

# **IMPLICIT FEEDBACK BY RECOGNISING FACIAL EXPRESSIONS**

A Project Report

Submitted in Partial Fulfillment of the

Requirements

for the Award of the Degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

Submitted

By

**Manohar Virati (17311A1210)**

**Nikitha Katakam (17311A1215)**

**Dheeraj Perumandla (17311A1232)**

**Sowmya Racha (17311A1237)**

Under the esteemed guidance of

**Dr. Sreedhar Potla**

Associate Professor



**Department of Information Technology.**

**Sreenidhi Institute of Science and Technology, Ghatkesar.**

## **CERTIFICATE**

This is to certify that the project work entitled “**Implicit Feedback by Recognising Facial ExpressionsExpressions** ” done by Manohar Virati, Nikitha Katakam, Dheeraj Perumandla, Sowmya Racha bearing the Rollnos: **17311A1210, 17311A1215, 17311A1232, 17311A1237** respectively are the students of Information Technology Department, is a record of bonafide work carried out by them. This project is done as a partial fulfillment of obtaining Bachelor of Technology Degree to be awarded by **Sreenidhi Institute of Science and Technology (SNIST), Ghatkesar.**

The matter embodied in this project report has not been submitted to any other university for the award of any other degree.

**Guide/Mentor**

**Dr. Sreedhar Potla.**

**Head of the Department**

**Dr. V. V. S. S. S. Balaram.**

## **ACKNOWLEDGEMENT**

We express our sincere thanks to **DR. P. NARASIMHA REDDY** director,

**DR. T. CH. SIVA REDDY** principal, **Sreenidhi Institute of Science and Technology**, Ghatkesar for the support and motivation provided to us.

Our sincere thanks to **DR. V. V. S. S. S. BALARAM**, Head of IT Department, SNIST and our guide **DR. SREEDHAR POTLA** for his valuable suggestions and also the faculty for the encouragement and guidance provided.

Manohar Virati (17311A1210)

Nikitha Katakam (17311A1215)

Dheeraj Perumandla (17311A1232)

Sowmya Racha (17311A1237)

# **Table of Contents**

1. Abstract.....	5
2. INTRODUCTION .....	6
3. PROJECT DESCRIPTION .....	7
A. Frontal Face Image Database.....	7
B. Data Pre-processing .....	7
C. Neural Network Design .....	9
1. Convolutional Layers : .....	10
2. Max Pooling : .....	10
3. Dense Layers : .....	10
4. Batch Normalization : .....	10
5. Training and Saving the Network : .....	10
6. Network Modules : .....	11
D. Application .....	11
F. masterControl.py Module .....	11
4. EVALUATION .....	12
Training Statistics .....	12
Tests on Native Dataset .....	12
Performance on Real-world Applications .....	14
5. OUTPUT .....	15
6. SUMMARY AND CONCLUSION.....	17
7. TOOLS, PACKAGES AND DATASETS USED:.....	17
8. REFERENCES.....	18

# ***1. Abstract***

**F**acial emotion recognition is an integral part of psychology, forensics and social media. In all these fields, a high degree of reliability of classification is imperative. Currently human judgement is used in these fields but humans are not always accurate. Therefore, there is a need for a reliable and quick way of identifying human expressions.

The recent advances in machine learning and pattern recognition has offered several algorithms to recognize human emotions. One such algorithms is the Convolutional Neural Network (CNN). The CNN is capable of high-speed image processing with excellent reliability. In this project one such CNN is used to recognize facial emotions. It is seen that with proper training this method can yield very high accuracy of classification.

## **2.INTRODUCTION**

**T**HERE are a total of seven human emotions which can be identified from facial expressions. They are anger, fear, disgust, happiness, sadness, surprise and contempt.

Recognition of such emotions is valuable in several fields and is currently done manually. But human judgement can be flawed in many cases. Errors can occur due to various factors like insufficient knowledge of facial expressions and their meaning, biased judgement, stress and the monotonous nature of the work. Such misinterpretations can have very severe results depending on the field. Therefore, it is necessary to come up with an automated solution to this problem with minimal errors, high speed and reproducible results.

Facial emotion recognition has long been a topic of study in machine learning and deep learning. Several machine learning algorithms such as Support Vector Machines (SVM), Naïve Bayes and Maximum entropy have been applied to detect human facial emotions. Deep learning algorithms such as Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) have also been used in the field of facial emotion recognition. In this project the Convolutional Neural Network has been used in order to perform facial emotion recognition.

Convolutional Neural Networks offer various advantages over an Artificial neural network for image processing applications. The CNN is a partially-connected network where each neuron is connected to only a subset of neurons from the previous layer. This subset of neurons is from a small region in the image. The key advantage in this approach is that since every neuron is connected to a small region from the previous layer, the spatial properties of the image are often maintained. This enables the network to easily identify patterns in an image. The second important advantage of a CNN over an ANN is that the number of training parameters for the CNN is much lesser due to the fact that each neuron has fewer connections with the previous layer this greatly improves training and classification speed and also combats the problem of overfitting.

In this project the CNN is used to analyze the complex facial expressions, analyze key patterns which correspond to the seven facial emotions, perform statistical analysis and determine the facial emotion which has the highest probability. An eighth class called Neutral is used in this project in order to handle scenarios where the facial emotion cannot be identified with a good degree of certainty.

# 3.PROJECT DESCRIPTION

## A. Frontal Face Image Database

This project requires a frontal face image dataset with a wide variety of facial expressions which correspond to the seven emotions. For this purpose, the Cohn-Kanade AU-Coded Expression Database was used. This database is one of the leading databases used for facial emotion recognition. It contains subjects with several facial features and varying degree of facial expressions classified into the seven emotion which this project aims to recognize.

## B. Data Pre-processing

Data which was downloaded from the Cohn-Kanade AU-Coded Expression Database was further processed to better suit the requirements of the project in the following ways.

### 1. Face Detection

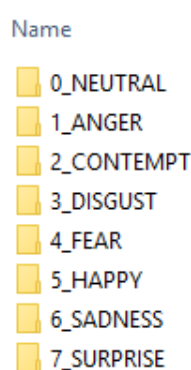
The CNN used in this project aimed to classify facial emotions and required only face portion of the images in the dataset. This cannot be done manually since the dataset contained thousands of images and it was also necessary to be able to automatically detect faces in the case of real time applications. In order to quickly detect and extract the face portion of the images, an LBP Cascade Classifier available in OpenCV was used. This classifier offers high speed classification and sufficient accuracy for real-time applications.

### 2. Manual Verification of Data

In order to ensure best results, the data must be manually inspected to find and remove any flaws before training the CNN. To facilitate this process, the detected faces were resized to 100X100 images and stored in a separate output location under directories named after the different emotions. In addition to this, a directory for neutral images were also created to house those faces whose expressions were deemed to be unclear.

Once the programs which populated these directories with faces were executed, the faces were manually inspected. The false positives from the LBT Cascade Classifier were removed. Faces with unclear emotions were moved to the neutral folder.

Figure 1: DIRECTORIES CONTAINING LABELED FACES



### **3. Data and Label Files**

After a thorough inspection it was found that some of the emotion directories contained fewer specimens than others. This could adversely affect stability of the CNN and will cause misclassifications. The classifier will be more prone to identify faces as belonging to the category which had the greatest number of samples during training. In order to overcome this problem, virtual sampling was done. Virtual sampling is a technique which improves the representation of classes with insufficient samples through re-sampling and adding some noise to the additional samples.

While training the network it will be highly inefficient to access the image files every time since this would involve system calls to access every image. To improve training speed, the images in all the eight emotion directories were consolidated into a file named as data.npy and their corresponding emotion labels were saved into a file called labels.npy. It is to be noted that in a gray scale image, the three RGB values of a pixel are always equal. Therefore, to remove this redundancy, only one of the three basic color values are stored in the data.npy file for each pixel.

### **4. Data Pre-processing Modules**

To achieve the above-mentioned functionality three python modules were created. The faceDetectorLayer.py module was made to leverage the LBP Cascade Classifier for frontal face detection available in OpenCV in order to detect the human faces in the given images. The returned images are grayscale images.

The ckPlusLayer.py module was built to navigate the CK+ dataset, use the faceDetectorInterface.py module to detect faces in the dataset and store the detected faces in separate output location under the appropriate emotion directories.

The dataInterface.py module was created to allow the user to control the data preprocessing steps and also to create the data.npy and labels.npy files.



### C. Neural Network Design

The Neural network consists of three convolutional layers and two fully connected dense layers. Each of the convolutional layers used filters of dimensions 3X3. The entire network was designed using Keras a python package which uses Tensor Flow as the backend package. The structure of the network is as seen in Figure 2.



### **1. Convolutional Layers :**

The first convolutional layer was connected to the input and it used a total of 32 filters, the second layer consisted of 64 filters and the third filter consisted of 128 filters. They all used ReLU as the activation function. Each of the convolutional layers were preceded by a batch normalization layer and succeeded by a max pooling layer.

### **2. Max Pooling :**

Max pooling is the process of reducing image size by substituting a group of matrix elements with a single element whose value is equal to the maximum value in the group. Max pooling was used in this project to reduce the size of the convolutional layer output matrices thus boosting training and prediction speed.

### **3. Dense Layers :**

The dense layers are fully connected layers which were connected after the third convolutional layer. The data was flattened before being sent to the dense layers. There was a total of two dense layers. The first dense layer contained 20 neurons each of which used ReLU activation function. The second dense layer contained 8 neurons with SoftMax activation. Each of the dense layers were preceded by a batch normalization layer. The second dense layer was the output layer and each neuron represent one of the eight classes.

### **4. Batch Normalization :**

The batch normalization layers are extremely vital for to the stability of the network. Without batch normalization, the network will train on high input values especially with the ReLU activation function. This will in turn result in large weights. If the network has large weights then even small changes in the input caused by noise will be amplified across the layers thus making the predictions extremely unstable. The batch normalization ensures that the data is normalized within a range of 0 and 1. This will in turn result in smaller network weights. Thus, noise will not be amplified and the network predictions will be stable.

### **5. Training and Saving the Network :**

The data and the corresponding labels were imported from the data.npy and labels.npy files respectively. The data and the labels were shuffled and split into Training, Testing and Validation sets in the ration 80:10:10 respectively. While training, the training data and the validation data were used. The training data was used to train the values of the network parameters while the validation was used to measure accuracy after each epoch in order to prevent overfitting. After the training was completed the model was tested on the

unknown test data and the performance was measured. Finally, the model was saved so that it can be used for predictions later.

## **6. Network Modules :**

The emotionRecognitionNetwork.py module is built to create and train the network according to given specification. This module also stores the network in a file and exposes interfaces for predictions to the cnnInterface.py module.

The cnnLayer.py module is used to specify network configurations, test the network and assess its performance and to expose prediction interface to the applications. This module also does the test, train and validation data split.

## **D. Application**

To demonstrate the functionality of the CNN, detectFaceInWebcam() method is available in applicationLayer.py module.

It takes the input from the webcam(can be changed to other sources) and detect the faces in it using faceDetectLayer.py module and later predicts the expressions of the detected faces using cnnLayer.py module, then reformats the image to show the predicted expressions and the reformatted image on the screen.

## **F. masterControl.py Module**

The masterControl.py module is used to provide a centralized user interface through which the entire project can be handled.

## **4.EVALUATION**

After the CNN was built, the network was put through several tests to comprehensively evaluate the performance of the network. These tests and the corresponding outcome are explained in detail below.

### ***Training Statistics***

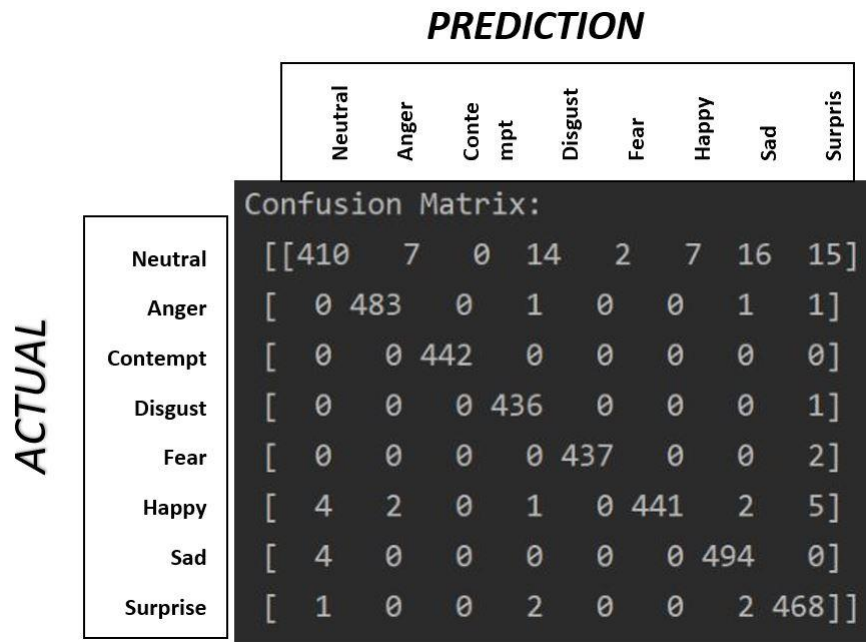
Proper training of the CNN is of paramount importance if the network is to have high prediction accuracy on unknown dataset. While training the network may begin to memorize the samples while omitting the key features which are common to the samples in order to improve accuracy on the training data. This causes the network to have poor prediction accuracy on unknown data. This may occur due to overtraining, insufficient training data or too many trainable variables in the network. This phenomenon is called overfitting.

In order to avoid overfitting, it is essential to keep track of the network's performance on unknown data and continue training only if there is an improvement in performance on both the training and the unknown data. For this purpose, 10% of the data was reserved as validation data

### ***Tests on Native Dataset***

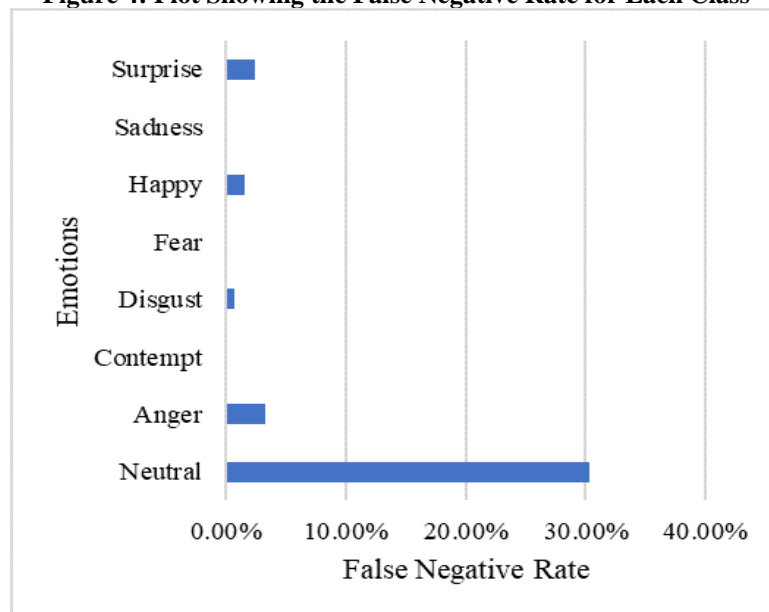
Before training the CNN the data 10% of the data from the CK+ database was reserved for testing purposes. This data was never used during any part of the network training process. Therefore, the performance of the network on this data proved as an excellent way of assessing the ability of the network to learn the critical facial features for emotion recognition without overfitting. The trained CNN network was used on the test data to predict the emotions. The output from the CNN contained the confidence of the prediction for each of the images. The emotion which was predicted with the highest confidence value was identified for each image and the results were compared with the ground truth. Based on this prediction, a confusion matrix was built and the overall accuracy of classification was determined.

Figure 3: Confusion Matrix



Based on the confusions matrix it is easily seen that the classifier performs extremely well on the seven original. The Neutral emotion category is the main source of errors. This particular category contains several false negatives. Therefore, a plot of the false negative rates for each class was created as seen in Figure 4.

Figure 4: Plot Showing the False Negative Rate for Each Class



The reason for the abnormally high false negatives on the Neutral class is that the Neutral class was a manually created class which contains faces with unclear or subtle emotions. There may be several miss classifications due to human error in this particular class since it is solely based on the opinions of a single person. The trained CNN is able to correctly identify several faces depicting valid emotions even among the Neutral images which are ambiguous to us. Further analysis and proper classification of the Neutral faces can yield much better results. The current overall accuracy of the CNN on unknown data from the native dataset is 95.96%. With proper classification of the faces misclassified due to human error could yield an accuracy of more than 97%.

### ***Performance on Real-world Applications***

The truest indication of a well-built system is its ability to solve real world problems. This holds true in the case of our CNN which was built with real world applications in mind. The LBP Cascade Classifier used for face detection ensures that the faces in the images are detected with speed and reliability. The well trained and simple CNN guarantees that the emotions of the detected faces will be classified with ease. Therefore this setup is well suited for real world applications where speed and accuracy of classification is of paramount importance.

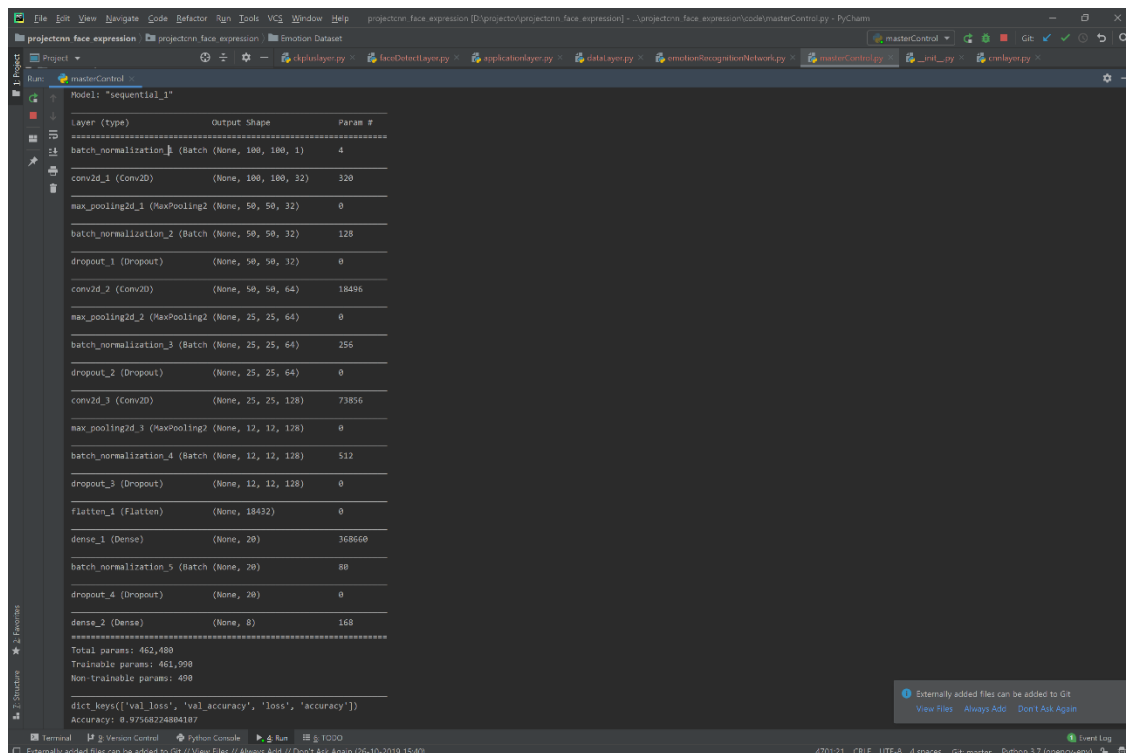
The CNN despite its good performance on unknown data has a lot of room for improvement. It is to be noted that the training dataset of the network primarily consisted of posed images. Therefore, the CNN has some difficulty identifying candid facial emotions in everyday images. The CNN categorizes people wearing glasses as having the expression of disgust. This is due to the fact that the nose piece between the eyes is identical to the wrinkled skin found in the same area. Similarly, the training dataset does not contain any subjects with beard or mustache. Therefore, the CNN is less confident while predicting the emotions of such subjects. These problems in the CNN network are easily remedied by altering the training data to contain more diverse samples. Therefore, it cannot be said that these are flaws in the approach of using for facial emotion recognition.

Another notable flaw is that the CNN was not very confident about facial emotions while classifying the emotions of faces in a video. This is because the images were non-posed and there was lip movement in most of the images since the subjects were trying to speak. This issue can be mitigated by combining the predictions from facial emotion recognition with those of vocal emotion recognition systems. But vocal emotion recognition is beyond the scope of this project.

## 5.OUTPUT

```
C:\Users\DHEERAJ SKYLARK\Anaconda3\envs\opencv-env\python.exe
2019-11-16 18:41:48.163104: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your
CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
=====Main Interface=====

Options:
1. Run Applications.
2. Process Data for training.
3. Build and test network.
4. Exit
Enter the option number : 3
```



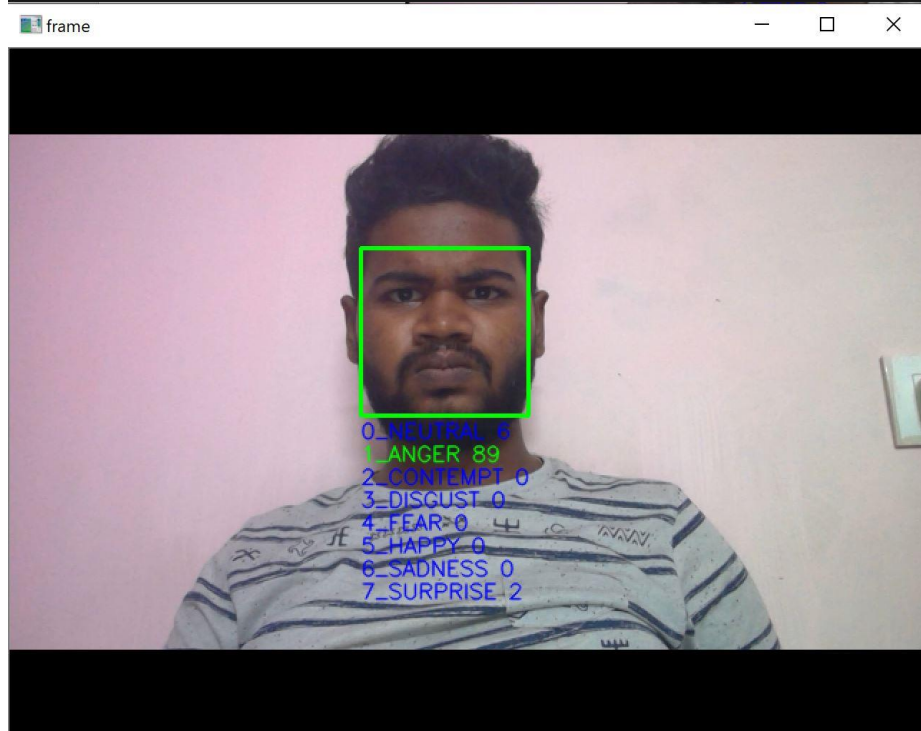
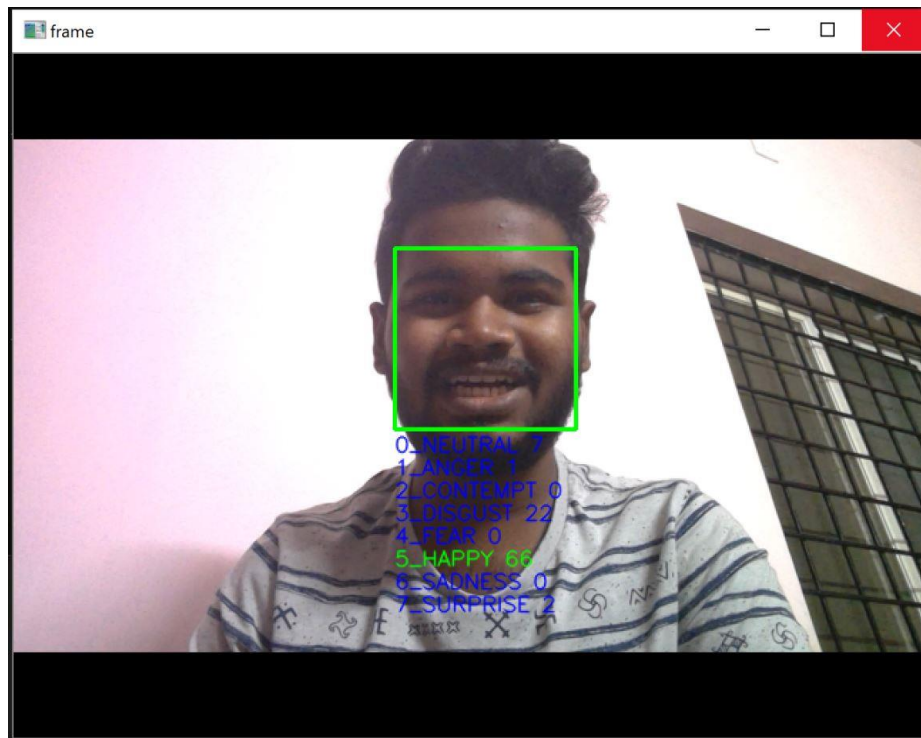
The screenshot shows the PyCharm IDE with a project named 'projecton face expression'. The 'Run' tab is active, displaying the architecture of 'Model: 'sequential\_1''. The architecture is a sequential model with the following layers and parameters:

Layer (type)	Output Shape	Param #
batch_normalization_1 (Batch Normalization)	(Batch Normalization, 100, 100, 1)	4
conv2d_1 (Conv2D)	(None, 100, 100, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 50, 50, 32)	0
batch_normalization_2 (Batch Normalization)	(Batch Normalization, 50, 50, 32)	128
dropout_1 (Dropout)	(None, 50, 50, 32)	0
conv2d_2 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 25, 25, 64)	0
batch_normalization_3 (Batch Normalization)	(Batch Normalization, 25, 25, 64)	256
dropout_2 (Dropout)	(None, 25, 25, 64)	0
conv2d_3 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 128)	0
batch_normalization_4 (Batch Normalization)	(Batch Normalization, 12, 12, 128)	512
dropout_3 (Dropout)	(None, 12, 12, 128)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 20)	368660
batch_normalization_5 (Batch Normalization)	(Batch Normalization, 20)	80
dropout_4 (Dropout)	(None, 20)	0
dense_2 (Dense)	(None, 8)	168

Summary statistics:

- Total params: 462,480
- Trainable params: 461,990
- Non-trainable params: 490

The model's accuracy is displayed as 0.97580224884187. The bottom status bar shows the file encoding as UTF-8 and the Python version as 3.7 (opencv-env).





## ***6.SUMMARY AND CONCLUSION***

To summarize, in this project a highly accurate Convolutional Neural Network to recognize emotions from facial expressions was built. The LBP cascade classifier available in OpenCV was used to detect faces in images. A CNN network was trained using the detected faces to identify a total of eight emotions. The concept of batch normalization was used in the network to regularize the weights and improve stability. Virtual sampling was used to improve class representation of those classes with insufficient data. The trained CNN was tested on unknown data from the same dataset used for training. The performance on test data was analysed.

The performance of the CNN on real-time applications was analysed. Further avenues for improving the performance of the network for real time applications was explored and documented.

## ***7.TOOLS, PACKAGES AND DATASETS USED:***

1. ANACONDA3
2. VISUAL STUDIO COMMUNITY
3. CV2(OPENCV)
4. KERAS
5. NUMPY
6. SKLEARN
7. OS
8. CK AND CK+ DATASET

## 8. REFERENCES

- J. F. Cohn, Z. Ambadar, and P. Ekman. Observer-based measurement of facial expression with the facial action coding system. *The handbook of emotion elicitation and assessment*, pp 203–221, 2007.
- P. Ekman and W. V. Friesen. Facial action coding system. 1977
- A. Mollahosseini, D. Chan, and M. H. Mahoor, “Going deeper in facial expression recognition using deep neural networks,” in 2016 *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–10, IEEE, 2016.
- T. Kanade, J. F. Cohn and Y. Tian, “Comprehensive database for facial expression analysis”, *Proc. 4th IEEE International Conf. on Automatic Face and Gesture Recognition (FG'00)*.
- T. Ahonen, A. Hadid, and M. Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037-2041, 2006.
- P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, “The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression”
  - <http://blog.contactsunny.com/data-science/how-to-split-your-dataset-to-train-and-test-datasets>
  - <https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/>
  - <https://www.tensorflow.org/learn>
  - [https://www.robots.ox.ac.uk/~vgg/research/very\\_deep/](https://www.robots.ox.ac.uk/~vgg/research/very_deep/)
  - <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>
  - <https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/>
  - <https://www.scipy.org/docs.html>
  - <https://becominghuman.ai/how-do-pretrained-models-work-11fe2f64eaa2>
  - <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>