TASK 2 : Face Detection using OpenCV in Python

- It involves building a system that can identify and locate faces within images or video streams using Deep Learning-based face detection methods available in OpenCV's library.
- The process includes loading the face detection model, processing the input image, and marking detected faces with bounding boxes.
- This technique is widely used in applications like surveillance systems, photo tagging, and user authentication.

Install The Required Packages required in this Project :

Open your Command Prompt/Anaconda Prompt and write the following commands one after the other.

```
pip install opencv-python
pip install imutils
pip install argparse
pip install pickle
pip install scikit-learn
pip install numpy
```

These two folders we will be using for this project execution

- **If we have to detect the face in a picture :**

Required code :

```
# USAGE
# python detect_faces.py --image rooster.jpg --prototxt deploy.prototxt.txt --model
res10_300x300_ssd_iter_140000.caffemodel

# import the necessary packages
import numpy as np
import argparse
import cv2
```

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# load the input image and construct an input blob for the image
# by resizing to a fixed 300x300 pixels and then normalizing it
image = cv2.imread(args["image"])
(h, w) = image.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0,
	(300, 300), (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the detections and
# predictions
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
	# extract the confidence (i.e., probability) associated with the
	# prediction
	confidence = detections[0, 0, i, 2]

	# filter out weak detections by ensuring the `confidence` is
	# greater than the minimum confidence
	if confidence > args["confidence"]:
		# compute the (x, y)-coordinates of the bounding box for the
```

```
            # object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # draw the bounding box of the face along with the associated
            # probability
            text = "{:.2f}%".format(confidence * 100)
            y = startY - 10 if startY - 10 > 10 else startY + 10
            cv2.rectangle(image, (startX, startY), (endX, endY),
                (0, 255, 0), 2)
            cv2.putText(image, text, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)

# show the output image
cv2.imshow("Output", image)
cv2.waitKey(0)
```

Set the directory where the folder is kept :

```
PS C:\project-1\PyPower_face-detection-20241218T125031Z-001> cd \project-1\PyPo
wer_face-detection-20241218T125031Z-001\PyPower_face-detection
```

PyPower_face-detection> python detect_faces.py --image sample1.jpg --prototxt deploy.prototxt.txt --model res10_300x300_ssd_iter_140000.caffemodel

Components in the Command

1. **detect_faces.py**:
    - The Python script contains the code to perform face detection.
2. **Command-Line Arguments**:
    - --image sample1.jpg: Specifies the input image (sample1.jpg) on which face detection will be performed.
    - --prototxt deploy.prototxt.txt: Refers to the **Prototxt** file defining the architecture of the neural network.
    - --model res10_300x300_ssd_iter_140000.caffemodel: The pre-trained **Caffe model** file containing the weights for the face detection network.

How It Works

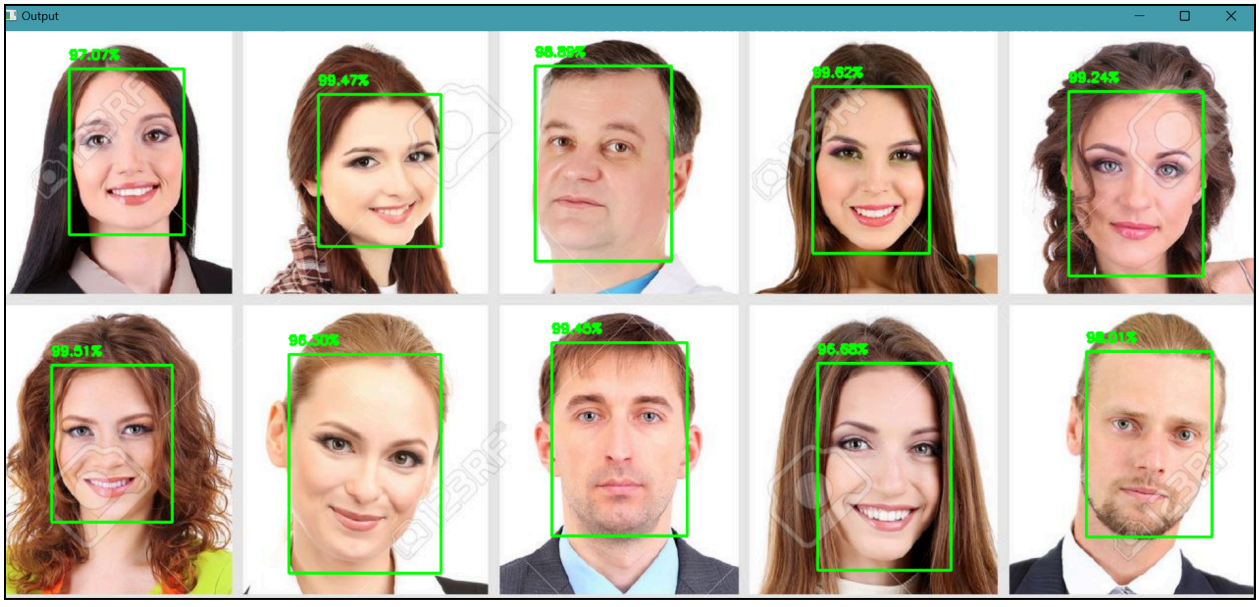
1. **Load the Neural Network**:

- The script uses OpenCV's cv2.dnn.readNetFromCaffe to load the network defined in deploy.prototxt.txt and its weights from res10_300x300_ssd_iter_140000.caffemodel.

2. **Preprocess the Image**:
   - The input image is resized to 300x300 (the input size for the model) and normalized before being passed to the network.
3. **Perform Inference**:
   - The network processes the image and outputs **bounding boxes** for detected faces along with confidence scores.
4. **Draw Bounding Boxes**:
   - The script iterates over the detections, checks if the confidence score exceeds a threshold (e.g., 0.5), and draws rectangles around the detected faces on the input image.
5. **Display or Save Output**:
   - The resulting image with detected faces is displayed or saved, depending on the script's configuration.

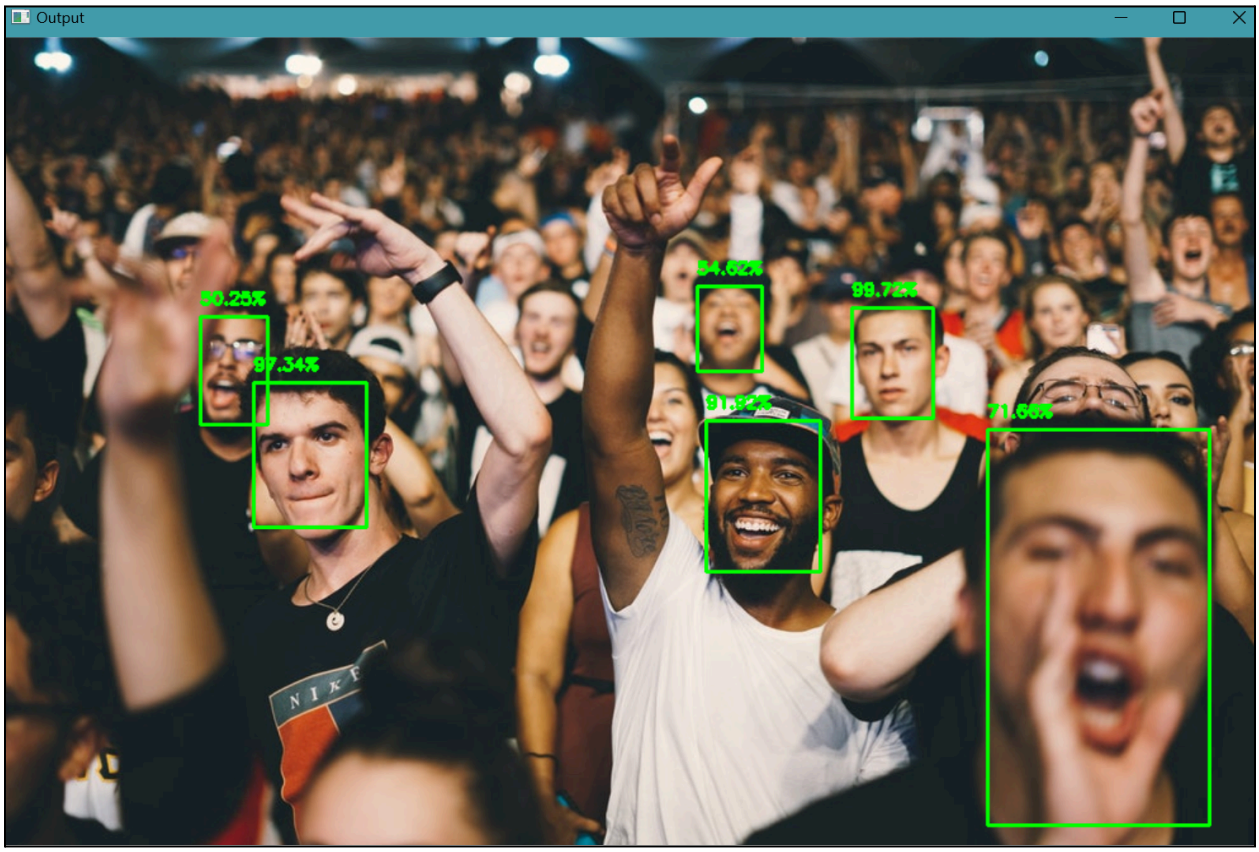
```
PS C:\project-1\PyPower_face-detection-20241218T125031Z-001\PyPower_face-detect
ion> python detect_faces.py --image sample1.jpg --prototxt deploy.prototxt.txt
--model res10_300x300_ssd_iter_140000.caffemodel
[INFO] loading model...
[INFO] computing object detections...
```

Output :

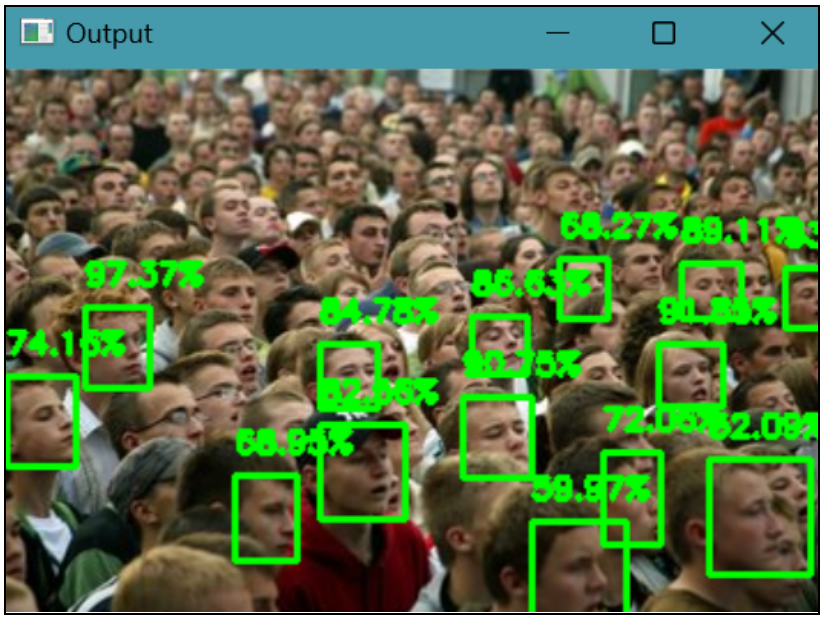


Lets try with different images :

```
PS C:\project-1\PyPower_face-detection-20241218T125031Z-001\PyPower_face-detect
ion> python detect_faces.py --image sample2.jpg --prototxt deploy.prototxt.txt
--model res10_300x300_ssd_iter_140000.caffemodel
```




```
PS C:\project-1\PyPower_face-detection-20241218T125031Z-001\PyPower_face-detect
ion> python detect_faces.py --image sample3.jpg --prototxt deploy.prototxt.txt
--model res10_300x300_ssd_iter_140000.caffemodel
[INFO] loading model...
[INFO] computing object detections...
```



- **If we have to detect the face in a video :**

Required code :

```
# USAGE
# python detect_faces_video.py --prototxt deploy.prototxt.txt --model
res10_300x300_ssd_iter_140000.caffemodel

# import the necessary packages
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

```
# initialize the video stream and allow the cammera sensor to warmup
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=720)

	# grab the frame dimensions and convert it to a blob
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,
		(300, 300), (104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the detections and
	# predictions
	net.setInput(blob)
	detections = net.forward()

	# loop over the detections
	for i in range(0, detections.shape[2]):
		# extract the confidence (i.e., probability) associated with the
		# prediction
		confidence = detections[0, 0, i, 2]

		# filter out weak detections by ensuring the `confidence` is
		# greater than the minimum confidence
		if confidence < args["confidence"]:
			continue

		# compute the (x, y)-coordinates of the bounding box for the
		# object
		box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
		(startX, startY, endX, endY) = box.astype("int")

		# draw the bounding box of the face along with the associated
		# probability
		text = "{:.2f}%".format(confidence * 100)
		y = startY - 10 if startY - 10 > 10 else startY + 10
		cv2.rectangle(frame, (startX, startY), (endX, endY),
			(0, 0, 255), 2)
		cv2.putText(frame, text, (startX, y),
```

```
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)

        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

```
PS C:\project-1\PyPower_face-detection-20241218T125031Z-001\PyPower_face-detect
ion> python detect_faces_video.py --prototxt deploy.prototxt.txt --model res10_
300x300_ssd_iter_140000.caffemodel
[INFO] loading model...
[INFO] starting video stream...
```

Output :

Camera will be opened > and your face will be detected 100%.