MASK AND SOCIAL DISTANCE DETECTION:-

Requirements:- TENSORFLOW, KERAS, YOLOV3, OPENCV, NUMPY, SKLEARN

Problem:-

1. If no person is in the video, it should alert “No Person”.
2. If the person walks in, detect the person and assign a unique ID.
3. Detect the person wearing a mask or not with alerts like “Mask Detected” and “No Mask Detected”.
4. If a new person walks in, repeat steps 2 and 3 for him/her.
5. If the two persons are standing close enough, it should alert ”Maintain Social Distancing”.

Problem Breakdown:-

- *Detecting people by YOLOv3*
  - Yolov3 uses darknet architecture which helps in predicting the objects (people in this problem) much faster.
  - Using blobFromImage helps in finding the objects by neglecting the background
  - construct a blob from the input frame and then perform a forward pass of the YOLO object detector, giving us our bounding boxes and associated probabilities
  - extract the class ID and confidence (i.e., probability) of the current object detection
  - filter detections by (1) ensuring that the object detected was a person and (2) that the minimum confidence is met
  - scale the bounding box coordinates back relative to the size of the image, keeping in mind that YOLO actually returns the center (x, y)-coordinates of the bounding box followed by the boxes' width and height
  - use the center (x, y)-coordinates to derive the top and and left corner of the bounding box
  - apply non-maxima suppression to suppress weak, overlapping bounding boxes
  - update our results list to consist of the person prediction probability, bounding box coordinates, and the Centroid.
- *Detecting mask*
  - Training the model with datasets of images with and without masks
  - Load the Images from the datasets and convert them to arrays and then preprocess images
  - Converting these preprocess images to numpy arrays and performing one-hot encoding (LabelBinarizer)
  - partition the data into training and testing splits using 80% of the data for training and the remaining 20% for testing
  - construct the training image generator for data augmentation
  - load the MobileNetV2 network, ensuring the head FC layer sets are left off (Base Mode l) and construct the head of the model that will be placed on top of the base model
  - place the head FC model on top of the base model (Model which will be trained)
  - compile and train the model
  - Serialize the model to the disk (Saving the prediction model)
- *Social Distancing*
  - ensure there are at least two people detections (required in order to compute our pairwise distance maps)
  - extract all centroids from the results and compute the Euclidean distances between all pairs of the centroids
  - check to see if the distance between any two centroid pairs is less than the configured number of pixels
  - update violation set with the indexes of the centroid pairs