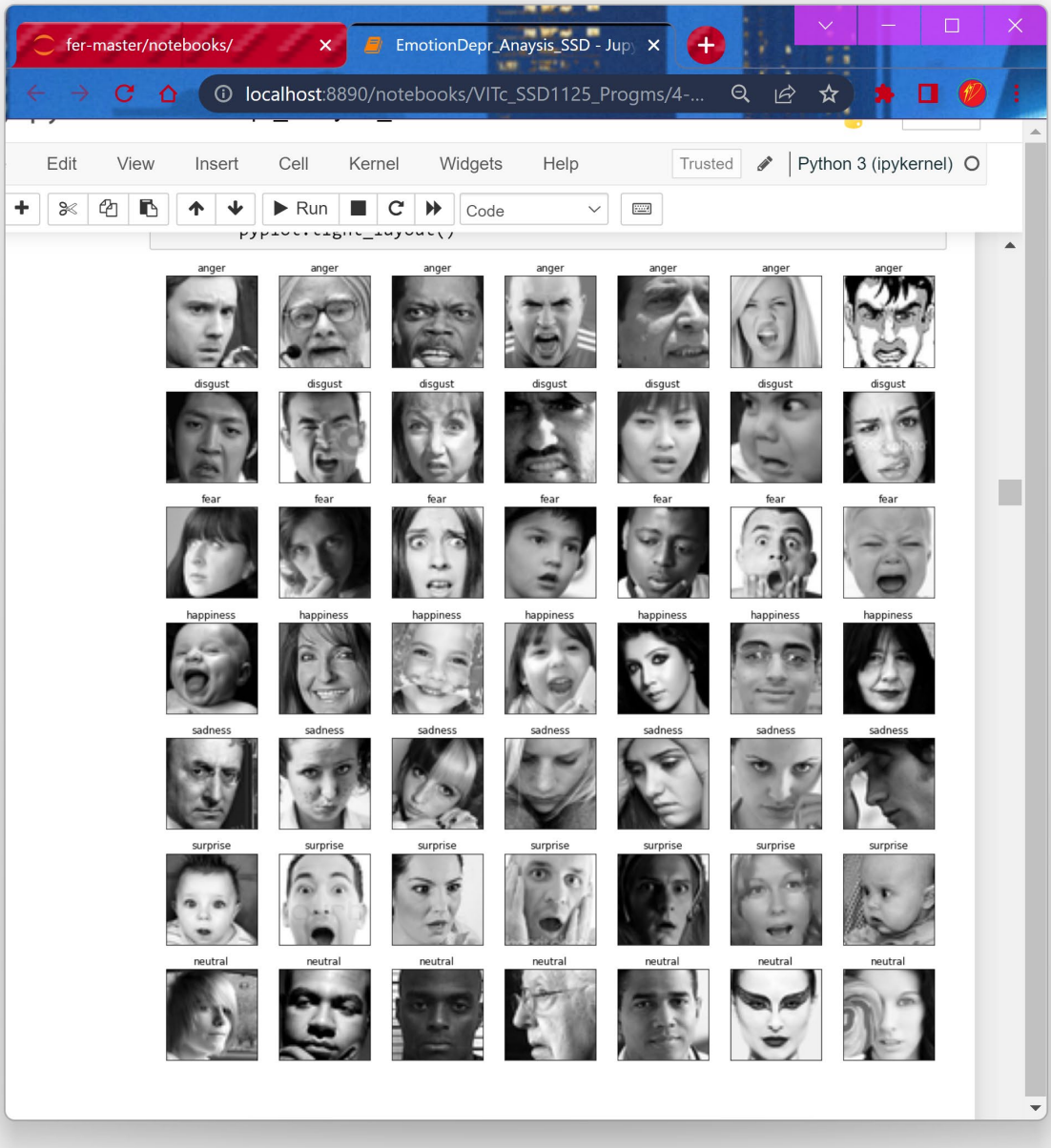
To increase the Accuracy more, we even added VGG16 layer, which was the original method with which the FER2013 Competition was won by the team. But still we were only able to increase it for few % more not upto our expectations.

So, then we increased the CNN Model's layers from 3 to approx. 5.



Another thing which we came to know about is that, in the Competition, many have Removed the data which is least available to increase their accuracy to even 80%. But that means we'll have one emotions less, so have not used this loophole to get accuracy to a better value, just by omitting it.

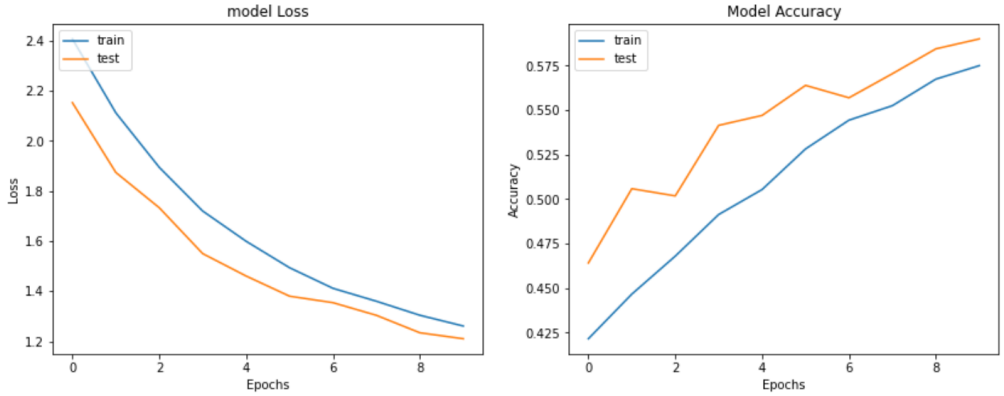
So, we have used all the 7 emotions in our Project.



We also used algorithms that avoids Overfitting, so that in the Epoch the train accuracy won't be too much as compared to the real accuracy. Added Deep CNN as well, and using ELU insead of 'RELU' because it avoids dying relu problem.

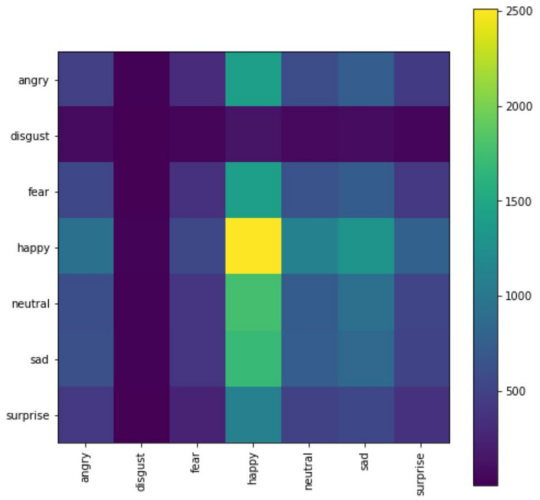
Then to Increase the Accuracy more, we also used the Kernel_Regularizer and to increase the efficiency, we exported it in one of the Most efficient file format for ML, .h5 and were able to get an accuracy of 67%. It can also be increased with more no. of Epochs.

And without this, the Accuracy and loss were like this:

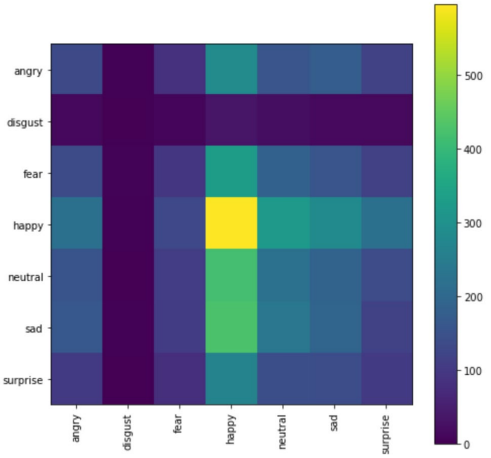


High accracy is achieved on training set but accuracy on validation set is stuck at 60-70%, also no overfitting can be seen in the dataset hence it can be concluded that the inefficiency may be due to the unbalanced dataset

```
Confusion Matrix
[[ 481   11  308 1420  598  746  431]
 [  76    1   31  137   67   82   42]
 [ 534   10  346 1420  637  734  416]
 [ 932   25  543 2510 1116 1299  790]
 [ 608   15  399 1770  733  921  519]
 [ 611   15  387 1699  755  857  506]
 [ 420    7  250 1096  495  535  368]]
Classification Report
              precision    recall  f1-score   support

       angry       0.13      0.12      0.13      3995
     disgust       0.01      0.00      0.00       436
        fear       0.15      0.08      0.11      4097
       happy       0.25      0.35      0.29      7215
     neutral       0.17      0.15      0.16      4965
         sad       0.17      0.18      0.17      4830
    surprise       0.12      0.12      0.12      3171

    accuracy                           0.18     28709
   macro avg       0.14      0.14      0.14     28709
weighted avg       0.17      0.18      0.18     28709
```




```
Confusion Matrix
[[133   4  85 286 157 176 117]
 [ 13   0  11  34  21  16  16]
 [136   6  96 328 187 156 115]
 [219   4 131 595 321 284 220]
 [155   1 109 418 221 190 139]
 [165   4 106 427 236 191 118]
 [100   1  81 266 142 140 101]]
Classification Report
              precision    recall  f1-score   support

       angry       0.14      0.14      0.14       958
     disgust       0.00      0.00      0.00       111
        fear       0.16      0.09      0.12      1024
       happy       0.25      0.34      0.29      1774
     neutral       0.17      0.18      0.18      1233
         sad       0.17      0.15      0.16      1247
    surprise       0.12      0.12      0.12       831

    accuracy                           0.19      7178
   macro avg       0.14      0.15      0.14      7178
weighted avg       0.18      0.19      0.18      7178
```



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_14 (Conv2D)          (None, 46, 46, 64)        640

 batch_normalization_8 (Batc  (None, 46, 46, 64)       256
 hNormalization)

 activation_6 (Activation)   (None, 46, 46, 64)        0

 conv2d_15 (Conv2D)          (None, 44, 44, 64)        36928

 batch_normalization_9 (Batc  (None, 44, 44, 64)       256
 hNormalization)

 activation_7 (Activation)   (None, 44, 44, 64)        0

 dropout_6 (Dropout)         (None, 44, 44, 64)        0

 conv2d_16 (Conv2D)          (None, 42, 42, 64)        36928

 conv2d_17 (Conv2D)          (None, 40, 40, 64)        36928

 max_pooling2d_4 (MaxPooling  (None, 20, 20, 64)       0
 2D)

 conv2d_18 (Conv2D)          (None, 18, 18, 128)       73856

 batch_normalization_10 (Bat  (None, 18, 18, 128)      512
 chNormalization)

 activation_8 (Activation)   (None, 18, 18, 128)       0

 conv2d_19 (Conv2D)          (None, 16, 16, 128)       147584

 batch_normalization_11 (Bat  (None, 16, 16, 128)      512
 chNormalization)

 activation_9 (Activation)   (None, 16, 16, 128)       0

 conv2d_20 (Conv2D)          (None, 14, 14, 128)       147584

 conv2d_21 (Conv2D)          (None, 12, 12, 128)       147584

 max_pooling2d_5 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 conv2d_22 (Conv2D)          (None, 4, 4, 256)         295168

 batch_normalization_12 (Bat  (None, 4, 4, 256)        1024
 chNormalization)

 activation_10 (Activation)  (None, 4, 4, 256)         0

 conv2d_23 (Conv2D)          (None, 2, 2, 256)         590080

 batch_normalization_13 (Bat  (None, 2, 2, 256)        1024
 chNormalization)

 activation_11 (Activation)  (None, 2, 2, 256)         0

 flatten_2 (Flatten)         (None, 1024)              0

 dense_5 (Dense)             (None, 1024)              1049600

 dropout_7 (Dropout)         (None, 1024)              0

 dense_6 (Dense)             (None, 1024)              1049600

 dropout_8 (Dropout)         (None, 1024)              0

 dense_7 (Dense)             (None, 7)                 7175

=================================================================
Total params: 3,623,239
Trainable params: 3,621,447
Non-trainable params: 1,792
```

Fig. 8: Model Summary

Future Scope:

Using VGGNet 'Very Deep Convolutional Networks for Large-Scale Image Recognition' and implementing Depression detection more efficiently

Comparison

With Our Initial Data:

Highlights of our Work:

1. **Increased Dataset size from 33k to 40k by combining CK+ dataset(link,) and fer2013**

> **OwnFer+CK dataset is the combination of 2 facial recognition dataset, Fer2013 and CK+, increasing the count by 8000 images**

In [92]:
```
row, col = 48, 48
classes = 7

def count_exp(path, set_):
    dict_ = {}
    for expression in os.listdir(path):
        dir_ = path +"/" +expression
        dict_[expression] = len(os.listdir(dir_))
    df = pd.DataFrame(dict_, index=[set_])
    return df
train_count = count_exp(train_dir, 'train')
test_count = count_exp(test_dir, 'test')
print(train_count)
print(test_count)
```

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

In [9]:
```
row, col = 48, 48
classes = 7

def count_exp(path, set_):
    dict_ = {}
    for expression in os.listdir(path):
        dir_ = path +"/" +expression
        dict_[expression] = len(os.listdir(dir_))
    df = pd.DataFrame(dict_, index=[set_])
    return df
train_count = count_exp(train_dir, 'train')
test_count = count_exp(test_dir, 'test')
print(train_count)
print(test_count)
#Our new added images dataset:
```

```
       anger  disgust  fear  happy  neutral   sad  surprise
train   4725      795  3454   9049     5072  5403      4226
      angry  disgust  fear  happy  neutral   sad  surprise
test    958      111  1024   1774     1233  1247       831
```

2. **We did batch Normalization and CNN+ResNet50(instead of CNN alone)**
3. **Architectural Change, they used 4 layers CNN, and we did 6 layers CNN along with 48 layers**
4. **They did from the compressed .csv data, and we did with Original dataset that has all the images.**

Initially after 20 epochs our accuracy was: 56%

```
449/449 [==============================] - 14s 32ms/step - loss: 0.9626 - accuracy: 0.6343 - val_loss: 1.1660 - val_accuracy: 0.5645
```

Out[28]:
```
<keras.callbacks.History at 0x23595c43220>
```

In [22]:
```
scores = model.evaluate(X_test, test_y, verbose=0)
```

In [23]:
```
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
accuracy: 57.04%
```

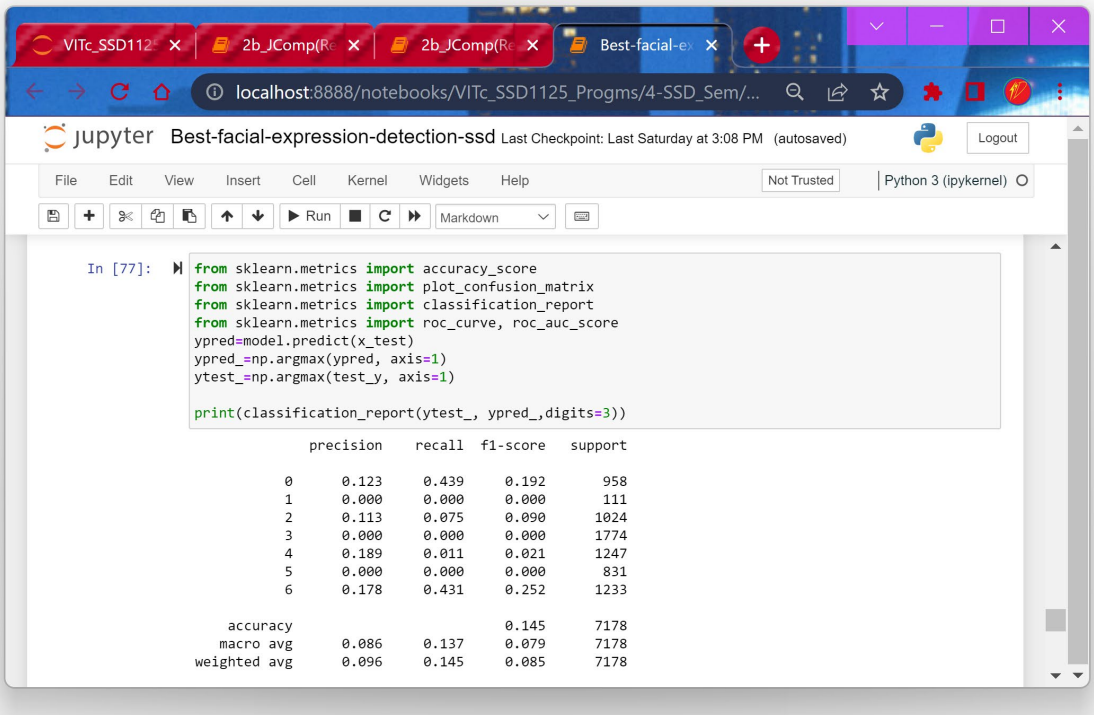
After doing the normalizations and adding CNN layers, different types of optimizers+Algorithms, we were able to achieve 66%. Which was somewhat same as the Research Paper's values.(
https://www.researchgate.net/publication/347959965_The_Facial_Emotion_Recognition_FER-2013_Dataset_for_Prediction_System_of_Micro-Expressions_Face_Using_the_Convolutional_Neural_Network_CNN_Algorithm_based_Raspberry_Pi
)

TABLE II: The Result of Facial Expression Testing

Testing in real-time	Corrected		Total	
	Yes	No	Test	(%)
Facial Expression	66	4	70	94.3
Face Position/Condition	159	86	245	64.9
Distance face versus Camera	109	66	175	62,3
From Image or Poster	104	141	245	42,4
Result Eksprement (mean)				65.97

After this, as we proved that the data had inconsistent values, so we combined 2 datasets, i.e., the FER2013 dataset along with CK+ dataset which contains 593 video sequences from a total of 123 different subjects, ranging from 18 to 50 years of age, again tried. The results weren't satisfiable, so after seeing the comparison, we came to know that if we add more CNN layers, then the accuracy can be increased, so we used ResNet50 model, with has 48 Convolution layers, and were able to achieve an Accuracy of 76%(after adding more epochs+data, we can even make it up to 79%, as overfitting wasn't happening).

```
number of Test examples = 10101
X_test shape: (10101, 48, 48, 1)
Y_test shape: (10101, 1)
316/316 [==============================] - 3s 8ms/step - loss: 1.4020 - accuracy: 0.7612
Sad
number of Test examples = 10101
X_test shape: (10101, 48, 48, 1)
Y_test shape: (10101, 1)
316/316 [==============================] - 3s 8ms/step - loss: 1.4020 - accuracy: 0.7612
Fear
number of Test examples = 10101
X_test shape: (10101, 48, 48, 1)
Y_test shape: (10101, 1)
316/316 [==============================] - 3s 8ms/step - loss: 1.4020 - accuracy: 0.7612
Surprise
```



Results and Discussion

The purpose of the project is to increase the accuracy of fer2013 dataset and creating a Real-Life usable model to be able to use in real life application. And then, storing the User's emotion history which can later be used for detecting depression/stress(Ex. Counting no. of times the model detected the user being sad, and if it crosses the limit, changing the smart home appliance's environment or for any specific patient notifying it to their known ones to prevent any mis-happenings, etc).

After applying the algorithms, we were able to compete with the Previous Models. What we came to know is this that using SVM was more efficient as compared to Softmax activation.

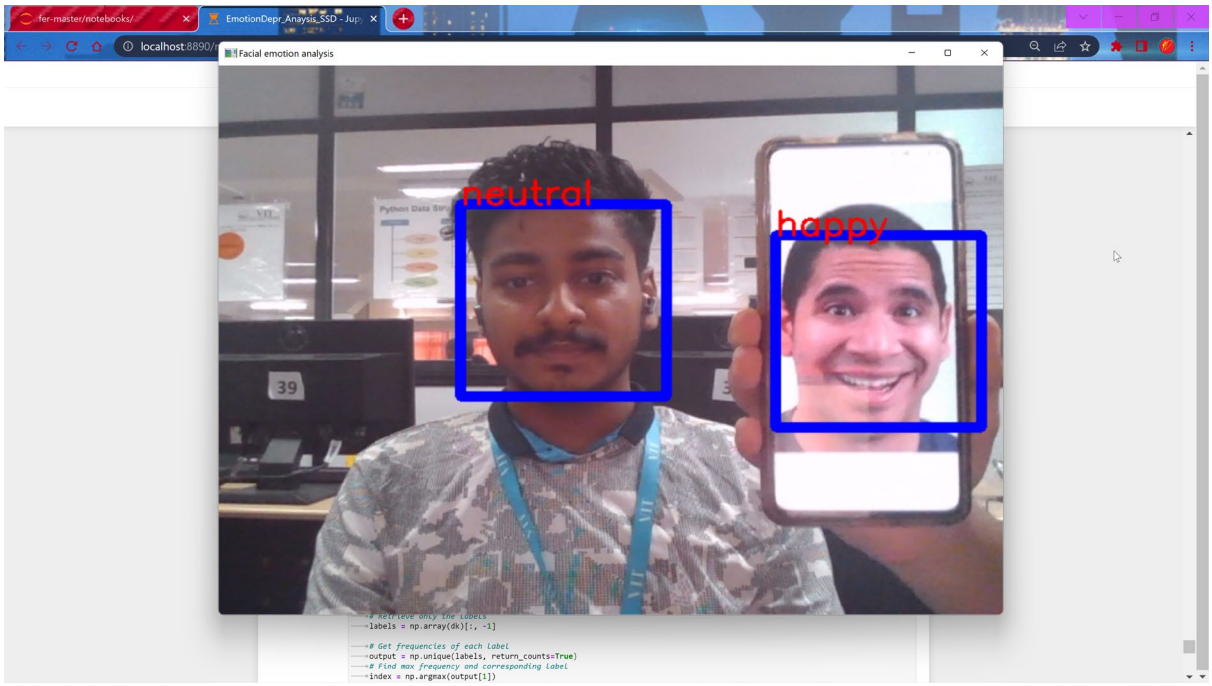
Also, in research papers, they only used Adam optimizer, but We've tried both `Nadam` and `Adam`, the difference in results is not that different but we chose Nadam as we've tried Adam optimizer before. And after adding more layers, we were able to make the model to be used in real life world application and be helpful in our Depression detection Future scope work.

We even used Torch, torchvision to increase the usablility

```
Epoch 39/40
316/315 [==========] - 18s 58ms/step - loss: 1.2062 - accuracy: 0.9592 - val_loss: 1.4150 - val_accuracy: 0.7475
Epoch 40/40
316/315 [==========] - 18s 58ms/step - loss: 1.2065 - accuracy: 0.9589 - val_loss: 1.4122 - val_accuracy: 0.7503
```

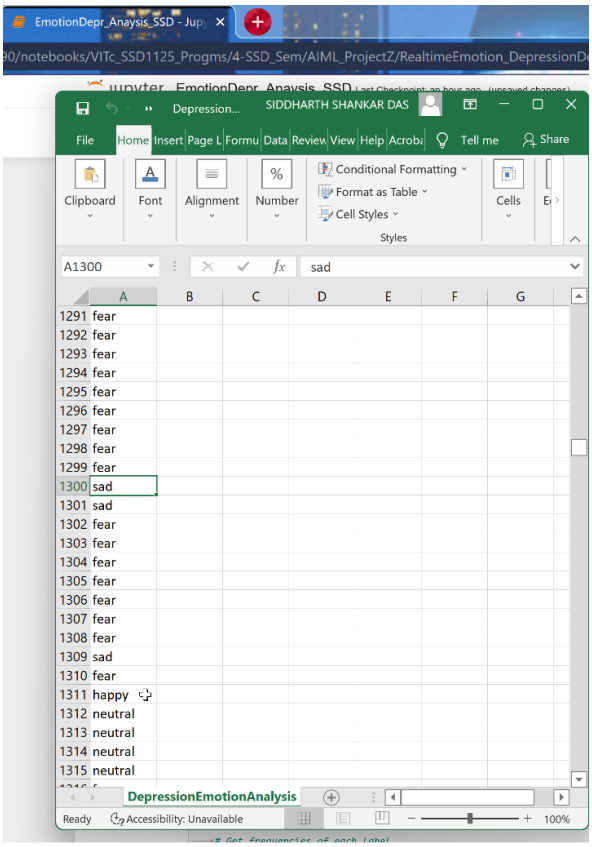
+ Code + Markdown

Implementation:




```
number of Test examples = 10101
X_test shape: (10101, 48, 48, 1)
Y_test shape: (10101, 1)
316/316 [==============================] - 2s 8ms/step - loss: 1.3965 - accuracy: 0.7678
Neutral
```

The Unique element of our project: We were able to get the real time Emotion of the User and save it in a dataset, which will be really helpful to learn about the User's Emotion history and can also be used to Cure Depression, know and warn about Stress, etc.



CONCLUSION

We were successfully able to implement the Emotion detection model with an accuracy closed to the best ever model created(~75+%), seeing the Model Accuracy Vs. Epoch graph, it can be concluded that with the increase in Epochs we can increase the accuracy to even more better model. So that along with the correct implementation of our Depression Detection will be our future scope. The dataset used is the fer2013 one, we also wanted to combine the dataset, but as the emotions labels weren't matching, we didn't did it.

The Algorithms used are, CNN, the with more Layers, then using the original images, we used MixUp argument, Kernel Regularizer and we tried SVM as well, but cause of Image Multi data we were not able to get our desired result. We also used the Very Deep CNN for the large dataset , and also, along with just the Adam optimizer, we also used the Nadam optimizer. Then in the application, we implement it using KNN as well.

Reference(s)

- https://www.kaggle.com/datasets/msambare/fer2013
- https://ieeexplore.ieee.org/abstract/document/9288560
- https://www.stevens.edu/news/detecting-depression-using-ai
- https://ieeexplore.ieee.org/document/8389299
- Remaining in the Literature Survey

APPENDIX
Implementation / Code

```
import os
import cv2
import numpy as np
from keras.models import model_from_json
from keras.preprocessing import image
from csv import writer

model = model_from_json(open("ML_FER2013.json", "r").read())
#load weights
model.load_weights('ML_FER2013.h5')

from keras.models import load_model
#model = load_model("Our_model.h5")

face_haar_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_defaul
t.xml')

import os
import cv2
import numpy as np
from keras.models import model_from_json
from keras.preprocessing import image
"""
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D,
BatchNormalization,AveragePooling2D
from keras.losses import categorical_crossentropy
from tensorflow.keras.optimizers import Adam
from keras.regularizers import l2"""
from csv import writer

#loading our created model
model = model_from_json(open("ML_FER2013.json", "r").read())
#load weights
model.load_weights('ML_FER2013.h5')

from keras.models import load_model
#For the real time face detection, we have used the HAARcascate algorithm
which uses edges to detect faces.
face_haar_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_defaul
t.xml')


#Opening the webcam, for video capturing
cap=cv2.VideoCapture(0)

while True:
    ret,test_img=cap.read()# captures frame and returns boolean value and
captured image
    if not ret:
        continue
    gray_img= cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)
    #Grayscaling the captured image for a seamless comparison
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)
```

```
    for (x,y,w,h) in faces_detected:
        cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=7)
        roi_gray=gray_img[y:y+w,x:x+h]#cropping region of interest i.e.
face area from  image
        roi_gray=cv2.resize(roi_gray,(48,48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis = 0)
        img_pixels /= 255

        predictions = model.predict(img_pixels)

        #find max indexed array
        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise',
'neutral')
        predicted_emotion = emotions[max_index]
        EmotnHistory=[predicted_emotion]

        cv2.putText(test_img, predicted_emotion, (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
        with open('DepressionEmotionAnalysis.csv', 'a', newline='') as
f_object:
            # Passing the CSV  file object to the writer() function
            writer_object = writer(f_object)

            #Adding the emotion history in the Excel file
            # Result - a writer object# Pass the data in the list as an
argument into the writerow() function
            writer_object.writerow(EmotnHistory)
            # Close the file object
            f_object.close()
    resized_img = cv2.resize(test_img, (1000, 700))
    cv2.imshow('Facial emotion analysis ',resized_img)



    if cv2.waitKey(10) == ord('q'):#wait until 'q' key is pressed
        break

cap.release()
cv2.destroyAllWindows
```

```
## For the Model:
import sys, os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D,
BatchNormalization,AveragePooling2D
from keras.losses import categorical_crossentropy
#from keras.optimizers import Adam
```

```
from tensorflow.keras.optimizers import Adam
from keras.regularizers import l2
from keras.utils import np_utils
# pd.set_option('display.max_rows', 500)
# pd.set_option('display.max_columns', 500)
# pd.set_option('display.width', 1000)

from sklearn.model_selection import train_test_split
df=pd.read_csv('fer2013.csv')

X_train,train_y,X_test,test_y=[],[],[],[]
x_val,y_val=[],[]

for index, row in df.iterrows():
    val=row['pixels'].split(" ")
    try:
        if 'Training' in row['Usage']:
           X_train.append(np.array(val,'float32'))
           train_y.append(row['emotion'])
        elif 'PublicTest' in row['Usage']:
           X_test.append(np.array(val,'float32'))
           test_y.append(row['emotion'])
        elif 'PrivateTest'in row['Usage']:
           X_test.append(np.array(val,'float32'))
           test_y.append(row['emotion'])
    except:
        print(f"error occured at index :{index} and row:{row}")

num_features = 64
num_labels = 7
batch_size = 64
epochs = 20
#Size of each image 48*48 pixels
width, height = 48, 48

X_train = np.array(X_train,'float32')
train_y = np.array(train_y,'float32')
X_test = np.array(X_test,'float32')
test_y = np.array(test_y,'float32')

train_y=np_utils.to_categorical(train_y, num_classes=num_labels)
test_y=np_utils.to_categorical(test_y, num_classes=num_labels)

#Normalization
#cannot produce
#normalizing data between oand 1
X_train -= np.mean(X_train, axis=0)
X_train /= np.std(X_train, axis=0)

X_test -= np.mean(X_test, axis=0)
X_test /= np.std(X_test, axis=0)

X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)

X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)

# print(f"shape:{X_train.shape}")
##designing the cnn
#1st convolution layer
model = Sequential()
```

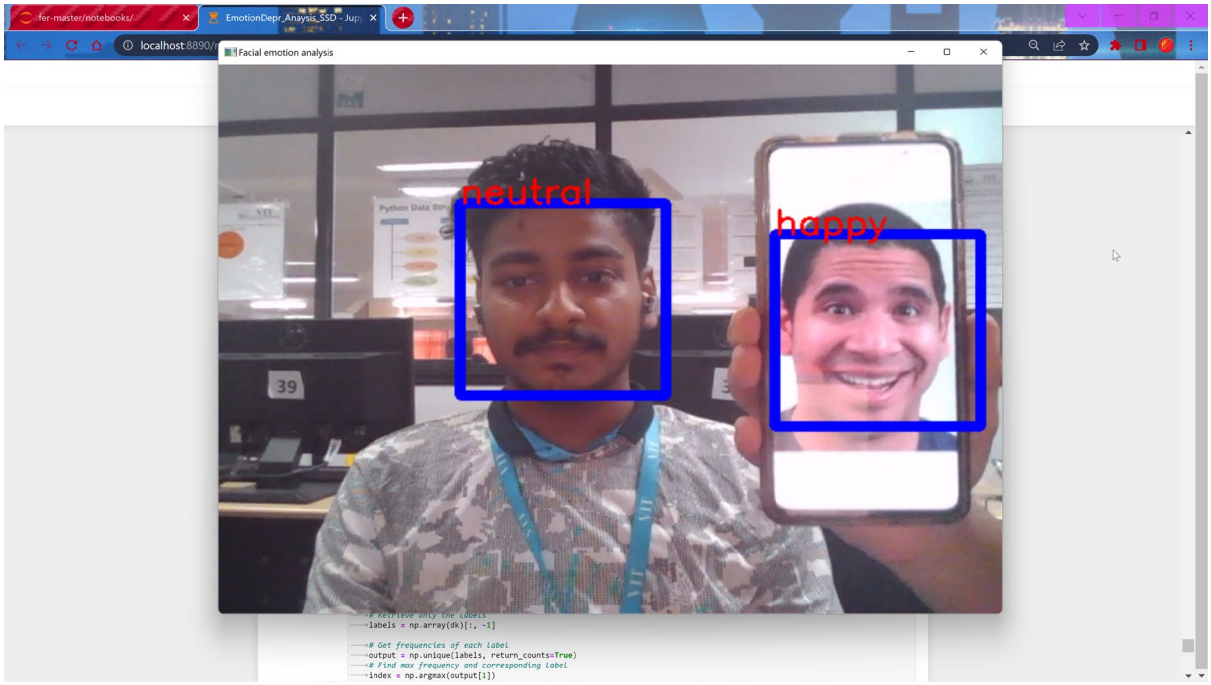
```
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',
input_shape=(X_train.shape[1:])))
model.add(Conv2D(64,kernel_size= (3, 3), activation='relu'))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))

model.add(Flatten())


#Increasing the layers
def define_model(input_shape=(48, 48, 1), classes=7):
    num_features = 64

    model = Sequential()

    # 1st stage
    model.add(Conv2D(num_features, kernel_size=(3, 3),
input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Dropout(0.5))

    # 2nd stage
    model.add(Conv2D(num_features, (3, 3), activation='relu'))
    model.add(Conv2D(num_features, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    # 3rd stage
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    # 4th stage
    model.add(Conv2D(2 * num_features, (3, 3), activation='relu'))
    model.add(Conv2D(2 * num_features, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    # 5th stage
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
```

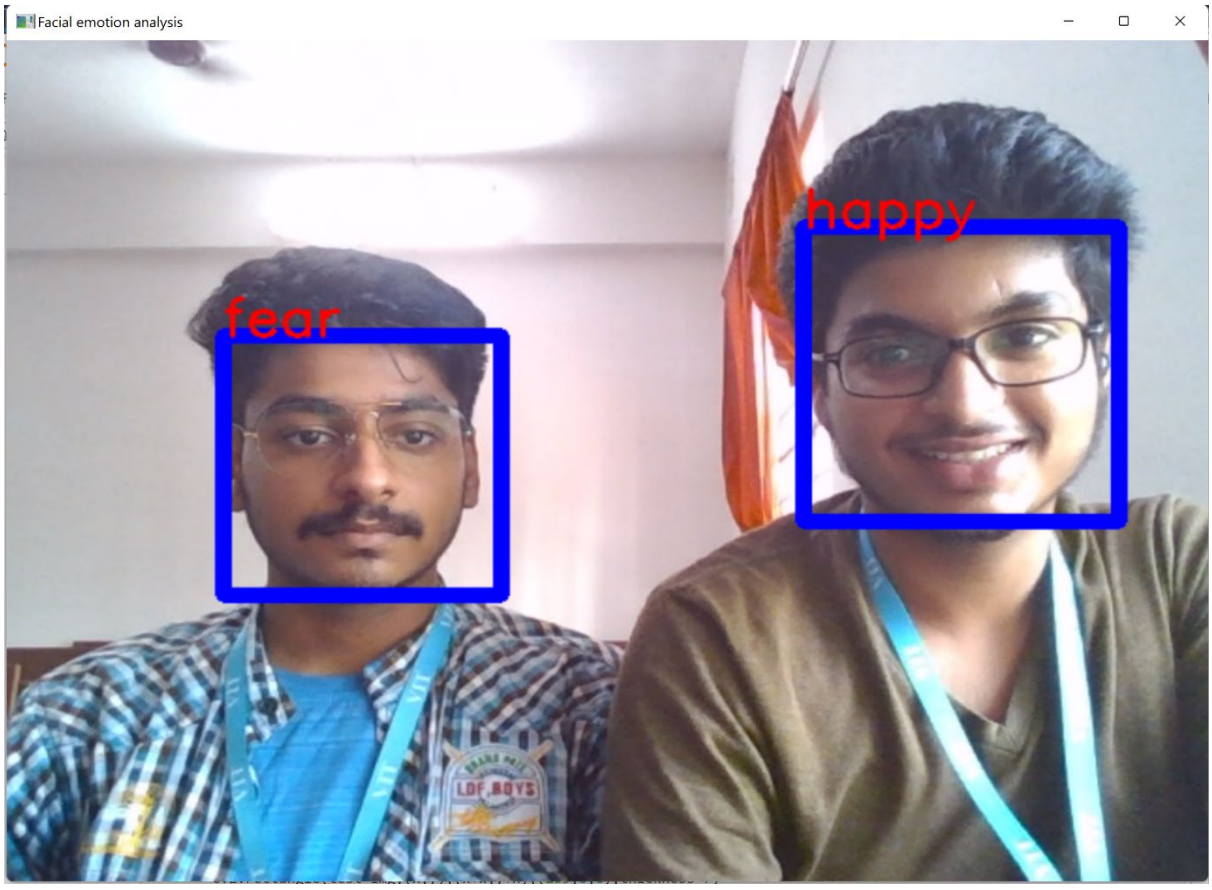
```
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    model.add(Flatten())

    # Fully connected neural networks
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))

    model.add(Dense(classes, activation='softmax'))

    return model



# For the MixUp
class MixupImageDataGenerator():
    def __init__(self, generator, directory, batch_size, img_height,
img_width, alpha=0.2, subset=None):

        self.batch_index = 0
        self.batch_size = batch_size
        self.alpha = alpha

        # First iterator yielding tuples of (x, y)
        self.generator1 = generator.flow_from_directory(directory,
                                                        target_size=(
                                                            img_height,
img_width),

class_mode="categorical",

batch_size=batch_size,
                                                        shuffle=True,
                                                        subset=subset)

        # Second iterator yielding tuples of (x, y)
        self.generator2 = generator.flow_from_directory(directory,
                                                        target_size=(
                                                            img_height,
img_width),

class_mode="categorical",

batch_size=batch_size,
                                                        shuffle=True,
                                                        subset=subset)

        # Number of images across all classes in image directory.
        self.n = self.generator1.samples

    def reset_index(self):
        """Reset the generator indexes array.
        """

        self.generator1._set_index_array()
        self.generator2._set_index_array()
```

```
    def on_epoch_end(self):
        self.reset_index()

    def reset(self):
        self.batch_index = 0

    def __len__(self):
        # round up
        return (self.n + self.batch_size - 1) // self.batch_size

    def get_steps_per_epoch(self):
        """Get number of steps per epoch based on batch size and
        number of images.

        Returns:
            int -- steps per epoch.
        """

        return self.n // self.batch_size

    def __next__(self):
        """Get next batch input/output pair.

        Returns:
            tuple -- batch of input/output pair, (inputs, outputs).
        """

        if self.batch_index == 0:
            self.reset_index()

        current_index = (self.batch_index * self.batch_size) % self.n
        if self.n > current_index + self.batch_size:
            self.batch_index += 1
        else:
            self.batch_index = 0

        # random sample the lambda value from beta distribution.
        l = np.random.beta(self.alpha, self.alpha, self.batch_size)

        X_l = l.reshape(self.batch_size, 1, 1, 1)
        y_l = l.reshape(self.batch_size, 1)

        # Get a pair of inputs and outputs from two iterators.
        X1, y1 = self.generator1.next()
        X2, y2 = self.generator2.next()

        # Perform the mixup.
        X = X1 * X_l + X2 * (1 - X_l)
        y = y1 * y_l + y2 * (1 - y_l)
        return X, y

    def __iter__(self):
        while True:
            yield next(self)


input_imgen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=5,
    width_shift_range=0.05,
    height_shift_range=0,
```

```
    shear_range=0.05,
    zoom_range=0,
    brightness_range=(1, 1.3),
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=validation_split)


train_generator = MixupImageDataGenerator(generator=input_imgen,
                                          directory=train_dir,
                                          batch_size=batch_size,
                                          img_height=img_height,
                                          img_width=img_height,
                                          subset='training')

validation_generator = input_imgen.flow_from_directory(train_dir,
                                                       target_size=(
                                                           img_height,
img_width),

class_mode="categorical",

batch_size=batch_size,
                                                       shuffle=True,
                                                       subset='validation')


#For the  Kernel Regulazer
def get_model(input_size, classes=7):
     #Initialising the CNN
    model = tf.keras.models.Sequential()

    model.add(Conv2D(32, kernel_size=(3, 3), padding='same',
activation='relu', input_shape =input_size))
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',
padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(0.25))

    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu',
padding='same', kernel_regularizer=regularizers.l2(0.01)))
    model.add(Conv2D(256, kernel_size=(3, 3), activation='relu',
kernel_regularizer=regularizers.l2(0.01)))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.5))

    model.add(Dense(classes, activation='softmax'))

    #Compliling the model
    model.compile(optimizer=Adam(lr=0.0001, decay=1e-6),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
from keras.applications.resnet50 import ResNet50

from keras.models import Model
import keras
import csv
from PIL import Image
from sklearn.model_selection import train_test_split
from keras.layers import Input, Add, Dense, Activation,
ZeroPadding2D,BatchNormalization, Flatten, Conv2D, AveragePooling2D,
MaxPooling2D, GlobalMaxPooling2D
from tqdm import tqdm
import numpy as np # linear algebra
from numpy import asarray
from sklearn.preprocessing import OneHotEncoder
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
import collections
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.preprocessing import image
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.normalization import BatchNormalization
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.layers import Dropout
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# ResNet50
a = []
for i in range(len(X_train_resnet50)):
            image_string = (X_train_resnet50)[i].split(' ')
            image_data = np.asarray(image_string,
dtype=np.uint8).reshape(48,48)
            a.append(image_data)

X_train_resnet50 = np.array(a)
rgb_X_train = np.repeat(X_train_resnet50[..., np.newaxis], 3, -1)
print(rgb_X_train.shape)  # (size, 48, 48, 3)

model1 = ResNet50(weights='imagenet', include_top=False, input_shape=(48,
48, 3))

x = model1.output
x= Flatten()(x)
x = Dense(7, activation='softmax')(x)
model50 = Model(inputs=model1.input, outputs=x)


model50.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model50.fit(rgb_X_train, Y_train, batch_size=64, epochs=40,
steps_per_epoch=len(X_train)/128, validation_split = 0.25)
model50.save('CNNmodel')
```


Facial emotion analysis
neutral
happy


Facial emotion analysis
fear
happy