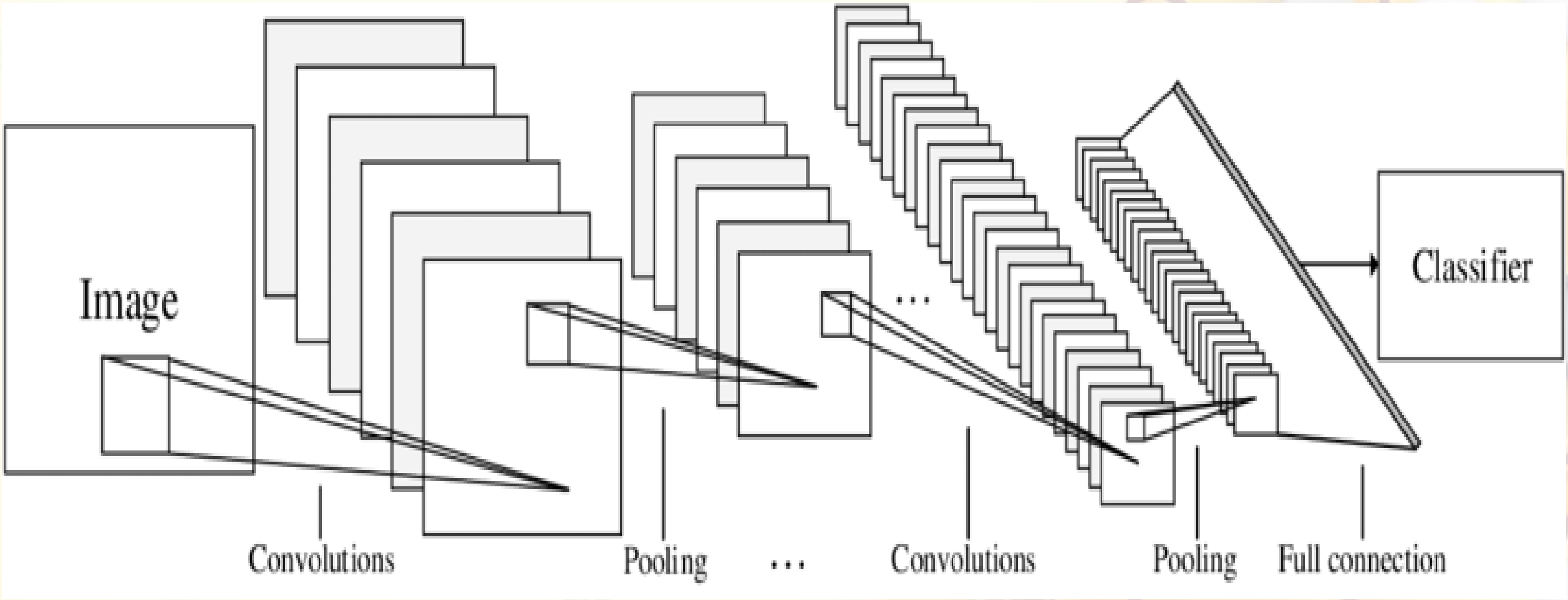
Topics to be covered in this presentation:-

- Why to recognise faces?
- Recap to theory (explained earlier)
- Convolution and Image padding
- Max Pooling and Image Flattening
- Testing
- .json file data
- Accuracy Calculation
- Saving values for confusion matrix generation
- Prediction in the video using live web camera
- Step 1 to Step 5 (Prediction)
- References

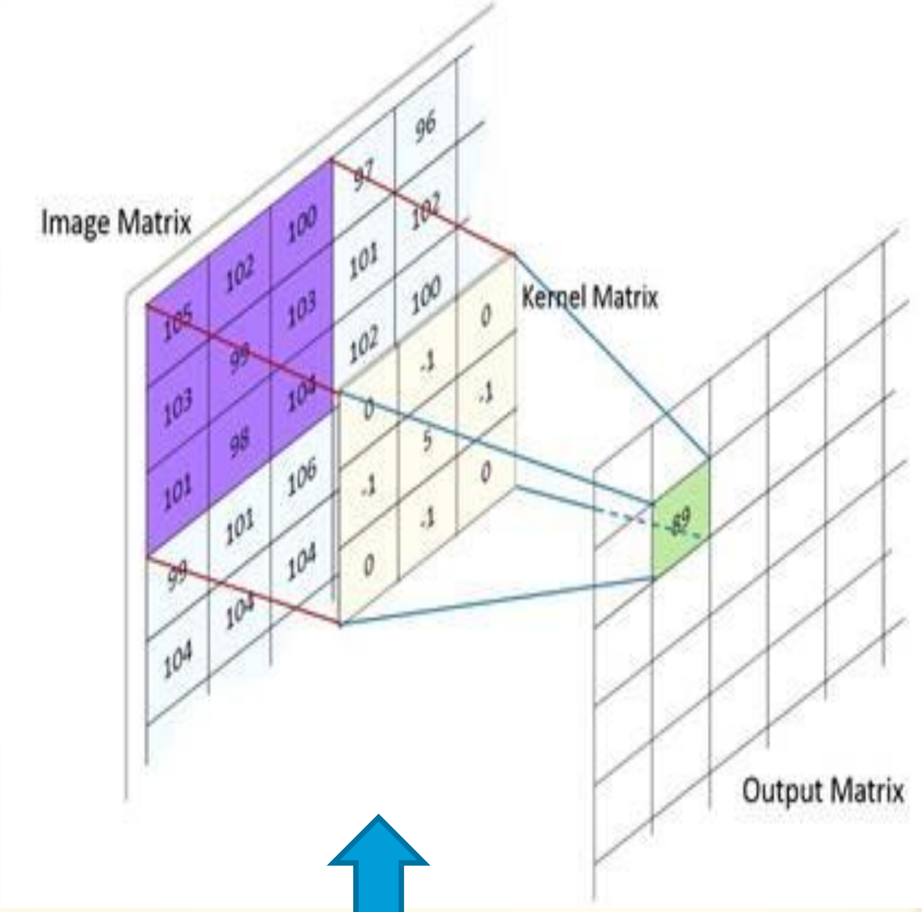
Why to recognise Faces :-

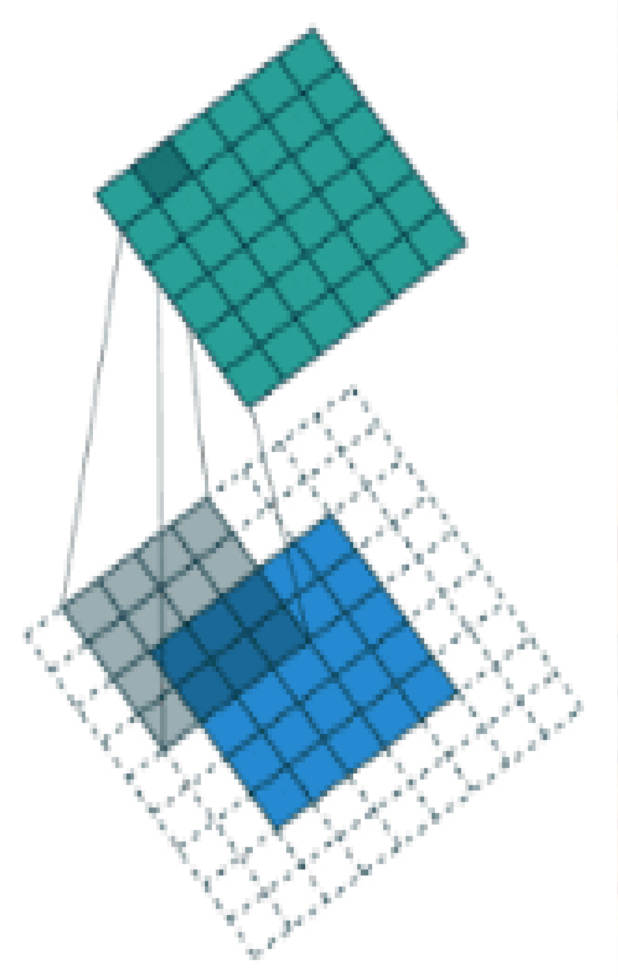
- The motivation behind choosing this topic specifically lies in the huge investments large corporations do in feedbacks and surveys but fail to get equitable response on their investments.
- Facial Expression Recognition through facial gestures is a technology that aims to improve product and services performance by monitoring customer facial expressions to certain products or service staff by their evaluation.

Recap to theory (explained earlier):



LeNet Architecture (Broad View)

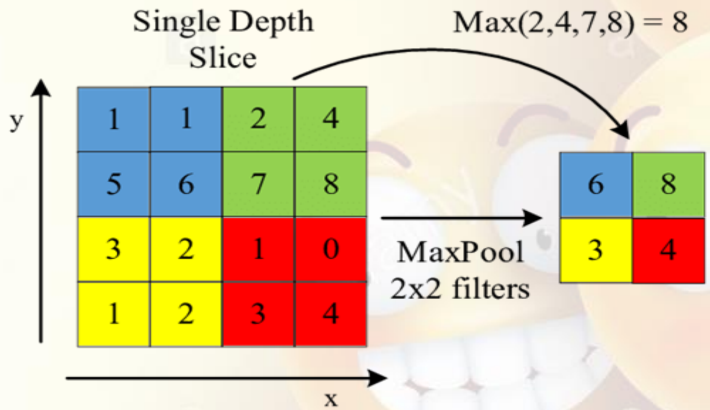




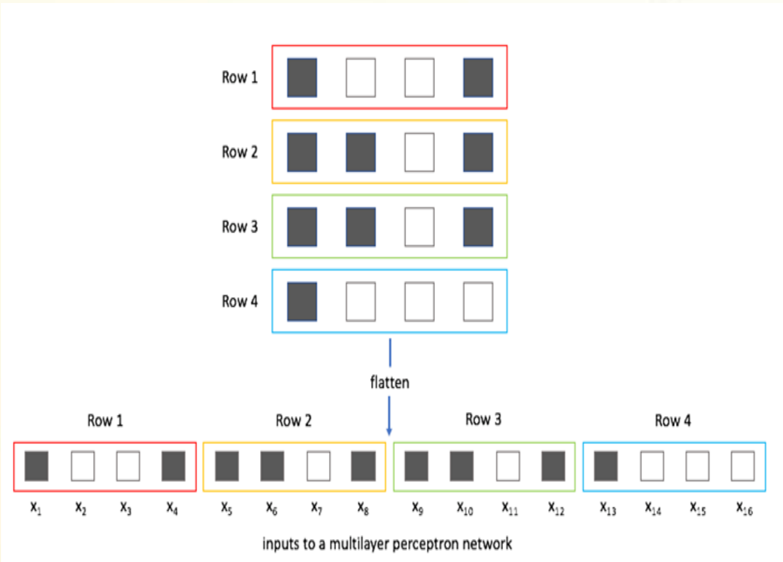
Step 1. Convolution

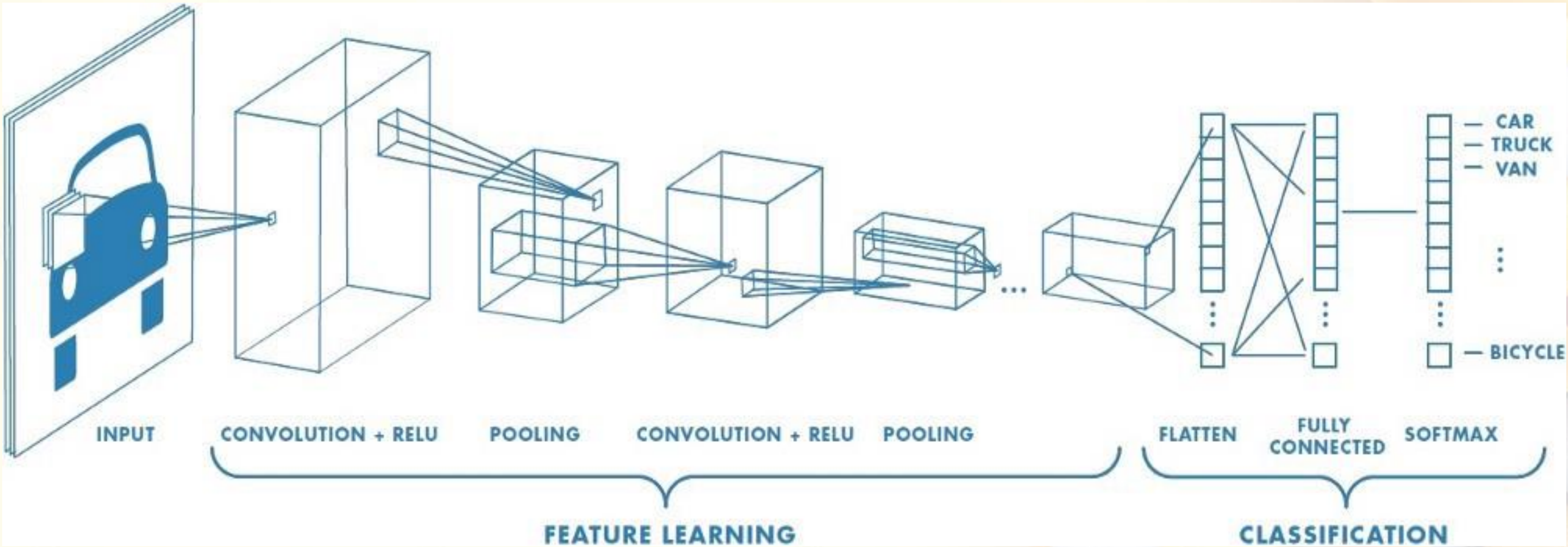
Step 2. Image Padding

Step 3. Max Pooling



Step 4. Image Flattening





Step 5. Feature Learning and Classification

Implementation of Convolution Neural Network

We have used Python Programming Language for implementation purpose.

Major Steps are:

- Data Preprocessing
- Feature Extraction
- Training and Validation
- Testing

4. Testing

Step 1: Load the .json and .h5 file

Loading .json file

```
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("model.h5")
print("Loaded model from disk")
```

Loading .h5 model

.json file data

sers > S Rani > Desktop > Facial Expression Recognition > {} model.json > ...


```
{"class_name": "Sequential", "keras_version": "2.2.4", "config": {"layers": [{"class_name": "Conv2D", "config": {"kernel_initializer":
{"class_name": "VarianceScaling", "config": {"distribution": "uniform", "scale": 1.0, "seed": null, "mode": "fan_avg"}}, "name": "conv2d_1",
"kernel_constraint": null, "bias_regularizer": null, "bias_constraint": null, "dtype": "float32", "activation": "relu", "trainable": true,
"data_format": "channels_last", "filters": 64, "padding": "valid", "strides": [1, 1], "dilation_rate": [1, 1], "kernel_regularizer":
{"class_name": "L1L2", "config": {"l2": 0.009999999776482582, "l1": 0.0}}, "bias_initializer": {"class_name": "Zeros", "config": {}},
"batch_input_shape": [null, 48, 48, 1], "use_bias": true, "activity_regularizer": null, "kernel_size": [3, 3]}}, {"class_name": "Conv2D",
"config": {"kernel_constraint": null, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"distribution": "uniform", "scale": 1.
0, "seed": null, "mode": "fan_avg"}}, "name": "conv2d_2", "bias_regularizer": null, "bias_constraint": null, "activation": "relu", "trainable":
true, "data_format": "channels_last", "padding": "same", "strides": [1, 1], "dilation_rate": [1, 1], "kernel_regularizer": null, "filters": 64,
"bias_initializer": {"class_name": "Zeros", "config": {}}, "use_bias": true, "activity_regularizer": null, "kernel_size": [3, 3]}},
{"class_name": "BatchNormalization", "config": {"beta_constraint": null, "gamma_initializer": {"class_name": "Ones", "config": {}},
"moving_mean_initializer": {"class_name": "Zeros", "config": {}}, "name": "batch_normalization_1", "epsilon": 0.001, "trainable": true,
"moving_variance_initializer": {"class_name": "Ones", "config": {}}, "beta_initializer": {"class_name": "Zeros", "config": {}}, "scale": true,
"axis": -1, "gamma_constraint": null, "gamma_regularizer": null, "beta_regularizer": null, "momentum": 0.99, "center": true}}, {"class_name":
"MaxPooling2D", "config": {"name": "max_pooling2d_1", "trainable": true, "data_format": "channels_last", "pool_size": [2, 2], "padding":
"valid", "strides": [2, 2]}}, {"class_name": "Dropout", "config": {"rate": 0.5, "noise_shape": null, "trainable": true, "seed": null, "name":
"dropout_1"}}, {"class_name": "Conv2D", "config": {"kernel_constraint": null, "kernel_initializer": {"class_name": "VarianceScaling", "config":
{"distribution": "uniform", "scale": 1.0, "seed": null, "mode": "fan_avg"}}, "name": "conv2d_3", "bias_regularizer": null, "bias_constraint":
null, "activation": "relu", "trainable": true, "data_format": "channels_last", "padding": "same", "strides": [1, 1], "dilation_rate": [1, 1],
"kernel_regularizer": null, "filters": 128, "bias_initializer": {"class_name": "Zeros", "config": {}}, "use_bias": true,
"activity_regularizer": null, "kernel_size": [3, 3]}}, {"class_name": "BatchNormalization", "config": {"beta_constraint": null,
"gamma_initializer": {"class_name": "Ones", "config": {}}, "moving_mean_initializer": {"class_name": "Zeros", "config": {}}, "name":
"batch_normalization_2", "epsilon": 0.001, "trainable": true, "moving_variance_initializer": {"class_name": "Ones", "config": {}},
"beta_initializer": {"class_name": "Zeros", "config": {}}, "scale": true, "axis": -1, "gamma_constraint": null, "gamma_regularizer": null,
"beta_regularizer": null, "momentum": 0.99, "center": true}}, {"class_name": "Conv2D", "config": {"kernel_constraint": null,
"kernel_initializer": {"class_name": "VarianceScaling", "config": {"distribution": "uniform", "scale": 1.0, "seed": null, "mode": "fan_avg"}},
"name": "conv2d_4", "bias_regularizer": null, "bias_constraint": null, "activation": "relu", "trainable": true, "data_format": "channels_last",
"padding": "same", "strides": [1, 1], "dilation_rate": [1, 1], "kernel_regularizer": null, "filters": 128, "bias_initializer": {"class_name":
"Zeros", "config": {}}, "use_bias": true, "activity_regularizer": null, "kernel_size": [3, 3]}}, {"class_name": "BatchNormalization", "config":
{"beta_constraint": null, "gamma_initializer": {"class_name": "Ones", "config": {}}, "moving_mean_initializer": {"class_name": "Zeros",
"config": {}}, "name": "batch_normalization_3", "epsilon": 0.001, "trainable": true, "moving_variance_initializer": {"class_name": "Ones",
"config": {}}, "beta_initializer": {"class_name": "Zeros", "config": {}}, "scale": true, "axis": -1, "gamma_constraint": null,
"gamma_regularizer": null, "beta_regularizer": null, "momentum": 0.99, "center": true}}, {"class_name": "MaxPooling2D", "config": {"name":
```


Ln 1, Col 1   Spaces: 4   UTF-8   CRLF

Step 2: Accuracy Calculation :

```
truey=[]
predy=[]
x = np.load('./modXtest.npy')
y = np.load('./modytest.npy')

yhat= loaded_model.predict(x)
yh = yhat.tolist()
yt = y.tolist()
count = 0

for i in range(len(y)):
    yy = max(yh[i])
    yyt = max(yt[i])
    predy.append(yh[i].index(yy))
    truey.append(yt[i].index(yyt))
    if(yh[i].index(yy)== yt[i].index(yyt)):
        count+=1

acc = (count/len(y))*100

```

Load the test files

Accuracy Calculation

Step 3: Saving the values for confusion matrix analysis

```
#saving values for confusion matrix and analysis
np.save('truey', truey)
np.save('predy', predy)
print("Predicted and true label values saved")
print("Accuracy on test set :"+str(acc)+"%")
```

Prediction in live video using Web Cam

Step 1: All the designed models are imported in the first step.

```
detection_model_path = 'haarcascade_frontalface_default.xml'
emotion_recognition_model_path = 'model.hdf5'
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_recognition_model_path)
emotions = ['angry', 'disgust', 'scared', 'happy', 'sad', 'surprised', 'neutral']
```

Step 2: System camera is accessed and switched on for accessing the face.

```
cv2.namedWindow('emotion_recognition')
camera = cv2.VideoCapture(0)  ## to use laptop camera
```

Step 3: Reading from video camera

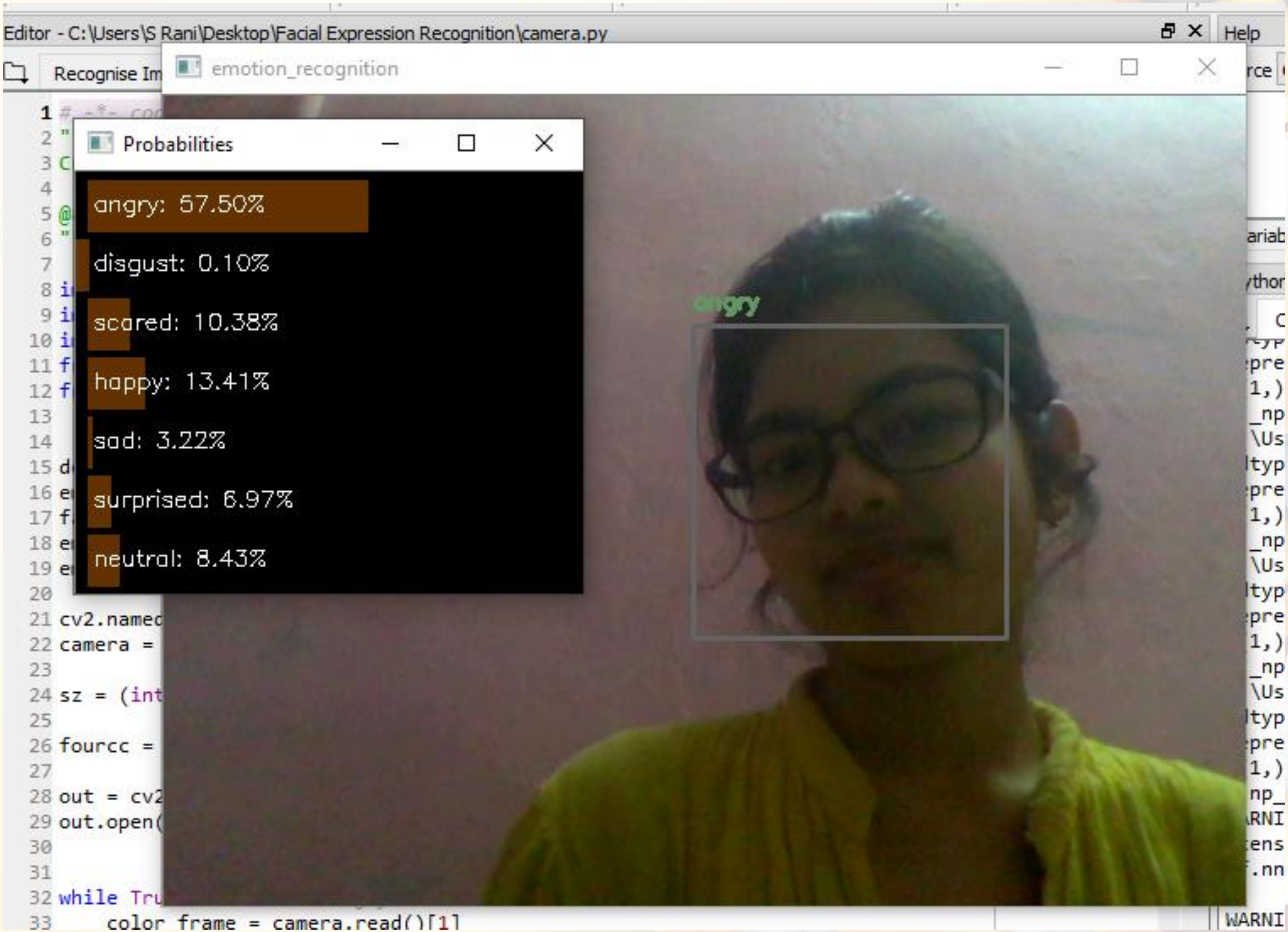
```
while True: # when reading from a video camera
    color_frame = camera.read()[1]
    color_frame = imutils.resize(color_frame,width=min(720, color_frame.shape[1]))


    gray_frame = cv2.cvtColor(color_frame, cv2.COLOR_BGR2GRAY)
    detected_faces = face_detection.detectMultiScale(gray_frame,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)

    canvas = np.zeros((250, 300, 3), dtype="uint8")
    frameClone = color_frame.copy()
```

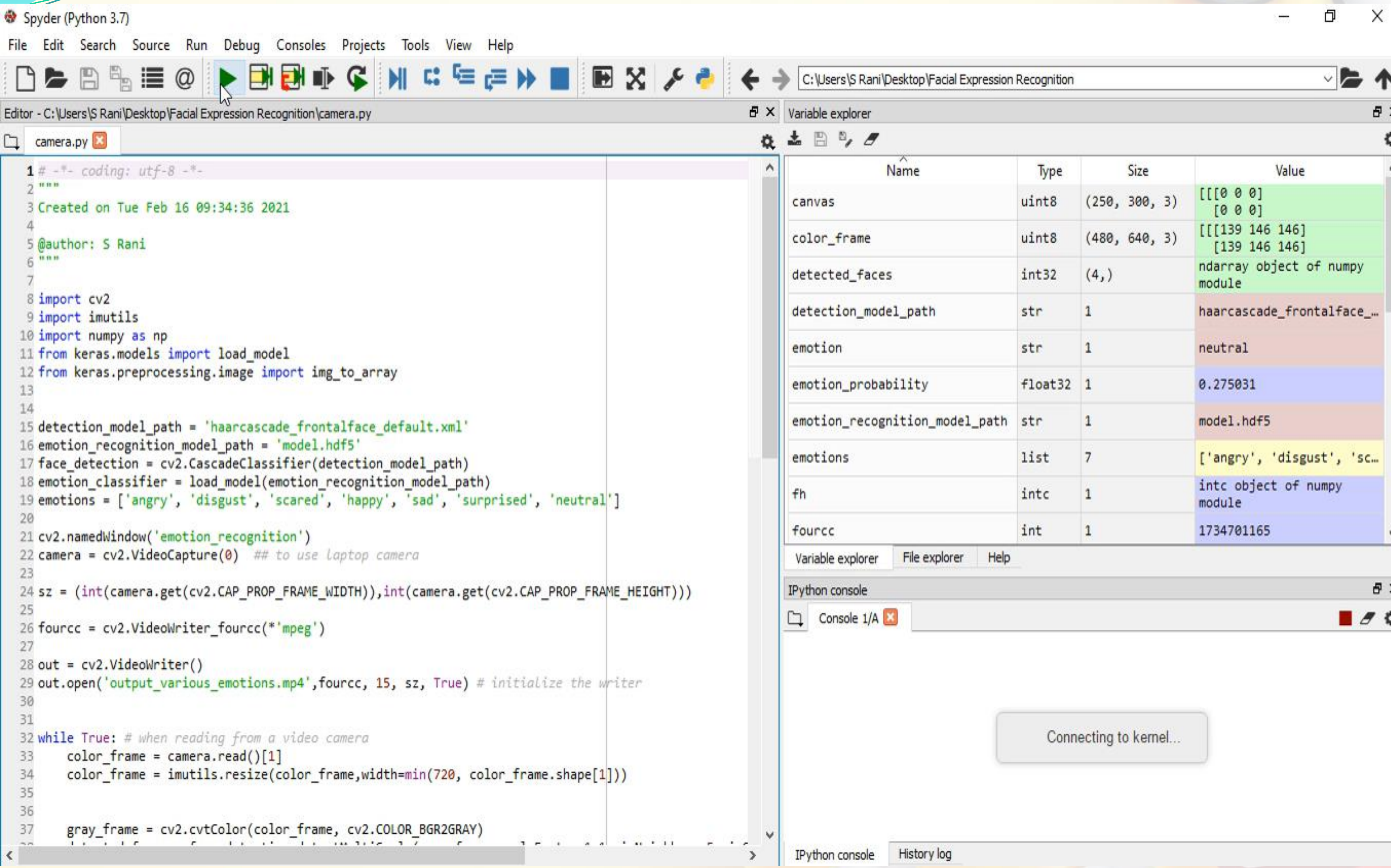
Step 4: Font and box is put over the face and prediction is done.

```
if len(detected_faces)>0:

    detected_faces = sorted(detected_faces, reverse=True, key=lambda x: (x[2]-x[0])*(x[3]-x[1]))[0] # if more than one faces
    (fx, fy, fw, fh) = detected_faces

    im = gray_frame[fy:fy+fh, fx:fx+fw]
    im = cv2.resize(im, (48,48))  # the model is trained on 48*48 pixel image
    im = im.astype("float")/255.0
    im = img_to_array(im)
    im = np.expand_dims(im, axis=0)

    preds = emotion_classifier.predict(im)[0]
    emotion_probability = np.max(preds)
    label = emotions[preds.argmax()]

    cv2.putText(color_frame, label, (fx, fy-10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
    cv2.rectangle(color_frame, (fx, fy), (fx + fw, fy + fh),(0, 0, 255), 2)
```

Step 5: Output image is shown frame by frame in continuous video.

```

    for (i, (emotion, prob)) in enumerate(zip(emotions, preds)):
        # construct the label text
        text = "{}: {:.2f}%".format(emotion, prob * 100)
        w = int(prob * 300)

        cv2.rectangle(canvas, (7, (i * 35) + 5), (w, (i * 35) + 35), (0, 50, 100), -1)
        cv2.putText(canvas, text, (10, (i * 35) + 23), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)
        cv2.putText(frameClone, label, (fx, fy - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (100, 150, 100), 2)
        cv2.rectangle(frameClone, (fx, fy), (fx + fw, fy + fh), (100, 100, 100), 2)

    out.write(frameClone)
    out.write(canvas)

    cv2.imshow('emotion_recognition', frameClone)
    cv2.imshow("Probabilities", canvas)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

camera.release()
out.release()
cv2.destroyAllWindows()
```


Editor - C:\Users\S Rani\Desktop\Facial Expression Recognition\camera.py
Help
Recognise Im
emotion_recognition
Probabilities
angry: 57.50%
disgust: 0.10%
scared: 10.38%
happy: 13.41%
sad: 3.22%
surprised: 6.97%
neutral: 8.43%
angry
cv2.named
camera =
sz = (int
fourcc =
out = cv2
out.open(
while Tru
color_frame = camera.read()[1]
WARNI


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\S Rani\Desktop\Facial Expression Recognition
Editor - C:\Users\S Rani\Desktop\Facial Expression Recognition\camera.py
camera.py
-*- coding: utf-8 -*-
"""
Created on Tue Feb 16 09:34:36 2021
@author: S Rani
"""
import cv2
import imutils
import numpy as np
from keras.models import load_model
from keras.preprocessing.image import img_to_array
detection_model_path = 'haarcascade_frontalface_default.xml'
emotion_recognition_model_path = 'model.hdf5'
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_recognition_model_path)
emotions = ['angry', 'disgust', 'scared', 'happy', 'sad', 'surprised', 'neutral']
cv2.namedWindow('emotion_recognition')
camera = cv2.VideoCapture(0) ## to use laptop camera
sz = (int(camera.get(cv2.CAP_PROP_FRAME_WIDTH)),int(camera.get(cv2.CAP_PROP_FRAME_HEIGHT)))
fourcc = cv2.VideoWriter_fourcc(*'mpeg')
out = cv2.VideoWriter()
out.open('output_various_emotions.mp4',fourcc, 15, sz, True) # initialize the writer
while True: # when reading from a video camera
color_frame = camera.read()[1]
color_frame = imutils.resize(color_frame,width=min(720, color_frame.shape[1]))
gray_frame = cv2.cvtColor(color_frame, cv2.COLOR_BGR2GRAY)
Variable explorer
Name Type Size Value
canvas uint8 (250, 300, 3) [[[0 0 0] [0 0 0]
color_frame uint8 (480, 640, 3) [[[139 146 146] [139 146 146]
detected_faces int32 (4,) ndarray object of numpy module
detection_model_path str 1 haarcascade_frontalface_…
emotion str 1 neutral
emotion_probability float32 1 0.275031
emotion_recognition_model_path str 1 model.hdf5
emotions list 7 ['angry', 'disgust', 'sc…
fh intc 1 intc object of numpy module
fourcc int 1 1734701165
Variable explorer File explorer Help
IPython console
Console 1/A
Connecting to kernel...
IPython console History log

Conclusion

- The system correctly identified the correct facial expression in 23815 of the 35887 images (66.36% of the cases).
- Committee neural networks offer a potential tool for image based mood detection. 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 66.36946224575091% , precision of 0.67, recall 0.57 and F1-score 0.60.

REFERENCES

- *Shan, C., Gong, S., & McOwan, P. W. (2005, September). Robust facial expression recognition using local binary patterns. In Image Processing, 2005. ICIP 2005. IEEE International Conference on (Vol. 2, pp. II-370). IEEE.*

- *Chibelushi, C. C., & Bourel, F. (2003). Facial expression recognition: A brief tutorial overview. CVonline: On-Line Compendium of Computer Vision, 9.*

- *"Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013.*

THANK YOU!