Lab – 8

CE004
20CEUOD004

Aim: Leveraging machine learning using Mahout like tools

1. Run recommendation of a sample dataset using Mahout Library.

```
from .social_distancing_config import NMS_THRESH
from .social_distancing_config import MIN_CONF
import numpy as np
import cv2

def detect_people(frame, net, ln, personIdx=0):

    (H, W) = frame.shape[:2]
    results = []
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),swapRB=True, crop=False)
    net.setInput(blob)
    layerOutputs = net.forward(ln)
    boxes = []
    centroids = []
    confidences = []

    
```

```
    for output in layerOutputs:
        for detection in output:
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            if classID == personIdx and confidence > MIN_CONF:
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int")
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                boxes.append([x, y, int(width), int(height)])
                centroids.append((centerX, centerY))
                confidences.append(float(confidence))

    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF, NMS_THRESH)

    if len(idxs) > 0:
        for i in idxs.flatten():
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            r = (confidences[i], (x, y, x + w, y + h), centroids[i])
            results.append(r)

    return results
```

```
#Set the path of “yolo-coco” library
MODEL_PATH = "yolo-coco"

MIN_CONF = 0.3
#set the minimum threshold
NMS_THRESH = 0.3

USE_GPU = False

#minimum safe distance between two people from each other.
MIN_DISTANCE = 100
```

```
from mylib import config, thread
from mylib.mailer import Mailer
from mylib.detection import detect_people
from imutils.video import VideoStream, FPS
from scipy.spatial import distance as dist
import numpy as np
import argparse, imutils, cv2, os, time, schedule


ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="",
    help="path to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="",
    help="path to (optional) output video file")
ap.add_argument("-d", "--display", type=int, default=1,
    help="whether or not output frame should be displayed")
args = vars(ap.parse_args())
#-------------------------------------------------------------------------#

# load the COCO class labels our YOLO model was trained on

labelsPath = os.path.sep.join([".\\core\\Socialdistancing\\yolo\\coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")


weightsPath = os.path.sep.join([".\\core\\Socialdistancing\\yolo\\yolov3.weights"])
configPath = os.path.sep.join([ ".\\core\\Socialdistancing\\yolo\\yolov3.cfg"])


# load our YOLO object detector trained on COCO dataset (80 classes)
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

```
# loop over the frames from the video stream
while True:
    # read the next frame from the file
    if config.Thread:
        frame = cap.read()

    else:
        (grabbed, frame) = vs.read()
        # if the frame was not grabbed, then we have reached the end of the stream
        if not grabbed:
            break

    # resize the frame and then detect people (and only people) in it
    frame = imutils.resize(frame, width=700)
    results = detect_people(frame, net, ln,
        personIdx=LABELS.index("person"))

    # initialize the set of indexes that violate the max/min social distance limits
    serious = set()
    abnormal = set()
```

```
    if len(results) >= 2:
        # extract all centroids from the results and compute the
        # Euclidean distances between all pairs of the centroids
        centroids = np.array([r[2] for r in results])
        D = dist.cdist(centroids, centroids, metric="euclidean")

        # loop over the upper triangular of the distance matrix
        for i in range(0, D.shape[0]):
            for j in range(i + 1, D.shape[1]):
                # check to see if the distance between any two
                # centroid pairs is less than the configured number of pixels
                if D[i, j] < config.MIN_DISTANCE:
                    # update our violation set with the indexes of the centroid pairs
                    serious.add(i)
                    serious.add(j)
                # update our abnormal set if the centroid distance is below max distance limit
                if (D[i, j] < config.MAX_DISTANCE) and not serious:
                    abnormal.add(i)
                    abnormal.add(j)

    # loop over the results
```

```
    for (i, (prob, bbox, centroid)) in enumerate(results):
        # extract the bounding box and centroid coordinates, then
        # initialize the color of the annotation
        (startX, startY, endX, endY) = bbox
        (cX, cY) = centroid
        color = (0, 255, 0)

        # if the index pair exists within the violation/abnormal sets, then update the color
        if i in serious:
            color = (0, 0, 255)
        elif i in abnormal:
            color = (0, 255, 255) #orange = (0, 165, 255)

        # draw (1) a bounding box around the person and (2) the
        # centroid coordinates of the person,
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.circle(frame, (cX, cY), 5, color, 2)
```

```
    # draw some of the parameters
    Safe_Distance = "Safe distance: >{} px".format(config.MAX_DISTANCE)
    cv2.putText(frame, Safe_Distance, (470, frame.shape[0] - 25),
        cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255, 0, 0), 2)
    Threshold = "Threshold limit: {}".format(config.Threshold)
    cv2.putText(frame, Threshold, (470, frame.shape[0] - 50),
        cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255, 0, 0), 2)

    # draw the total number of social distancing violations on the output frame
    text = "Total serious violations: {}".format(len(serious))
    cv2.putText(frame, text, (10, frame.shape[0] - 55),
        cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 255), 2)

    text1 = "Total abnormal violations: {}".format(len(abnormal))
    cv2.putText(frame, text1, (10, frame.shape[0] - 25),
        cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)

#-----------------------------Alert function----------------------------------#
    if len(serious) >= config.Threshold:
        cv2.putText(frame, "-ALERT: Violations over limit-", (10, frame.shape[0] - 80),
            cv2.FONT_HERSHEY_COMPLEX, 0.60, (0, 0, 255), 2)
```