Devanshu Surana
PC-23, 1032210755
Panel C, Batch C1

AIES Lab Assignment 8

Aim: Implement and Understand working of Neural Networks for a real-life application: Face Recognition with Python.

Objective: To study and implement face recognition using Python and the open source library openCV.

Theory: Machine learning is a broader concept involving algorithms that learn patterns and make predictions while Deep learning is a subset of ML that uses neural networks with multiple layers to learn and make decisions.

OpenCV (Open Source Computer Vision) is an open source library for computer vision and image processing tasks.
It provides tools for image and video analysis, including functions for face detection, object recognition and image manipulation.

Neural Networks are computational models inspired by the human brain, consisting of interconnected nodes that process info.

Input: Input an Image with a human face in it.
Output: Algorithm will detect faces of all humans present in image.

Algorithm: Neural Network

FAQ's

1.) Explain Cascade and Classifier in detail.

→ a) Cascade: A series of stage where each stage eliminates regions that are not likely to ~~certain~~ the contain the object of interest, improving efficiency.

b) classifier: A ML model that decides whether a given region of an image contains the object based on features.

2) What are cascades provided by OpenCV? Write in brief.

→ 1) Haar Cascade (Used for face detection) - Based on Haar-like features, which are simple rectangular features used to identify objects.

2) LBP (Local Binary Pattern) Cascade (Efficient for face detection) - Utilizes local binary patterns to describe texture and appearance of an image.

3) HOG (Histogram of Oriented Gradients) Cascade (Effective for human detection) - Focuses on distribution of intensity gradient in an img, effective for detecting humans, capturing shape and structure of objects.

G
29/11/23

CODE:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Sample input data (features)
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

# Sample output data (labels)
y = np.array([0, 1, 1, 0])

# Build a simple feedforward neural network
model = Sequential()
model.add(Dense(4, input_dim=2, activation='relu'))  # Hidden layer with 4 neurons and
ReLU activation
model.add(Dense(1, activation='sigmoid'))          # Output layer with 1 neuron and
Sigmoid activation

# Compile the model
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=500, verbose=0)

# Test the model
sample_input = np.array([[0, 0]])
predicted_output = model.predict(sample_input)
print(f"Sample Input: {sample_input}")
print(f"Predicted Output: {predicted_output}")
```

```
Input:
[[0 0]]

Output:
[[0.03575368]]
```