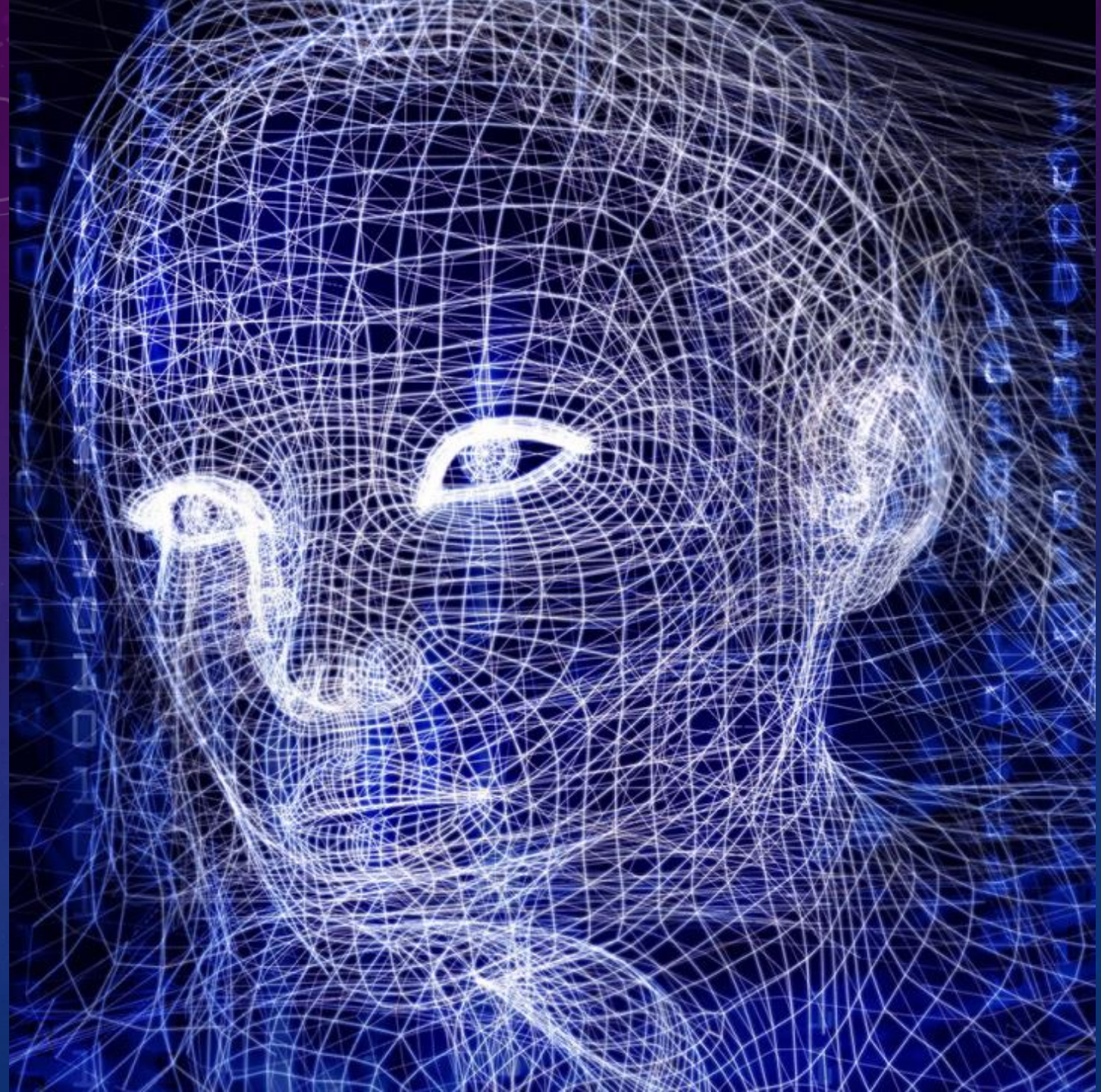
LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

CH5 OBJECT DETECTION WITH YOLO



徐繼聖

Gee-Sern Jison Hsu

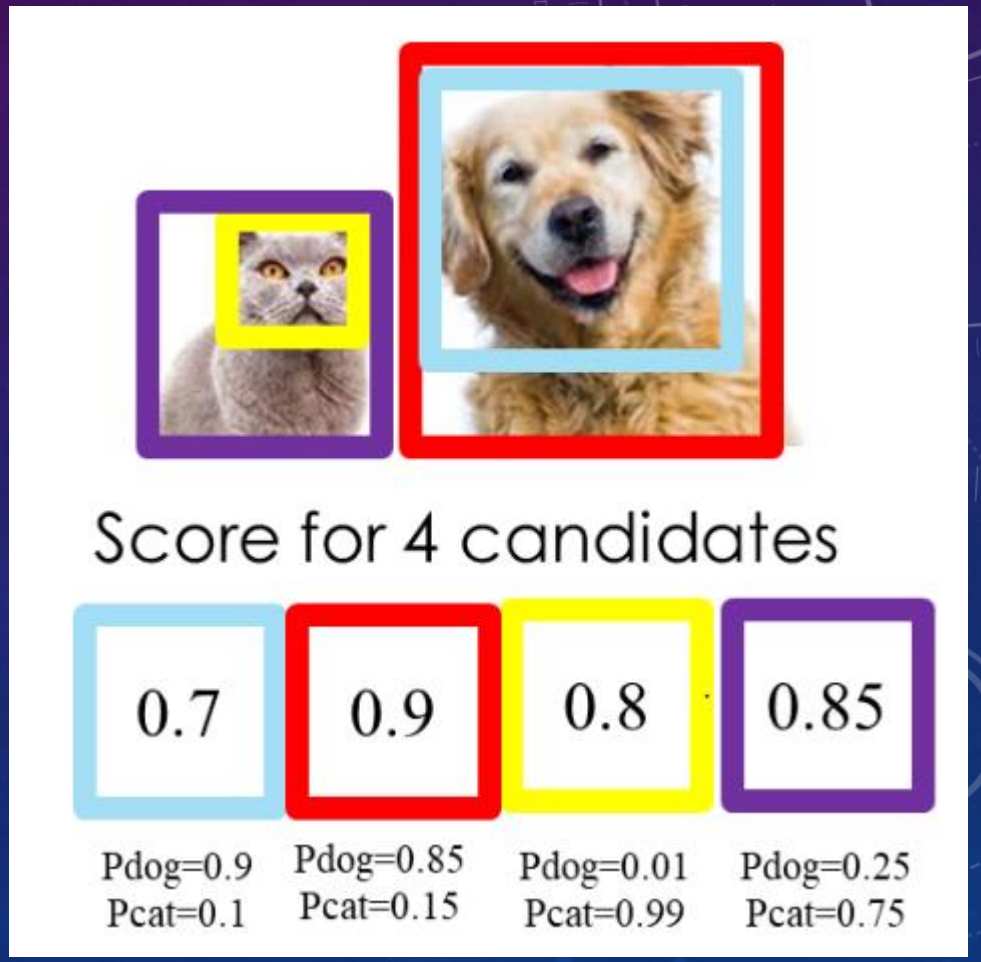
National Taiwan University of Science and Technology

NMS

PLEASE CALCULATE BY HAND. IN CASE YOU RUN INTO ISSUES, TAKE A LOOK AT PAGE 14

Exercise 5.1 – Non-Maximum Suppression (NMS)

- Please do the same steps from the example in the slides and decide the bounding box of each objects in the figure.
- Please write down result in MS Word to Moodle



Example 5-2 K Means

Example 5-2 K-Means

- In the improvement of yolov2, the calculation of the anchors box is through k-means. In this exercise, we will visualize k-means to understand how it works.

Example 5-2 K-Means

The algorithm works as follows:

1. First we initialize k points, called means.
2. We categorize each items to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

Example 5-2 K-Means

```
seed_num = 3
dot_num = 20
```

3 clusters and 20data

```
#initial element
x = np.random.randint(0, 1000, dot_num)
y = np.random.randint(0, 1000, dot_num)

#initial cluster center
kx = np.random.randint(0, 1000, seed_num)
ky = np.random.randint(0, 1000, seed_num)



#2 point distance
def dis(x, y, kx, ky):
    return int(((kx-x)**2 + (ky-y)**2)**0.5)
```

```
#Group each element
def cluster(x, k, kx, ky):
    team = []
    for i in range(3):
        team.append([])
    mid_dis = 99999999
    for i in range(dot_num):
        for j in range(seed_num):
            distant = dis(x[i], y[i], kx[j], ky[j])
            if distant < mid_dis:
                mid_dis = distant
                flag = j
        team[flag].append([x[i], y[i]])
        mid_dis = 99999999
    return team
```

→ Number of cluster

Example 5-2 K-Means

```
 #k-means Grouping
def kmeans(x, y, kx, ky, fig):
    team = cluster(x, y, kx, ky)
    nkx, nky = re_seed(team, kx, ky)

    # plot: nodes connect to seeds
    cx = []
    cy = []
    line = plt.gca()
    for index, nodes in enumerate(team):
        for node in nodes:
            cx.append([node[0], nkx[index]])
            cy.append([node[1], nky[index]])
        for i in range(len(cx)):
            line.plot(cx[i], cy[i], color='r', alpha=0.3)
        cx = []
        cy = []
```

Exercise 5-2 – K-Means exercise

1. Please adjust the number of clusters and data by yourself and observe the changes.

- The needed data “k-means.py” is given on Moodle.

Please write down result and your code in MS Word to Moodle

YOLOv4 with COLAB

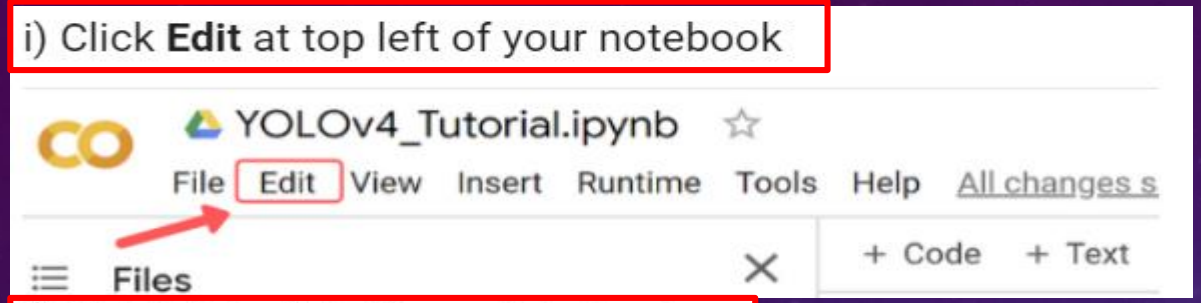
TO MAKE THE SETUP EASY FOR EVERYONE, WE WILL USE GOOGLE COLAB.

Example 5-3 – Yolov4

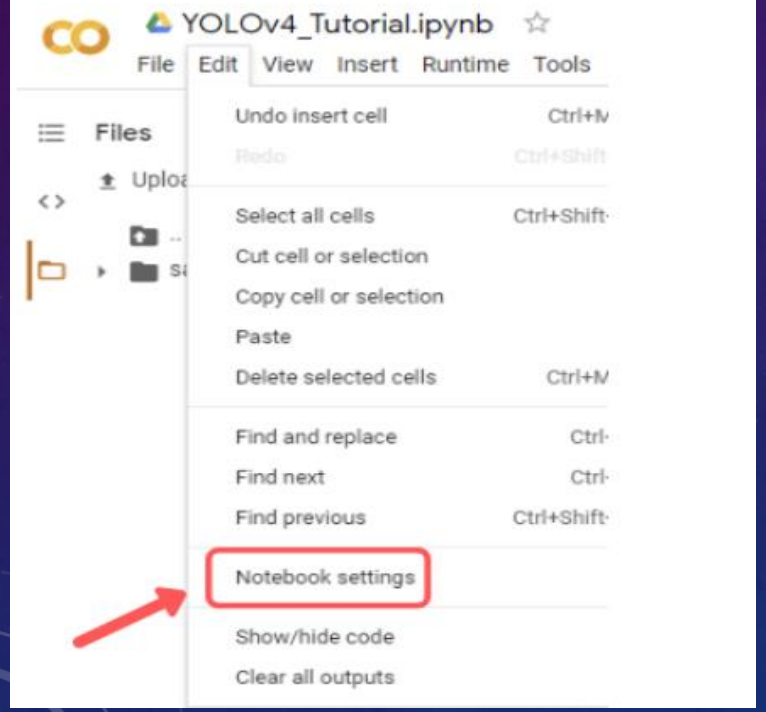
Last time, you used Facebooks's detectron2 for our experiments. This time we will use the famous YOLO network to detect objects of our interest. Again, we will test different pre-trained models. The jupyter notebook comes with a detailed description of commands and guidance.

Example 5-3 Yolov4 – Enabling GPU

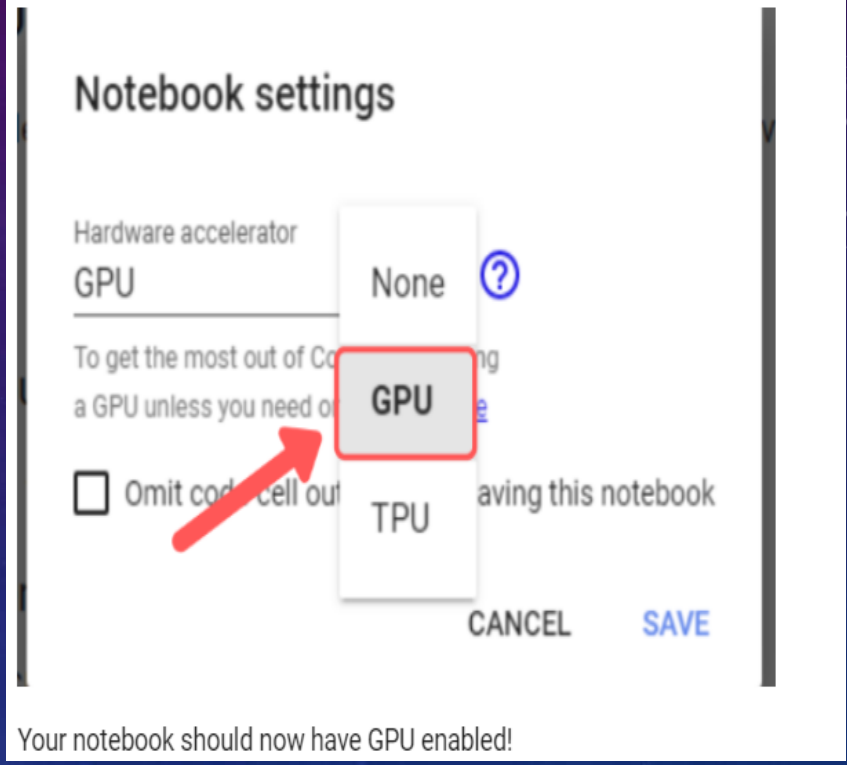
i) Click **Edit** at top left of your notebook



ii) Click **Notebook Settings** within dropdown



iii) Under 'Hardware Accelerator' select **GPU** and then hit **Save**



Your notebook should now have GPU enabled!

- make sure that pytorch, cuda and the GPU are ready to go

Example 5-3 Yolov4 – Cloning and Building Darknet

- The following cells will clone darknet from AlexeyAB's famous repository, adjust the Makefile to enable OPENCV and GPU for darknet and then build darknet.

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

```
# 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
```

```
# make darknet
!make
```

Example 5-3 Yolov4 – What is darknet ?

- Darknet is an open source neural network framework written in C and CUDA.
- It is fast, easy to install, and supports CPU and GPU computation
- You can read more about what Darknet can do right here:

https://pjreddie.com/darknet/

Example 5-3 Yolov4 – Download pretrained weights

- YOLOv4 has been trained already on the coco dataset which has 80 classes that it can predict. We will grab these pretrained weights so that we can run YOLOv4 on these pretrained classes and get detections.

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
```

- We are chosing the pretrained weights
- There are several other weights available

https://github.com/AlexeyAB/darknet#pre-trained-models

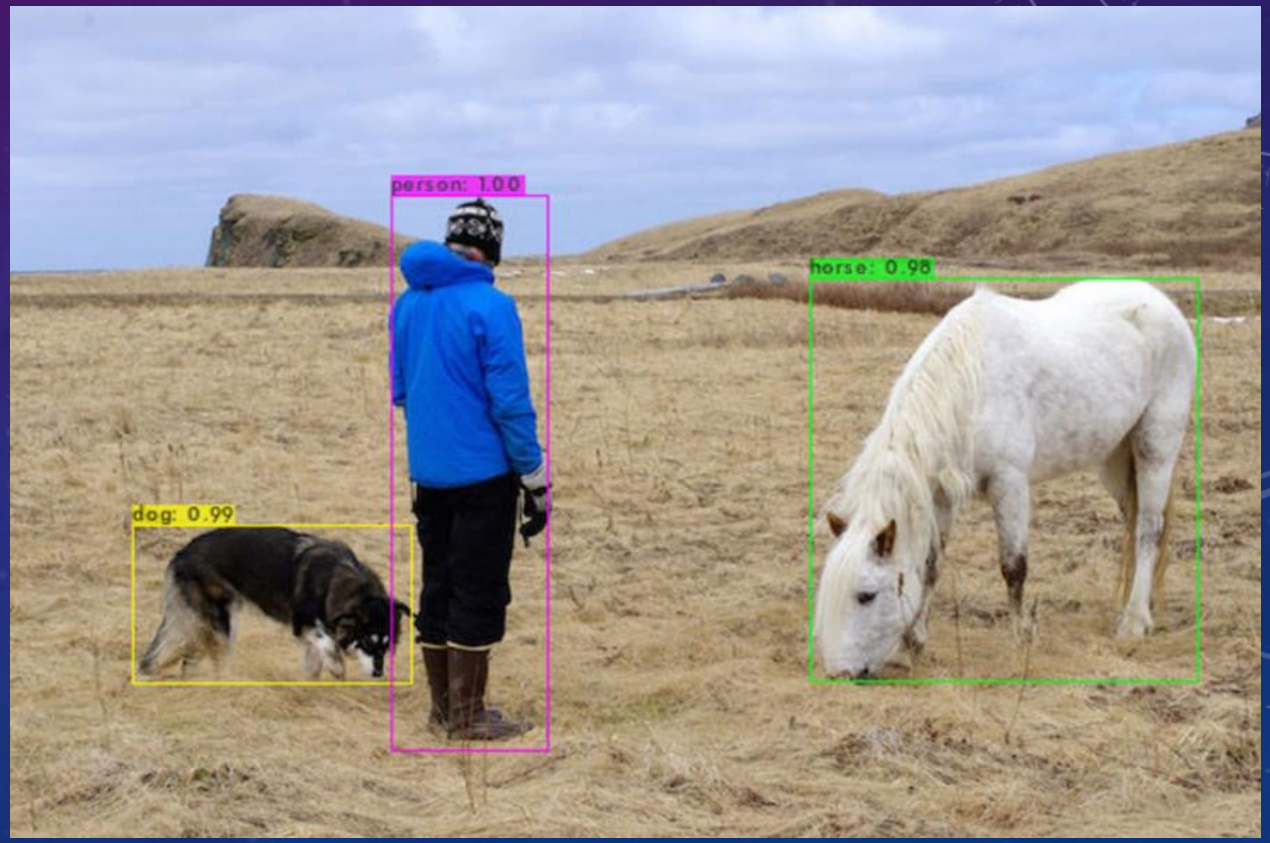
Example 5-3 Yolov4 – Inference on a example image

```
# run darknet detection on test images
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg
```

Test data

Pretrain model weights

The pipeline is working.
Lets try on our own image.



Example 5-3 Yolov4 – Inference on a example image

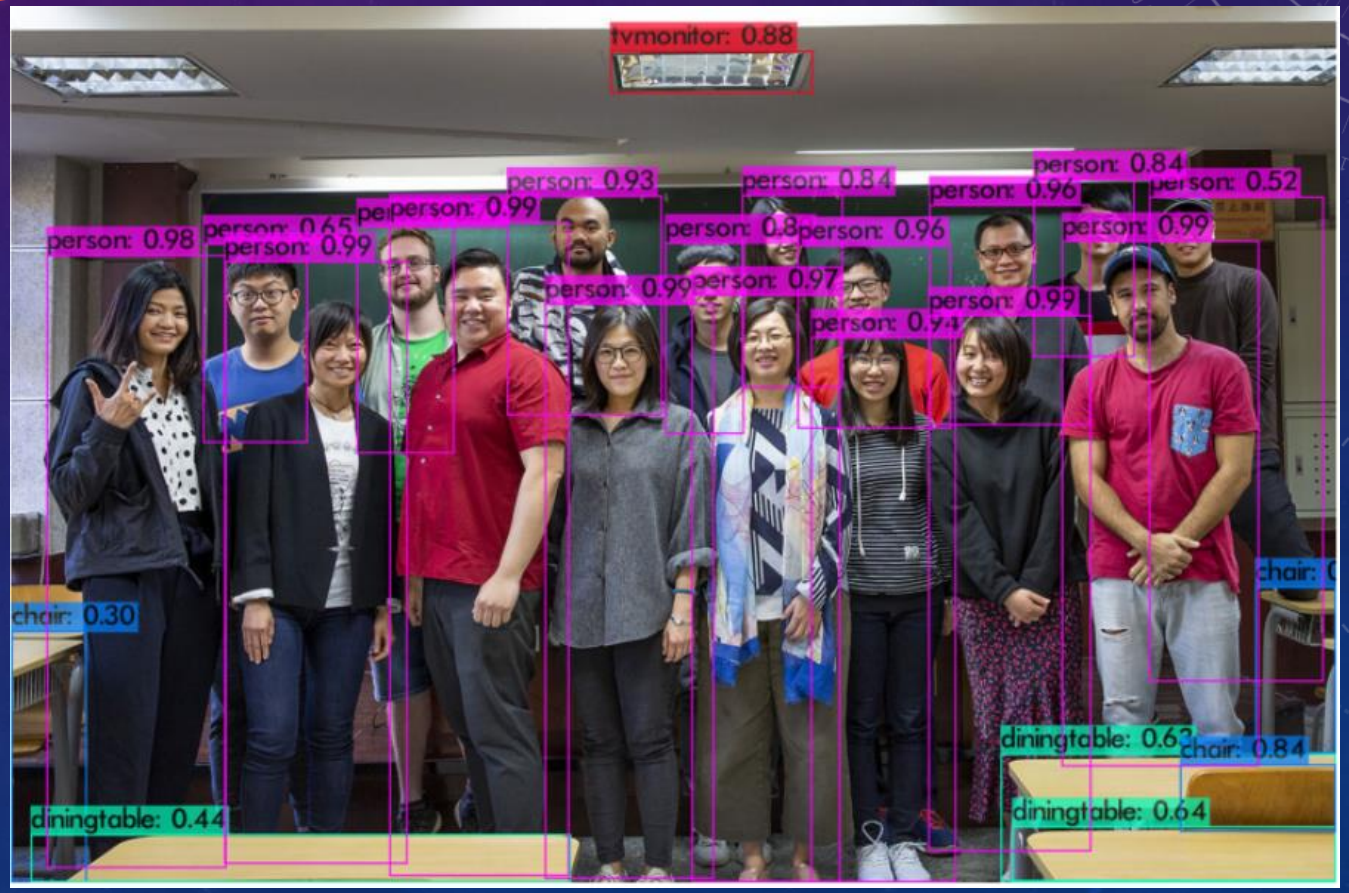
Try on our own image.

```
# run darknet with YOLOv4 on your personal image! (note yours will not be called highway.jpg so change the name)
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights ../class.jpg
imShow('predictions.jpg')
```

Test data

Pretrain model weights

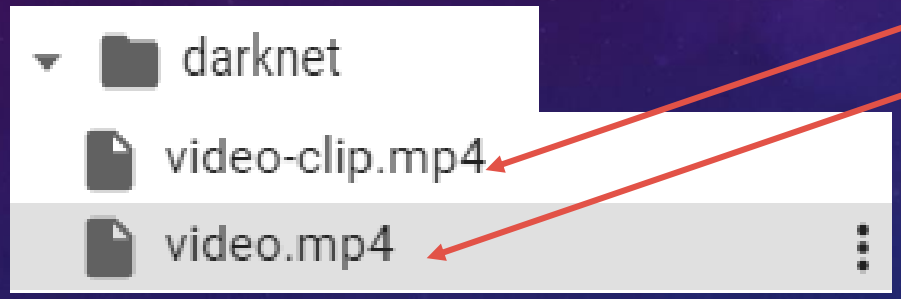
- Yolo is also doing great for crowded images and does not miss a single person.



Example 5-3 Yolov4 – Video Input

Download the video, and crop 6 seconds for processing.

```
!pip install youtube-dl
!pip uninstall -y opencv-python-headless opencv-contrib-python
!apt install python3-opencv    # the one pre-installed have some issues
!youtube-dl https://www.youtube.com/watch?v=l18TgCZ0plk -f 22 -o video.mp4
!ffmpeg -i video.mp4 -t 00:00:06 -c:v copy video-clip.mp4
```




```
!./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights -dont_show /content/darknet/video-clip.mp4 -i 0 -out_filename results.avi
```

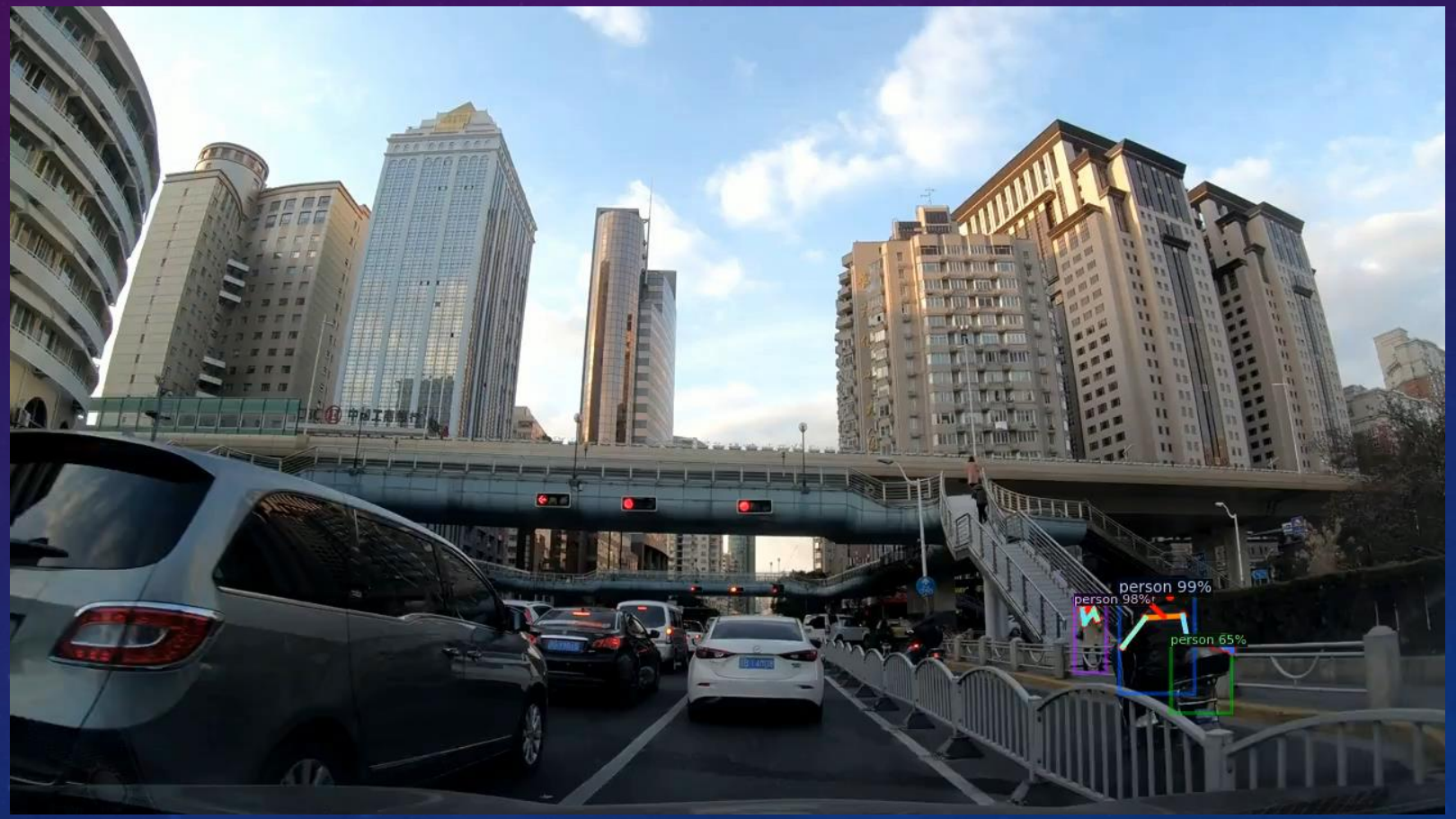
Pretrain model weights

Your video data

Output

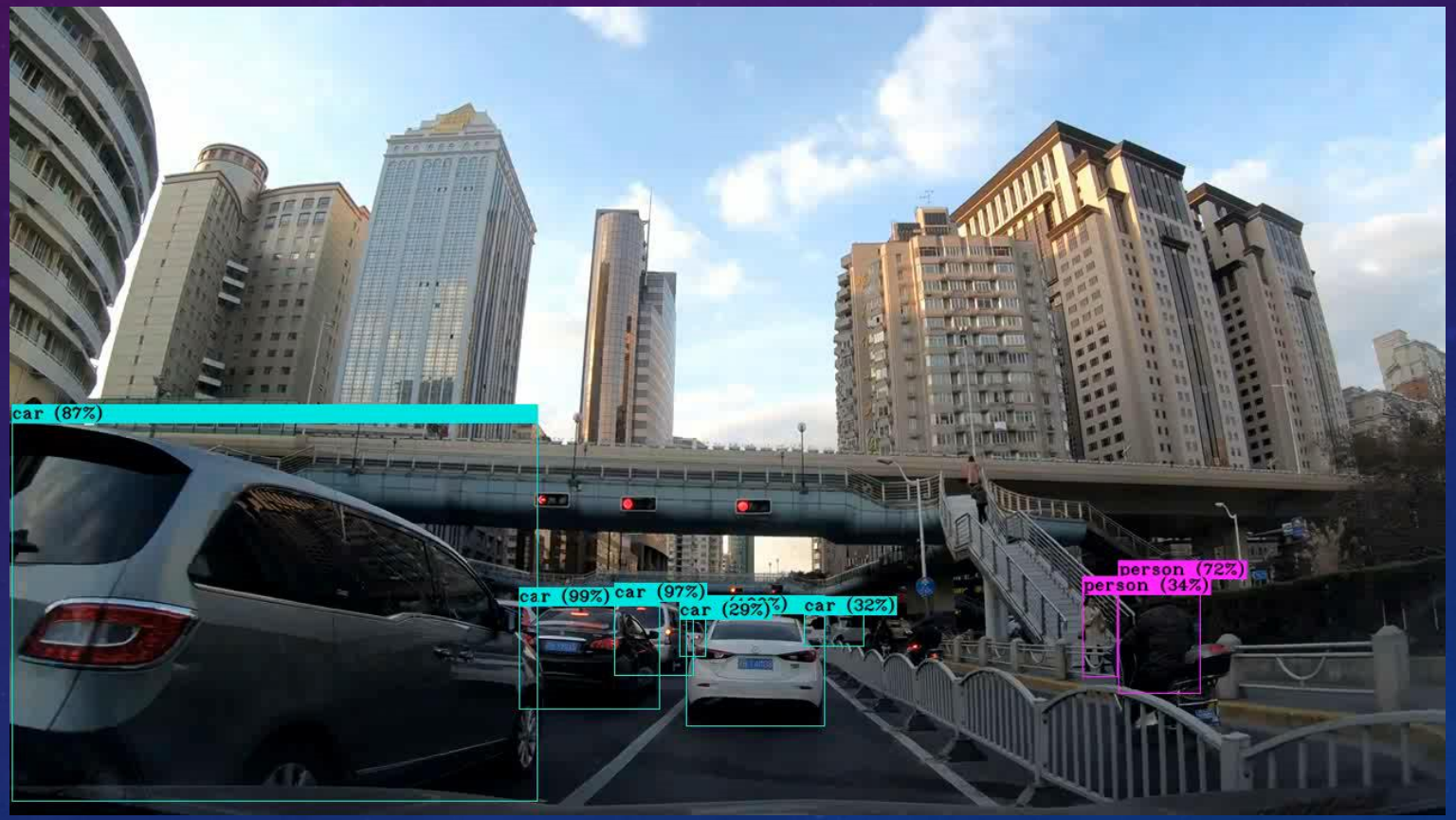
Example 5-3 Yolov4 – Video Input

- Do you remember last weeks example with Detectron2? Let’s see the video



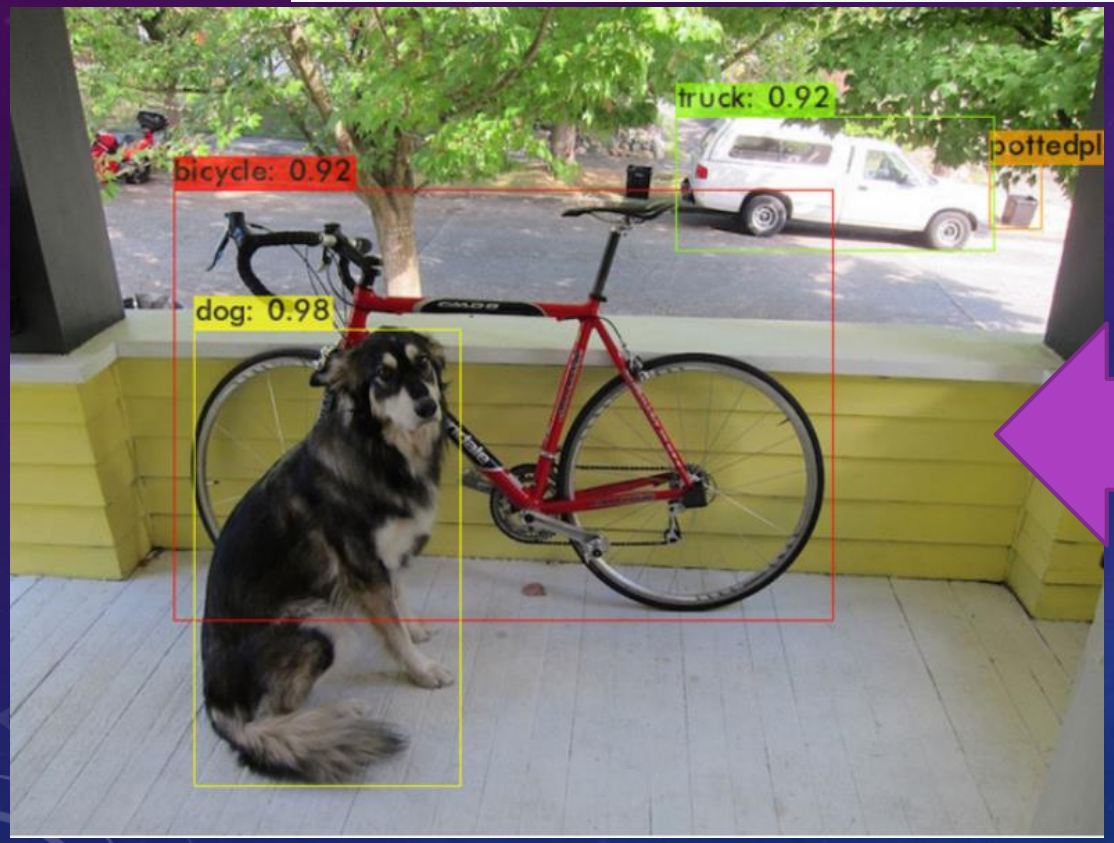
Example 5-3 Yolov4 – Video Input

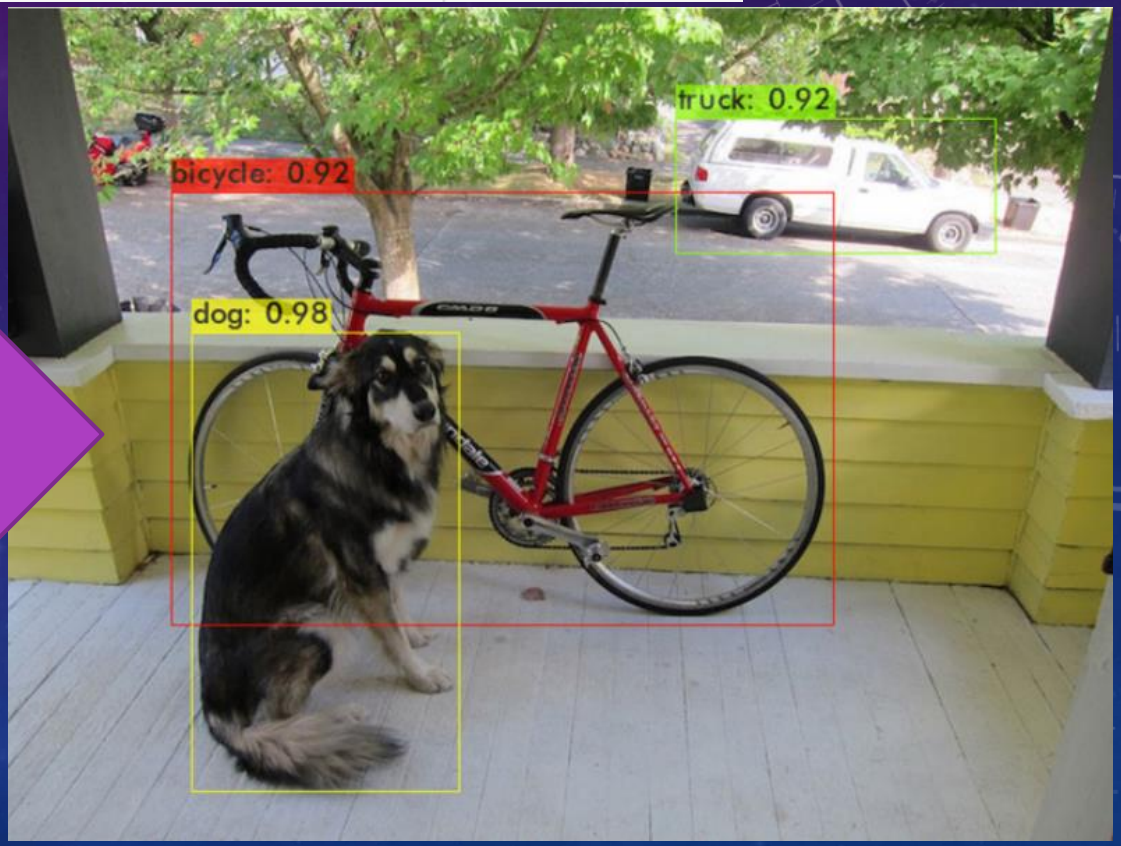
- We tried the same video with YOLO. Let’s see the results



Example 5-3 Yolov4 – Flags – Changed thresholds – Spot the difference!

```
# same detections but ran with the threshold flag set to 0.5 (pottedplant is no longer detected!)
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg -thresh 0.5
imShow('predictions.jpg')
```

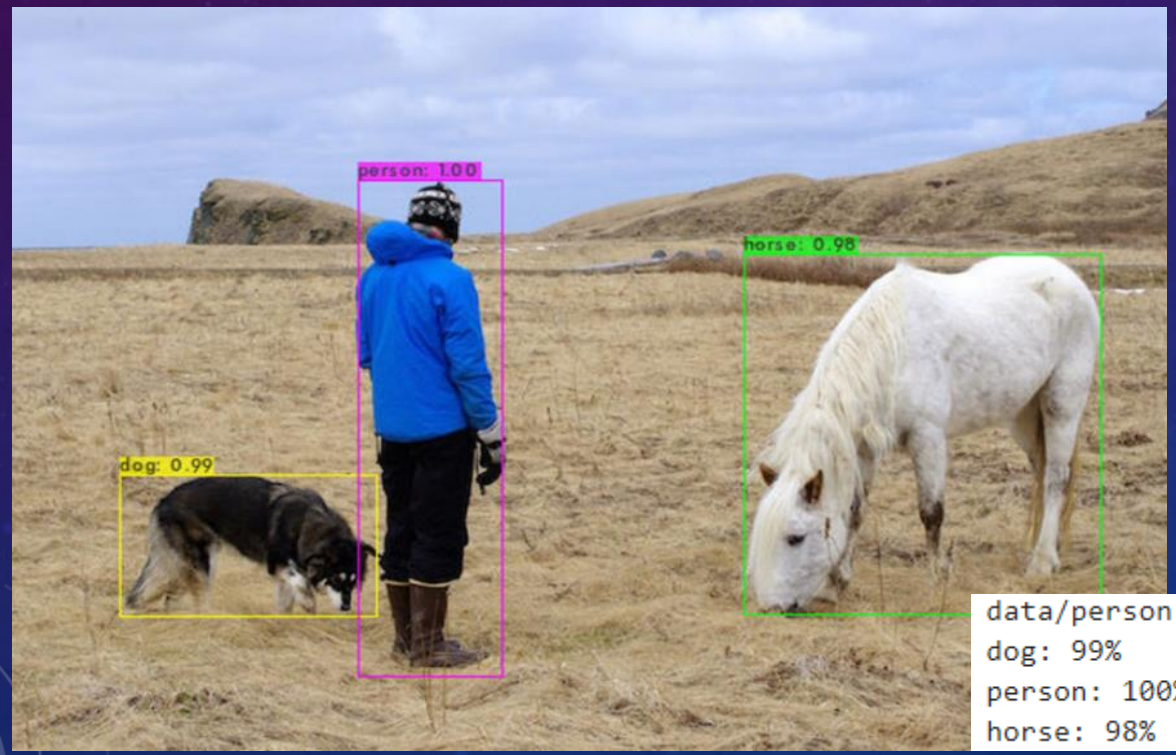




Example 5-3 Yolov4 – Flags – Outputting Bounding Box Coordinates

You can output bounding box coordinates for each detection with the flag '-ext_output'. This external outputs flag will give you a few extra details about each detection within an image.

```
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg -ext_output
imShow('predictions.jpg')
```




```
data/person.jpg: Predicted in 54.798000 milli-seconds.
dog: 99%        (left_x:   62   top_y:  265   width:  142   height:   80)
person: 100%    (left_x:  194   top_y:   98   width:   78   height:  281)
horse: 98%      (left_x:  406   top_y:  140   width:  196   height:  204)
```

Exercise 5-3 - Yolov4

- Please download “YOLO_tutorial.ipynb” and open it with Colab
- Take some of the pictures that you have taken for other detectors from previous classes. Put them inside the YOLO detector and compare their performance
- Choose the video you chose last week and run it inside the YOLO detection pipeline. Compare it to Detectron2.
- Change the pre-trained weights to a different pretraining, e.g. coco or the tiny weights, and compare with the same footage
- Play around with the threshold flag and make an experiment as above, where you see a detection disappearing or appearing
- Finally, it is quite common to test on multiple images at once. Therefore, use the last part of the provided notebook and the 5 images by yourself, make your so-called file list, run your predictions, put them in a .json file and upload it to moodle.