San Jose State University

Department of Electrical Engineering

EE104, Fall 2022, Pham

Laboratory Assignment #8

# Objectives

This lab demonstrates your ability to enhance recognition capability based on an existing Yolov5 dataset.

This lab also demonstrates your ability to create an arcade-class game with sound and graphic setting to allow a challenge dance competition between two game players.

# Grading

Refer to the section **Python Programming** for grading criteria.

# Bibliography

I would like to acknowledge the Python open-source community and respective suppliers for making the material available.

References:

<https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4>

<https://wandb.ai/onlineinference/YOLO/reports/Collect-and-Label-Images-to-Train-a-YOLOv5-Object-Detection-Model-in-PyTorch--VmlldzoxMzQxODc3>

<https://github.com/ultralytics/yolov5/blob/d68afedb32fb5f3b632f67f2cbea2c89a145f0ad/data/coco128.yaml#L12-L13>

<https://www.youtube.com/watch?v=80Q3HIBy7Qg>

<https://github.com/AarohiSingla/yolov5>

<https://towardsdatascience.com/convert-pascal-voc-xml-to-yolo-for-object-detection-f969811ccba5>

# Download, Installation, and Licensing

## Install necessary Python packages

You will follow all steps from this tutorial to demonstrate that your computer can recognize images from your PC built-in camera (or an USB camera), from a picture, or from YouTube: <https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4>

## Install the tool to label custom images

You will follow all steps from this tutorial up to Creating your YAML file to demonstrate that you can add labels to your new images: <https://wandb.ai/onlineinference/YOLO/reports/Collect-and-Label-Images-to-Train-a-YOLOv5-Object-Detection-Model-in-PyTorch--VmlldzoxMzQxODc3>

(i.e. you will not upload your files to wanb.ai website).

## Download a Yolov5 COCO dataset

Download and unzip the COCO (Common Object in COntext) dataset from here: <https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip>

According to this website, you will see that this dataset has the following 80 classes:

<https://github.com/ultralytics/yolov5/blob/d68afedb32fb5f3b632f67f2cbea2c89a145f0ad/data/coco128.yaml#L12-L13>

# class names

names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',

'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',

'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',

'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',

'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',

'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',

'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',

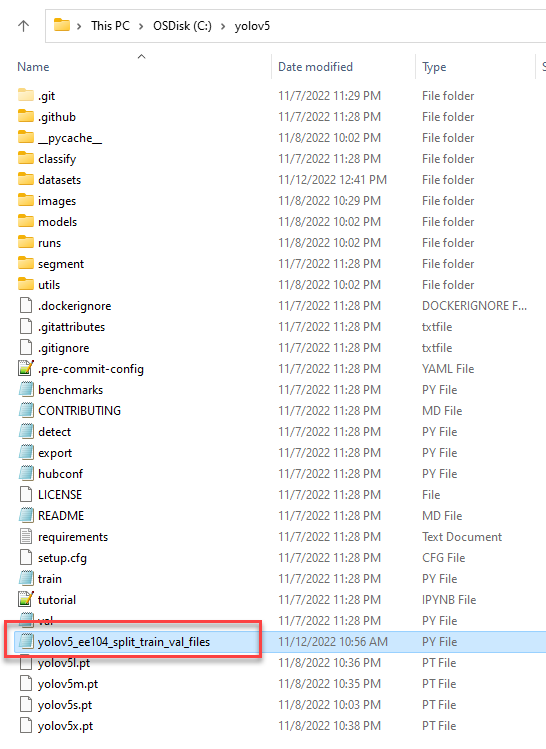
'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',

'hair drier', 'toothbrush' ]

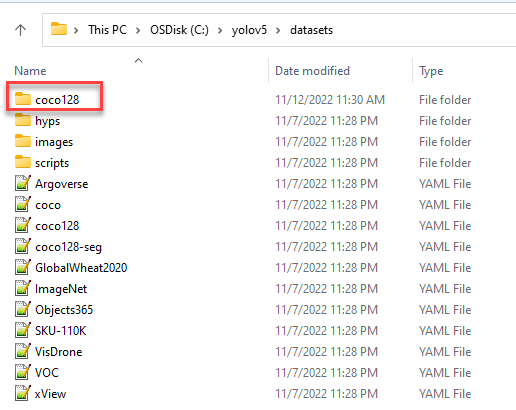
# Directory File Structure

Below is the directory file structure that the sample codes given in this lab is using. You can either create the same directory file structure or modify the code after your actual paths.

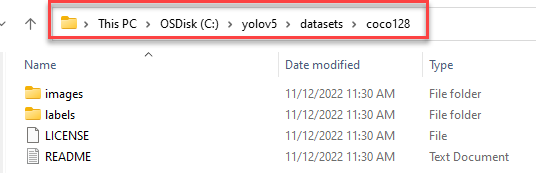
Note that there is a file **yolov5\_ee104\_split\_train\_val\_files.py** that you will run later to split the COCO dataset into 80/20 train/val ratio. You can download this file from Canvas and save it to the C:\yolov5 directory.



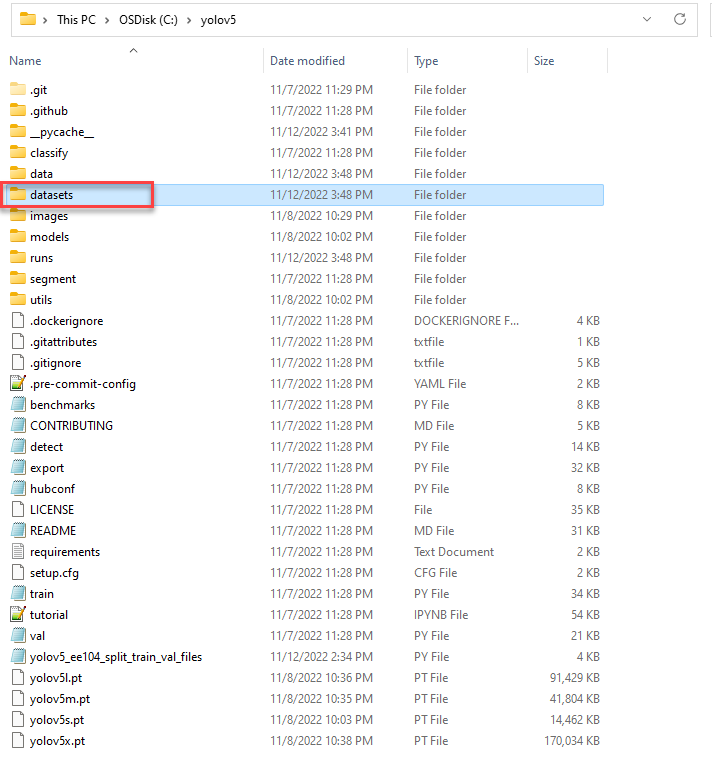
Unzip the COCO dataset that you downloaded earlier, and move the folder coco128 into this subfolder:



Now you should have this subdirectories inside the coco128 directory:



At the top level, you should see your new directory:

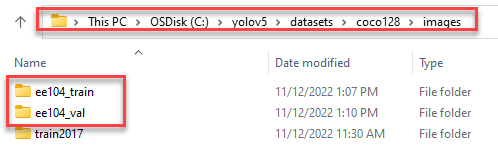


**Create directories for your EE104 Train and Val files:**

Now you must manually create the followings sub-directories if they are not already existing:

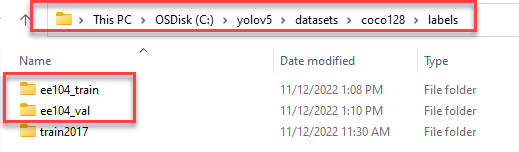
'C:/yolov5/datasets/coco128/images/ee104\_train'

'C:/yolov5/datasets/coco128/images/ee104\_val'



'C:/yolov5/datasets/coco128/labels/ee104\_train'

'C:/yolov5/datasets/coco128/labels/ee104\_val'



# Data Cleaning

1. Find the extra image files and extra txt files

Inspect two COCO source directories, you will see that

C:\yolov5\datasets\coco128\images\train2017 has 2 images that do not have the same txt files in the

C:\yolov5\datasets\coco128\labels\ee104\_val directory.

Also C:\yolov5\datasets\coco128\labels\ee104\_val directory has 2 extra txt files with no equivalent images from the train2017 directory.

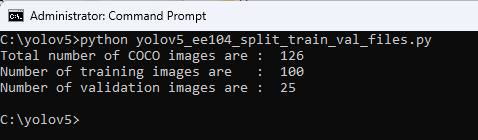
1. Clean those files: You will have to delete the extra files from both directories above. If not, the step below will not run correctly.

# Split the COCO source files into 80% Train & 20% Val directories

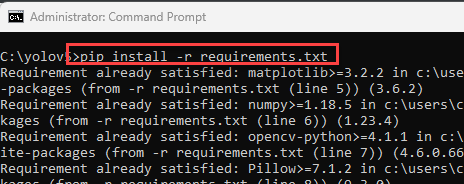
Note: The script below will not run correctly if you did not do the Data Cleaning step above.

1. Now you will run the provided file below to split the COCO source files into 80% for Train and 20% for Validation.

C:\yolov5>python yolov5\_ee104\_split\_train\_val\_files.py



1. Validate to see that the previously empty ee104\_train and ee140\_val directories are filled with images that you just copy over.
2. From C:\yolov5, run again the command below to ensure all requirements are installed:



# Train the Model

1. Add one new class:

Note that the trained model you just created is able to recognized the followings as described in the YAML file

# 80 class names

names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',

'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',

'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',

'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',

'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',

'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',

'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',

'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',

'hair drier', 'toothbrush' ]

**Now you want to add one more class of your choice to the 80 existing classes above.**

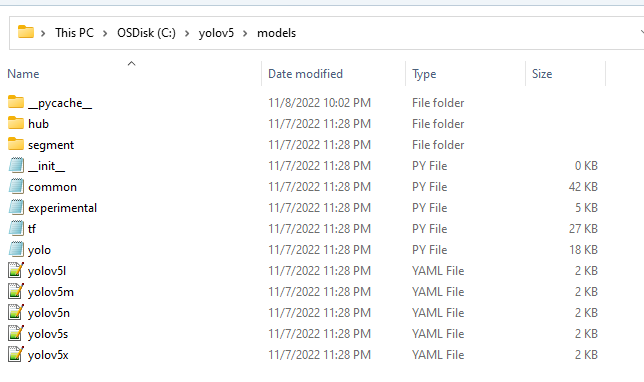
1. Modify the YAML file accordingly to add the new class.
2. Add more pictures and labels as appropriate to the ee104\_train and ee104\_val directories. You can use the labeling tool from the link below

<https://wandb.ai/onlineinference/YOLO/reports/Collect-and-Label-Images-to-Train-a-YOLOv5-Object-Detection-Model-in-PyTorch--VmlldzoxMzQxODc3>

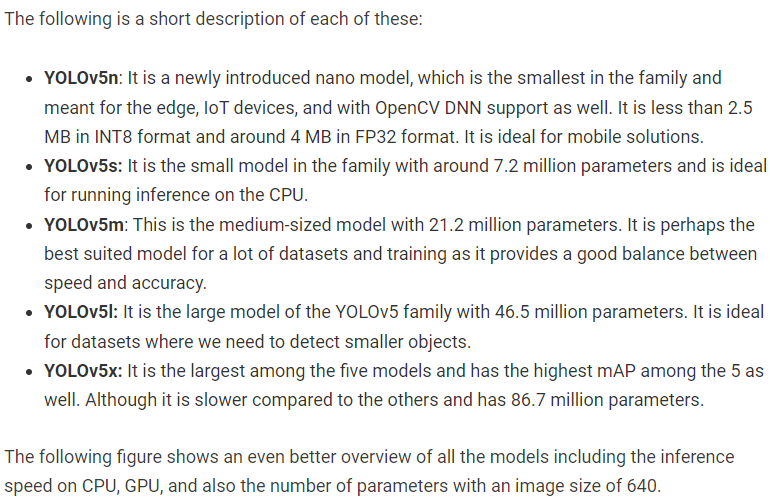
or from Roboflow:[**https://roboflow.com/annotate**](https://roboflow.com/annotate)

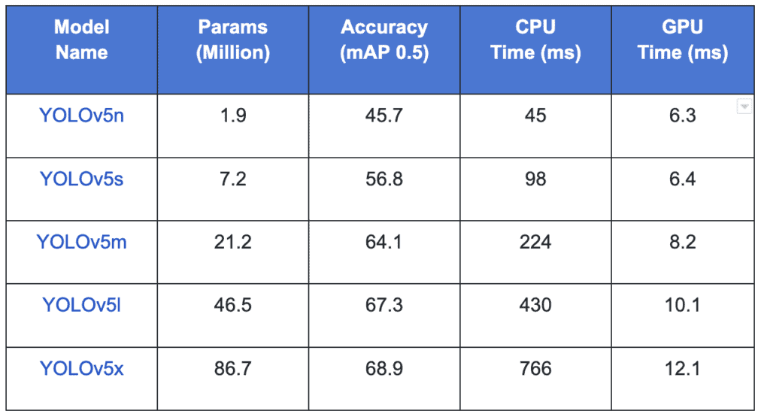
1. Train the model:

Note that the example below shows that we use the yolov5s.pt, but you can use the bigger model for higher accuracy but it will run much longer and requires more CPU & memory resources.



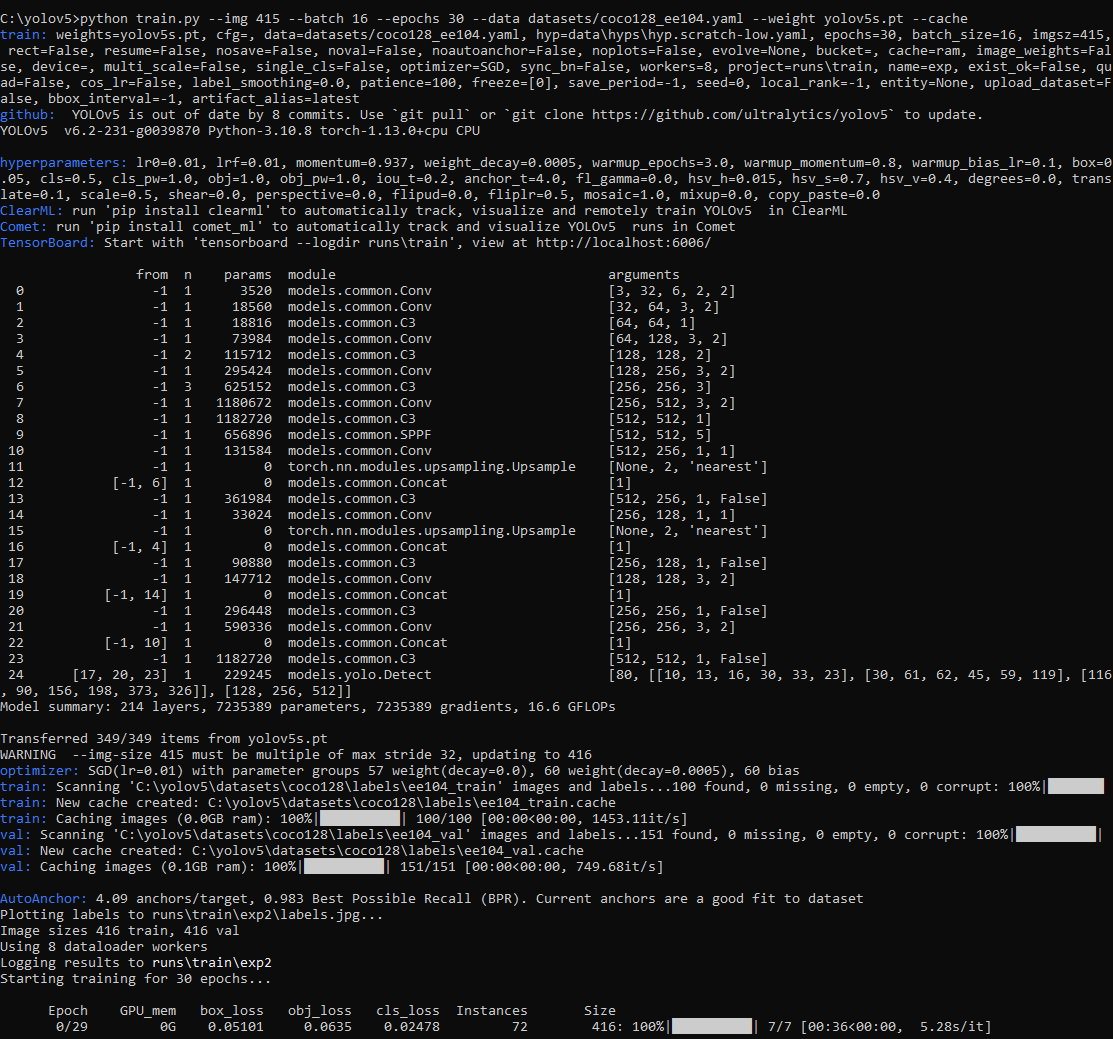
See more here: <https://learnopencv.com/custom-object-detection-training-using-yolov5/>

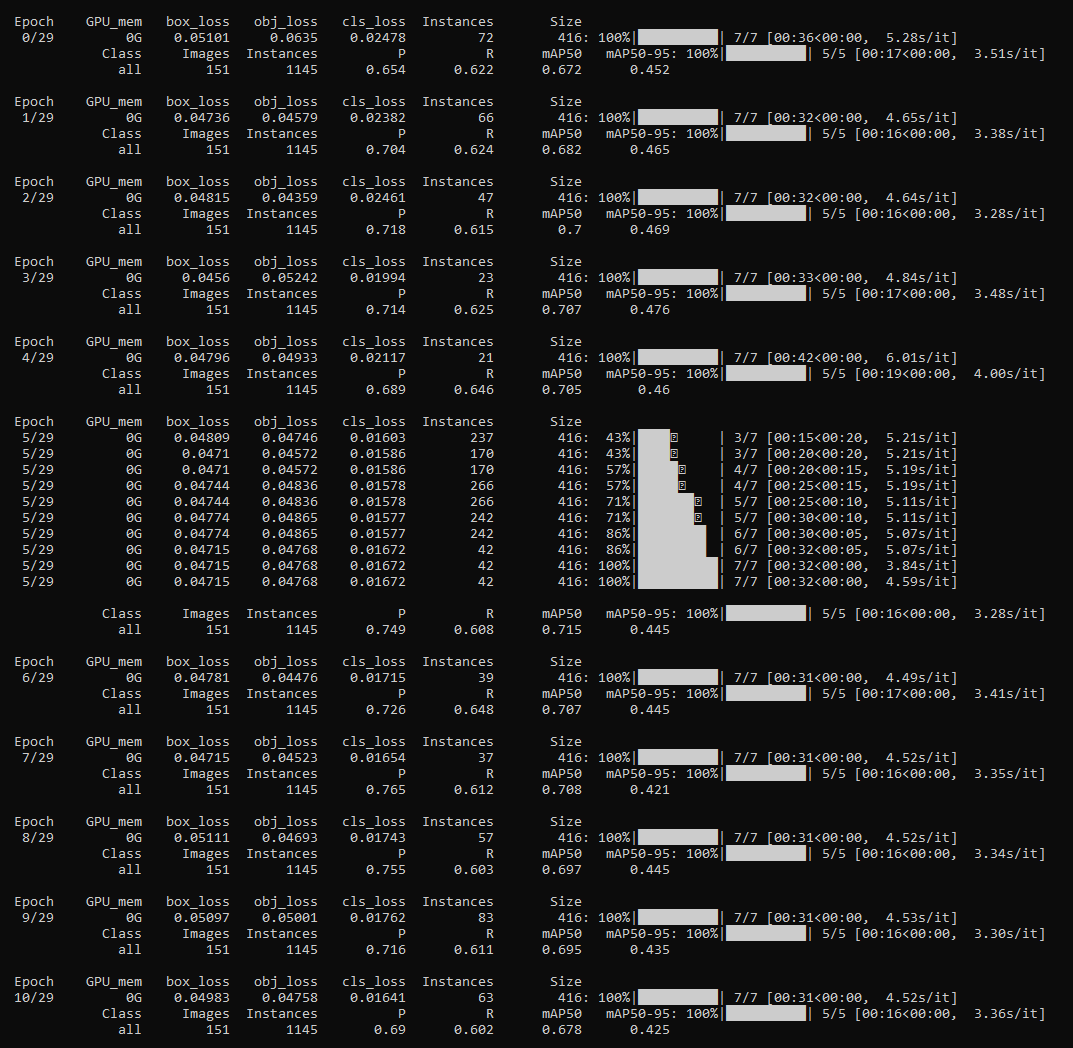




From the command line, run this command:

**python train.py --img 415 --batch 16 --epochs 30 --data datasets/coco128\_ee104.yaml --weight yolov5s.pt --cache**

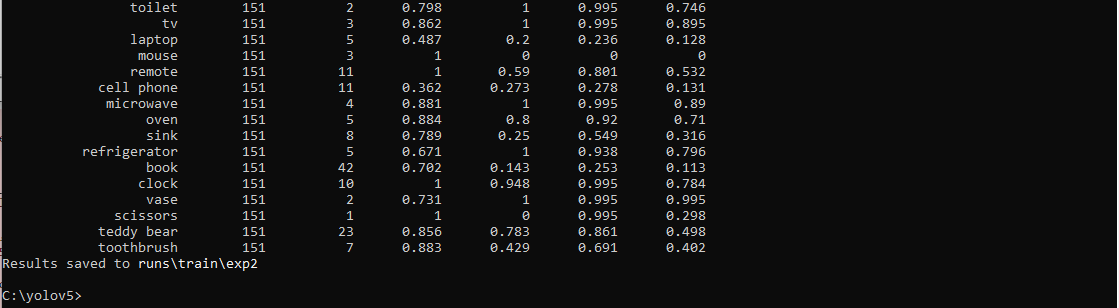




…

# 

# 

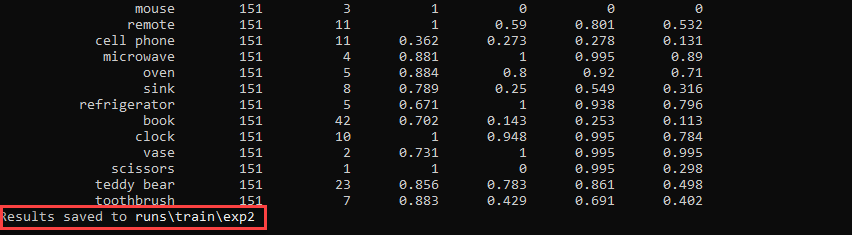


# Your Trained Model

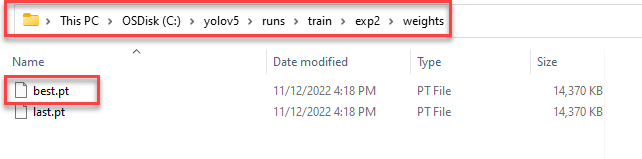
Now you can find your trained models:

If you already run one time, then the next saved model is in the **\exp2** folder.

Every time you run, the results will be saved to the next **\expN+1** folder.

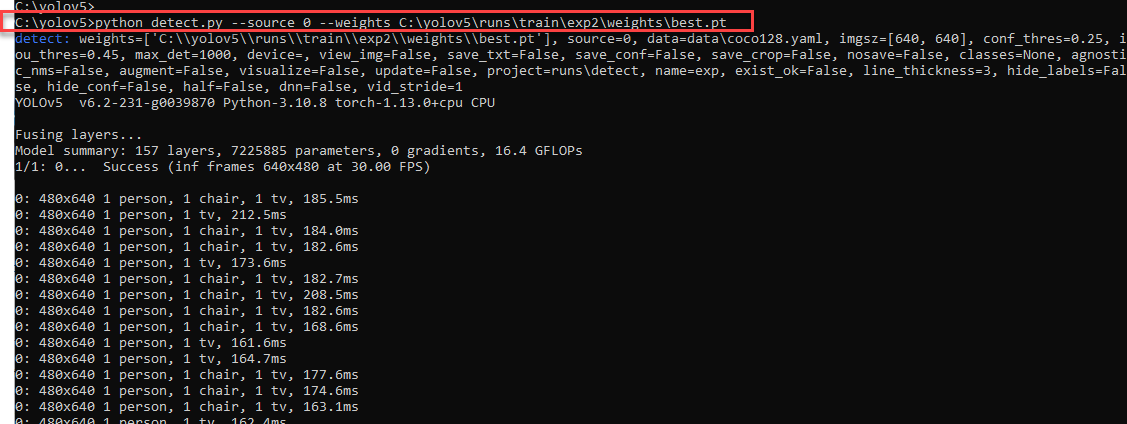


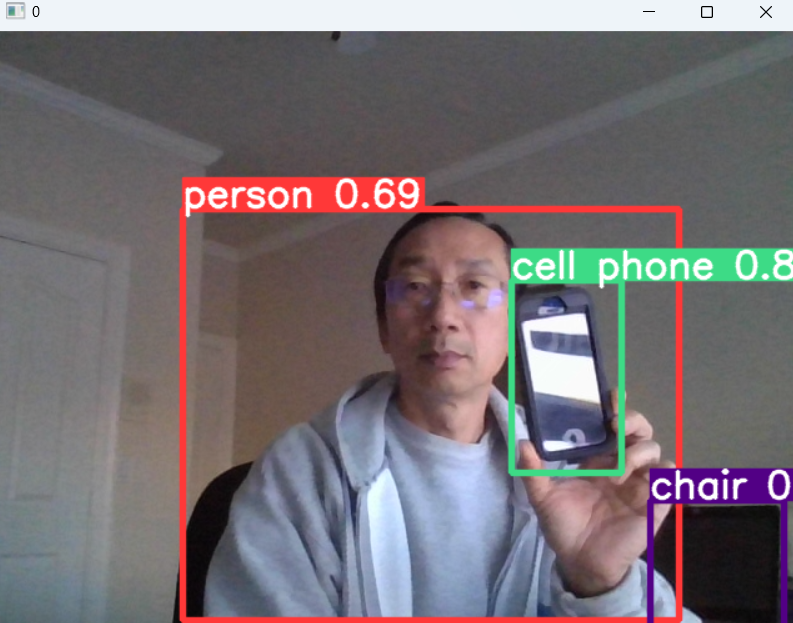
You will use the best.pt model:



This is the validation command you will use to make use of your best.pt model

C:\yolov5>python detect.py --source 0 --weights C:\yolov5\runs\train\exp2\weights\best.pt 跑這行





# Your Deliverables

**Validate that your model detects your added class of object.**

For example, if you added a new class to detect a *whale* if pictures of whales (i.e. *whale1*.jpg, *whale2*.jpg, …., *whaleN*.jpg along with their annotations *whale1*.txt, *whale2*.txt, …, *whaleN*.txt) then now you should be able to recognize yourself holding something looks like a whale and the AI should recognize that whale object you are holding. **Each of the team member should demonstrate you are holding your new class object (i.e. a whale as in this example).**

# Python Programming

#### Lab Submission

|  |  |  |
| --- | --- | --- |
| **Program or Requirement** | **Use Case** | **Earned Score / Max Score** |
| Demonstration Video | You must submit a demonstration video or your score for this lab will be zero |  |
| Document file & check in your README file, documentation, video and codes to Github | README is a brief user guide and developer documentation so that the user can install the proper python packages and knows how to execute your program.  The documentation section can contain sample screenshots with explanation. | \_\_\_\_\_ / 10 |
| Training the new class | **Add and train successfully one more class of your choice to the 80 existing classes. Add a minimum 30 pictures (1 point/ each picture).**  **To increase the accuracy, you can add as many picture as you want.** | \_\_\_\_\_ / 30 |
| Recognize your new class | **Each of the team member should demonstrate two times that you are holding your new class object and your model now recognizes that new object. Use a different samples of the same class for each time (15 points / each time)**  **Points will be granted separately for each team member although the same video is being submitted.** | \_\_\_\_\_ / 30 |
| Game Development – Dance Challenge | Use cases for entertainment and educational applications: Leverage the base code from chapter Dance Challenge and add your own Hacks and Tweaks to add the following features: A Longer Dance, Play Against A Friend, Change the Music (10 points each) | \_\_\_\_\_ / 30 |
|  | **TOTAL** | **100%** |

That’s all for this lab. Hopefully you found it useful and increase your interest in the Python world! See you in the next lab.

# Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. You are required to demonstrate a working design with a narrated video demonstration. You are also required to submit an archive of your project in the form of a ZIP file to submit to Canvas. You also need to check in your code and files to Github.

You will submit your zip file to the instructor through Canvas by the due date and time. If the class will be on campus, then you will expect to demonstrate in the classroom. If we ever have to go back to an online mode, turn in your archive to Canvas along with a narrated video capturing the screen of your computer running your program demonstration. If your program is not completely functional by the due date, you should demonstrate and turn in what you have accomplished to receive partial credit. See the syllabus for the late penalty guideline