

Dissertation on
**FACIAL EMOTION DETECTION-BASED CONTENT
MODIFICATION IN INTELLIGENT TUTORING SYSTEM USING
DEEP LEARNING**

*Thesis submitted towards partial fulfillment
of the requirements for the degree of*

Master of Technology in IT (Courseware Engineering)

Submitted by
SHANKAR LAYEK

EXAMINATION ROLL NO.: M4CWE22016
UNIVERSITY REGISTRATION NO.: 154498 of 2020 - 2021

Under the guidance of
DR. SASWATI MUKHERJEE
School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India
2022

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled “**Facial Emotion Detection-Based Content Modification in Intelligent Tutoring System using Deep Learning**” is a bonafide work carried out by **Shankar Layek** under our supervision and guidance for partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering) in School of Education Technology, during the academic session 2021-2022.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN – FISLM
Jadavpur University,
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination

for evaluation of Thesis

** Only in case the thesis is approved.

**DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC
ETHICS**

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME : SHANKAR LAYEK
EXAMINATION ROLL NUMBER : M4CWE22016
REGISTRATION NUMBER : 154498 of 2020 - 2021
THESIS TITLE : FACIAL EMOTION DETECTION-
BASED CONTENT MODIFICATION
IN INTELLIGENT TUTORING
SYSTEM USING DEEP LEARNING

SIGNATURE:

DATE:

ACKNOWLEDGEMENT

I feel fortunate while presenting this dissertation at **School of Education Technology, Jadavpur University, Kolkata**, in the partial fulfillment of the requirement for the degree of **M.Tech in IT (Courseware Engineering)**.

I hereby take this opportunity to show my gratitude towards my mentor, **Dr. Saswati Mukherjee**, who has guided and helped me with all possible suggestions, support, aspiring advice and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Prof. Ranjan Parekh, Director of School of Education Technology** for his timely encouragement, support and advice. I would also like to thank **Prof. Matangini Chattopadhyay and Mr. Joydeep Mukherjee** for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from M.Tech in IT (Courseware Engineering) and Master in Multimedia Development. I do wish to thank all the departmental support staffs and everyone else who has different contributions to this dissertation. Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieves this height.

Regards

.....

Shankar Layek
Examination Roll No : M4CWE22016
Registration No: 154498 of 2020 – 2021
M.Tech in IT (Courseware Engineering)
School of Education Technology
Jadavpur University
Kolkata : 700032

Date:

CONTENTS

Topic	Page No.
Executive Summary	1
1. Introduction	2
1.1. Problem Statement	2
1.2. Objectives	2
2. Background Concept	3-8
2.1. Intelligent Tutoring System	3
2.2. Facial Emotion Recognition	3
2.2.1. Filtering	4
2.2.2. Convolution Neural Network	4
3. Literature Survey	9-12
4. Proposed Methodology	13-22
5. Experimentation and Result	23-24
6. Conclusion and Future Scope	25
Reference	26-27
Appendix	28-43

EXECUTIVE SUMMARY

The present work proposes a design of an intelligent tutoring system which mainly focuses on modification of learning material according to the student emotion. In the proposed work different facial emotions of the student are captured while the student is involved in the system for learning. The facial emotion recognition system predicts the emotion from the captured image. The content is changed or modified as per the emotion in the tutoring system. The proposed work has predicted various emotions of the student effectively.

1. INTRODUCTION

1.1. PROBLEM STATEMENT

FACIAL EMOTION DETECTION-BASED CONTENT MODIFICATION IN INTELLIGENT TUTORING SYSTEM USING DEEP LEARNING.

1.2. OBJECTIVES

The objectives of proposed work are:

- Modification of learning tutorial by detecting student emotion using deep learning models.
- Selection of an efficient CNN architecture that would predict different kind of emotions with better accuracy.

2. BACKGROUND CONCEPT

2.1. Intelligent Tutoring System

It is a computer based teaching system which provides tutoring material automatically according to a learner's performance. It doesn't require any human teacher. The system track student's performance and according to the performance it automatically adjust the tutoring material. In primary stage the system gives some questions or problems to the learner. Based on the answer the system can understand about the learner's strength and weakness. The system mostly has 3 basic components [1].

- A. The Domain model
- B. The Student model
- C. The Tutoring model

The *domain model or expert knowledge model* contains theories, rules and problem solving strategies.

The *student model* keeps information about the student. It keeps record what the student knows. The student submits some assignment. From that assignment it knows about the student's strong and weak knowledge zone of that topic [2]. It is updated throughout the learning process.

The *tutoring model* considers data from the domain and student models and decides the type of strategies and actions of the tutoring system. During the problem solving process if any student needs help, the system provides hints for the next step. The system detects when the learner deviates from the learning process. It also provides quick feedback to the student.

2.2. Facial Emotion Recognition

In human communication, facial emotion is an important aspect because it helps other person to understand the internal state of mind of an individual. Through emotions one can assess the mental state of a human being and its level of communications amongst themselves [3]. It plays an important role in computer based teaching system. Now a day, convolution neural network became popularized for image classification

in deep learning architectures. The structure of convolution neural network is built like human brain, which consists of artificial neurons with several hidden layers. Input images are feed to artificial neurons and weight is multiplied with the pixel of an input image. After adding bias, activation function is applied with it [4]. Huge amount of data is feed to the convolution neural network, to make the model highly accurate.

2.2.1. Filtering Process

It is a process of adjusting the colors of the pixels to change the appearance of the image. It increases the contrast of the image, strengthen certain features or remove some features. Filtering technique is used to reduce noise from the images. High-pass filter is applied to reduce low frequency components of the images. It makes the image sharp which contains the edges. Filtering process is applied with the convolution neural network to increase the accuracy of facial emotion recognition system.

2.2.2. Convolution Neural Network

It is a specialized type of deep learning algorithm that takes input images and some mathematical operation called convolution is done on the image to differentiate one from the other as shown in Fig. [1]. A digital image consists of pixel values in matrix form. The mathematical calculation is carried out in the images which is matrix multiplication on the layers of the image.

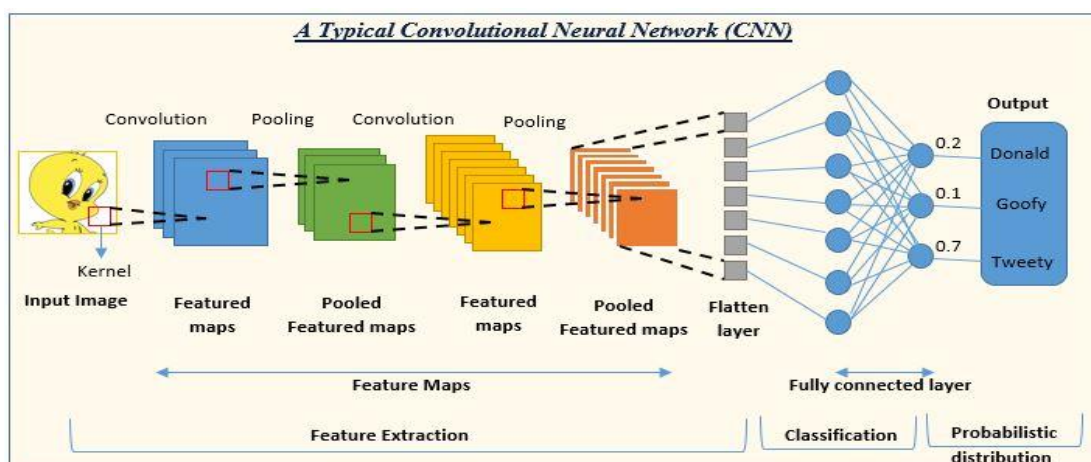


Figure 1. CNN structure

Basic architecture of CNN -

a. Input image -

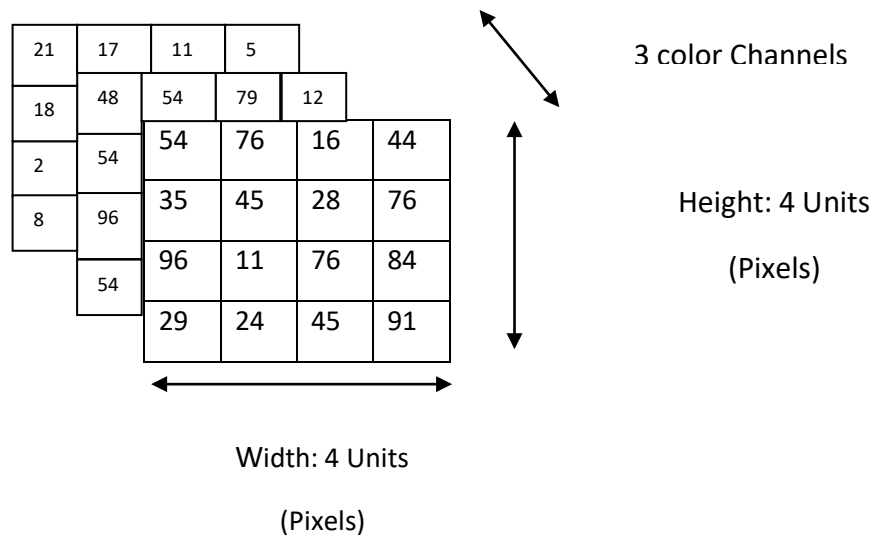


Figure 2. Input image

The input RGB image consists of three color planes – Red, Green and Blue as given in Fig. [2]. It consists of pixel value at every point. The image is represented by height * width * no of channel. Height and width is number of pixels along height and width respectively.

b. Convolution layer - It is the core building block of a CNN because majority of calculation is done in this layer. It requires input image, feature map and filter. It has filter or kernel, which moves across the matrix of the image as shown in Fig. [3] and check for a certain feature is present or not. This process is called convolution.

The feature detector is a 2-D array of weights. Generally 3x3 matrix is used as a feature detector.

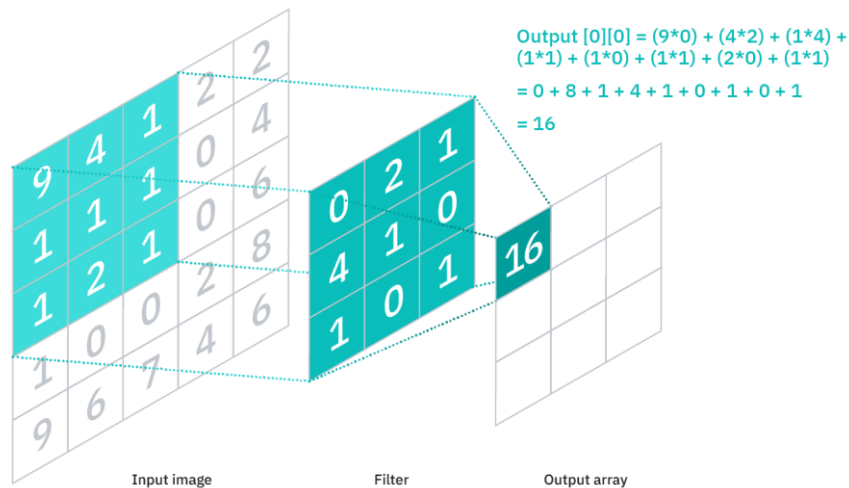


Figure 3. Convolution operation

It is observed from the picture that the filter is applied to a portion of an image. Between the input pixel and filter a dot product is calculated. The result is sent to the output array. Then the filter shifts by a stride. This process repeats until the kernel sweep over the entire image. The result of dot product gives the final result in matrix form.

c. **Pooling layer** - The dimensions of the feature map is reduced by the pooling layers. It sweeps a filter over the entire image matrix. An aggregation function is applied with the filter to get the output. There are 2 types of pooling:

i. **Max pooling** –

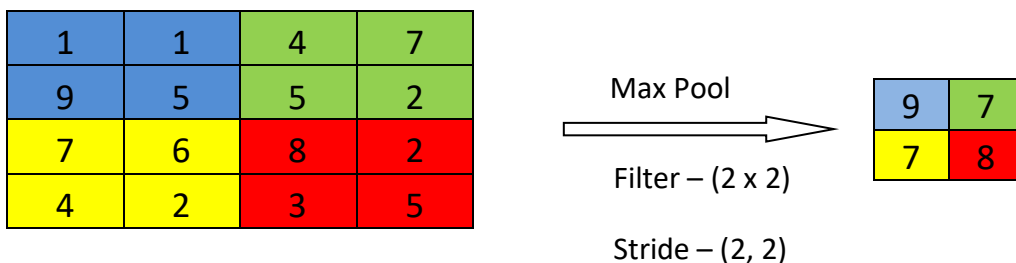


Figure 4. Max pooling operation

As the filter moves over the image, it selects the max pixel value. These selected values make the output array as given in Fig. [4].

ii. **Average pooling –**

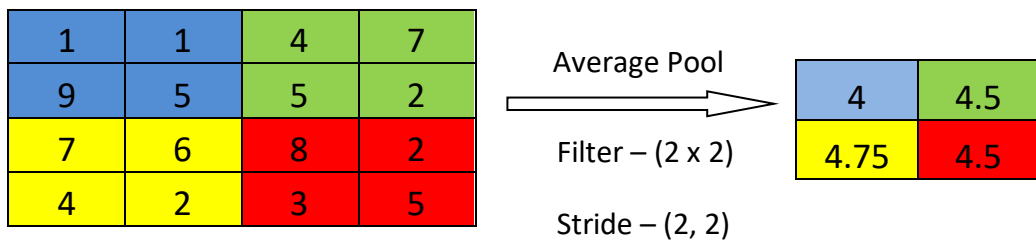


Figure 5. Average pooling operation

As the filter moves over the image, average value is calculated and sends to the output array as shown in Fig. [5].

d. ReLU layer - ReLU denotes rectified linear unit, and it is defined as an activation function given in Eq. [1].

$$y = \max(0, x) \text{ -----Equation 1}$$

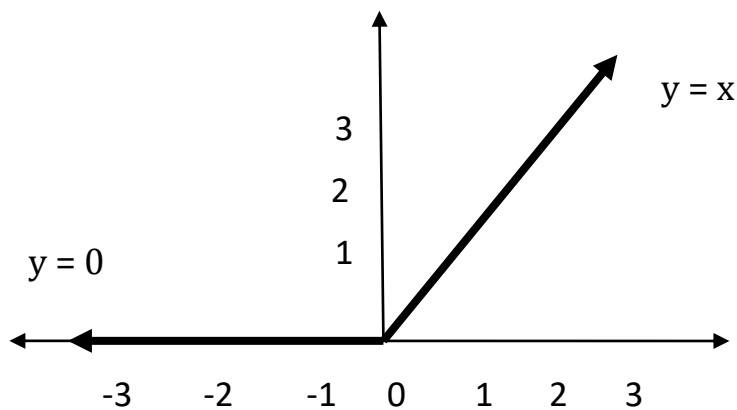


Figure 6. ReLU operation

It gives zero output for all negative value and keeps the positive value same as given in Fig. [6].

e. Fully connected layer – In fully connected layer each neuron in one layer is attached to another neuron of another layer. When the

flattened matrix goes through a fully connected layer it classifies the images.

f. Loss layer - It specifies how the predicted output is close to actual value.

g. Softmax function -

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{-----Equation 2}$$

This is the formula of Softmax function as shown in Eq [2]. It determines the probability of each class and gives the output between 0 and 1. It is applied just before the output layer.

Parameters -

- a. **Kernel size** - The kernel is the total number of pixels that have been processed together. It is expressed like 2 x 2 or 3 x 3.
- b. **Padding** - During calculation some pixel values are lost in output. To recover the lost portion, some '0' valued pixel are added on the image border which produces equally size output.
- c. **Stride** - In each iteration, the analysis window moves a certain number of pixel which is called stride. A stride of 2 means the kernel is displaced by 2 pixels from the predecessor.

3. LITERATURE SURVEY

Madupu et al. [5] designed an automatic emotion recognition system to classify facial expressions using convolution neural network. Firstly, *High Boost filtering* was applied to an image for noise reduction. Thereby strengthening the high frequency components and allowing low frequency component to pass. Then *SURF Feature Extraction* was applied combining four steps. First integral image was computed followed by Hessian Matrix Determinant. After calculating the normalization of the determinant, interest points were computed. After applying SURF feature extraction, images were feed into the *Convolution Neural Network* for training. It predicted the image and gave the result in the form of prediction value. CNN and BPNN (Back Propagation Neural Network) process were used for training the dataset, which achieved 91% accuracy in CNN model and 88% accuracy in BPNN model.

Poulose et al. [6] proposed foreground extraction based facial emotion recognition system using Xception model. Firstly images were converted to grey images after reducing computational complexity and background noise. The size of the images was considered was 48 x 48 pixels. Adam optimizer was used with learning rate of 0.02 and batch size of 128 with 150 epochs. The proposed method has achieved 97.51% accuracy and 94.14% accuracy without foreground feature extraction.

OZDEMIR et al. [7] designed a real time facial emotion recognition using LeNet CNN architecture. 3 datasets JAFFE, KDEF and custom dataset were used to train the model. Firstly, facial parts were detected by using the Haar Cascade filter. Then all the images were converted to 64 x 64 pixel images. LeNet architecture included 2 convolution layers, 2 max-pooling layers and one fully connected layer. Kernel size of 2 x 2 was applied with stride 2. Keras and TensorFlow libraries were used to train the LeNet CNN architecture. The epoch

number was set to 500. The accuracy achieved by the model was 96.43% and the training loss was 8.87%. The proposed LeNet model accuracy was good in predicting neutral, fear, surprise emotions but accuracy drops in detecting the sad emotion.

Zhou et al. [8] proposed a face and gender recognition system using Convolution Neural Network. It composed of two modules: face recognition module and gender recognition module. The training of face recognition and age recognition were done separately. Results from both the output are combined for prediction. LFW, YTF and VGGFace2 dataset were used to train CNN. For gender recognition module, Adience dataset was used. The proposed neural network was similar to ResNet50 to train face and gender datasets. It was a 50-layer network. After scaling the size of the image is converted to 224 x 224 pixels. Instead of fully connected layer Global Average Pool (GAP) was used to reduce the network size. It also increased training speed and accuracy of the model. After that Softmax layer was used to predict the emotion. After training of 200 epochs in LFW, YTF and VGGFace2 dataset the average validation accuracy was 99.33%, 99.83% and 93.22% respectively and the average validation loss was 2.6%, 8.9% and 5.93% respectively. The Adience dataset was trained in gender recognition module. The average validation accuracy was 93.22% with 100 epochs and the average validation loss was 27.81%.

Kaviya P et al. [9] designed a facial emotion analysis system of group images. Two different datasets mainly, FER-2013 and custom dataset were used. It has 4 processing methods i.e. pre-processing, feature extraction, group facial emotion prediction and speech synthesizer. At the *Pre-processing* stage the Haar filter was used to identify faces from the static and dynamic images. The images were converted to gray scale of size 48 x 48 pixels. These images were trained in Convolution Neural Network for *feature extraction* to predict the right emotion. The CNN architecture has convolution layer, max-pooling layer and fully connected layer. ReLU function was used as an activation function. At last, for downsampling a 2x2 max-pooling operation was performed. The *group*

emotion was predicted using calculated value of weighted average of individual emotion. Microsoft *speech synthesizer* was used to get the output in audio format. The proposed CNN model achieved 65% test accuracy in FER-2013 dataset and 60% test accuracy in custom dataset.

S. Pandey et al. [10] designed a facial emotion recognition system using deep learning with the help of a standalone based approach. The VGG-16 sequential model was improved by adding some method via `add()` method. The model has one input tensor and one output tensor in each stack of layers. The model was made of four convolution stages and three fully connected layers. The convolution layer was used to extract the features and Fully connected layers were used to classify the input images. Instead of Rectified Linear Unit (ReLU), ELU (Exponential Linear Unit) was used as an activation function to avoid gradient dispersion problem and increase the training speed. Max pooling gave max score from the convolution block. Batch normalization was used to speed up the learning process. The data passed through first fully connected layer, followed by flatten function, ELU activation, batch normalization and dropout layer. Second layer has the same layer without the flatten function. The third fully connected layer was dense layer which has Softmax function that predicted the output and gave a value in the range of 0 to 1. Stochastic gradient descent (SGD) is used as an optimizer. With 100 epochs, 76.62% accuracy was achieved.

R. Priya et al. [11] proposed an emotion recognition system using LeNet and MobileNet CNN architectures. Both the architectures consisted of two types of layers: convolution layer and max pooling layer. With the help of filter/kernel the input image was converted to the specified size of the CNN. They used two types of deep learning models: one was non-pretrained model such as LeNet and other was pretrained model MobileNet. The LeNet architecture consists of two sets of convolutional, activation and max pooling layers, with flattening convolution layer, two fully-connected layers and a Softmax classifier. The MobileNet architecture has a convolutional layer with stride 2, depth-wise layer with stride 2 and point-wise layer that doubles the number of channels. The

accuracy achieved in LeNet model and MobileNet model were 95% and 96% respectively.

R. Gill et al. [12] proposed a deep learning approach for emotion recognition using facial landmark approach using Dlib open source library. This deep metric learning technique is used to recognize faces using a landmark model. As per the Dlib library, there are 68 landmarks in a human face. Those main landmarks are contour points of eye and eyebrow, pupils, nostrils, nose tips, mouth contour points etc. It has achieved 93% accuracy.

4. Proposed Methodology

In this proposed work, a tutoring system is designed where the facial expressions of a student is detected and based on his/her emotion the content materials (like video, text notes) are modified. 7 different types of emotions are identified; those are angry, disgust, fear, happy, neutral, sad, surprise. A convolution neural network is implemented to identify the learner's various emotions. If the model predicts the emotion as sad then the teaching material is changed and a simpler material is provided to the learner that might change the learner's emotion. Similarly if the prediction outcome of emotion is happy or neutral then there is no need to change the learning material in the ITS.

In the first stage, the learner registers in the system by providing the necessary details for login. After login the learning material is provided. Several images of the learner are captured during the entire process of learning. The captured images are then saved in a folder which is later used by the CNN model to predict the facial emotion. On the basis of the predicted emotion, the material in the ITS is modified.

Python programming language and its libraries like - Numpy, Pandas, CV2, OS, Tensorflow, Matplotlib are used to design the facial emotion recognition system. Different CNN architectures have been implemented using filtering method in order to achieve improved accuracy.

The proposed method consists of five components. a. Data collection
b. Filtering c. Preprocessing, d. Processing, e. Postprocessing and
f. Outcome.

- a. **Data collection** – Data is collected from 2 different datasets; those are CK+48 and FER-2013 dataset. After segregation, the images with proper expression of different people with different age group, emotions and poses are collected [13]. Images are also captured from webcam and saved in a folder. Both images are added in the dataset. 7 emotions are considered mainly angry, disgust, fear, happy, sad, surprise and neutral. Total 1890 images are taken for training and 70 images for testing, details shown in table [1].

Emotions	Number of Image Sample
Angry	270
Disgust	270
Fear	270
Happy	270
Neutral	270
Sad	270
Surprise	270

Table 1. Image dataset

b. Filtering - High-pass filter is applied to remove the low frequency components of the image. Firstly, RGB image is converted to BGR image as it is the default format in OpenCV. Finally the BGR images are converted to gray images. After comparing the histogram of BGR image and gray image it is found that gray histogram is decreased by (1/3) rd than the color image. Gray scale transformation (inverse operation in gray scale image) is applied in the gray image to make the bright pixels dark and dark pixels bright. A constant is added on the image so that the entire pixel became brighter. After that the contrast of the image is increased. Fourier transformation is applied in the gray scale image to find the frequency domain of the image. It blurs the gray scale image by Gaussian filter with kernel size of 10. High-pass filtering is applied to find the edges of the image. This section removes low frequency component of the image, resulting in a sharpened image which contains the edges.

Algorithm –

```
image = io.imread("location of image")  
image_2 = cv.cvtColor(image, cv.COLOR_BGR2RGB)  
  
plt.hist(image_2.ravel(), bins = 256, range = [0, 256])  
  
gray_image = cv.cvtColor(image_2, cv.COLOR_BGR2GRAY)  
  
im2 = 255 - gray_image
```

```
im3 = (100.0/255)*gray_image + 100
im4 = 255.0*(gray_image/255.0)**2
def histeq(im, nbr_bins = 256):
    imhist, bins = np.histogram(im.flatten(), nbr_bins, [0, 256])
    cdf = imhist.cumsum()
    cdf = imhist.max()*cdf/cdf.max()
    cdf_mask = np.ma.masked_equal(cdf, 0)
    cdf_mask = (cdf_mask - cdf_mask.min())*255/(cdf_mask.max() -
    cdf_mask.min())
    cdf = np.ma.filled(cdf_mask, 0).astype('uint8')
    return cdf[im.astype('uint8')]

img_array = histeq(im4)
cv2_imshow(img_array)
plt.hist(img_array.ravel(), bins = 256, range = [0, 256])
```

This process is applied in every image before training all architectures. This process helps to increase the detection accuracy of facial emotion recognition system.

- c. Preprocessing** – After the filtering process Haar cascade library is used to remove those unwanted pixels which are outside the range of facial emotion. The pixel values of the images are put in multi-dimensional array format. Different image size is considered for implementing different CNN architectures. The label name is assigned to each image for better understanding.
- d. Processing** – The pixels which are in multi-dimensional array format are then normalized by dividing the each pixel value with 255.0 so that each pixel value lies in the range between 0 and 1. This normalized matrix is used to train different architectures.

Different CNN Architectures

There are six different architectures are used in order to achieve improved accuracy. Those are –

i. VGG16 –

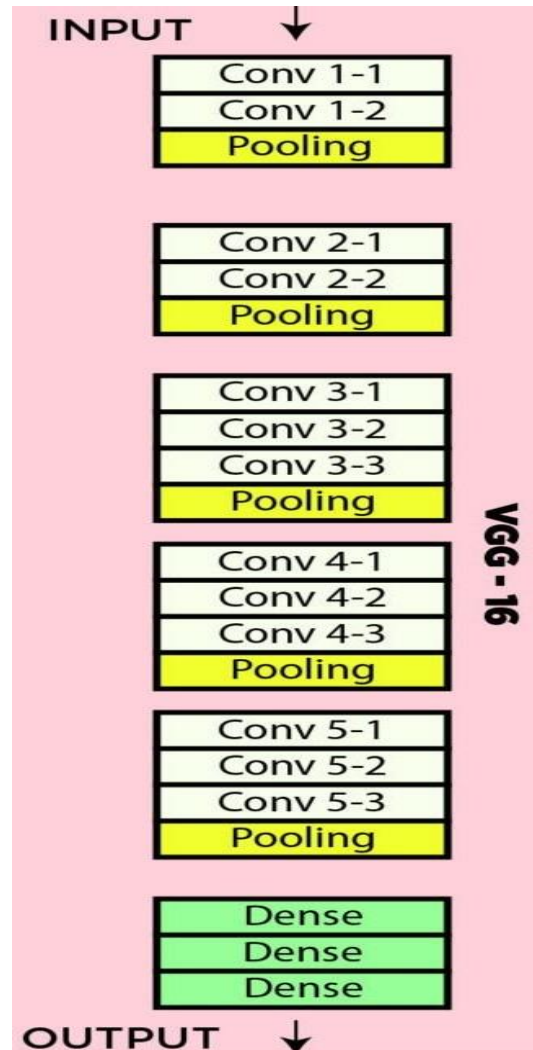


Figure 7. VGG16 Architecture

The details of VGG16 architecture is shown in Fig. [7]. After scaling the size of the image size is converted to 224 x 224 pixels. Total number of trainable parameter is 13,47,93,670.

ii. InceptionV3 –

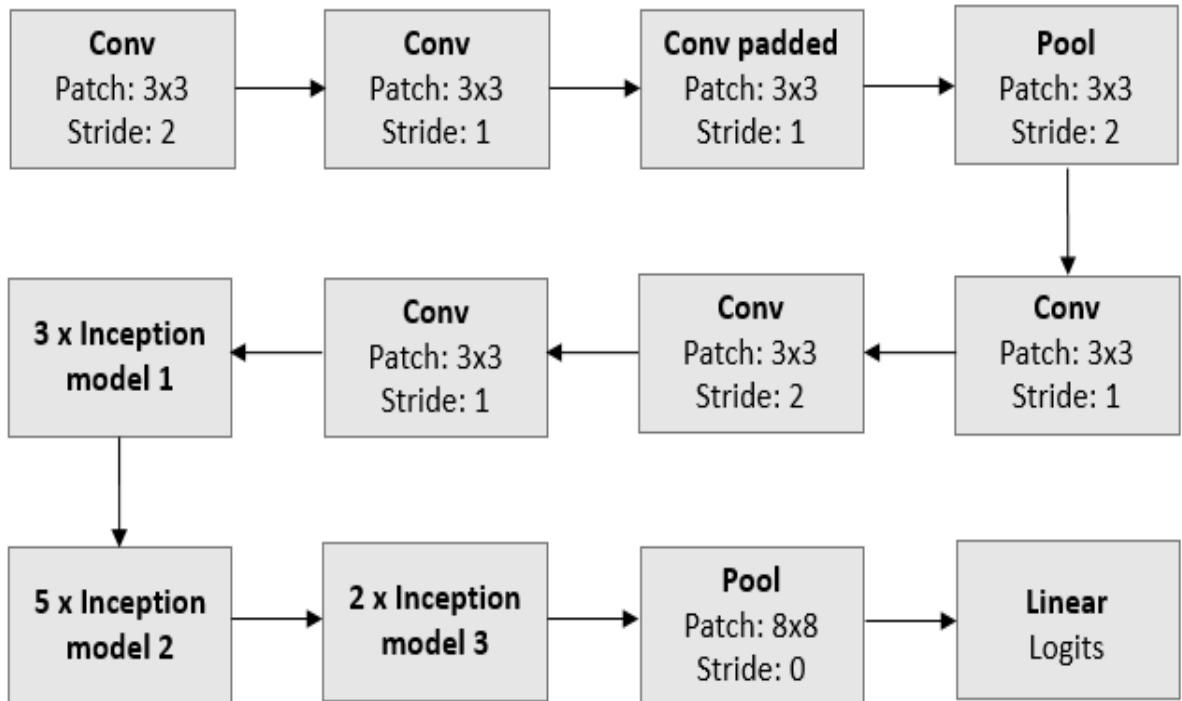


Figure 8. InceptionV3 Architecture

The details of InceptionV3 architecture is given in Fig. [8].After scaling the size of the image is converted to 299 x 299 pixels. Total number of trainable parameter is 2,20,39,330.

iii. ResNet50 –

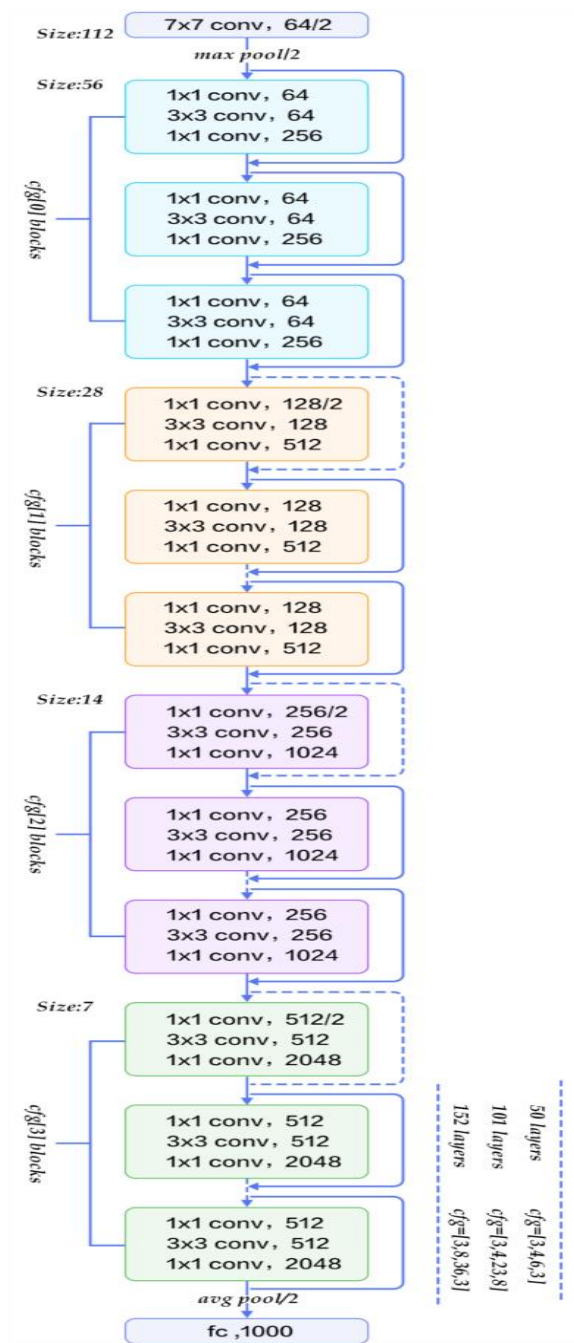


Figure 9. ResNet50 Architecture

The details of ResNet50 architecture is shown in Fig. [9].After scaling the size of the image is converted to 224 x 224 pixels. [14]. Total number of trainable parameter is 2,38,05,570.

iv. Xception –

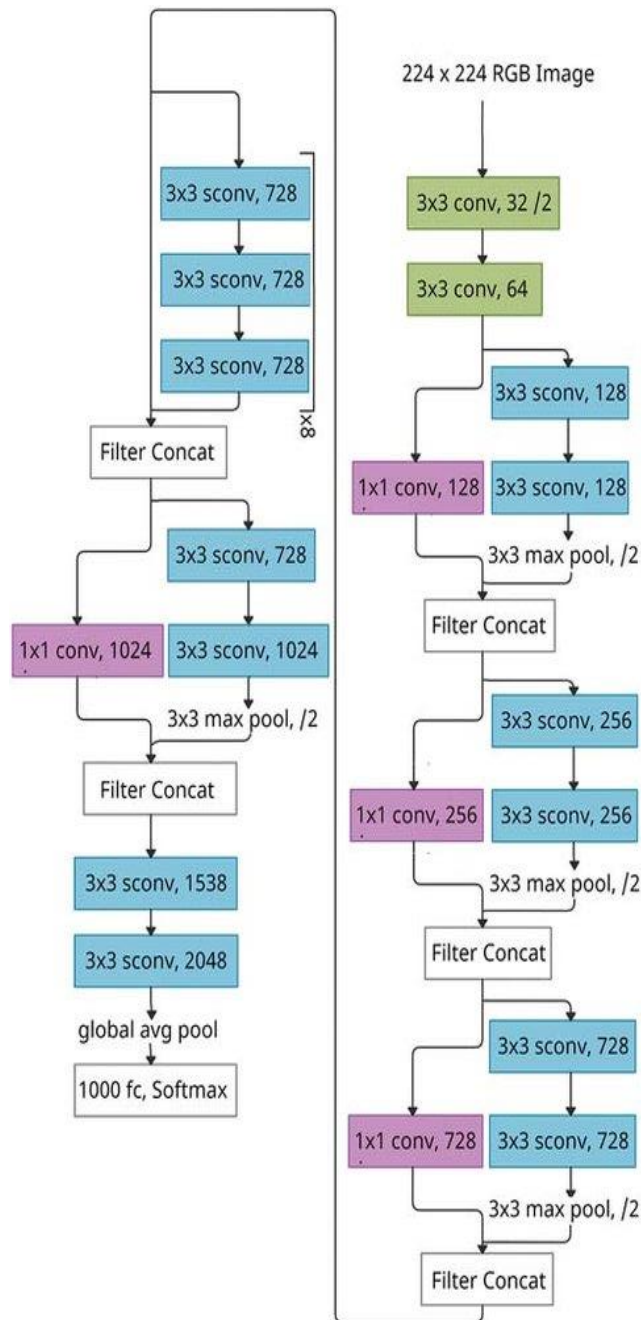


Figure 10. Xception Architecture

As in Fig. [10] image size is scaled to 299 x 299 pixels for Xception architecture. Total number of trainable parameter is 2,10,77,930 [15].

v. **DenseNet201 -**

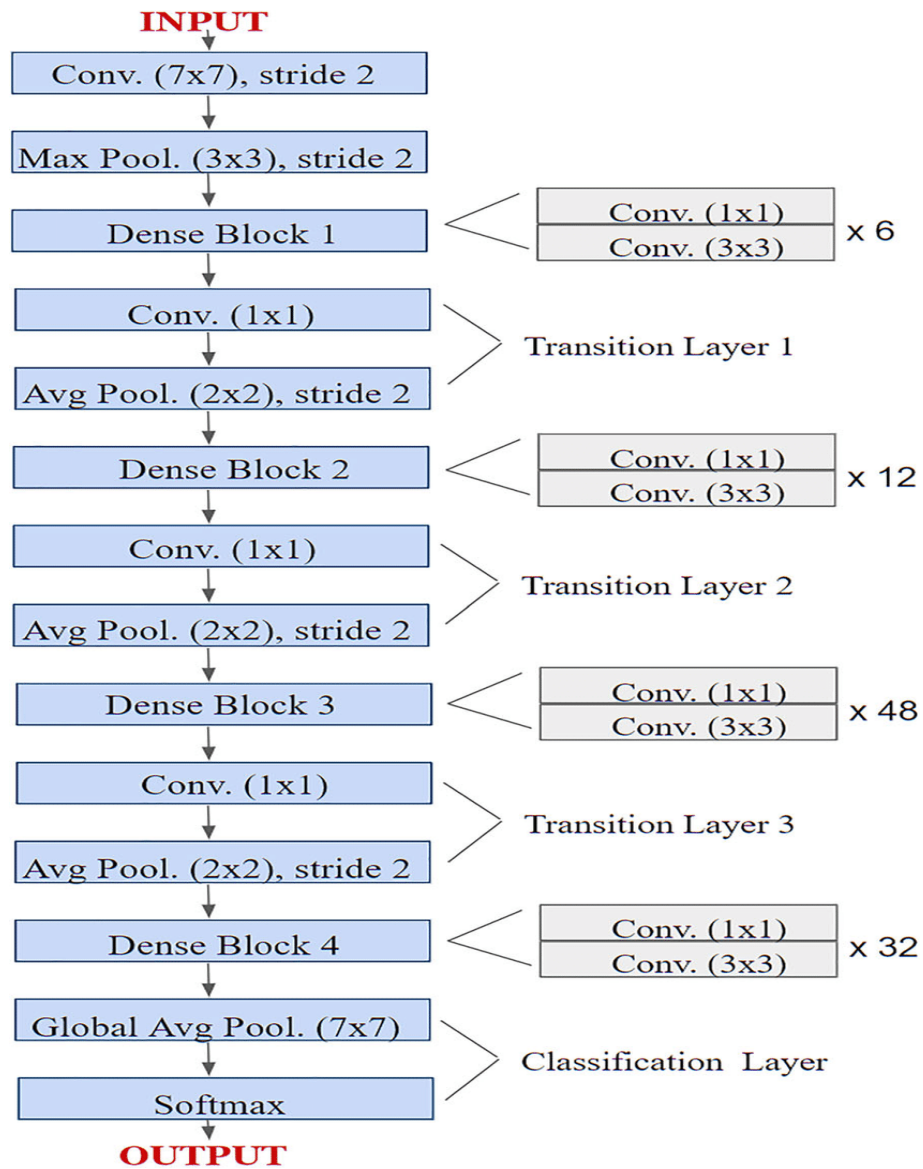


Figure 11. DenseNet201 Architecture

The details of DenseNet201 architecture is given in Fig. [11].After scaling the size of the image is converted to 224 x 224 pixels. Total number of trainable parameter is 18,347,500 [15].

vi. MobileNetV2 –

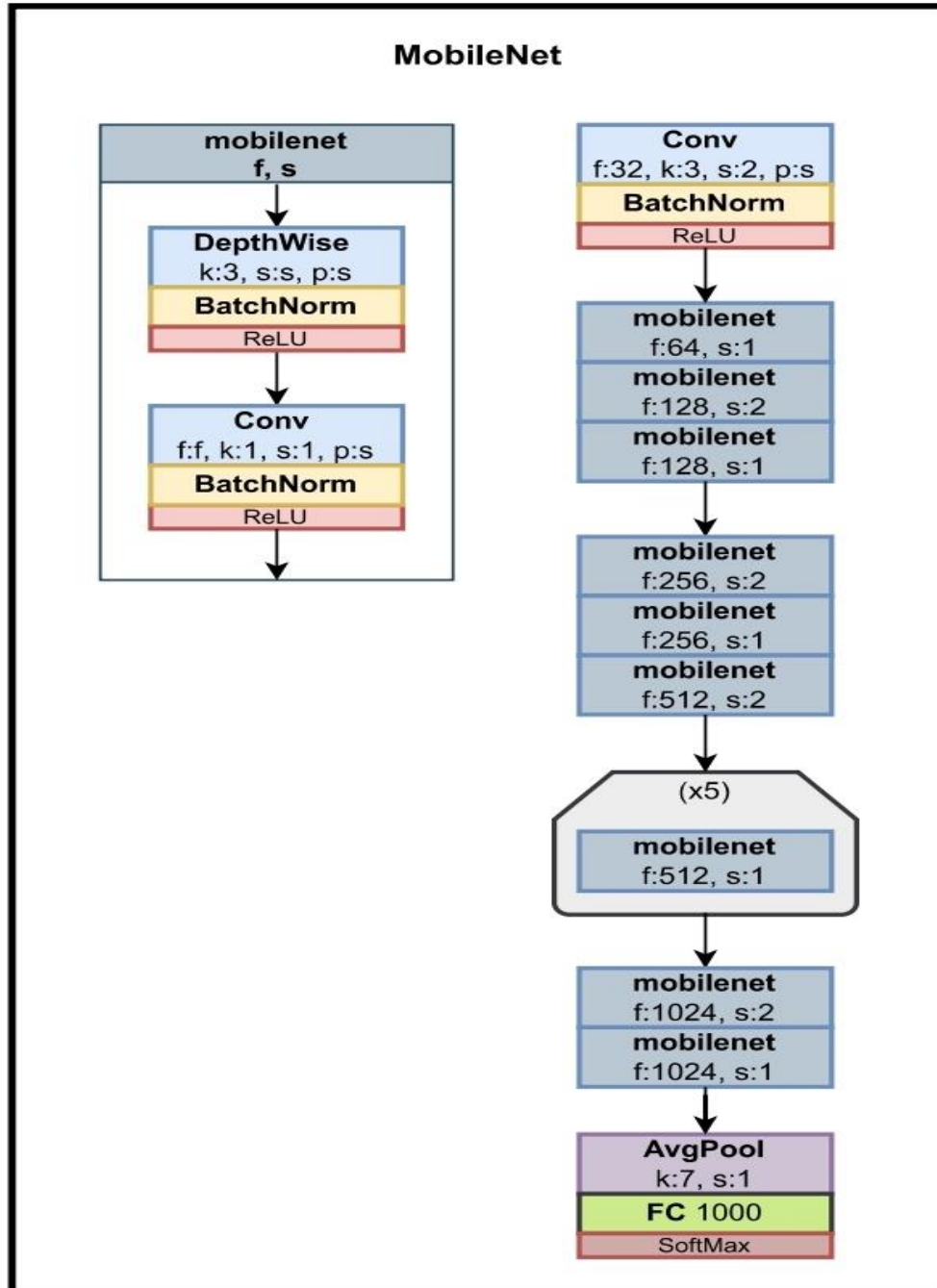


Figure 12. MobileNet Architecture

The details of MobileNet architecture is shown in Fig. [12].After scaling the size of the image is converted to 224 x 224 pixels. Total number of trainable parameter is 2,396,550 [16].

- e. Postprocessing** - The last `global_average_pooling` layer has 1000 classes in all architectures but only 7 classes are required for the facial emotion recognition system. So it is removed from all architectures. The dense layer is added after the `base_output`, which reduced the number of classes to 128. Relu activation function is added with the output, followed by Dense layer to reduce the number of classes to 64. Finally Softmax function is added to reduce the number of classes to 7. Adam optimizer is used for optimization.

- f. Outcome** – Softmax function is applied just before the output layer. It determines the probability of each emotion of an input image and gives the output between 0 and 1. It. The emotion which has maximum probability is the predicted emotion.

5. Experimentation and Result

Performance of Different Architectures - All the 6 architectures is trained with 40 epochs to achieved improved accuracy as an accuracy equation given in Eq. [3].

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}} * 100\% \quad \text{.....Equation 3}$$

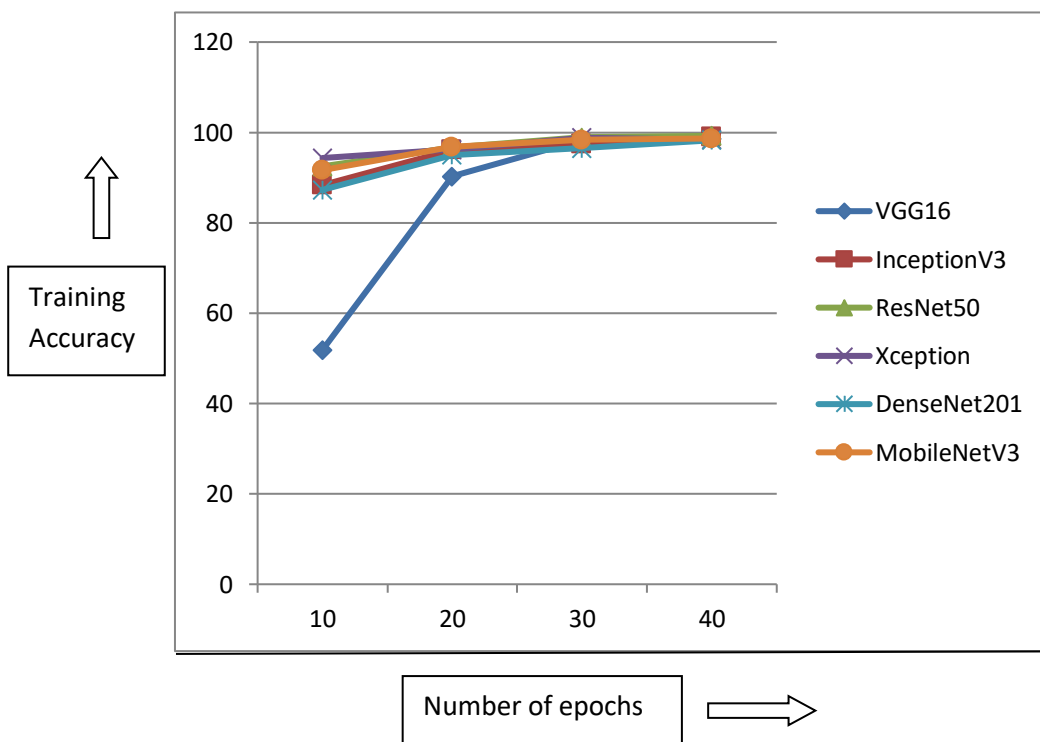


Figure 13. Training-accuracy (with filtering) of different architecture vs. Epoch

From the Fig. [13], it has found that in the first 20 epochs the training accuracy is improved sharply. After 30 epochs the accuracy tends to increase slowly and near 40 epochs the accuracy is almost stops in improving further.

Different Architectures	Testing Accuracy (without filtering)	Testing Accuracy (with filtering)
VGG16	85.51	95.65
InceptionV3	95.65	97.1
ResNet50	95.12	96.81
Xception	81.16	92.75
DenseNet201	97.09	98.55
MobileNetV2	24.64	36.38

Table 2. Testing accuracy results of different architectures

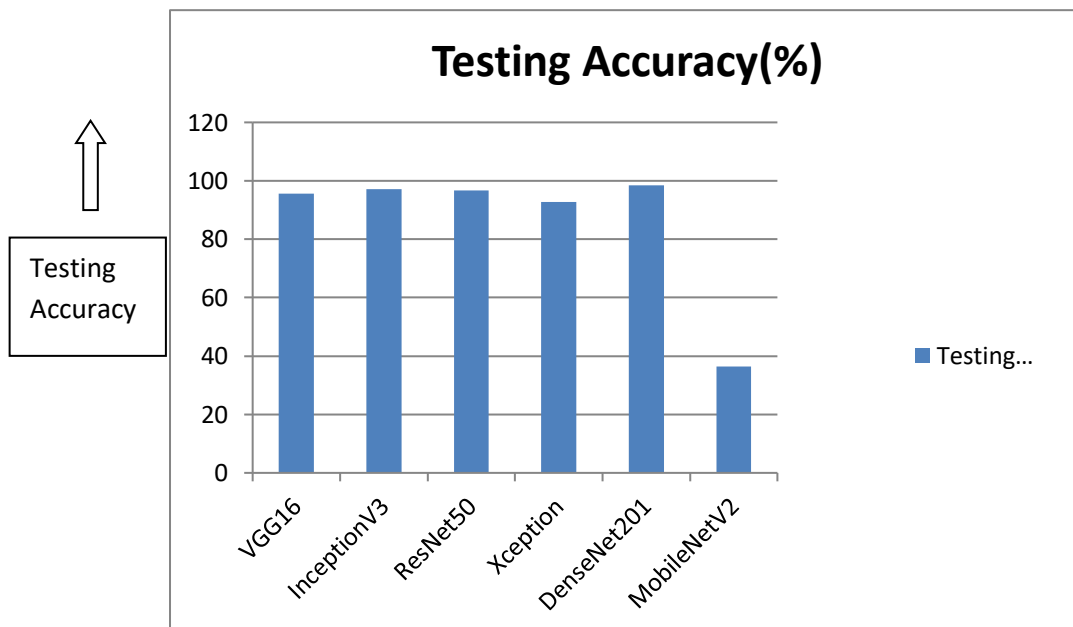


Figure 14. Testing-accuracy (with filtering) vs. different architectures

The testing accuracy with filtering and without filtering of different architectures has shown in table [2] and Fig. [14]. After comparing the accuracy of different architectures, it has found that filtering method increased the accuracy of all architectures. DenseNet201 has achieved max accuracy of 98.55% with filtering and 97.09% without filtering.

6. Conclusion

The proposed approach is based on real-time recognition of facial emotion. Facial expressions of seven different emotions of learners are analyzed. To extract the features, images are pre-processed and then it is trained in different CNN architectures. The performance of different architectures is measured with the same dataset. After comparing the result, it has found that DenseNet201 is more accurate than any other CNN architectures. It has observed that, the accuracy of all architectures has increased using filtering. The accuracy achieved in DenseNet201 is 98.55% using filtering method.

Future Scope

Facial emotion identification is still a difficult problem to solve. Accuracy of different critical applications can be improved by exploring new methods. In the proposed method the system performance can be further enhanced by considering different face angle and illumination factor.

Reference

- [1] A. Almasri, A. Ahmed, N. Al-masri, Y. Sultan, A. Mahmoud, I. Zaqout, A. Akkila and S. Naser, "Intelligent Tutoring Systems Survey for the Period 2000- 2018," *International Journal of Academic Engineering Research (IJAER)*, vol.3, issue 5, pp. 21-37, 2019.
- [2] A. Naser, "Predicting learners performance using artificial neural networks in linear programming intelligent tutoring system," *International Journal of Artificial Intelligence & Applications (IJAAIA)*, vol. 3, no. 2, 2012.
- [3] A. John, A. MC, A. Ajayan, S. Sanoop and V. Kumar, "Real-Time Facial Emotion Recognition System With Improved Preprocessing and Feature Extraction", *Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1328-1333, 2020.
- [4] E. Pranav, S.Kamal and S. Chandran, "Facial Emotion Recognition Using Deep Convolutional Neural Network," *6th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pp. 317-320, 2020.
- [5] R. Madupu, C. Kothapalli, V. Yarra and S. Harika, "Automatic Human Emotion Recognition System using Facial Expressions with Convolution Neural Network," *Fourth International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1179-1183, 2020.
- [6] A. Poulose, C. Reddy, J. Kim and D. Han, "Foreground Extraction Based Facial Emotion Recognition Using Deep Learning Xception Model," *Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 356-360, 2021.
- [7] M. Ozdemir, B.Elagoz, A. Alaybeyoglu and R. Sadighzadeh, "Real Time Emotion Recognition from Facial Expressions Using CNN Architecture," *Medical Technologies National Congress (TIPTKNO)*, pp. 1-4, 2019.
- [8] Y. Zhou, H. Ni, F. Ren and X. Kang, "Face and Gender Recognition System Based on Convolutional Neural networks," *International Conference on Mechatronics and Automation*, pp. 1091-1095, 2019.
- [9] P. Kaviya and T. Arumugaprakash, "Group Facial Emotion Analysis System Using Convolutional Neural Network," *Fourth International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 643-647, 2020.
- [10] S. Pandey and S. Handoo, "Facial Emotion Recognition using Deep Learning," *IEEE International Mobile and Embedded Technology*

Conference (MECON), pp.689-694, 2022.

[11] R. Priya, M. Hanmandlu and S. Vasikarla, "Emotion Recognition using Deep Learning," *IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, vol. 49, pp.69-78, 2021.

[12] R. Gill, J. Singh, "A Deep Learning Approach for Real Time Facial Emotion Recognition," *IEEE 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 497-501, 2021.

[13] L. Sun, X. Shen and J. Dai, "Facial emotion recognition based on LDA and Facial Landmark Detection," *2nd International Conference on Artificial Intelligence and Education (ICAIE)*, pp. 64-67, 2021.

[14] Z. Zhang, M. Yi, J. Xu, R. Zhang and J. Shen, "Two-stage Recognition and Beyond for Compound Facial Emotion Recognition," *IEEE International Conference on Automatic Face and Gesture Recognition*, vol. 10, no. 22, pp. 2847, 2020.

[15] H. Hardjadinata, R. Oetama and I. Prasetiawan, "Facial Expression Recognition Using Xception And DenseNet Architecture," *6th International Conference on New Media Studies (CONMEDIA)*, pp. 60-65, 2021.

[16] F. Zhang, Q. Li, Y. Ren, H. Xu, Y. Song and S. Liu, "An Expression Recognition Method on Robots Based on MobilenetV2-SSD," *IEEE 6th International Conference on Systems and Informatics (ICSAI)*, pp. 118-122, 2019.

Appendix


Overview of Intelligent Tutoring System as in Fig. [15-18].



Figure 15. Front Page

ALEKS

Sign Up



Firstname :

Lastname :

Gender : Male Female

Email :

Password :

[Already Registered](#) [Log In](#)

Figure 16. Sign Up Page

ALEKS

Login



Email:

Password:

[New Registration](#)

Figure. 17 Login Page

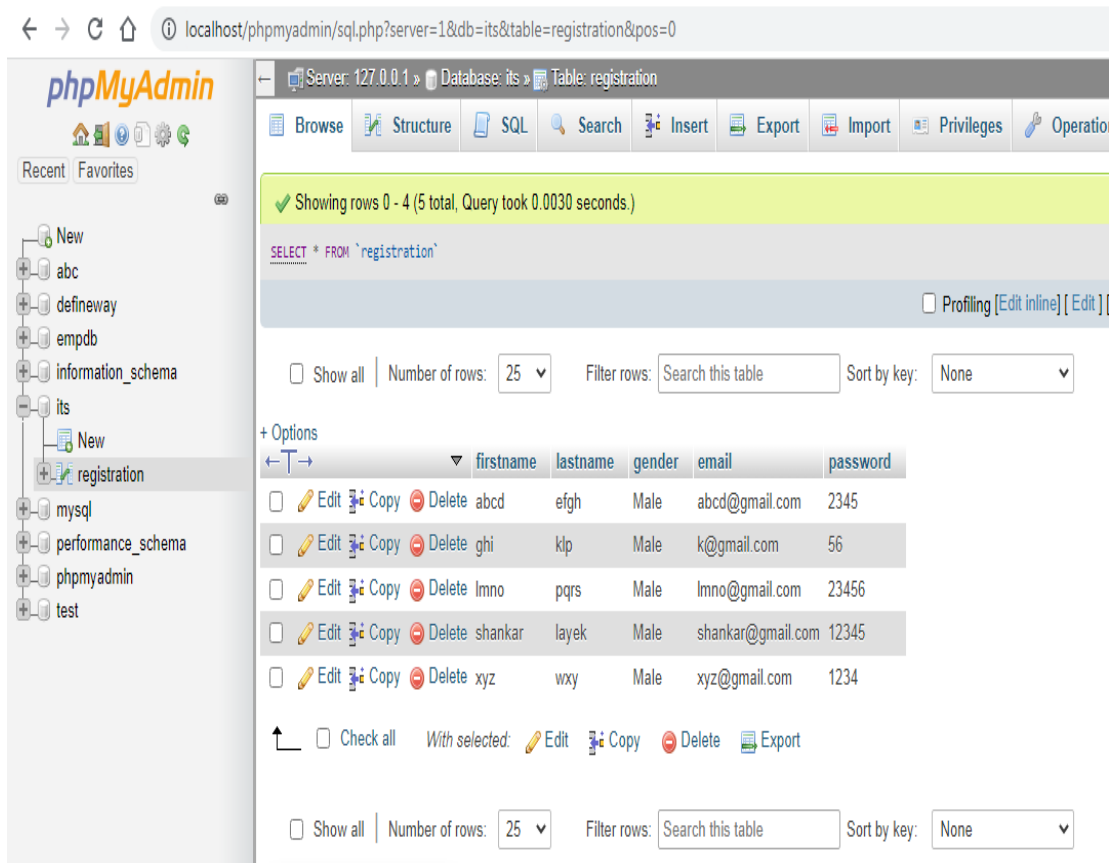


Figure. 18 Database

Facial Emotion Recognition using DenseNet201 Architecture with filtering

```
✓ [1] from google.colab import drive  
3s drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
✓ [2] import tensorflow as tf  
8s import cv2  
import os  
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
from google.colab.patches import cv2_imshow  
from skimage import io  
from PIL import Image
```

```
✓ [3] gray_image = io.imread("/content/drive/MyDrive/train.zip (Unzipped Files)/train/0/0.png")  
0s
```

```
✓ [4] im2 = 255 - gray_image  
0s
```

```
✓ [5] im3 = (100.0/255)*gray_image + 100  
0s
```

```
✓ [6] im4 = 255.0*(gray_image/255.0)**2  
0s
```

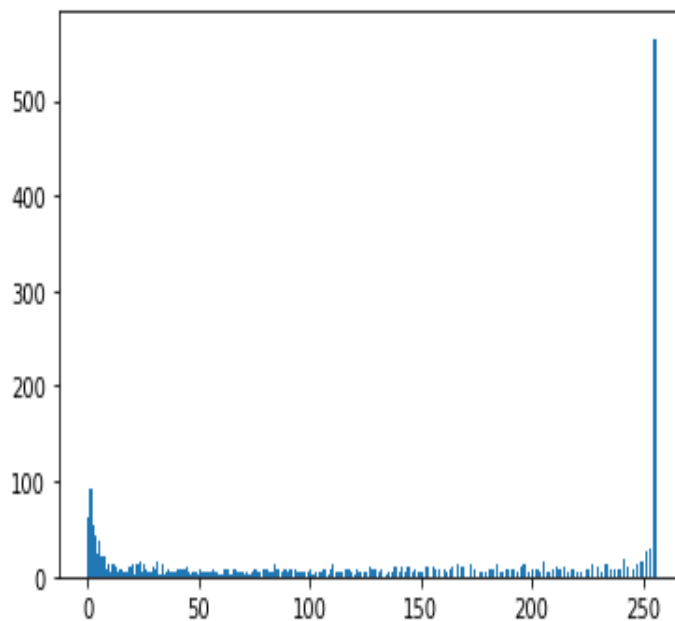
```

✓ [7] def histeq(im, nbr_bins = 256):
0s      """ Histogram equalization of a grayscale image. """
      imhist, bins = np.histogram(im.flatten(), nbr_bins, [0, 256])
      cdf = imhist.cumsum()
      cdf = imhist.max()*cdf/cdf.max()
      cdf_mask = np.ma.masked_equal(cdf, 0)
      cdf_mask = (cdf_mask - cdf_mask.min())*255/(cdf_mask.max() - cdf_mask.min())
      cdf = np.ma.filled(cdf_mask, 0).astype('uint8')
      return cdf[im.astype('uint8')]

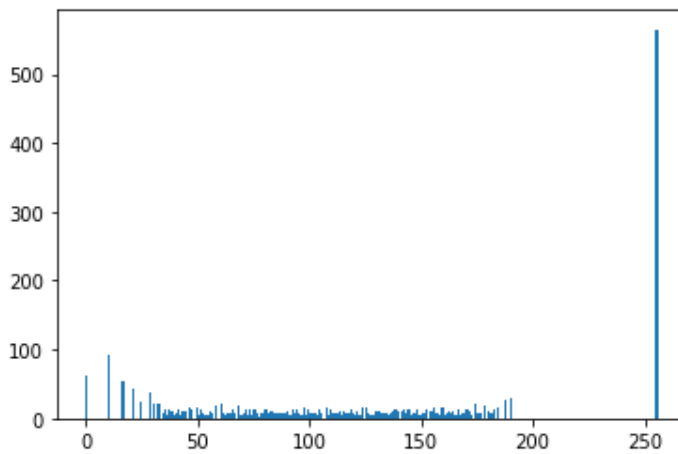
img_array = histeq(im4)
    
```

```

✓ ▶ plt.hist(im4.ravel(), bins = 256, range = [0, 256])
0s    plt.show()
    
```



```
✓ [9] plt.hist(img_array.ravel(), bins = 256, range = [0, 256])
1s plt.show()
```



```
✓ [10] img_array.shape
0s (48, 48)
```

```
✓ [11] print (img_array)
0s
[[38 28 21 ... 37 46 51]
 [32 21 16 ... 35 40 44]
 [32 28 21 ... 35 41 44]
 ...
 [16 46 60 ... 51 75 68]
 [ 0 16 55 ... 52 73 71]
 [ 0  0 36 ... 53 70 68]]
```

```
✓ [12] Datadirectory_train = "/content/drive/MyDrive/train.zip (Unzipped Files)/train"
0s
```

```
✓ [13] Datadirectory_test = "/content/drive/MyDrive/test.zip (Unzipped Files)/test"
0s
```

```
✓ [14] Classes_train = ["0", "1", "2", "3", "4", "5", "6"]
0s
```

```
✓ [15] Classes_test = ["0", "1", "2", "3", "4", "5", "6"]
0s
```

```
✓ [16] for category in Classes_train:
0s     path = os.path.join(Datadirectory_train, category)
     for img in os.listdir(path):
         img_array_train = cv2.imread(os.path.join(path, img))
         break
     break
```

```

✓ [17] for category in Classes_test:
0s     path = os.path.join(Datadirectory_test, category)
        for img in os.listdir(path):
            img_array_test = cv2.imread(os.path.join(path,img))
            break
        break

✓ [18] img_size= 224
0s     new_array_train= cv2.resize(img_array_train, (img_size,img_size))

✓ [19] img_size= 224
0s     new_array_test= cv2.resize(img_array_test, (img_size,img_size))

✓ [20] new_array_train.shape
0s
        (224, 224, 3)

✓ [21] new_array_test.shape
0s
        (224, 224, 3)

✓ [22] training_Data = []
0s

def create_training_Data():
    for category in Classes_train:
        path = os.path.join(Datadirectory_train, category)
        class_num = Classes_train.index(category)
        for img in os.listdir(path):
            try:
                gray_image = cv2.imread(os.path.join(path,img))
                im2 = 255 - gray_image
                im3 = (100.0/255)*gray_image + 100
                im4 = 255.0*(gray_image/255.0)**2
                def histeq(im, nbr_bins = 256):
                    imhist, bins = np.histogram(im.flatten(), nbr_bins, [0, 256])
                    cdf = imhist.cumsum()
                    cdf = imhist.max()*cdf/cdf.max()
                    cdf_mask = np.ma.masked_equal(cdf, 0)
                    cdf_mask = (cdf_mask - cdf_mask.min())*255/(cdf_mask.max() - cdf_mask.min())
                    cdf = np.ma.filled(cdf_mask, 0).astype('uint8')
                    return cdf[im.astype('uint8')]
                im5 = histeq(im4)
                new_array_train= cv2.resize(im5, (img_size,img_size))
                training_Data.append([new_array_train,class_num])
            except Exception as e:
                pass
    
```



```

✓ [23] testing_Data = []
0s

def create_testing_Data():
    for category in Classes_test:
        path = os.path.join(Datadirectory_test, category)
        class_num = Classes_test.index(category)
        for img in os.listdir(path):
            try:
                gray_image = cv2.imread(os.path.join(path,img))
                im2 = 255 - gray_image
                im3 = (100.0/255)*gray_image + 100
                im4 = 255.0*(gray_image/255.0)**2
            def histeq(im, nbr_bins = 256):
                imhist, bins = np.histogram(im.flatten(), nbr_bins, [0, 256])
                cdf = imhist.cumsum()
                cdf = imhist.max()*cdf/cdf.max()
                cdf_mask = np.ma.masked_equal(cdf, 0)
                cdf_mask = (cdf_mask - cdf_mask.min())*255/(cdf_mask.max() - cdf_mask.min())
                cdf = np.ma.filled(cdf_mask, 0).astype('uint8')
                return cdf[im.astype('uint8')]
            im5 = histeq(im4)
            new_array_test= cv2.resize(im5, (img_size,img_size))
            testing_Data.append([new_array_test,class_num])
        except Exception as e:
            pass
    
```

```

✓ [24] create_training_Data()
16s

print(len(training_Data))
    
```

1787

```

✓ [25] create_testing_Data()
14s

print(len(testing_Data))
    
```

69

```

✓ [26] temp_train = np.array(training_Data)
0s
    
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: VisibleDeprecationWarning: Creating an ndarray
 """Entry point for launching an IPython kernel.

```

✓ [27] temp_test = np.array(testing_Data)
0s
    
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: VisibleDeprecationWarning: Creating an ndarray
 """Entry point for launching an IPython kernel.

```
✓ [28] temp_train.shape
0s
      (1787, 2)
```

```
✓ [29] temp_test.shape
0s
      (69, 2)
```

```
✓ [30] import random
0s
      random.shuffle(training_Data)
```

```
✓ [31] import random
0s
      random.shuffle(testing_Data)
```

```
✓ [32] X_train = []
0s      y_train = []
      for features,label in training_Data:
          X_train.append(features)
          y_train.append(label)

      X_train = np.array(X_train).reshape(-1, img_size, img_size, 3)
```

```
✓ [33] X_test = []
0s      y_test = []
      for features,label in testing_Data:
          X_test.append(features)
          y_test.append(label)

      X_test = np.array(X_test).reshape(-1, img_size, img_size, 3)
```

```
✓ [34] X_train.shape
0s
      (1787, 224, 224, 3)
```

```
✓ [35] X_test.shape
0s
      (69, 224, 224, 3)
```

```
✓ [36] X_train = X_train/255.0;
1s
```

```
✓ [37] X_test = X_test/255.0;
0s
```

```
✓ [38] type(y_train)
0s
      list
```

```

0s [39] type(y_test)
      list

0s [40] y_train[0]
      4

0s [41] y_test[0]
      2

0s [42] Y_train= np.array(y_train)

0s [43] Y_test = np.array(y_test)

0s [44] Y_train.shape
      (1787,)

0s [45] Y_test.shape
      (69,)

0s [46] import tensorflow as tf
      from tensorflow import keras
      from tensorflow.keras import layers

0s [47] model = tf.keras.applications.DenseNet201()

      Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201
      82526208/82524592 [=====] - 1s 0us/step
      82534400/82524592 [=====] - 1s 0us/step

21s [48] model.summary()

      conv5_block30_2_conv (Conv2D) (None, 7, 7, 32) 36864 ['conv5_block30_1_relu[0][0]']
      conv5_block30_concat (Concatenate) (None, 7, 7, 1856) 0 ['conv5_block29_concat[0][0]',
      'conv5_block30_2_conv[0][0]']
      conv5_block31_0_bn (BatchNormalization) (None, 7, 7, 1856) 7424 ['conv5_block30_concat[0][0]']
      conv5_block31_0_relu (Activation) (None, 7, 7, 1856) 0 ['conv5_block31_0_bn[0][0]']
      conv5_block31_1_conv (Conv2D) (None, 7, 7, 128) 237568 ['conv5_block31_0_relu[0][0]']
    
```

```

✓ [48] conv5_block32_concat (Concaten (None, 7, 7, 1920) 0 ['conv5_block31_concat[0][0]',
21s ate) 'conv5_block32_2_conv[0][0]']

bn (BatchNormalization) (None, 7, 7, 1920) 7680 ['conv5_block32_concat[0][0]']

relu (Activation) (None, 7, 7, 1920) 0 ['bn[0][0]']

avg_pool (GlobalAveragePooling (None, 1920) 0 ['relu[0][0]']
2D)

predictions (Dense) (None, 1000) 1921000 ['avg_pool[0][0]']

```

```

=====
Total params: 20,242,984
Trainable params: 20,013,928
Non-trainable params: 229,056
=====

```

```

✓ [49] base_input = model.layers[0].input
1s

```

```

✓ [50] base_output = model.layers[-2].output
1s

```

```

✓ [51] base_output
1s

```

```

<KerasTensor: shape=(None, 1920) dtype=float32 (created by layer 'avg_pool')>

```

```

✓ [52] final_output = layers.Dense(128)(base_output)
0s
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(64)(final_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(7,activation='softmax')(final_output)

```

```

✓ [53] final_output
0s

```

```

<KerasTensor: shape=(None, 7) dtype=float32 (created by layer 'dense_2')>

```

```

✓ [54] new_model = keras.Model(inputs = base_input, outputs = final_output)
0s

```

```

✓ [55] new_model.summary()
18s

```

```

conv5_block30_concat (Concaten (None, 7, 7, 1856) 0 ['conv5_block29_concat[0][0]',
ate) 'conv5_block30_2_conv[0][0]']

conv5_block31_0_bn (BatchNorma (None, 7, 7, 1856) 7424 ['conv5_block30_concat[0][0]']
lization)

```

```

✓ [55] activation (Activation)      (None, 128)      0      ['dense[0][0]']
18s
      dense_1 (Dense)              (None, 64)      8256   ['activation[0][0]']
      activation_1 (Activation)    (None, 64)      0      ['dense_1[0][0]']
      dense_2 (Dense)              (None, 7)       455    ['activation_1[0][0]']
    
```

```

=====
Total params: 18,576,583
Trainable params: 18,347,527
Non-trainable params: 229,056
    
```

```

✓ [56] new_model.compile(loss="sparse_categorical_crossentropy", optimizer = "adam", metrics = ["accuracy"])
0s
    
```

```

✓ ▶ Y_train[20]
0s
    
```

```

↳ 6
    
```

```

✓ ▶ new_model.fit(X_train, Y_train, epochs = 40)
21m
    
```

```

↳ Epoch 1/40
56/56 [=====] - 76s 665ms/step - loss: 1.1819 - accuracy: 0.5280
Epoch 2/40
56/56 [=====] - 32s 569ms/step - loss: 0.8891 - accuracy: 0.6618
Epoch 3/40
56/56 [=====] - 31s 561ms/step - loss: 0.7673 - accuracy: 0.7016
Epoch 4/40
56/56 [=====] - 31s 557ms/step - loss: 0.6830 - accuracy: 0.7436
Epoch 5/40
56/56 [=====] - 31s 562ms/step - loss: 0.6102 - accuracy: 0.7727
Epoch 6/40
56/56 [=====] - 31s 559ms/step - loss: 0.5187 - accuracy: 0.8001
Epoch 7/40
56/56 [=====] - 31s 560ms/step - loss: 0.4846 - accuracy: 0.8208
Epoch 8/40
56/56 [=====] - 31s 561ms/step - loss: 0.4330 - accuracy: 0.8415
Epoch 9/40
56/56 [=====] - 31s 563ms/step - loss: 0.3498 - accuracy: 0.8639
Epoch 10/40
56/56 [=====] - 31s 560ms/step - loss: 0.3522 - accuracy: 0.8729
Epoch 11/40
56/56 [=====] - 31s 559ms/step - loss: 0.3429 - accuracy: 0.8841
Epoch 12/40
56/56 [=====] - 31s 558ms/step - loss: 0.2145 - accuracy: 0.9188
Epoch 13/40
56/56 [=====] - 31s 559ms/step - loss: 0.2365 - accuracy: 0.9171
Epoch 14/40
56/56 [=====] - 31s 560ms/step - loss: 0.2069 - accuracy: 0.9261
    
```

```

21m ✓ 56/56 [=====] - 31s 559ms/step - loss: 0.1410 - accuracy: 0.9507
Epoch 18/40
56/56 [=====] - 31s 560ms/step - loss: 0.1289 - accuracy: 0.9563
Epoch 19/40
56/56 [=====] - 31s 559ms/step - loss: 0.1402 - accuracy: 0.9485
Epoch 20/40
56/56 [=====] - 31s 559ms/step - loss: 0.1413 - accuracy: 0.9502
Epoch 21/40
56/56 [=====] - 31s 559ms/step - loss: 0.1092 - accuracy: 0.9630
Epoch 22/40
56/56 [=====] - 31s 559ms/step - loss: 0.0947 - accuracy: 0.9670
Epoch 23/40
56/56 [=====] - 31s 559ms/step - loss: 0.0764 - accuracy: 0.9731
Epoch 24/40
56/56 [=====] - 31s 559ms/step - loss: 0.0742 - accuracy: 0.9737
Epoch 25/40
56/56 [=====] - 31s 559ms/step - loss: 0.0879 - accuracy: 0.9703
Epoch 26/40
56/56 [=====] - 31s 560ms/step - loss: 0.0667 - accuracy: 0.9787
Epoch 27/40
56/56 [=====] - 31s 559ms/step - loss: 0.0842 - accuracy: 0.9709
Epoch 28/40
56/56 [=====] - 31s 559ms/step - loss: 0.0894 - accuracy: 0.9703
Epoch 29/40
56/56 [=====] - 31s 559ms/step - loss: 0.0676 - accuracy: 0.9782
Epoch 30/40
56/56 [=====] - 31s 559ms/step - loss: 0.1007 - accuracy: 0.9653
Epoch 31/40
56/56 [=====] - 31s 558ms/step - loss: 0.0806 - accuracy: 0.9714

21m ✓ [58] Epoch 32/40
56/56 [=====] - 31s 560ms/step - loss: 0.0354 - accuracy: 0.9894
Epoch 33/40
56/56 [=====] - 31s 559ms/step - loss: 0.0154 - accuracy: 0.9938
Epoch 34/40
56/56 [=====] - 31s 559ms/step - loss: 0.0285 - accuracy: 0.9916
Epoch 35/40
56/56 [=====] - 31s 559ms/step - loss: 0.0232 - accuracy: 0.9922
Epoch 36/40
56/56 [=====] - 31s 559ms/step - loss: 0.0307 - accuracy: 0.9916
Epoch 37/40
56/56 [=====] - 31s 559ms/step - loss: 0.0648 - accuracy: 0.9793
Epoch 38/40
56/56 [=====] - 31s 558ms/step - loss: 0.0799 - accuracy: 0.9742
Epoch 39/40
56/56 [=====] - 31s 558ms/step - loss: 0.0895 - accuracy: 0.9742
Epoch 40/40
56/56 [=====] - 31s 559ms/step - loss: 0.0440 - accuracy: 0.9826
<keras.callbacks.History at 0x7f9a3c0e0d90>

5s ✓ new_model.evaluate(X_test, Y_test)
3/3 [=====] - 5s 979ms/step - loss: 0.0660 - accuracy: 0.9855
[0.06603308767080307, 0.9855072498321533]

```

```
frame = cv2.imread("/content/drive/MyDrive/happy boy (2).jpg")
```

```
frame.shape
```

```
(533, 800, 3)
```

```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f3584eee250>
```



```
face_Cascade = cv2.CascadeClassifier('/content/drive/MyDrive/haarcascade_frontalface_default.xml')
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
gray.shape
```

```
(533, 800)
```

```
face_Cascade.empty()
```

```
False
```

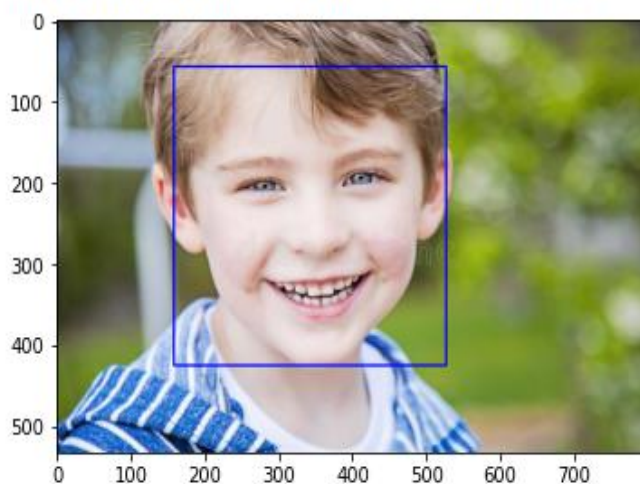
```
faces = face_Cascade.detectMultiScale(gray,1.1,4)
for x,y,w,h in faces:
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = frame[y:y+h, x:x+w]
    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
    faces = face_Cascade.detectMultiScale(roi_gray)
    if len(faces) == 0:
        print("Face not detected")
    else:
        for (ex,ey,ew,eh) in faces:
            face_roi = roi_color[ey: ey+eh, ex:ex + ew]
```

```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f35851c9650>
```

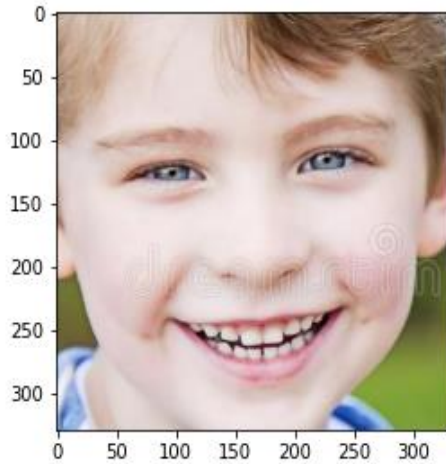
```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f35851c9650>
```




```
plt.imshow(cv2.cvtColor(face_roi, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f3584dff750>
```



```
final_image = cv2.resize(face_roi, (224,224))
final_image = np.expand_dims(final_image, axis = 0)
final_image = final_image/255.0
```

```
Predictions = new_model.predict(final_image)
```

```
P = np.round(Predictions, 4)
```

```
P
```

```
array([[0.    , 0.    , 0.0115, 0.9885, 0.    , 0.    , 0.    ]],
      dtype=float32)
```

```
np.argmax(Predictions)
```

```
3
```

.....THE END.....