INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 4 - V7I4-1385)
Available online at: https://www.ijariit.com

Facial emotion recognition and detection

Pranav S.
pranav.suresh565@gmail.com
New Horizon College of Engineering, Bengaluru, Karnataka

Praveen C.
praveenct1237@gmail.com
New Horizon College of Engineering, Bengaluru, Karnataka

Niveditha C. B.
nivedithaniveditha1151@gmail.com
New Horizon College of Engineering, Bengaluru, Karnataka

Asha Rani Borah
nhce.asha.borah@gmail.com
New Horizon College of Engineering, Bengaluru, Karnataka

***Abstract*— Face and Emotion recognition presents a challenging problem in the field of image analysis and computer vision. The security of information is becoming very significant and difficult. Security cameras are presently common in airports, Offices, University, ATM, and Bank and in any locations with a security system. Face recognition is a biometric system used to identify or verify a person from a digital image. The Face recognition system should be automatically able to detect a face and their emotions. This system would be able to perform for multiple people at a time.**

Keywords: *Face recognition, Emotion recognition, CNN, Deep learning, Multimodal.*

1. INTRODUCTION

Facial Emotion are can be described as the feelings that one human being express every day and in fact every second. This plays an important fact in conveying the persons feeling and his emotion at that instant.

Security cameras are presently common in airports, Offices, University, ATM, and Bank and in any locations with a security system. Face recognition is a biometric system used to identify or verify a person from a digital image. Face Recognition system is used in security. Face recognition system are able to automatically detect a face in an image. This involves extracts its features and then recognize it, regardless of lighting, expression, illumination, ageing, transformations doing these tasks with high accuracy and speed is a difficult task.

The application which is being developed will help the user to recognize others emotions instantly regardless of the external conditions and also detect emotions instantantly. Inputs will be given to the system in the form an live camera feed or through an image which then will be used for the recognition.

This application is built using the CNN (Convolution Neural Networks) algorithm. Which has multiple layers and through each layer processing with different kernel filters. Emotions that are able to recognized.

2. SYSTEM ANALYSIS

2.1 Existing System

The previously developed systems for facial emotion recognition uses AFERS. In existing system, classification is done through simple image processing to classify images only. The main drawback of this isit has very high recourse costs for its implementation. Computing the Hash Value for a large data file can be very stressful on some of the devices which are used to run the application on small devices such as phones.

2.2 Proposed System

In our project a FER model is proposed, this makes use of convolution features. In our work we try to focus on the importance of deep convolution network features. convolution networks can be used to enhance the performance of FER. The impact of deep neural features on the task is being studied. Features that extracted from the convolution layer of the pretrained model are obtained and then is taken as the input for the classifier algorithm. We propose that usesdeep convolution methods to represent the image instead of hand crafted methods. The main downside is that they require huge datasets to obtain their optimal performance. Pre trained models are used instead of having large datasets.Feature extraction is done using a pre trained convolution networks model and for classification deep neural networks are used.

3. HARDWARE AND SOFTWAREREQUIREMENTS

3.1 Hardware Requirements

Processor :500 MHz
Processor RAM :2 GB
Hard Disk : 100 GB System
type : 64 Bit OS

3.2 Software Requirements

Operatingsystem: WindowsXPorHigher

FrontEnd :PyQT
IDE: Spyder

4. DIAGRAMS

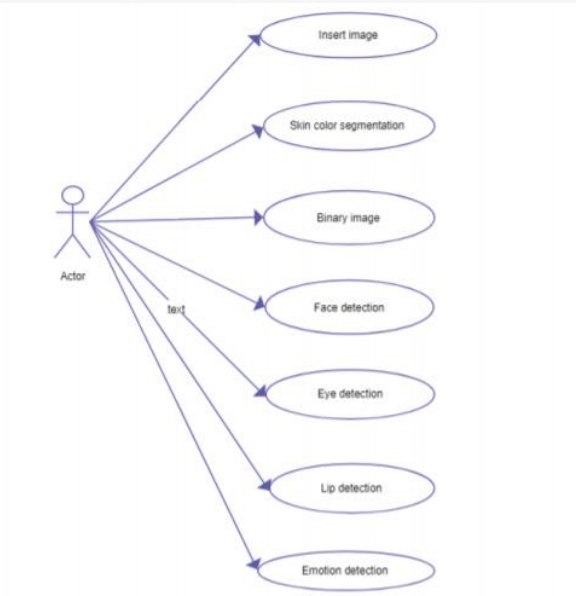


Fig.1. Use Case

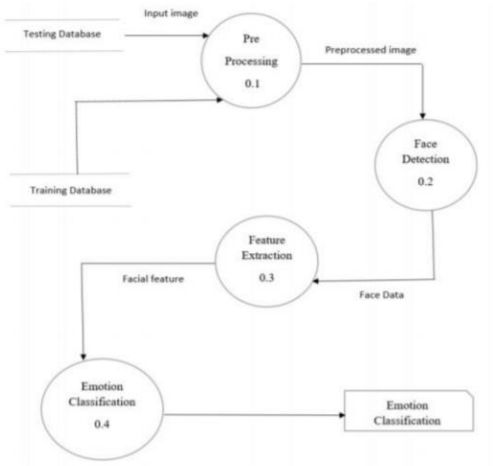


Fig.2. High Level Design

5. SYSTEM IMPLEMENTATION

5.1 CNN Model Creation

The model is creation is done using the Keras library to implement the CNN Layers and the Model is created on online Kaggle kernel as the facial emotion model creation requires a lot of CPU power and compilation time for Rendering epochs to higher accuracy. Total of 7 different layers are present to differentiate between 7 different emotions and sequential modal is used, for model creation. For kernel filters Conv2D is used with appropriate filter sizes to differentiate between the layers. Factors like BactchNormalization, Activation, MaxPooling2D, Dropout are specified in each layer.

5.2 Fitting Model to Test and training Data

We need to fit our CNN Model to our training and validation data set in orderto create our Model that can be used for facial emotion recognition. Keras optimizers are used to fitting the test and training data set. EarlyStopping and ReduceLRonPlateau functions are used as stopping conditions when appropriate accuracy is reached

5.3 Training and Testing

In order to create the training and test data set Kaggle is being used. Facial emotion data set that contains validation and training data sets with 3000+ images for each of the 7 emotions are used to train our model.ImageDataGenerator is used for data creation. Color mode used for the model is grayscale and classmode is categorical along with shuffle .

5.4 Algorithm Implementation

Naïve Bayes algorithm is used for predicting as it is more efficient in dealing with huge data. Sklearn provides libraries for dealing with machine learning algorithm. GausianNB is imported from sklearn.naive_bayes.Classifeier is an object of GaussianNB. Once the text data is converted to numbers with the help of label encoding the text and respective data are stored in dictionaries which can be used for converting the newly obtained data's text to there respective numbers without calling label encoder again.The model is fitted by calling fit function whose parameters are x_train and y _train.X_train

contains the details which help in deciding the y_train values. Y_train contains the output i.e Job Role.Once the model is fitted we can predict the Job role by calling predict function by passing x_test as aparameter.

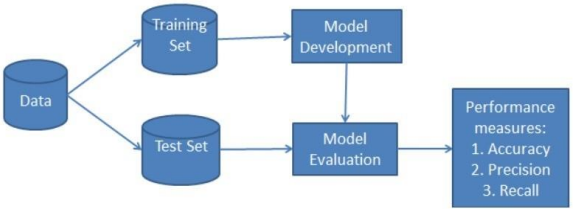


Fig.6. Naive Bayes Algorithm

5.5 Accuracy

Accuracy of the built model can be found by using accuary_score which is available in sklearn.metrics.y_test and predicted variables which store the prediction are passed as a parameter to accuary_score.The accuracy can be increased by improving the data passed and there form.

5.6 Result

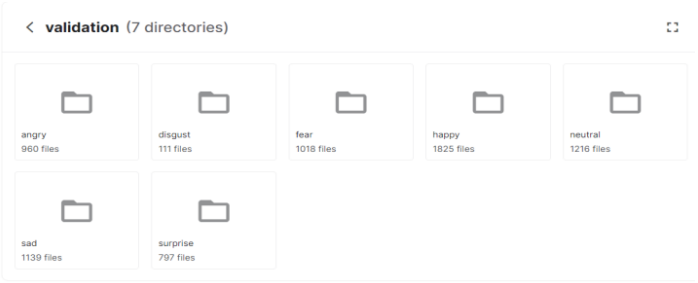
The data given by the user is evaluated in the backend using the algorithm built and displays the Suggested Job Role to the user in the front end.

5.7 Libraries

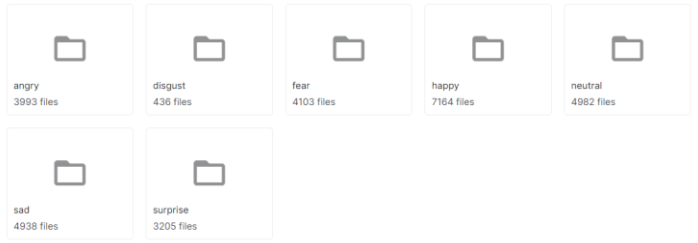
Libraries that are getting used in the model creation are matplotlib, numpy, pandas, seaborn, os, and several deep learning libraries from keras. Keras: Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow.It was developed to make implementing deep learning models as fast and easy as possible for research and development.It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

5.8 Dataset

The dataset for this algorithm is from Kaggle. The dataset is divided into training and test. where each contains seven classes representing the seven different emotions – happy, sad, angry, disgusted, surprised, neutral and fearful. The dataset contains the images in grayscale affixed sizes, so that is it will be easy during the analysis process. This dataset contains several thousands of images which makes the recognition of more accurate and work in poor lighting conditions.



Validation dataset



Training Dataset

6. MODEL

In building of the model there CNN is used. This algorithm is built using Convolutional neural network and opencv library, which detects the facial expressions of the person through the webcam and captures the image if the facial expression is found. The convolutional neural network is a deep learning , feed forward neural network used for analyzing the images. In this project, the keras optimizers is used for detecting the seven facial expressions. The keras optimizers extracts the features from images in the dataset and finds a certain pattern which is used for identifying the expression in real faces through video camera. Opencv is used for capturing the images from the camera. Keras module's optimizers are used to import Adam, SGD and RMSprop which are used for the model creation. Adam is an optimizing algorithm, Adam optimization is also a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. SGD Is used to perform frequent updates with a high variance. Rmsprop is used to maintain a discounted average of the square of gradients and divide the gradient by root of this average.

```
from keras.optimizers import Adam,SGD,RMSprop


no_of_classes = 7

model = Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
```

Sequential model is used for the creation of this model. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.In our model there are mainly 4 different layers and 2 other layers than are connect to the other layers.

Methods present in the model creation:

Conv2D: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

- BatchNormalization: Batch normalization is a technique designed to automatically standardize the inputs to a layer in a deep learning neural network.batch normalization to accelerate the training of deep learning neural networks in Python with Keras.
- Activation: Applies the rectified linear unit activation function.With default values, this returns the standard ReLU activation: max(x, 0), the element-wise maximum of 0 and the input tensor.Modifying default parameters allows you to use non-zero thresholds, change the max value of the activation, and to use a non-zero multiple of the input for values below the threshold.
- MaxPooling2D: Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map.The maximum pooling operation can be added to the worked example by adding the MaxPooling2D layer provided by the Keras API.
- Dropout: Dropout is a technique which is used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.
- Dense: Dense layer is a regular deep connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

  output = activation(dot(input, kernel) + bias)
- Compile: The compilation is the final step in creating a model. Once the compilation is done, we can move on to training phase. After this step the summary is printed.

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

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


opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256
_________________________________________________________________
activation (Activation)      (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 512)       2048
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
_________________________________________________________________
```


6.1 Fitting Model With Dataset

In this function of the model creation of the CNN model the model created is fitted on to the dataset from the Kaggle Library. It is the process of making the model to learn the relationship between the outcome and the predictors. The model is fitted with the train data, testing data. The model trainedfor 48 epochs and the weights of the trained model are stored in H5 file. Keras callbacks are used A callback is a set of functions to be applied at given stages of the training procedure. This includes stopping training when you reach a certain accuracy/loss score, saving your model as a checkpoint after each successful epoch, adjusting the learning rates over time, and more.

Methods that are used are:

- ModelCheckpoint: ModelCheckpoint is a Keras callback to save model weights or entire model at a specific frequency or whenever a quantity (for example, training loss) is optimum when compared to last epoch/batch. It allows us to specify a quantity to monitor, such as loss or accuracy on training or validation dataset.
- EarlyStopping: Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.Sub methods arguments used are monitor: quantity to be monitored. min_delta: threshold for measuring the optimum, to only focus on significant changes. Verbose: used to update messages 0,1.
- ReduceLRonPlateau: Reduce learning rate when a metric has stopped improving.

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.Sub methods arguments used are monitor: quantity to be monitored. min_delta: threshold for measuring the optimum, to only focus on significant changes. Verbose: used to update messages 0,1.

Model.fit_generator is used to create the model (h5 file) along with the specifics specified in fitting of the dataset.
The model is after fit_generator method the epochs are created along with the accuracy of the model.

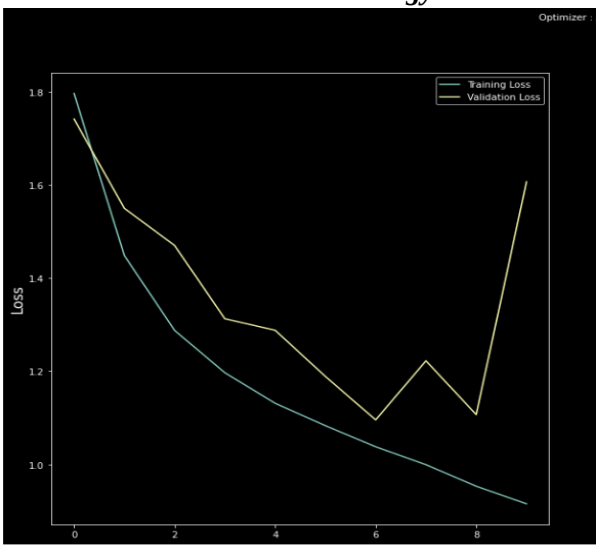
6.2 Accuracy And Loss Plot

MatplotLib is used to the plotting of accuracy and loss of the model. A loss function is used to optimize a machine learning algorithm. The loss is calculated on training and validation and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or validation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization.

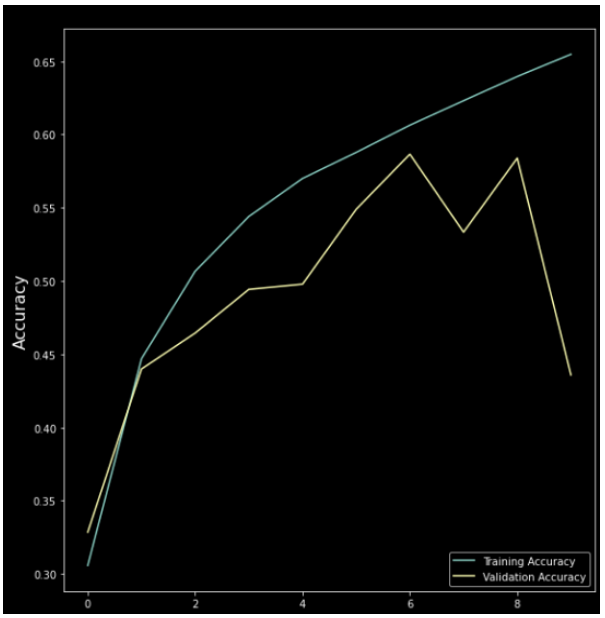
An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage. It is the measure of how accurate your model's prediction is compared to the true data.

We can create plots from the collected history data. 2 Plots are created:

- A plot of accuracy on the training and validation datasets over training epochs.
- A plot of loss on the training and validation datasets over training epochs.

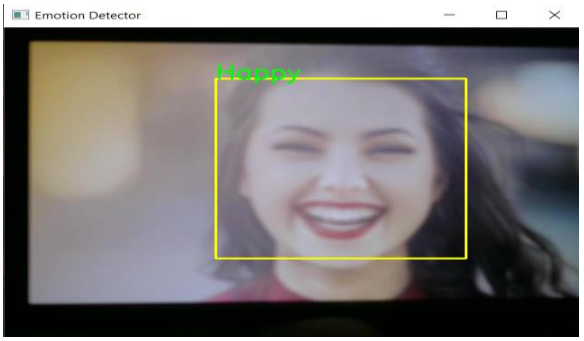


The optimizers used for the plotting of loss graph is the adam optimizer. It is a scholastic gradint descent method.Based on adaptive estimation of first-order and second-order moments. Loss plot From the plot of loss, we can see that the model has comparable performance on both train and validation datasets (labeled test). If these parallel plots start to depart consistently, it might be a sign to stop training at an earlier epoch.

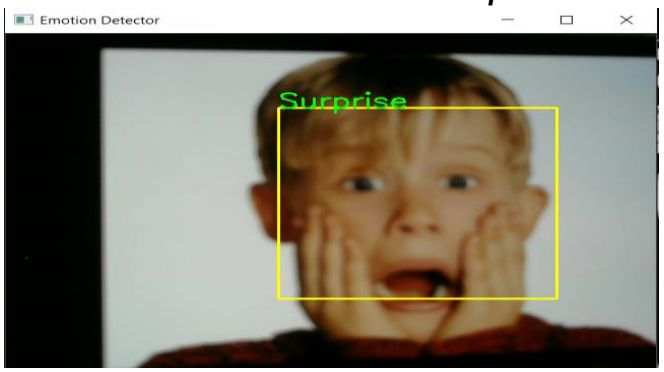


From the plot of accuracy, we can see that the model could probably be trained a little more as the trend for accuracy on both datasets is still rising for the last few epochs. We can also see that the model has not yet over-learned the training dataset, showing comparable skill on both datasets.

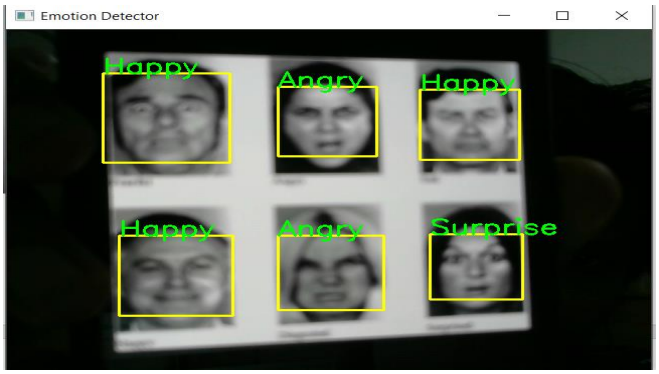
7. SANPSHOTS



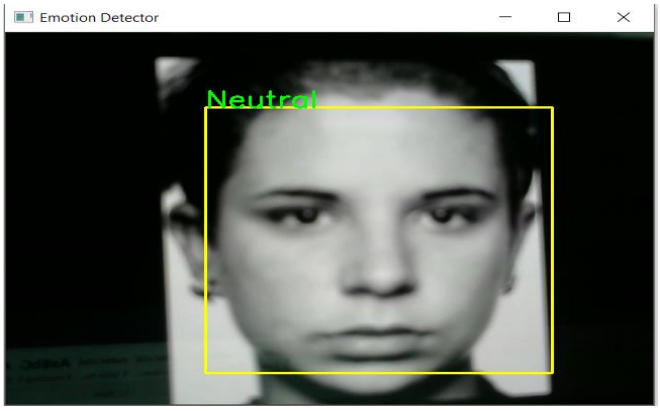
Happy Emotion recognized



Surprise Emotion detected



Multiple Faces and Emotions are detected



Neutral Emotion is detected

8. CONCLUSION

A prototype of a facial emotion detection prediction system is developed using CNN. The system extracts expressions that are present in the human face. The models are first trained and are validated against a test dataset. Prediction, Identification, Classification, Optimization that are considered to be the goals of deep learning are defined based on the data exploration. The goals of deep learning are evaluated against the trained model. The application can be used in surveys to get instant feedback from the customer, evaluate the emotions of interviewees when under a interview, used for theft detections and security as well along with many other uses.

9. FUTURESCOPE

A more powerful application can be developed where facial emotions can be more deeply classified and the algorithm can be modified to make it useable in multiple areas. Like drowsiness detector in cars, theft detection in homes and many more. We all know with continues development in deep learning the algorithm can be more to have higher accuracy and many more features.

10. ACKNOWLEDGMENT

We take the opportunity to thank our mentor Ms Asha Rani Borah and teachers from New Horizon College of Engineering, Bangalore for the most effective and valuable guidance. They have very helpful and been up front to motivate and encourage us for bringing out this project successfully and showed great sagacity during the development of the project.

11. REFERENCES

[1] Michel, P., & El Kaliouby, R. (2003, November). Real time facial expression recognition in video using support vector machines. In Proceedings of the 5th international conference on Multimodal interfaces (pp. 258-264). ACM.

[2] Bettadapura, V. (2012). Face expression recognition and analysis: the state of the art. arXiv preprint arXiv:1203.6722.

[3] Bhatt, M., Drashti, H., Rathod, M., Kirit, R., Agravat, M., &Shardul, J. (2014). A Studyof Local Binary Pattern Method for Facial Expression Detection. arXiv preprintarXiv:1405.6130.

