



Department of Computer Science and Engineering  
College of Engineering, Guindy  
Anna University, Chennai – 600 025

## **EMOTION DETECTION FROM** **FACIAL EXPRESSION AND HEART RATE**

<b>NAME</b>	<b>REG.NO</b>	<b>E-MAIL</b>	<b>MOBILE NO</b>
MURALI R	2019103038	muralirajendran835@g mail.com	9361176681
RAGHURAJ S V	2019103563	raghusvr32@gmail.com	9750723889

**INSTRUCTOR: Dr AROCKIA XAVIER ANNIE R**

## Contents

<b>1.ABSTRACT</b>	4
<b>2.INTRODUCTION</b>	4
<b>3.LITERATURE SURVEY</b>	5
<b>4. PROBLEM STATEMENT</b>	5
<b>5. PROBLEM SOLUTION</b>	
5.1 Data Set:-	6
5.2 Input :-	6
5.3 Approach:-	7
<b>6. ARCHITECTURE</b>	
6.1 Abstract Architecture	7
6.2 Explanation of The Architecture	8
<b>7. DETAILED EXPLAiNATION</b>	
7.1 Preprocessing	9
7.2 Feature Extraction	9
7.3 Model Training	10
7.4 Classification	10
<b>8. IMPLEMENTATION</b>	
8.1 Initial Set-up:	10
8.2 Code:	10
8.2.1 Importing required modules	11
8.2.2 Reading data file	12
8.2.3 Emotion prediction function	12
8.2.4 Preprocessing data	13
8.2.5 Building CNN model	15
8.2.6 Training the CNN model	15
8.2.7 Saving CNN model	16
8.2.8 Building SVM model for HR	16

---

8.2.9 Code to predict output	17
<b>9. RESULT AND ANALYSIS</b>	
9.1 Output from code	18
9.2 Accuracy	19
9.3 Accuracy and loss graph for CNN	19
9.4 Accuracy of particular emotion	19
<b>10. CONCLUSION AND FUTURE ENHANCEMENTS</b>	
10.1 Conclusion	19
10.2 Future Enhancements	20
<b>11. LIST OF REFERENCES</b>	20
<b>12.APPENDIX A:</b>	21
<b>13.APPENDIX B: ABBREVIATIONS</b>	21

## 1.ABSTRACT

Facial expression recognition is a topic of great interest in most fields from artificial intelligence and gaming to marketing and healthcare. The goal of our project is to classify images of human faces into one of seven basic emotions. A number of different models were experimented with, including decision trees and neural networks before arriving at a final Convolutional Neural Network (CNN) model. CNNs work better for image recognition tasks since they are able to capture special features of the inputs due to their large number of filters. The proposed model consists of six convolutional layers, two max pooling layers and two fully connected layers. Upon tuning of the various hyper parameters, this model achieved a final accuracy of 0.60. We also aim in predicting the emotion from heart rate using the SVM algorithm.

## 2.INTRODUCTION

Facial emotions are important factors in human communication that help to understand the intentions of others. In general, people infer the emotional state of other people, such as joy, sadness and anger, using facial expressions and vocal tones. Facial expressions are one of the main information channels in interpersonal communication. Therefore, it is natural that facial emotion research has gained a lot of attention over the past decade with applications in perceptual and cognitive sciences . Interest in automatic Facial Emotion Recognition (FER) has also been increasing recently with the rapid development of Artificial Intelligent (AI) techniques. They are now used in many applications and their exposure to humans is increasing. To improve Human Computer Interaction (HCI) and make it more natural, machines must be provided with the capability to understand the surrounding environment, especially the intentions of humans. Machines can capture their environment state through cameras and sensors. In recent years, Deep Learning (DL) algorithms have proven to be very successful in capturing environment states . Emotion detection is necessary for machines to better serve their purpose since they deliver information about the inner state of humans. A machine can use a sequence of facial images with DL techniques to determine human emotions.

### **3.LITERATURE SURVEY:**

Gargesha and Kuchi [1] has proposed a approach based on Multi Layer Perceptrons and Radial Basis Function Networks (MLP and RBF networks) which classify the seven basic types of emotions: Neutral, Happiness, Sadness, Anger, Fear, Surprise and Disgust. The geometric coordinates of the Facial Characteristic Points (FCPs) for the image, Euclidean distances for the contour points for the image and the differences in inter-feature distances are the only data fed into the neural network. Since the classification is done from the given image and the neutral face, the approach is based on the assumption that the neutral face image corresponding to each image is available to the system.

Lisetti and Rumelharts' neural network [2] classifies emotions based on signaled emotions and the level of expressiveness. The neural network is capable of dealing with the areas in the face which can carry out independent muscle movements: brow/forehead, eyelids and base of nose. The hidden layer in the neural network is connected with each part of the face and with each output unit. This approach works on two emotions: neutral and smiling in order to study how variations in expressiveness affect the overall performance of the system.

Raghuvanshi A. et al have built a Facial expression recognition system upon recent research to classify images of human faces into discrete emotion categories using convolutional neural networks [3].

Alizadeh, Shima, and Azar Fazel have developed a Facial Expression Recognition system using Convolutional Neural Networks based on the Torch model [4].

### **4.PROBLEM STATEMENT**

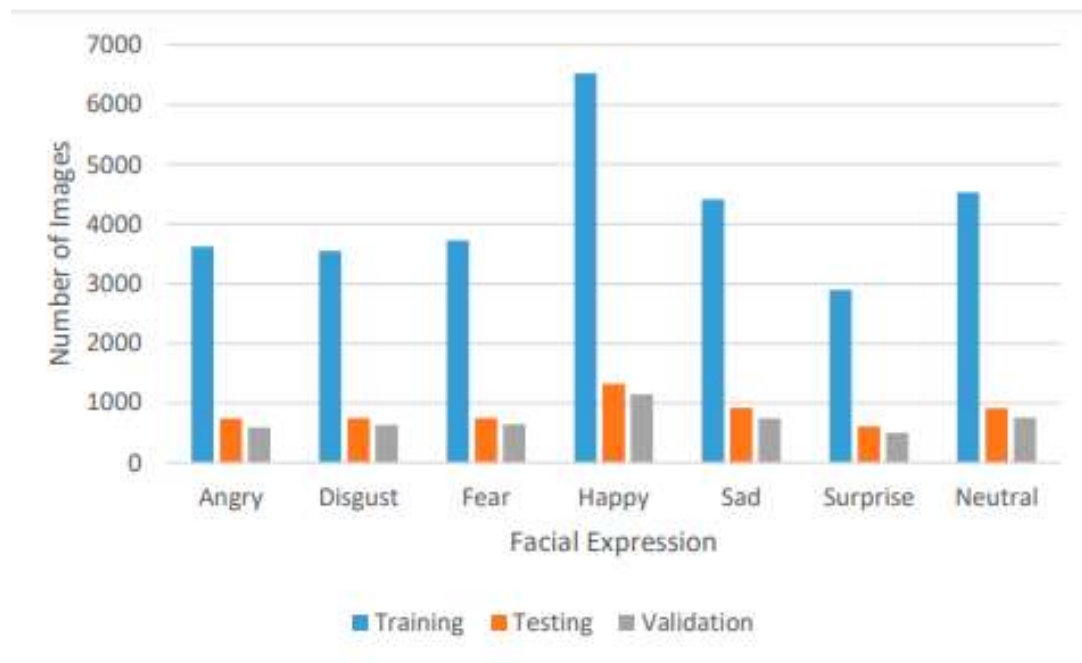
Human emotions and intentions are expressed through facial expressions and deriving an efficient and effective feature is the fundamental component of the facial expression system. Automatic recognition of facial expressions can be an important component of natural human-machine interfaces; it may also be used in behavioral science and in clinical practice. An automatic Facial Expression Recognition system needs to solve the following problems: detection and location of faces in a cluttered scene, facial feature extraction, and facial expression classification. There is also a relationship between heart rate and emotions. It aims to find the emotions for a particular heart rate.

## 5. PROBLEM SOLUTION

### 5.1 Data Set:-

The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) is used for the training and testing. It comprises pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. Dataset has a training set of 35887 facial images with facial expression labels.. The dataset has class imbalance issues, since some classes have a large number of examples while some have few. We customized the dataset by adding heart rate for each entry.

The first 2304 columns represent the 48x48 pixels. The 2305th column has the heart rate.



### 5.2 Input :-

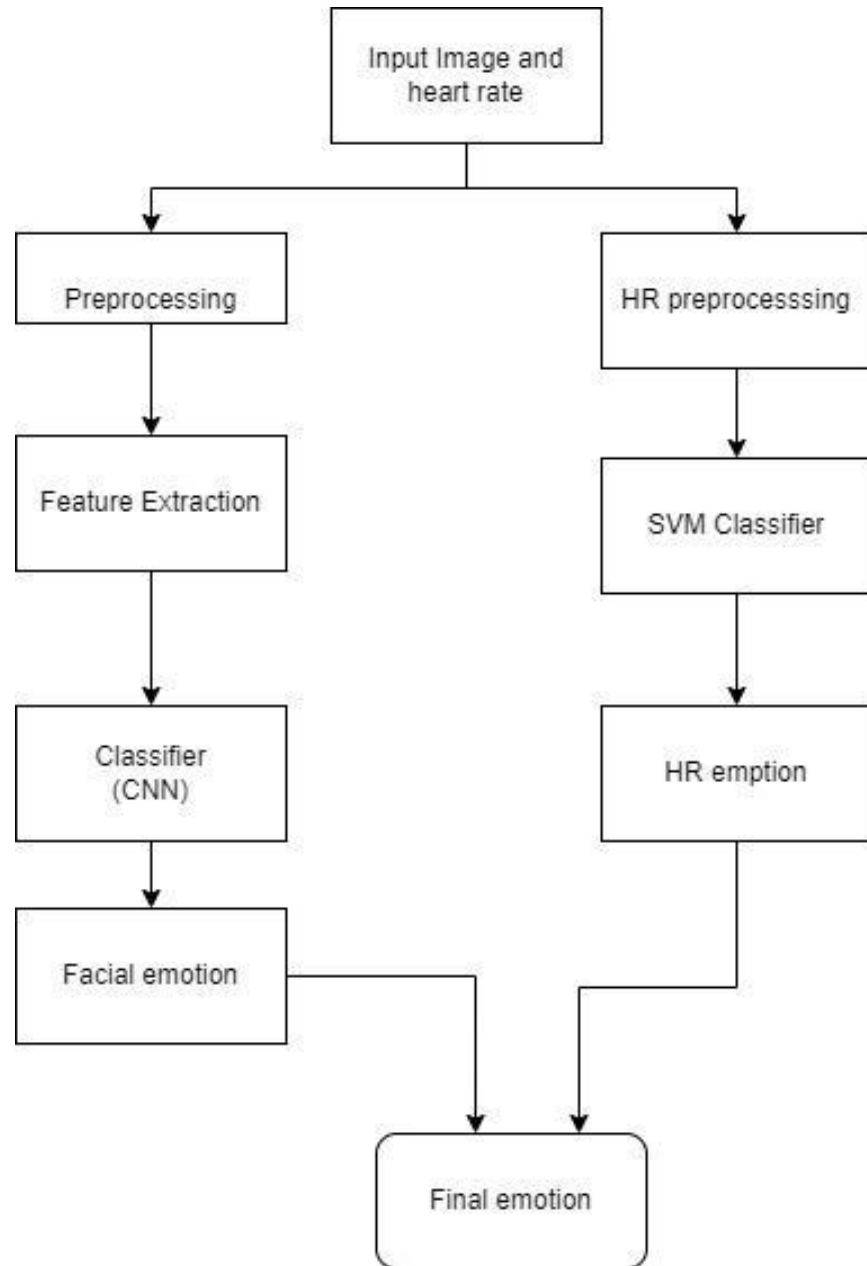
The input is a 48x48 pixel image and heart rate.

### 5.3 Approach:-

We are using support vector machine (SVM) and Convolutional Neural Network(CNN) algorithms.

## 6. ARCHITECTURE

### 6.1 ABSTRACT ARCHITECTURE



## **6.2 EXPLANATION OF THE ARCHITECTURE**

### **CNN**

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the letter's input matrix. This product is usually the Fresenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

### **Support Vector Machine (SVM) Algorithm**

Support vector machine is by many a simple and highly favoured algorithm, as it produces large accuracy with less computing power. The support vector machine algorithm's objective is to locate a hyper plane in an N-dimensional space (N- the number of features) that separately identifies the relevant data. Data was first split into training and testing data. Then, the SVM model was built. Since SVM is a binary algorithm, one of the strategies has been used to reduce the multiclass classification problem to multiple binary classification problem, which is a one vs all strategy. This technique includes learning a single classifier per class, with the class instances being positive instances and the remainder instances being negative. This layout allows the ground classifiers to generate a real-valued trust result for its decision, instead of only a class label.



## **7. DETAILED EXPLANATION**

### **7.1 PREPROCESSING:**

This module involves splitting the records in the .csv file into input and output and thus helping us train and design the model. The first 2304 columns in the dataset represent the 48x48 pixels of the face image. The 2305th column represents the heart rate for each record. The remaining columns represent the output i.e., the emotion for a particular entry.

#### **Image pre-processing:**

First, 1-2304 column data is extracted and is reshaped to ( 48, 48) ,so that it is converted into image and can be fed as input for the CNN model. The columns 2306-2312 have a value '1' among them in a particular column and the rest of the columns are '0'.The index at which the value '1' is present indicates the emotion. For example, if '1' is at the 0th index, its "angry". They serve as labels in training the CNN model.

#### **HR preprocessing:**

The 2305th column data is extracted, which serves as input to the SVM model to predict emotion from heart rate. The 2313th column has numbers from 1-7, each representing a particular emotion. It serves as a label for SVM models.

### **7.2FEATURE EXTRACTION**

Feature Extraction is one of the most interesting steps of image processing to reduce the efficient part of an image or dimensionality reduction of interesting parts of an image as a compact feature vector. Feature reduction representation is useful when the image size is large and required to rapidly complete tasks such as image matching and retrieval. This part is handled by our CNN Layer which is used for feature detection and for selection we use Max Pooling and dropout Layer.

### **7.3 MODEL TRAINING**

It is done after model curation which is done on the basis of all above information since we are using deep learning most of the task or almost all tasks are handled by our model itself so to perform all above operation we define layers in our model. After curation of Model we train our model by setting our Hyper-parameters. After training part is performed our model is ready for the process of Image Classification.

### **7.4 CLASSIFICATION**

It consists of a model that contains predefined patterns that are compared with detected objects to classify them in a proper category. Classification will be executed on the basis of spectral defined features such as density, texture etc. Image classification is performed using Convolution Neural Networks and Deep Learning. Heart rate classification is performed using SVM.

## **8. IMPLEMENTATION**

### **8.1 INITIAL SET-UP:**

Google Colab:

- Open Google Colab in your browser (windows, Linux, Mac).

## 8.2 CODE SNIPPETS:

### 8.2.1 Importing required modules

```
[1] from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg19 import VGG19
from keras.layers import Flatten, Dense, Dropout, Conv2D, MaxPool2D, BatchNormalization, LSTM, CuDNNLSTM, MaxPooling2D
from keras.models import Sequential
from keras import regularizers
import keras

from keras.layers import TimeDistributed
from keras.layers import GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.optimizers import Adam
import tensorflow as tf

import matplotlib.pyplot as plt
import cv2
import numpy as np
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Code Text

## 8.2.2 Reading data file

```
✓ 34s ▶ #LOADING DATASET FROM DRIVE
trainingset = np.loadtxt("drive/My Drive/fer2013_training_onehot.csv",delimiter=',')
testingset = np.loadtxt("drive/My Drive/fer2013_publictest_onehot (1).csv",delimiter=',')
print("training set")
print(trainingset)

print("testing set")
print(testingset)
```

training set

```
[[ 70.  80.  82. ...  0.  0.  1.]
 [151. 150. 147. ...  0.  0.  1.]
 [231. 212. 156. ...  0.  0.  3.]
 ...
 [ 74.  81.  87. ...  0.  0.  5.]
 [222. 227. 203. ...  0.  0.  1.]
 [195. 199. 205. ...  0.  0.  5.]]
```

testing set

```
[[254. 254. 254. ...  0.  0.  1.]
 [156. 184. 198. ...  0.  0.  2.]
 [ 69. 118.  61. ...  0.  0.  5.]
 ...
 [255. 255. 255. ...  0.  0.  5.]
 [ 33.  25.  31. ...  0.  0.  5.]
 [ 61.  63.  59. ...  0.  0.  5.]]
```

## 8.2.3 Emotion prediction function

```
▶ def get_emotion(indx):

    if indx == 0:
        return 'angry'
    elif indx == 1:
        return 'disgust'
    elif indx == 2:
        return 'fear'
    elif indx == 3:
        return 'happy'
    elif indx == 4:
        return 'sad'
    elif indx == 5:
        return 'surprise'
    elif indx == 6:
        return 'neutral'
```

## 8.2.4 Preprocessing data

```
▶ n_inputs = 2304
  n_classes = 7
  img_dim = 48

  #SPLITTING DATASET INTO TRAINING AND TESTING FOR FACIAL IMAGE
  x_training = trainingset[:, 0:n_inputs]
  y_training = trainingset[:, n_inputs+1:n_inputs + n_classes+1]

  x_testing = testingset[:,0:n_inputs]
  y_testing = testingset[:, n_inputs + 1 : n_inputs + n_classes + 1]

  x_training = x_training.reshape(x_training.shape[0], 48, 48)
  x_training = np.expand_dims(x_training, axis=3)

  x_testing = x_testing.reshape(x_testing.shape[0], 48, 48)
  x_testing = np.expand_dims(x_testing, axis=3)

  ##SPLITTING DATASET INTO TRAINING AND TESTING FOR HEART RATE
  x_training1 = trainingset[:, n_inputs]
  y_training1 = trainingset[:, -1]

  x_testing1 = testingset[:, n_inputs]
  y_testing1 = testingset[:, -1]
```

```

▶ print(x_training.shape)
  print(y_training.shape)

  print(x_testing.shape)
  print(y_testing.shape)

  sample = x_training[5, :]
  sample= sample.reshape(48, 48)

  plt.imshow(sample, cmap='gray')
  plt.show()

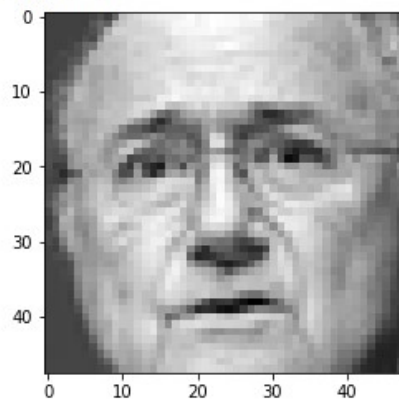
  print('facial emotion: %s' % get_emotion(np.argmax(y_training[5, :])))

```

```

↳ (28709, 48, 48, 1)
   (28709, 7)
   (3589, 48, 48, 1)
   (3589, 7)

```



facial emotion: fear

## 8.2.5 Building CNN model

```
#BUILDING CNN MODEL FOR IMAGE PROCESSING
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape = (48,48,1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_regularizer=regularizers.L2(0.001)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))
opt = Adam(learning_rate=0.0001, decay=10e-6)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

## 8.2.6 Training the CNN model

```
#TRAINING THE CNN MODEL
batch_size = 128
n_epochs = 500
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20)

History = model.fit(x_training, y_training, batch_size=batch_size, epochs=n_epochs,
                    validation_data=(x_testing, y_testing), shuffle=True,
                    callbacks=[early_stopping])
```

```
Epoch 24/500
225/225 [=====] - 21s 95ms/step - loss: 0.6802 - accuracy: 0.9553 - val_loss: 2.2326 - val_accuracy: 0.6043
Epoch 25/500
225/225 [=====] - 21s 95ms/step - loss: 0.6627 - accuracy: 0.9590 - val_loss: 2.1983 - val_accuracy: 0.6133
Epoch 26/500
225/225 [=====] - 22s 96ms/step - loss: 0.6432 - accuracy: 0.9610 - val_loss: 2.2027 - val_accuracy: 0.6186
Epoch 27/500
225/225 [=====] - 21s 95ms/step - loss: 0.6326 - accuracy: 0.9609 - val_loss: 2.2396 - val_accuracy: 0.6163
Epoch 28/500
225/225 [=====] - 22s 97ms/step - loss: 0.6251 - accuracy: 0.9614 - val_loss: 2.2626 - val_accuracy: 0.6085
Epoch 29/500
225/225 [=====] - 21s 95ms/step - loss: 0.6173 - accuracy: 0.9628 - val_loss: 2.1853 - val_accuracy: 0.6169
```



## 8.2.7 Saving CNN model

```
[15] scores = model.evaluate(x_testing, y_testing)
      print('%s: %.2f%%' % (model.metrics_names[1], scores[1]*100))

      model.save('fer2013.h5')
      model.save_weights('fer2013_weights.h5')

113/113 [=====] - 1s 10ms/step - loss: 2.1853 - accuracy: 0.6169
accuracy: 61.69%
```

```
[16] model.load_weights('fer2013_weights.h5')
      scores = model.evaluate(x_testing, y_testing)
      print('%s: %.2f%%' % (model.metrics_names[1], scores[1]*100))


113/113 [=====] - 1s 11ms/step - loss: 2.1853 - accuracy: 0.6169
accuracy: 61.69%
```


## 8.2.8 Building SVM model for training heart rate


```
[13] #BUILDING A SVM MODEL FOR TRAINING HEART RATE
      from sklearn.svm import SVC

      model1=SVC()
      x_training1=x_training1.reshape(-1,1)
      y_training1=y_training1.reshape(-1,1)
      x_testing1=x_testing1.reshape(-1,1)

      model1.fit(x_training1,y_training1)
      y_pred=model1.predict(x_testing1)
```

 /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: Data y = column\_or\_1d(y, warn=True)

```
 from sklearn.metrics import accuracy_score
      print(accuracy_score(y_pred,y_testing1))
```

 0.4505433268319866



## 8.2.9 Code to predict output

```
img_idx = np.uint32(np.random.rand()*(testingset.shape[0] - 1))
print('testing image index: ',img_idx)
i_sample = x_testing[img_idx, :]
i_sample = i_sample.reshape(48, 48)

pred_cls = model.predict(i_sample.reshape(1, 48, 48, 1))

h_sample=x_testing1[img_idx,:]
pred_cls_h= model1.predict(h_sample.reshape(-1,1))

print('heart rate :',h_sample)

classes_x=np.argmax(pred_cls,axis=1)
plt.imshow(i_sample, cmap='gray')
plt.show()
print('result from image processing')
print('> true emotion: %s\n> predicted emotion: %s' % ( get_emotion(np.argmax(y_testing[img_idx, :])),
                                                    get_emotion(classes_x)))

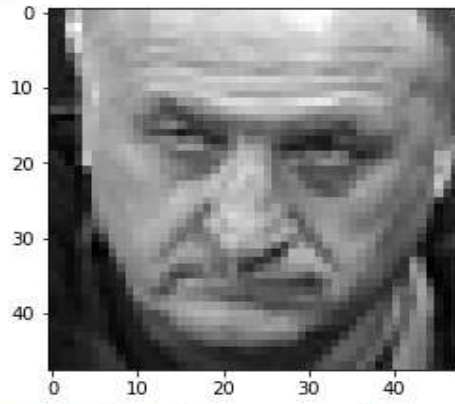
h_sample=x_testing1[img_idx,:]
pred_cls_h= model1.predict(h_sample.reshape(-1,1))

print('result from heart rate')
print('> true emotion: %s\n> predicted emotion: %s' % ( get_emotion(y_testing1[img_idx]-1),
                                                    get_emotion(pred_cls_h-1)))
```

## 9. RESULT AND ANALYSIS

### 9.1 OUTPUT FROM THE CODE

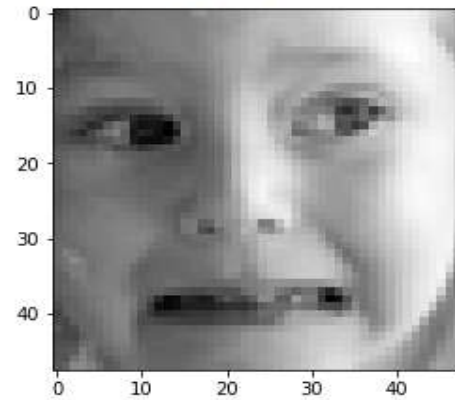
```
testing image index: 2700  
heart rate : [64.]
```



```
result from image processing  
> true emotion: sad  
> predicted emotion: sad  
result from heart rate  
> true emotion: sad  
> predicted emotion: sad
```

---

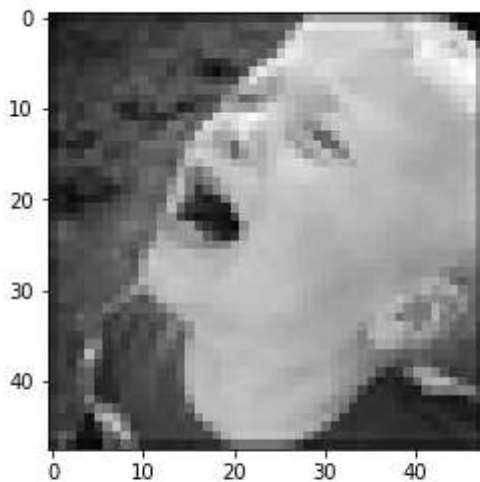
```
testing image index: 1773  
heart rate : [86.]
```



```
result from image processing  
> true emotion: fear  
> predicted emotion: fear  
result from heart rate  
> true emotion: fear  
> predicted emotion: fear
```

---

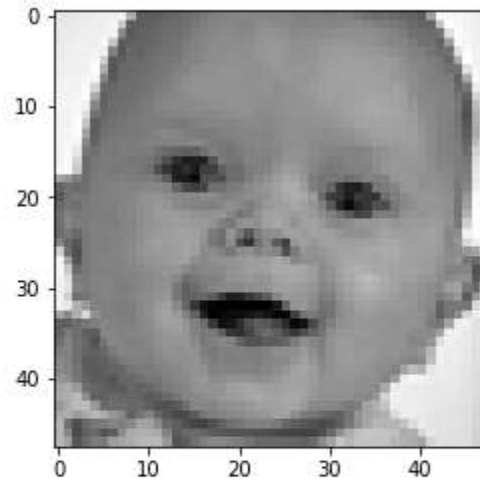
```
testing image index: 3333  
heart rate : [67.]
```



```
result from image processing  
> true emotion: sad  
> predicted emotion: happy  
result from heart rate  
> true emotion: sad  
> predicted emotion: sad
```

---

```
testing image index: 1691  
heart rate : [98.]
```



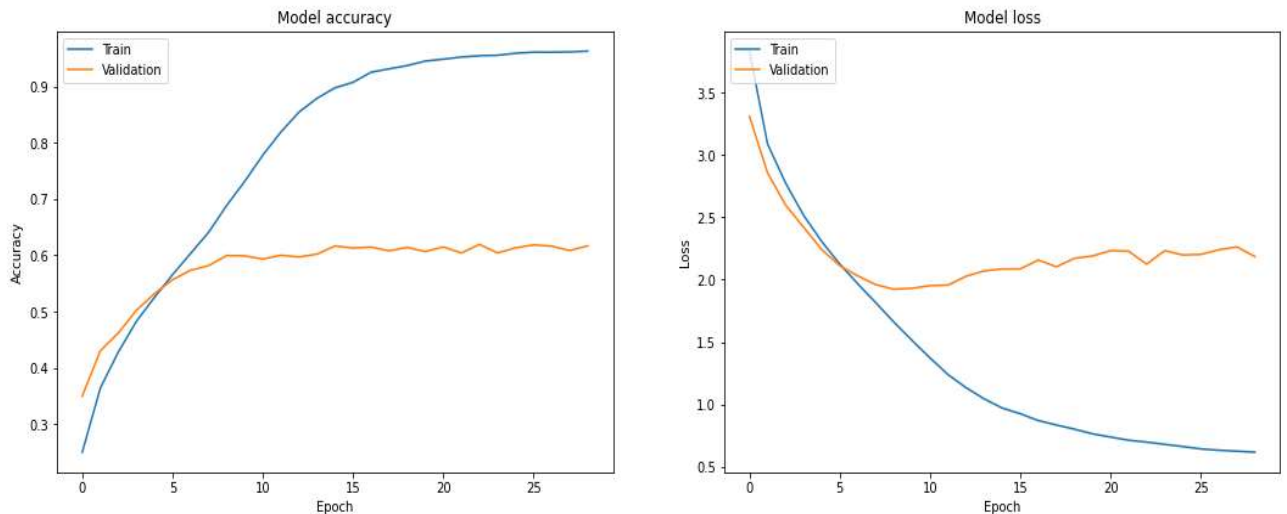
```
result from image processing  
> true emotion: happy  
> predicted emotion: happy  
result from heart rate  
> true emotion: happy  
> predicted emotion: angry
```

---

## 9.2 Accuracy

The accuracy of the testing dataset using the CNN model for image processing is 61.69%, while the accuracy of the SVM model for heart beat is 45.04%.

## 9.3 Accuracy and loss graph plot of CNN model



## 9.4 Accuracy of particular emotion

- > Accuracy 51.18% for <angry>
- > Accuracy 55.36% for <disgust>
- > Accuracy 42.54% for <fear>
- > Accuracy 82.35% for <happy>
- > Accuracy 49.92% for <sad>
- > Accuracy 75.18% for <surprise>
- > Accuracy 58.98% for <neutral>

# 10. CONCLUSION AND FUTURE ENHANCEMENTS

## 10.1 Conclusion

In this project, a LeNet architecture based six layer convolution neural network is implemented to classify human facial expressions i.e. happy, sad, surprise, fear, anger, disgust, and neutral. The system has been evaluated using Accuracy, Precision, Recall and F1-score. The classifier achieved accuracy of 60.17 % , precision of 0.61, recall 0.61 and F1-score 0.61.

## 10.2 Enhancement

For this project, we trained all the models from scratch using CNN packages. Here we used only grayscale images, in the future work, we would like to extend our model to color . This will allow us to investigate the efficacy of pre-trained models such as AlexNet or VGGNet for facial emotion recognition..

## 11. LIST OF REFERENCES

- [1] M. Gargsha, P. Kuchi, and I. D. K. Torkkola. Facial expression recognition using artificial neural networks. <http://www.public.asu.edu/~pkuchi/expressionrecognition.pdf>. In *EEE 511: Artificial Neural Computation Systems*, 2010.
- [2] C. L. Lisetti and D. E. Rumelhart. Facial expression recognition using a neural network. In *Proceedings of the Eleventh International FLAIRS Conference*. Menlo Park, pages 328–332. AAAI Press, 2008.
- [3] Raghuvanshi, Arushi, and Vivek Choksi. "Facial Expression Recognition with Convolutional Neural Networks." Stanford University, 2016
- [4] Alizadeh, Shima, and Azar Fazel. "Convolutional Neural Networks for Facial Expression Recognition." Stanford University, 2016
- [5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
- [6] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
- [7] <https://www.kaggle.com/datasets/msambare/fer2013>

## 12. APPENDIX A:

The undersigned acknowledge they have completed implementing the project “**EMOTION DETECTION FROM FACIAL EXPRESSION AND HEART RATE**” and agree with the approach it presents.

Signature: \_\_\_\_\_ Date:     /06/2022    

MURALI R  
Name: \_\_\_\_\_

Signature: \_\_\_\_\_ Date:     /06/2022    

RAGHURAJ S V  
Name: \_\_\_\_\_

## 13. APPENDIX B: ABBREVIATIONS

The following table provides definitions for terms relevant to this document.

Term	Abbreviation
CNN	Convolutional Neural Network
SVM	Support Vector Machine
ROI	Region of Interest