Compile YOLOv4 model

```
# Clone social_distance project
!git clone https://github.com/Andresmps/Simulation_project
```

```
# Clone darknet
!git clone https://github.com/AlexeyAB/darknet
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 14518 (delta 0), reused 3 (delta 0), pack-reused 14500
Receiving objects: 100% (14518/14518), 13.24 MiB | 19.68 MiB/s, done.
Resolving deltas: 100% (9864/9864), done.
```

```
# Change makefile to have GPU and OPENCV enabled
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

```
/content/darknet
```

```
# verify CUDA
!/usr/local/cuda/bin/nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
```

```
!export PATH=$PATH:/opt/VirtualGL/bin
!export PATH=$PATH:/usr/local/cuda-10.1/bin
!export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-10.1/lib64
```

```
!./build.sh
```

```
-- Autodetected CUDA architecture(s):  7.5
-- Building with CUDA flags: -gencode;arch=compute_75,code=sm_75
-- Your setup supports half precision (it requires CC >= 7.0)
--   ->  darknet is fine for now, but uselib_track has been disabled!
--   ->  Please rebuild OpenCV from sources with CUDA support to enable it
-- CMAKE_CUDA_FLAGS: -gencode arch=compute_75,code=sm_75 --compiler-options " -Wall -Wno-unused-re
-- Configuring done
-- Generating done
-- Build files have been written to: /content/darknet/build_release
[ 98%] Built target darknet
```

```
[ 98%] Built target dark
[100%] Built target uselib
Install the project...
-- Install configuration: "Release"
-- Up-to-date: /content/darknet/libdarknet.so
-- Up-to-date: /content/darknet/include/darknet/darknet.h
-- Up-to-date: /content/darknet/include/darknet/yolo_v2_class.hpp
-- Up-to-date: /content/darknet/uselib
-- Up-to-date: /content/darknet/darknet
-- Up-to-date: /content/darknet/share/darknet/DarknetTargets.cmake
-- Installing: /content/darknet/share/darknet/DarknetTargets-release.cmake
-- Up-to-date: /content/darknet/share/darknet/DarknetConfig.cmake
-- Installing: /content/darknet/share/darknet/DarknetConfigVersion.cmake
```

```
# Download pre-trained YOLOv4 weights
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
```

```
import darknet
```

```
help(darknet)
# import social_distance_monitor_using_yolov4 as yolov4
```

```
|      _ctypes.Structure
|      _ctypes._CData
|      builtins.object
|
|  Data descriptors defined here:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  bbox
|      Structure/Union member
|
|  classes
|      Structure/Union member
|
|  embedding_size
|      Structure/Union member
|
|  embeddings
|      Structure/Union member
|
|  mask
|      Structure/Union member
|
|  objectness
|      Structure/Union member
|
|  points
|      Structure/Union member
|
```

```
|  prob
|      Structure/Union member
|
|  sim
|      Structure/Union member
|
|  sort_class
|      Structure/Union member
|
|  track_id
|      Structure/Union member
|
|  uc
|      Structure/Union member
|
|  ----------------------------------------------------------------------
|  Methods inherited from _ctypes.Structure:
|
|  __init__(self, /, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __new__(*args, **kwargs) from _ctypes.PyCStructType
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
```

```
# file_input = 'OxfordTownCenter.mp4'
# yolov4.YOLO()
```

```
import matplotlib.pyplot as plt
import numpy as np
import darknet
import random
import math
import time
import cv2
import os


from itertools import combinations
from PIL import Image
from ctypes import *


def euclidean_distance(p1, p2):
    distance = math.sqrt(p1 ** 2 + p2 ** 2)
    return distance


def convertBack(x, y, w, h):
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))
    return xmin, ymin, xmax, ymax
```

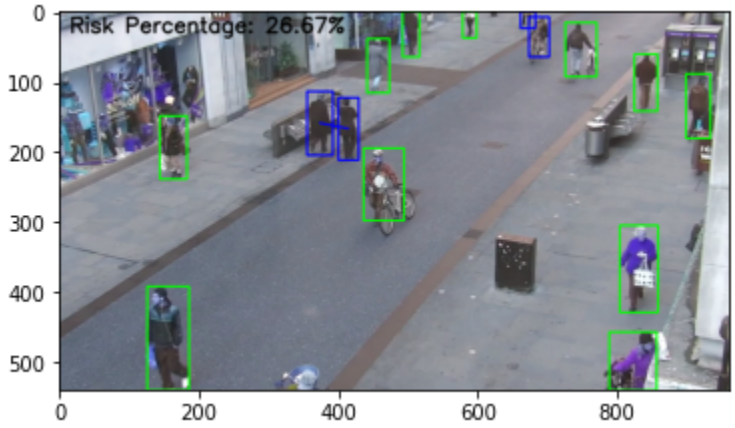
```
    return xmin, ymin, xmax, ymax


def cvDrawBoxes(detections, img):
    if len(detections) > 0:
        centroid_dict = dict()
        object_id = 0
        for label, confidence, bbox in detections:
            if label == 'person' and float(confidence) > 50:
                x, y, w, h = (bbox[0], bbox[1], bbox[2], bbox[3])
                print(f"Person # {object_id}, located: ({int(x)}, {int(y)}) " +
                      f"confidence: {confidence}")
                xmin, ymin, xmax, ymax = convertBack(float(x), float(y),
                                                     float(w), float(h))
                centroid_dict[object_id] = (int(x), int(y), xmin,
                                            ymin, xmax, ymax)
                object_id += 1


        red_zone_list = list()
        red_line_list = list()

        for (id1, p1), (id2, p2) in combinations(centroid_dict.items(), 2):
            dx, dy = p1[0] - p2[0], p1[1] - p2[1]
            distance = euclidean_distance(dx, dy)
            if distance < 60.0:
                if id1 not in red_zone_list:
                    red_zone_list.append(id1)
                    red_line_list.append(p1[0:2])
                if id2 not in red_zone_list:
                    red_zone_list.append(id2)
                    red_line_list.append(p2[0:2])
        for idx, box in centroid_dict.items():
            if idx in red_zone_list:
                cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]),
                              (255, 0, 0), 2)
            else:
                cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]),
                              (0, 255, 0), 2)
        amount_people = len(centroid_dict.keys())
        amount_bad_people = len(red_zone_list)

        print(f"\nTotal number of people {amount_people}\n" +
              f"Total number of people who break social distancing measure " +
              f"{amount_bad_people}\nPeople ids who break social " +
              f"distancing measure {red_zone_list}\n")

        risk_percentage = round((amount_bad_people/amount_people)*100, 2)
        text = f"Risk Percentage: {str(risk_percentage)}%"
        location = (15, 30)
        cv2.putText(img, text, location, cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0, 0, 0), 2, cv2.LINE_AA)

        for check in range(0, len(red_line_list) - 1):
```

```
            start_point = red_line_list[check]
            end_point = red_line_list[check + 1]
            check_line_x = abs(end_point[0] - start_point[0])
            check_line_y = abs(end_point[1] - start_point[1])
            if (check_line_x < 75) and (check_line_y < 25):
                cv2.line(img, start_point, end_point, (255, 0, 0), 2)
    return img, risk_percentage


netMain = None
metaMain = None
altNames = None


def YOLO(file_input):
    global metaMain, netMain, altNames
    configPath = "./cfg/yolov4.cfg"
    weightPath = "./yolov4.weights"
    metaPath = "./cfg/coco.data"
    if not os.path.exists(configPath):
        raise ValueError("Invalid config path `" +
                         os.path.abspath(configPath) + "`")
    if not os.path.exists(weightPath):
        raise ValueError("Invalid weight path `" +
                         os.path.abspath(weightPath) + "`")
    if not os.path.exists(metaPath):
        raise ValueError("Invalid data file path `" +
                         os.path.abspath(metaPath) + "`")

    network, class_names, class_colors = darknet.load_network(configPath,
                                                              metaPath,
                                                              weightPath,
                                                              batch_size=1)

    cap = cv2.VideoCapture(file_input)
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    new_height, new_width = frame_height // 2, frame_width // 2
    file_output = file_input.replace('.mp4', '_out.avi')
    file_output = file_output.replace('Input', 'Output')
    out = cv2.VideoWriter(file_output, cv2.VideoWriter_fourcc(*"MJPG"), 10.0,
                          (new_width, new_height))

    darknet_image = darknet.make_image(new_width, new_height, 3)
    c = 0
    total_risk = 0
    while True:
        prev_time = time.time()
        ret, frame_read = cap.read()
        if (not ret) or (c > 150):  # Number of frames for the resulting video
            break                   # If you want all the output comment it, but
        c += 1                      # it will last a lot.
        print(f"** Frame # {c} **\n")
        frame_rgb = cv2.cvtColor(frame_read, cv2.COLOR_BGR2RGB)
```

```
        frame_resized = cv2.resize(frame_rgb, (new_width, new_height),
                                   interpolation=cv2.INTER_LINEAR)

        darknet.copy_image_from_bytes(darknet_image, frame_resized.tobytes())
        detections = darknet.detect_image(network, class_names, darknet_image,
                                          thresh=0.25)

        image, risk = cvDrawBoxes(detections, frame_resized)
        total_risk += risk
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        out.write(image)
        # print(1 / (time.time() - prev_time))

        img = Image.fromarray(image, 'RGB')
        plt.imshow(img)
        plt.show()
        img.close()

    total_risk = round(total_risk/c, 2)
    cap.release()
    out.release()
    print("** End of Detection **\n** The percentage risk for this video is " +
          f"about {total_risk}%\nPlease look at the output of the system" +
          " in the file located in Simulation_project/data/Output **")
```

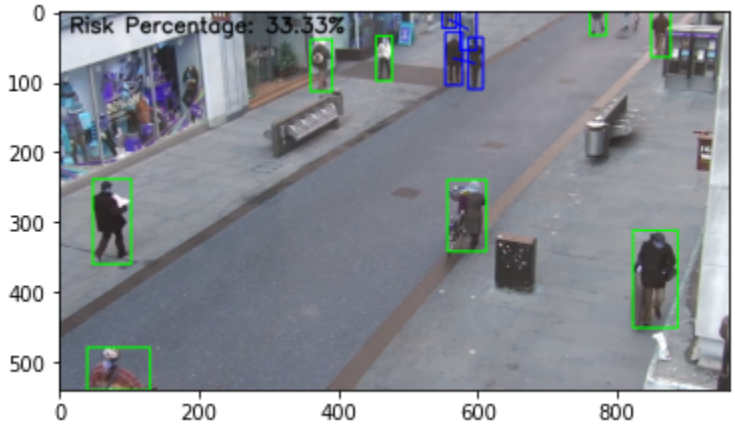
```
file_input = '/content/Simulation_project/data/Input/OxfordTownCenter.mp4'
YOLO(file_input)
```

```
** Frame # 1 **

Person # 0, located: (504, 32) confidence: 61.38
Person # 1, located: (671, 11) confidence: 76.59
Person # 2, located: (915, 135) confidence: 84.44
Person # 3, located: (588, 18) confidence: 85.65
Person # 4, located: (687, 36) confidence: 86.06
Person # 5, located: (163, 193) confidence: 88.95
Person # 6, located: (821, 498) confidence: 91.17
Person # 7, located: (464, 246) confidence: 92.79
Person # 8, located: (456, 77) confidence: 95.1
Person # 9, located: (840, 101) confidence: 96.5
Person # 10, located: (155, 466) confidence: 97.83
Person # 11, located: (373, 159) confidence: 98.01
Person # 12, located: (830, 367) confidence: 98.52
Person # 13, located: (414, 167) confidence: 98.68
Person # 14, located: (747, 54) confidence: 98.74

Total number of people 15
Total number of people who break social distancing measure 4
People ids who break social distancing measure [1, 4, 11, 13]
```




```
** Frame # 2 **

Person # 0, located: (504, 32) confidence: 61.38
Person # 1, located: (671, 11) confidence: 76.59
Person # 2, located: (915, 135) confidence: 84.44
Person # 3, located: (588, 18) confidence: 85.65
Person # 4, located: (687, 36) confidence: 86.06
Person # 5, located: (163, 193) confidence: 88.95
Person # 6, located: (821, 498) confidence: 91.17
Person # 7, located: (464, 246) confidence: 92.79
Person # 8, located: (456, 77) confidence: 95.1
Person # 9, located: (840, 101) confidence: 96.5
Person # 10, located: (155, 466) confidence: 97.83
Person # 11, located: (373, 159) confidence: 98.01
Person # 12, located: (830, 367) confidence: 98.52
Person # 13, located: (414, 167) confidence: 98.68
Person # 14, located: (747, 54) confidence: 98.74

Total number of people 15
Total number of people who break social distancing measure 4
People ids who break social distancing measure [1, 4, 11, 13]
```

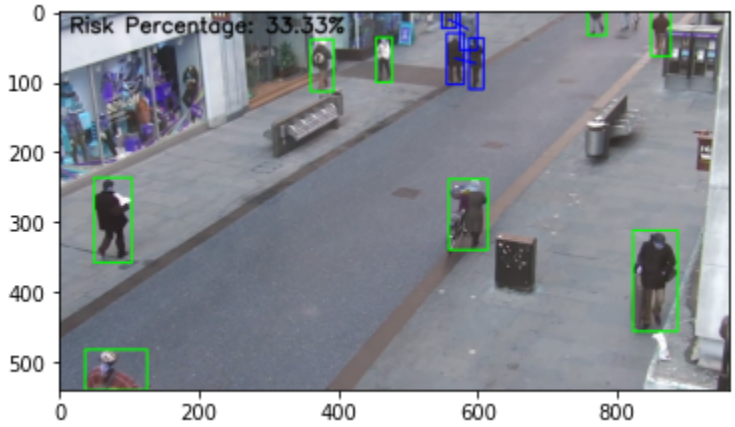
People ids who break social distancing measure [0, 1, 6, 7]



** Frame # 150 **

Person # 0, located: (558, 11) confidence: 60.58
Person # 1, located: (586, 26) confidence: 68.8
Person # 2, located: (81, 510) confidence: 70.43
Person # 3, located: (770, 17) confidence: 78.36
Person # 4, located: (376, 76) confidence: 85.74
Person # 5, located: (861, 31) confidence: 85.86
Person # 6, located: (567, 67) confidence: 94.77
Person # 7, located: (597, 74) confidence: 95.88
Person # 8, located: (585, 289) confidence: 95.91
Person # 9, located: (465, 68) confidence: 96.15
Person # 10, located: (76, 297) confidence: 96.23
Person # 11, located: (853, 384) confidence: 97.29

Total number of people 12
Total number of people who break social distancing measure 4
People ids who break social distancing measure [0, 1, 6, 7]

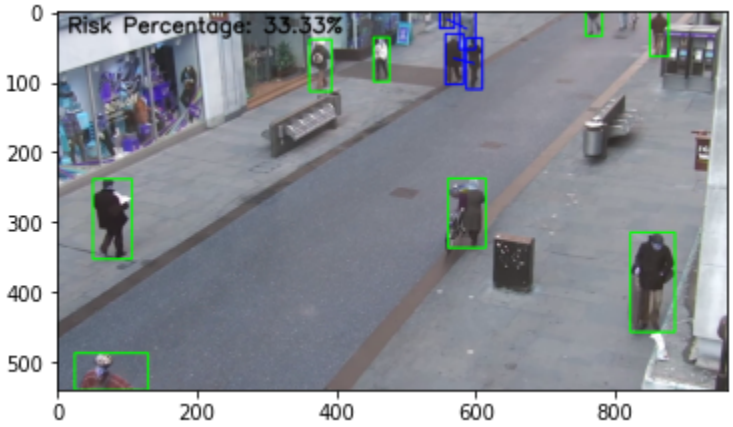


** Frame # 151 **

Person # 0, located: (558, 11) confidence: 50.11
Person # 1, located: (587, 25) confidence: 62.82
Person # 2, located: (77, 513) confidence: 73.24
Person # 3, located: (377, 76) confidence: 76.07
Person # 4, located: (769, 17) confidence: 80.9
Person # 5, located: (862, 31) confidence: 83.69
Person # 6, located: (587, 287) confidence: 93.85
Person # 7, located: (464, 68) confidence: 95.08
Person # 8, located: (568, 67) confidence: 95.15
Person # 9, located: (597, 74) confidence: 95.21
Person # 10, located: (79, 295) confidence: 97.05

Person # 11, located: (852, 385) confidence: 98.45

Total number of people 12
Total number of people who break social distancing measure 4
People ids who break social distancing measure [0, 1, 8, 9]



** End of Detection **
** The percentage risk for this video is about 42.64%
Please look at the output of the system in the file located in Simulation_project/data/Output **