ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

XỬ LÝ ẢNH SỐ VÀ VIDEO SỐ

IMAGE AND VIDEO PROCESSING

Lab04: Yolo and Darknet

Giảng viên lý thuyết: Lý Quốc Ngọc

Giảng viên hướng dẫn: Nguyễn Mạnh Hùng

Sinh viên thực hiện:

Họ và tên: Nguyễn Văn Hiếu

MSSV: 20127498

Lớp học phần: 20TGMT1

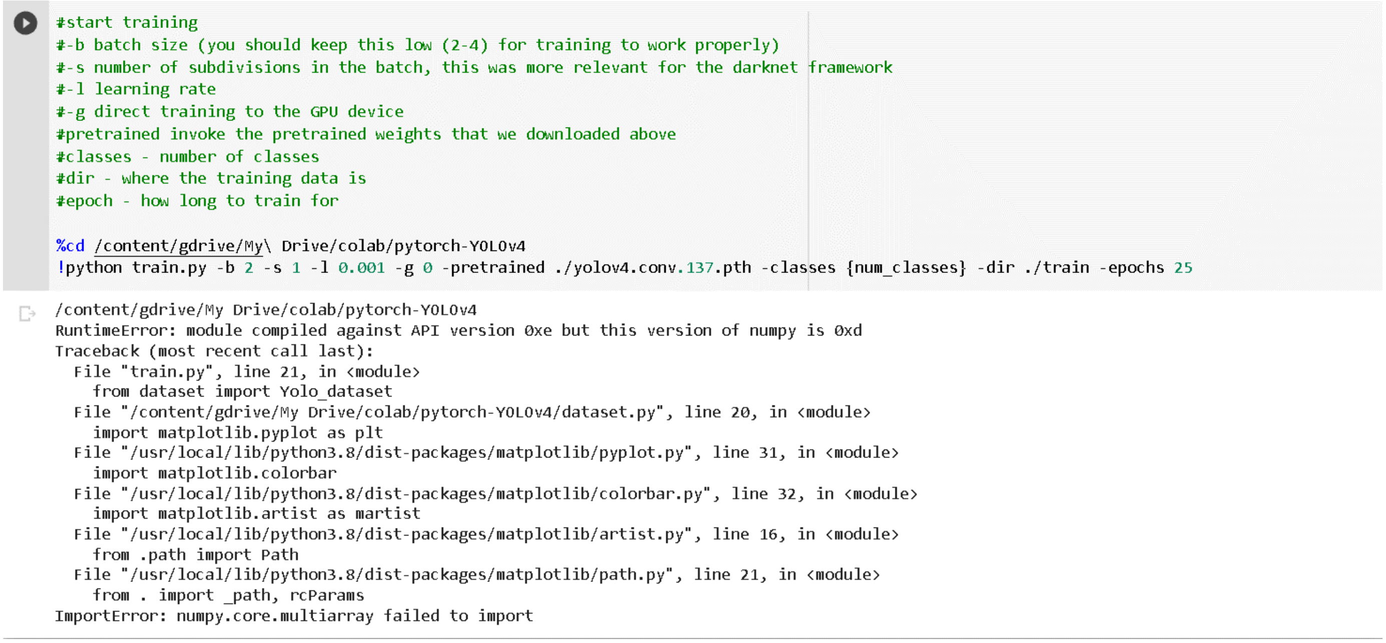
TP.HCM, ngày 7 tháng 12 năm 2022

I. Using YOLOv4 Pytorch Model

1. Old dataset (Chess Pieces Dataset)

- During the running process, I encountered some errors in the Numpy library and files like train.py and models.py

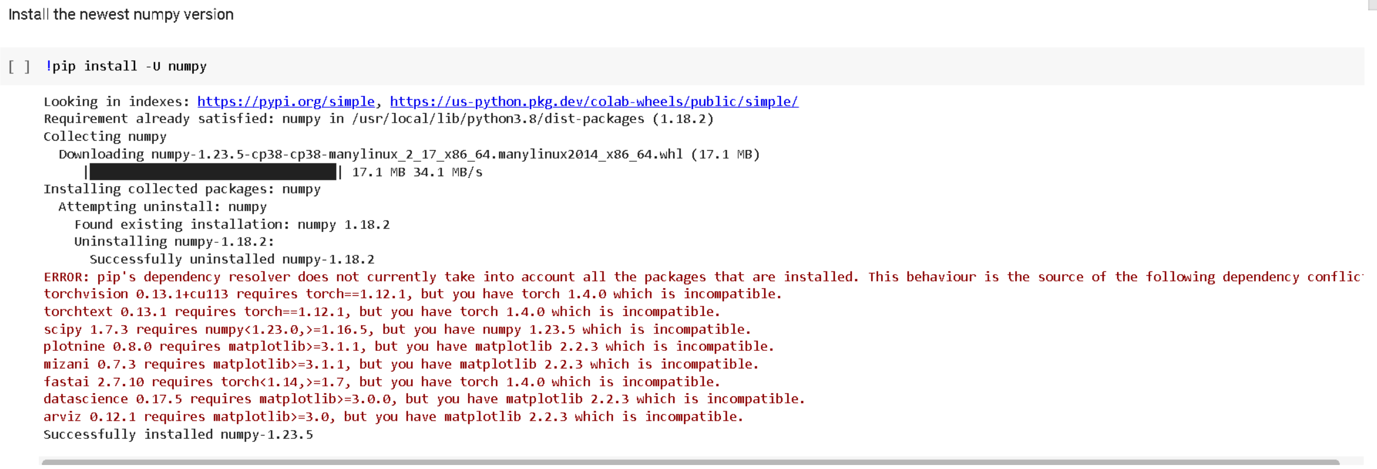
- In step start training, I encountered the following error:



```
#start training
#-b batch size (you should keep this low (2-4) for training to work properly)
#-s number of subdivisions in the batch, this was more relevant for the darknet framework
#-l learning rate
#-g direct training to the GPU device
#pretrained invoke the pretrained weights that we downloaded above
#classes - number of classes
#dir - where the training data is
#epoch - how long to train for

%cd /content/gdrive/My\ Drive/colab/pytorch-YOLOv4
!python train.py -b 2 -s 1 -l 0.001 -g 0 -pretrained ./yolov4.conv.137.pth -classes {num_classes} -dir ./train -epochs 25
```

```
/content/gdrive/My Drive/colab/pytorch-YOLOv4
RuntimeError: module compiled against API version 0xe but this version of numpy is 0xd
Traceback (most recent call last):
  File "train.py", line 21, in <module>
    from dataset import Yolo_dataset
  File "/content/gdrive/My Drive/colab/pytorch-YOLOv4/dataset.py", line 20, in <module>
    import matplotlib.pyplot as plt
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/pyplot.py", line 31, in <module>
    import matplotlib.colorbar
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/colorbar.py", line 32, in <module>
    import matplotlib.artist as martist
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/artist.py", line 16, in <module>
    from .path import Path
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/path.py", line 21, in <module>
    from . import _path, rcParams
ImportError: numpy.core.multiarray failed to import
```

- To solve the Numpy library error, I added the following command line in the source code (remember to restart the runtime):

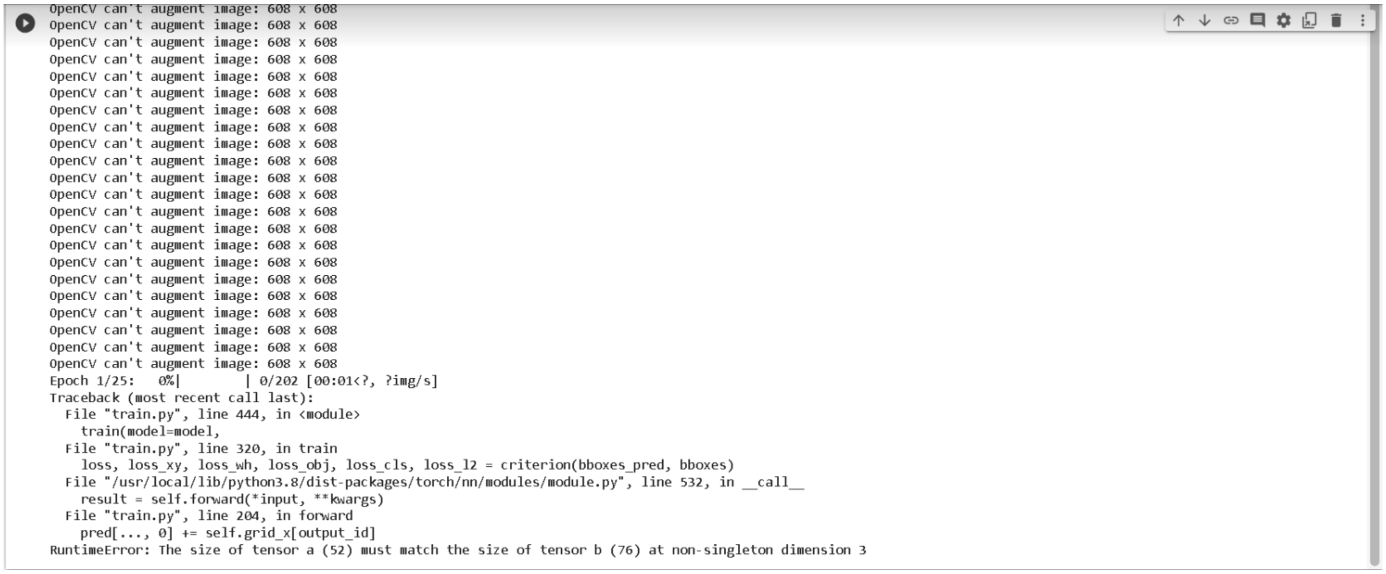


Install the newest numpy version

```
[ ] !pip install -U numpy
```

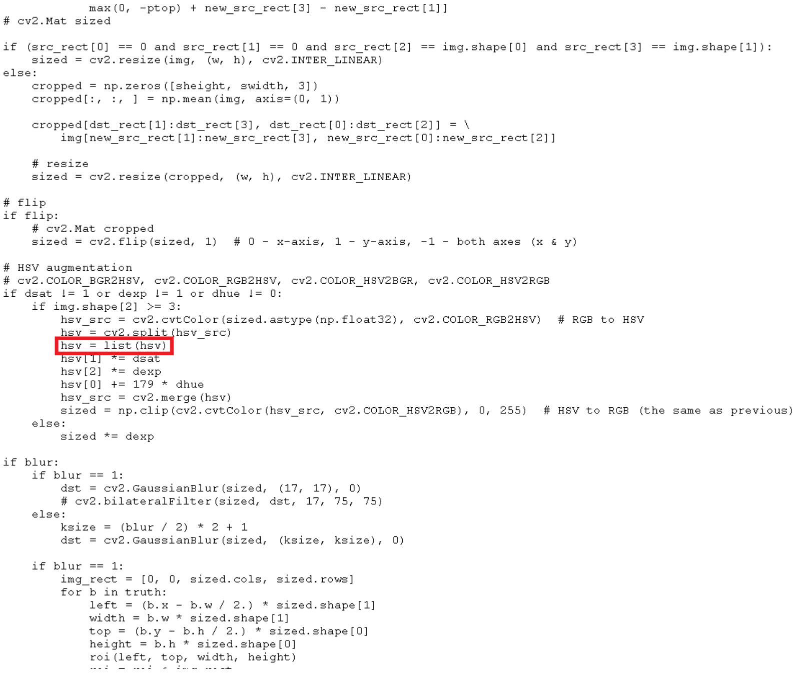
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.18.2)
Collecting numpy
  Downloading numpy-1.23.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
     |████████████████████████████████| 17.1 MB 34.1 MB/s
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.18.2
    Uninstalling numpy-1.18.2:
      Successfully uninstalled numpy-1.18.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflic
torchvision 0.13.1+cu113 requires torch==1.12.1, but you have torch 1.4.0 which is incompatible.
torchtext 0.13.1 requires torch==1.12.1, but you have torch 1.4.0 which is incompatible.
scipy 1.7.3 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.23.5 which is incompatible.
plotnine 0.8.0 requires matplotlib>=3.1.1, but you have matplotlib 2.2.3 which is incompatible.
mizani 0.7.3 requires matplotlib>=3.1.1, but you have matplotlib 2.2.3 which is incompatible.
fastai 2.7.10 requires torch<1.14,>=1.7, but you have torch 1.4.0 which is incompatible.
datascience 0.17.5 requires matplotlib>=3.0.0, but you have matplotlib 2.2.3 which is incompatible.
arviz 0.12.1 requires matplotlib>=3.0, but you have matplotlib 2.2.3 which is incompatible.
Successfully installed numpy-1.23.5
```

- Then, I encountered the following error:



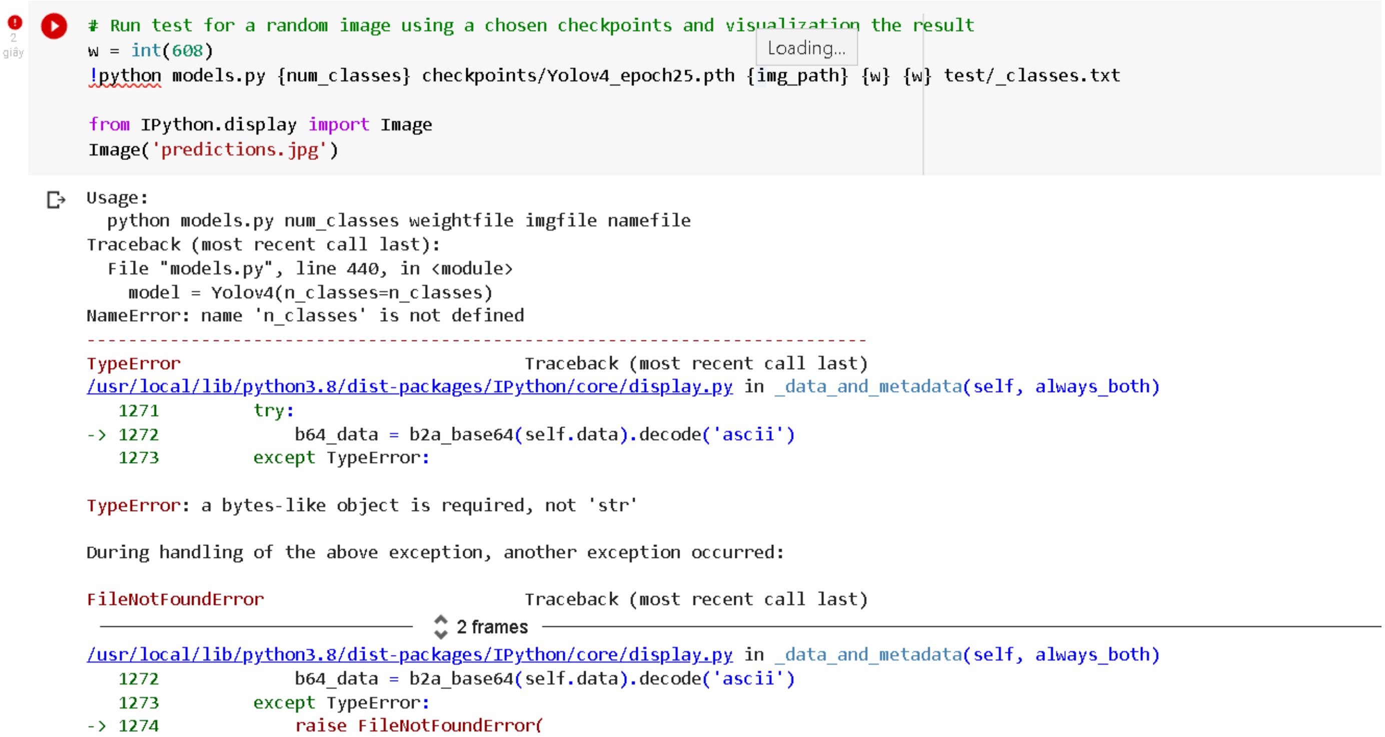
```
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
OpenCV can't augment image: 608 x 608
Epoch 1/25:   0%|          | 0/202 [00:01<?, ?img/s]
Traceback (most recent call last):
  File "train.py", line 444, in <module>
    train(model=model,
  File "train.py", line 320, in train
    loss, loss_xy, loss_wh, loss_obj, loss_cls, loss_l2 = criterion(bboxes_pred, bboxes)
  File "/usr/local/lib/python3.8/dist-packages/torch/nn/modules/module.py", line 532, in __call__
    result = self.forward(*input, **kwargs)
  File "train.py", line 204, in forward
    pred[..., 0] += self.grid_x[output_id]
RuntimeError: The size of tensor a (52) must match the size of tensor b (76) at non-singleton dimension 3
```

- To fix it, I adjusted the file dataset.py in location: My Drive/colab/pytorch-YOLOv4 by adding the line: hsv = list(hsv) (The corrected dataset.py file I will leave in the source code)



```
            max(0, -ptop) + new_src_rect[3] - new_src_rect[1]]
# cv2.Mat sized

if (src_rect[0] == 0 and src_rect[1] == 0 and src_rect[2] == img.shape[0] and src_rect[3] == img.shape[1]):
    sized = cv2.resize(img, (w, h), cv2.INTER_LINEAR)
else:
    cropped = np.zeros([sheight, swidth, 3])
    cropped[:, :, ] = np.mean(img, axis=(0, 1))

    cropped[dst_rect[1]:dst_rect[3], dst_rect[0]:dst_rect[2]] = \
        img[new_src_rect[1]:new_src_rect[3], new_src_rect[0]:new_src_rect[2]]

    # resize
    sized = cv2.resize(cropped, (w, h), cv2.INTER_LINEAR)

# flip
if flip:
    # cv2.Mat cropped
    sized = cv2.flip(sized, 1)  # 0 - x-axis, 1 - y-axis, -1 - both axes (x & y)

# HSV augmentation
# cv2.COLOR_BGR2HSV, cv2.COLOR_RGB2HSV, cv2.COLOR_HSV2BGR, cv2.COLOR_HSV2RGB
if dsat != 1 or dexp != 1 or dhue != 0:
    if img.shape[2] >= 3:
        hsv_src = cv2.cvtColor(sized.astype(np.float32), cv2.COLOR_RGB2HSV)  # RGB to HSV
        hsv = cv2.split(hsv_src)
        hsv = list(hsv)
        hsv[1] *= dsat
        hsv[2] *= dexp
        hsv[0] += 179 * dhue
        hsv_src = cv2.merge(hsv)
        sized = np.clip(cv2.cvtColor(hsv_src, cv2.COLOR_HSV2RGB), 0, 255)  # HSV to RGB (the same as previous)
    else:
        sized *= dexp

if blur:
    if blur == 1:
        dst = cv2.GaussianBlur(sized, (17, 17), 0)
        # cv2.bilateralFilter(sized, dst, 17, 75, 75)
    else:
        ksize = (blur / 2) * 2 + 1
        dst = cv2.GaussianBlur(sized, (ksize, ksize), 0)

    if blur == 1:
        img_rect = [0, 0, sized.cols, sized.rows]
        for b in truth:
            left = (b.x - b.w / 2.) * sized.shape[1]
            width = b.w * sized.shape[1]
            top = (b.y - b.h / 2.) * sized.shape[0]
            height = b.h * sized.shape[0]
            roi(left, top, width, height)
```

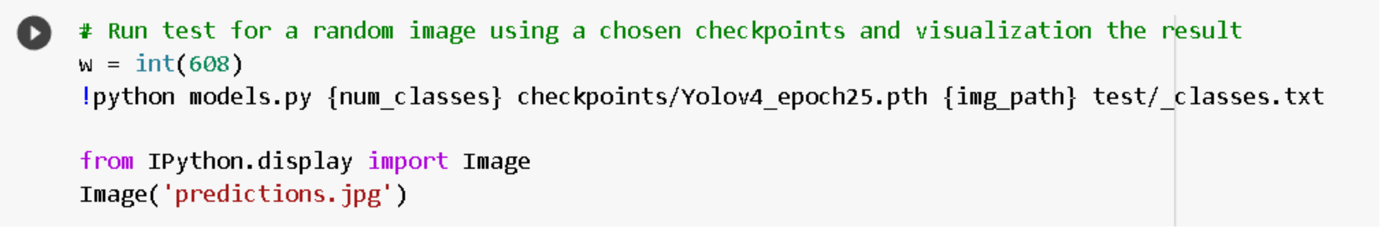
- In step run test, I encountered the following error:



```
# Run test for a random image using a chosen checkpoints and visualization the result
w = int(608)
!python models.py {num_classes} checkpoints/Yolov4_epoch25.pth {img_path} {w} {w} test/_classes.txt

from IPython.display import Image
Image('predictions.jpg')
```

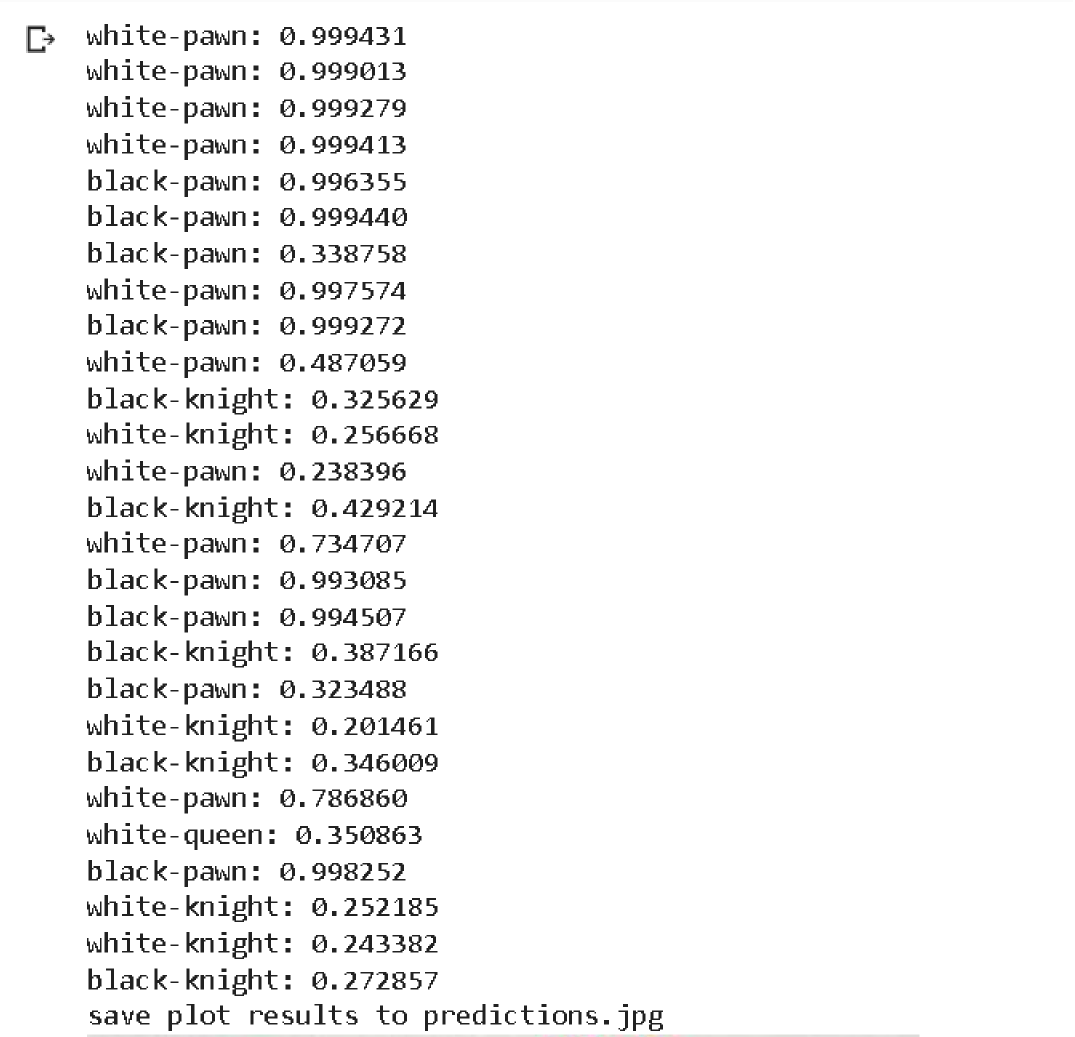
```
Usage:
  python models.py num_classes weightfile imgfile namefile
Traceback (most recent call last):
  File "models.py", line 440, in <module>
    model = Yolov4(n_classes=n_classes)
NameError: name 'n_classes' is not defined
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/IPython/core/display.py in _data_and_metadata(self, always_both)
   1271         try:
-> 1272             b64_data = b2a_base64(self.data).decode('ascii')
   1273         except TypeError:

TypeError: a bytes-like object is required, not 'str'

During handling of the above exception, another exception occurred:

FileNotFoundError                         Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/IPython/core/display.py in _data_and_metadata(self, always_both)
   1272             b64_data = b2a_base64(self.data).decode('ascii')
   1273         except TypeError:
-> 1274             raise FileNotFoundError(
```

- This error occurs because the command line parameters do not coincide with the models.py file, so I corrected as follows:

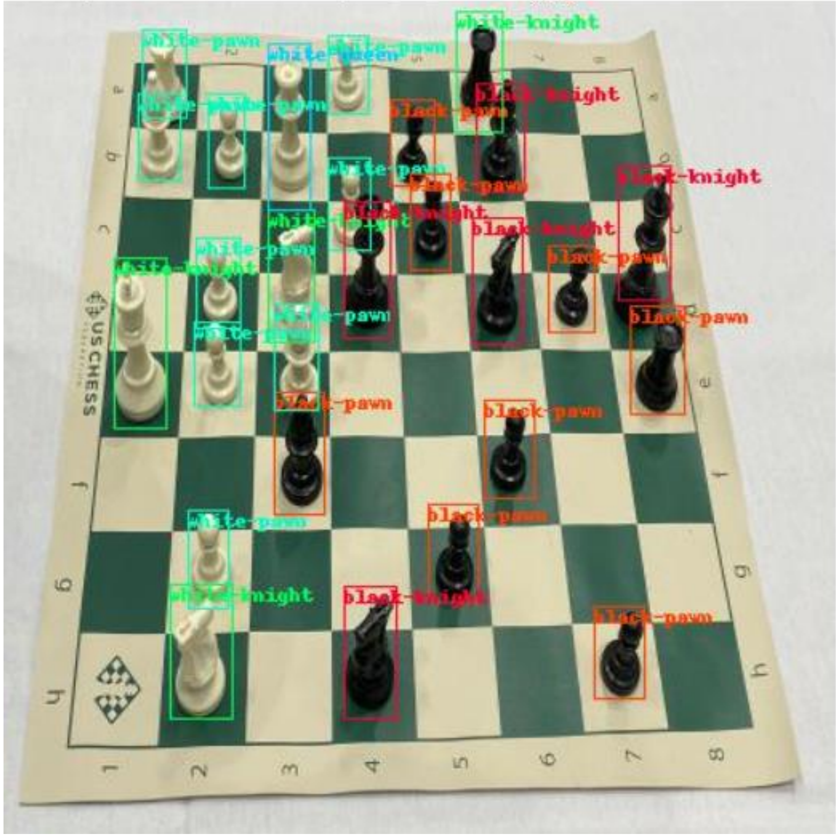


```
# Run test for a random image using a chosen checkpoints and visualization the result
w = int(608)
!python models.py {num_classes} checkpoints/Yolov4_epoch25.pth {img_path} test/_classes.txt

from IPython.display import Image
Image('predictions.jpg')
```

- Running step by step, according to the available source code, we will get the final result as follows:

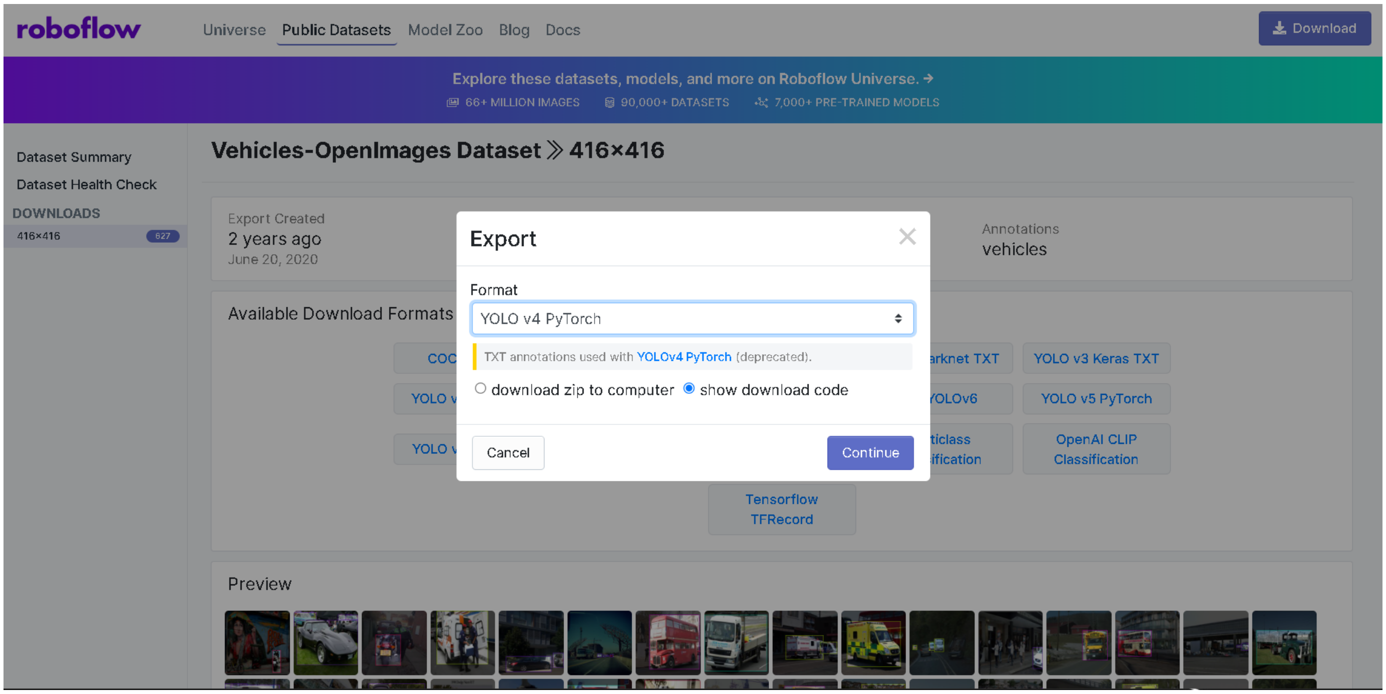


```
white-pawn: 0.999431
white-pawn: 0.999013
white-pawn: 0.999279
white-pawn: 0.999413
black-pawn: 0.996355
black-pawn: 0.999440
black-pawn: 0.338758
white-pawn: 0.997574
black-pawn: 0.999272
white-pawn: 0.487059
black-knight: 0.325629
white-knight: 0.256668
white-pawn: 0.238396
black-knight: 0.429214
white-pawn: 0.734707
black-pawn: 0.993085
black-pawn: 0.994507
black-knight: 0.387166
black-pawn: 0.323488
white-knight: 0.201461
black-knight: 0.346009
white-pawn: 0.786860
white-queen: 0.350863
black-pawn: 0.998252
white-knight: 0.252185
white-knight: 0.243382
black-knight: 0.272857
save plot results to predictions.jpg
```



2. Using new dataset (Vehicles Dataset)

- Vehicles Dataset is a set of vehicles that contain 627 photos divided into3 folder train, validation and testing. In each folder also contains _annotations.txt file (labeling and class number) and _classes.txt (classes) respectively. Each image has dimensions 416 x 416.
- Dataset link: https://public.roboflow.com/object-detection/vehicles-openimages
- Select the format for YOLO V4 Pytorch:



Vehicles-OpenImages Dataset » 416×416

Export Created
2 years ago
June 20, 2020

Annotations
vehicles

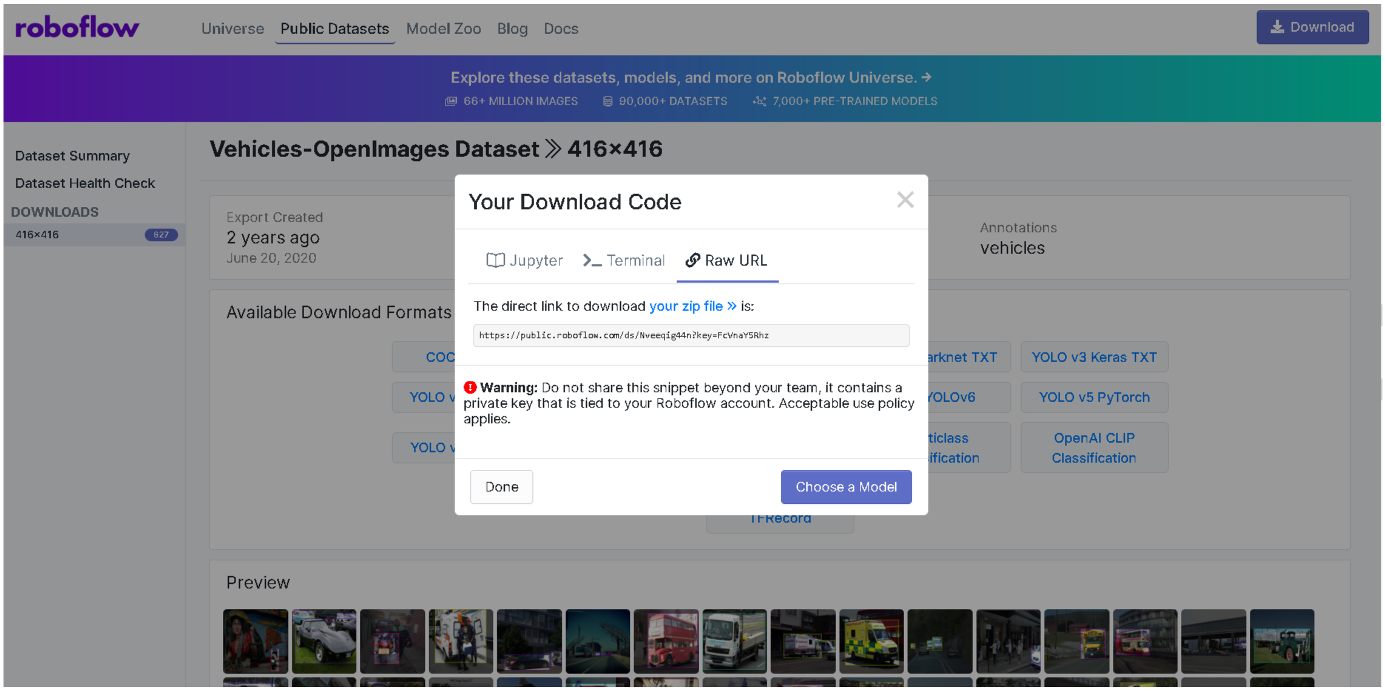
Export

Format
YOLO v4 PyTorch

TXT annotations used with YOLOv4 PyTorch (deprecated).

○ download zip to computer ● show download code

- Click “show download code” and Continue to get the download link:



Vehicles-OpenImages Dataset » 416×416

Export Created
2 years ago
June 20, 2020

Annotations
vehicles

Available Download Formats

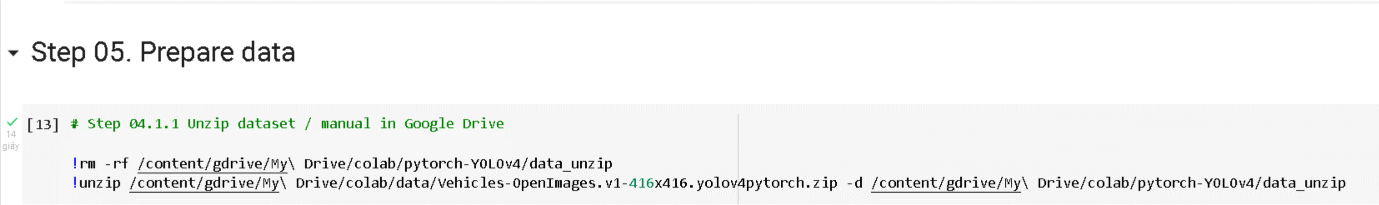
Your Download Code

The direct link to download your zip file » is:

```
https://public.roboflow.com/ds/Nveeqig44n?key=FcVnaY5Rhz
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

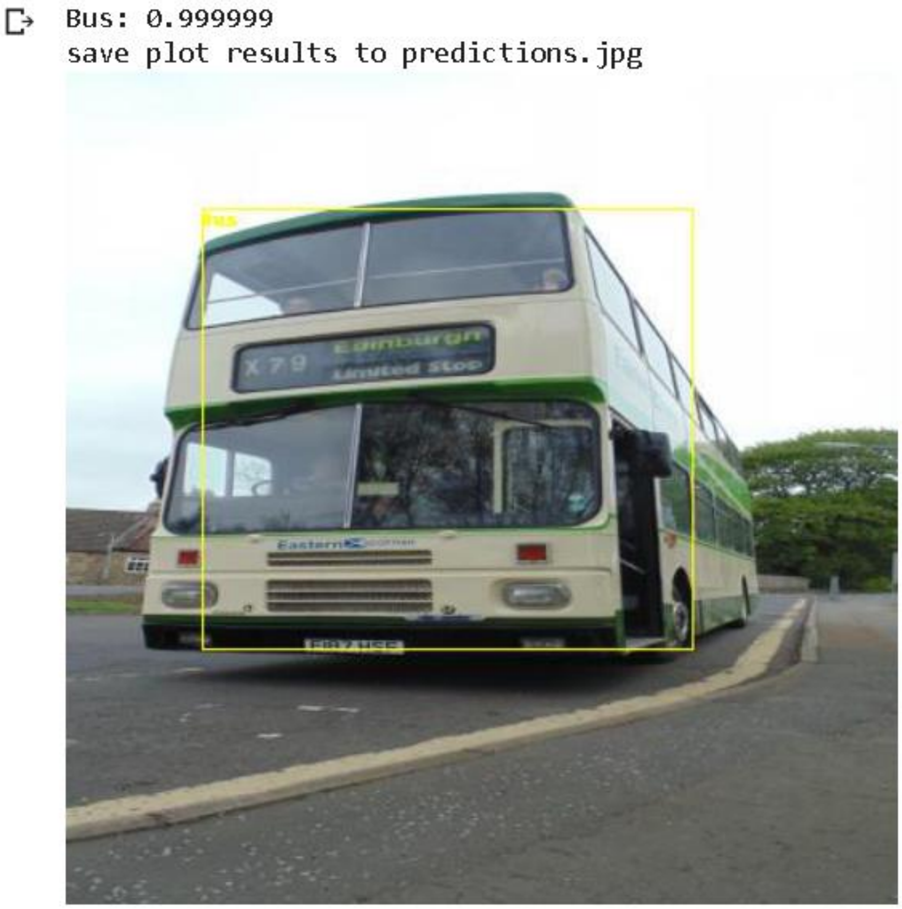
- Use Google Drive to store data as well as train results.\
- The next steps do the same with the old dataset
- Particularly, the data preparation step we will often change the zip file name of the new data instead of the old data:



Step 05. Prepare data

```
[13] # Step 04.1.1 Unzip dataset / manual in Google Drive

     !rm -rf /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip
     !unzip /content/gdrive/My\ Drive/colab/data/Vehicles-OpenImages.v1-416x416.yolov4pytorch.zip -d /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip
```

- In step run test, we will encounter similar errors with the old dataset and we will correct the same error.
- Finally, we will get the following result:

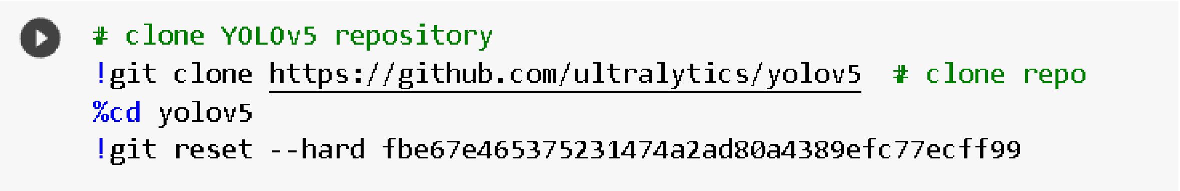


```
Bus: 0.999999
save plot results to predictions.jpg
```

II. Using YOLOv5 Pytorch Model

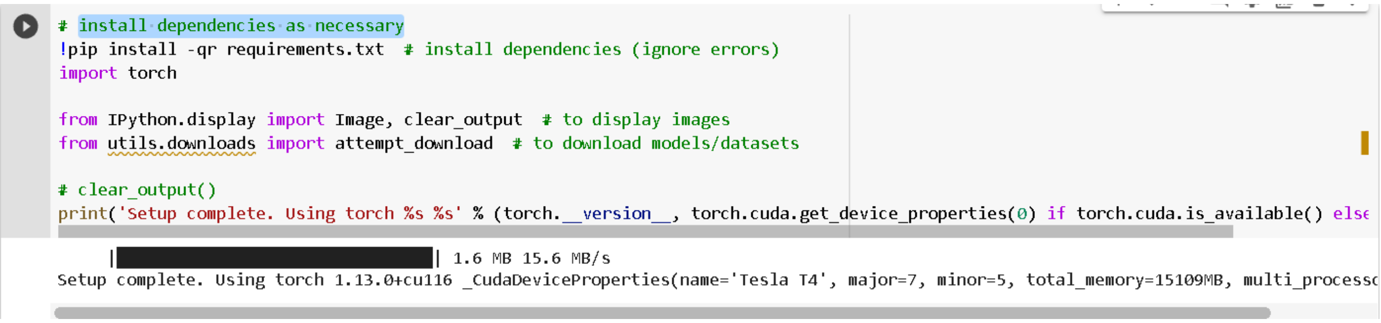
1. Using old dataset (Chess Pieces Dataset)

- To take the model, you can visit: https://roboflow.com/model/yolov5
- Clone YOLOv5 repo:



```
# clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
!git reset --hard fbe67e465375231474a2ad80a4389efc77ecff99
```

- Install dependencies as necessary:

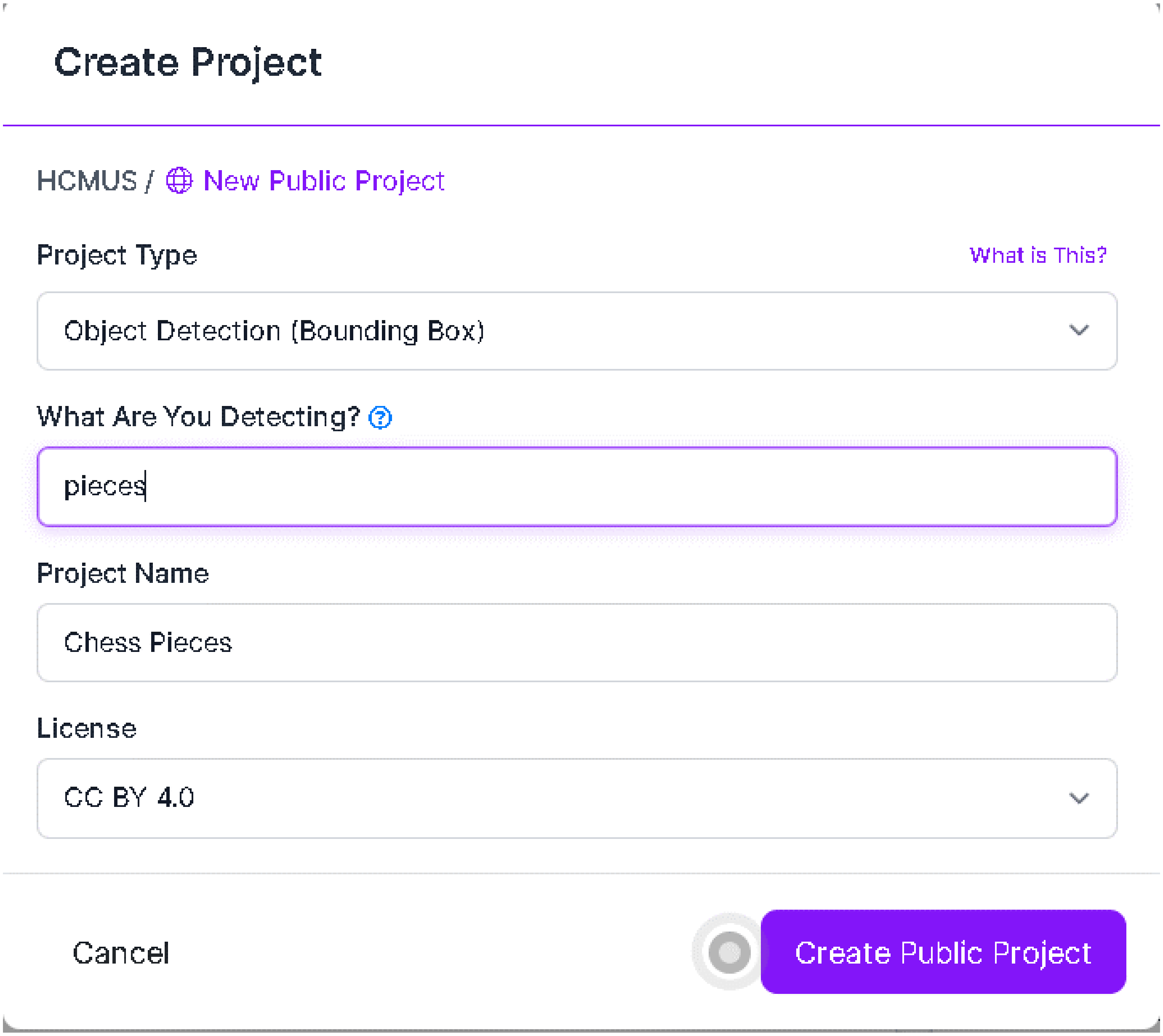


```
# install dependencies as necessary
!pip install -qr requirements.txt  # install dependencies (ignore errors)
import torch

from IPython.display import Image, clear_output  # to display images
from utils.downloads import attempt_download  # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else
```

```
     |████████████████████████████████| 1.6 MB 15.6 MB/s
Setup complete. Using torch 1.13.0+cu116 _CudaDeviceProperties(name='Tesla T4', major=7, minor=5, total_memory=15109MB, multi_processo
```

- Preparing dataset: Following the link to get the download code from roboflow: https://app.roboflow.com/?model=yolov5&ref=roboflow-yolov5

- Then you need to create a project for your dataset:



Create Project

HCMUS / New Public Project

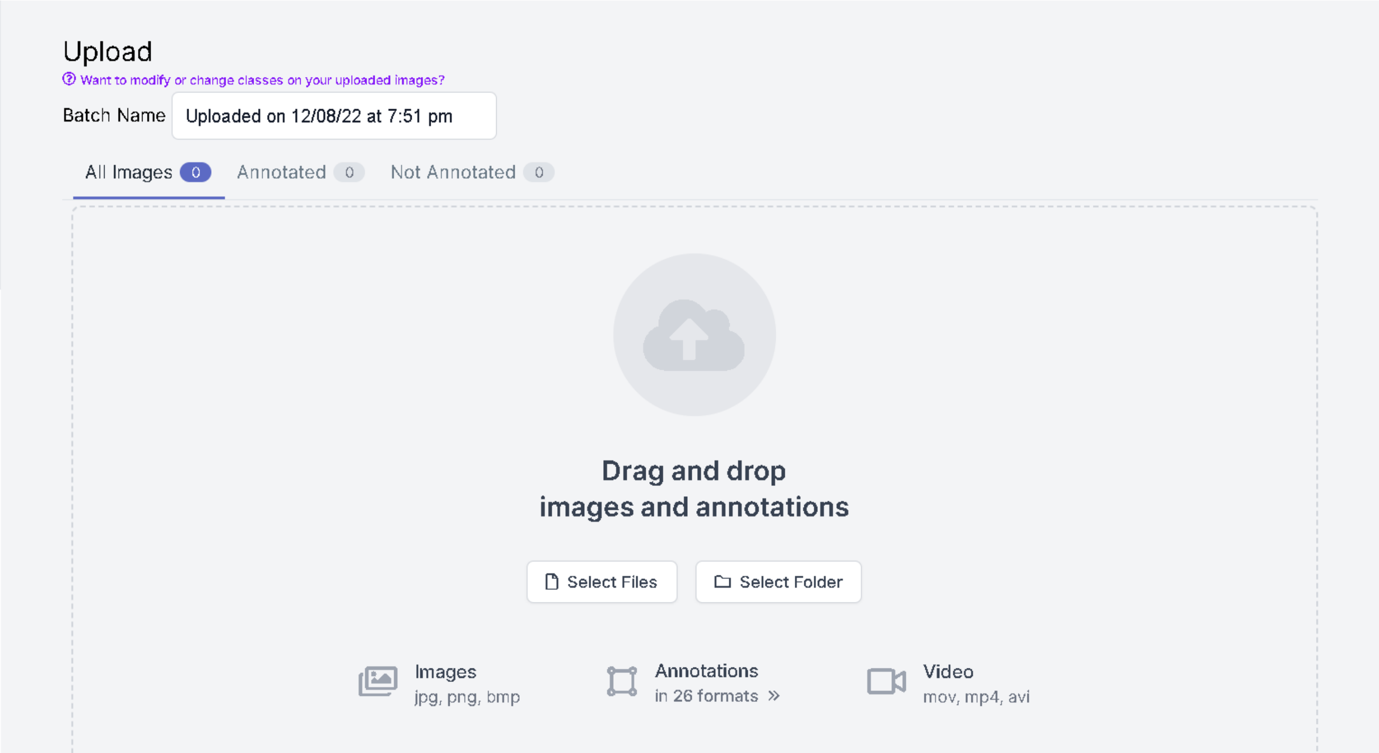
Project Type: Object Detection (Bounding Box)

What Are You Detecting?: pieces

Project Name: Chess Pieces

License: CC BY 4.0

- Next, drag and drop the image and annotation from your dataset:

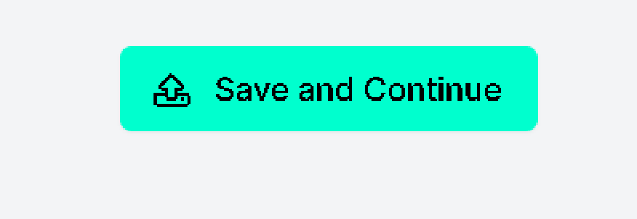


Upload

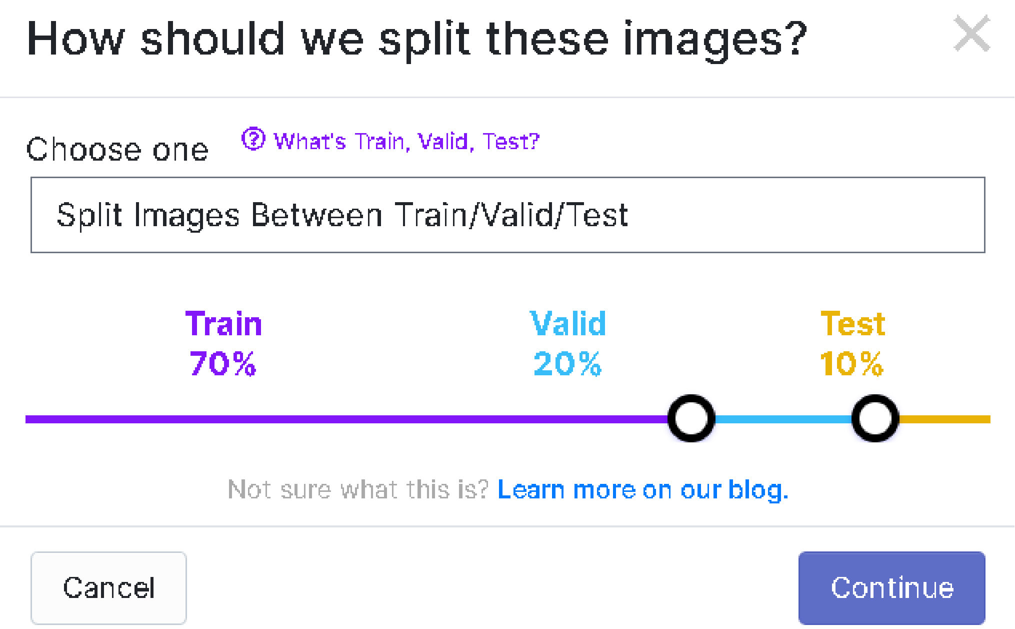
Want to modify or change classes on your uploaded images?

Batch Name: Uploaded on 12/08/22 at 7:51 pm

- After uploading, click Save and Continue:



- Click Continue to split image:



How should we split these images?

Choose one: Split Images Between Train/Valid/Test

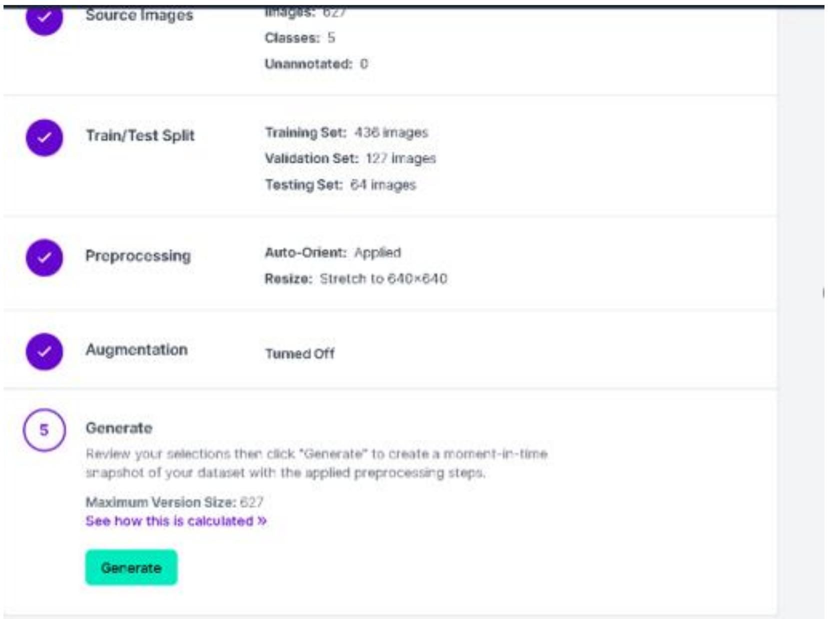
Train 70%

Valid 20%

Test 10%

What's Train, Valid, Test?
Not sure what this is? Learn more on our blog.
Cancel
Continue

- Adjust the necessary parameters and click Generate:



✓	Source Images	Images: 627 Classes: 5 Unannotated: 0
✓	Train/Test Split	Training Set: 436 images Validation Set: 127 images Testing Set: 64 images
✓	Preprocessing	Auto-Orient: Applied Resize: Stretch to 640×640
✓	Augmentation	Turned Off

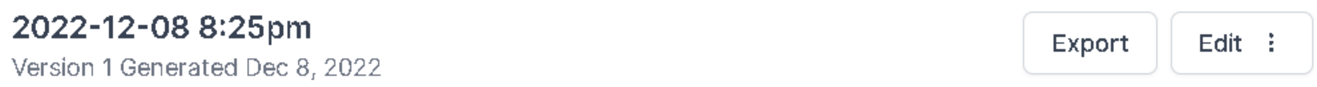
5 Generate

Review your selections then click "Generate" to create a moment-in-time snapshot of your dataset with the applied preprocessing steps.

Maximum Version Size: 627

See how this is calculated »

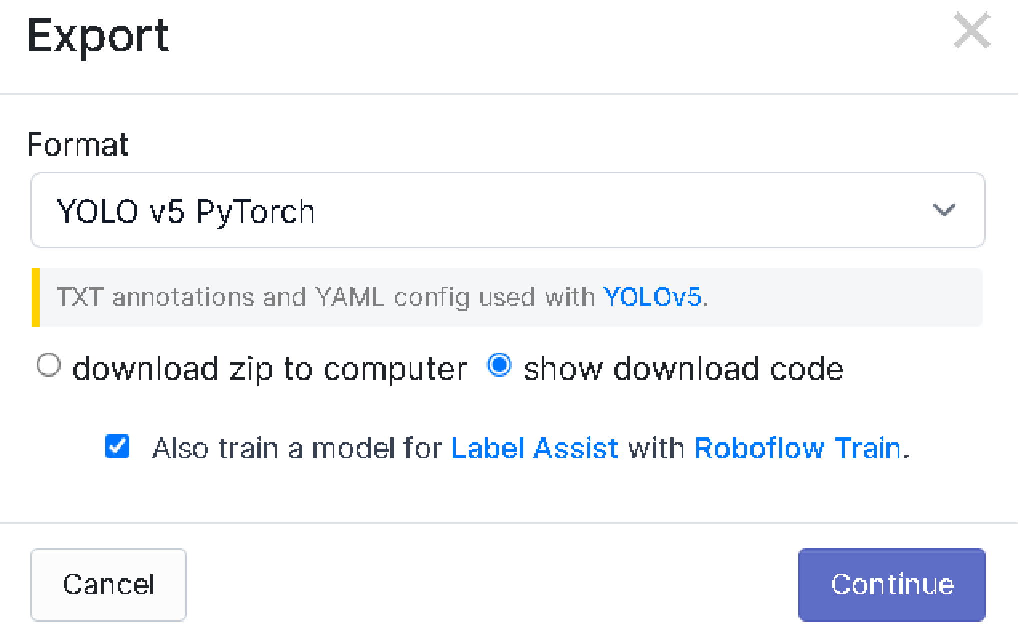
- Finally, click Export :



2022-12-08 8:25pm

Version 1 Generated Dec 8, 2022

- Choose the YOLO v5 Pytorch format and click Continue:



Export

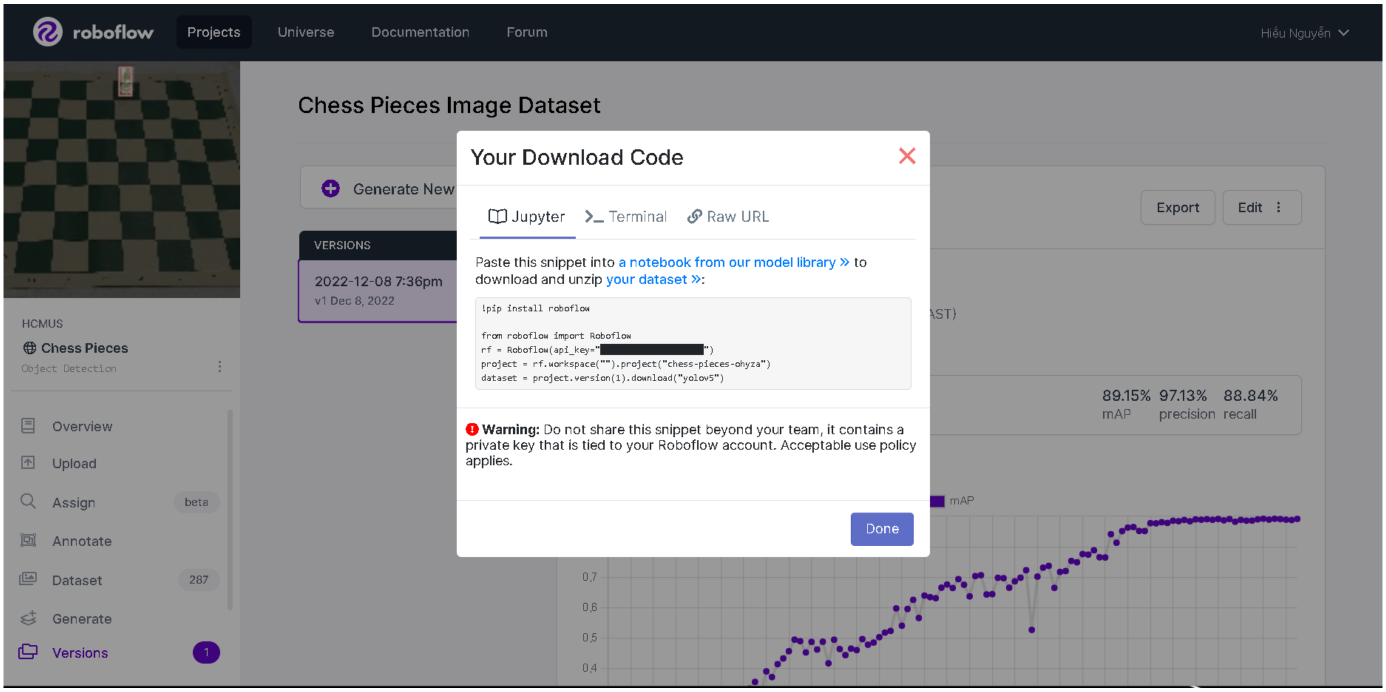
Format: YOLO v5 PyTorch

TXT annotations and YAML config used with YOLOv5.

○ download zip to computer ● show download code

☑ Also train a model for Label Assist with Roboflow Train.

- You will create a dataset and get the download code below:



Your Download Code

Jupyter | Terminal | Raw URL

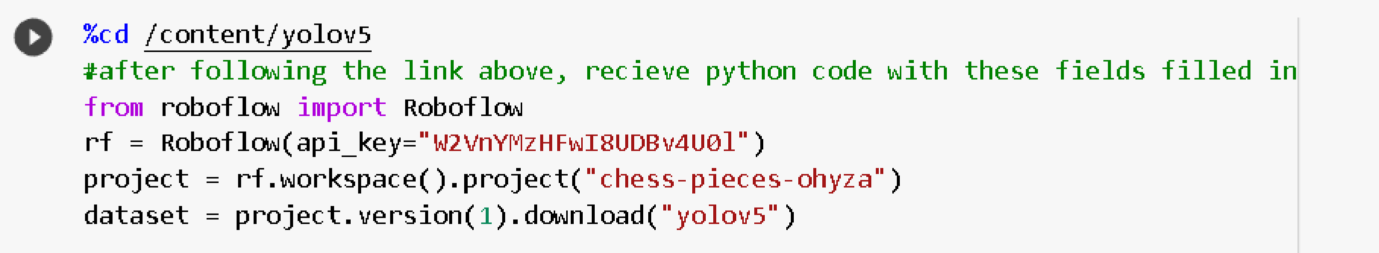
Paste this snippet into a notebook from our model library » to download and unzip your dataset »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("").project("chess-pieces-ohyza")
dataset = project.version(1).download("yolov5")
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

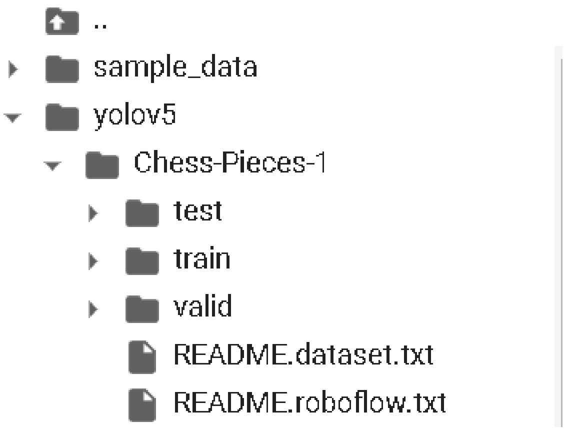
Done

- Copy and paste it in next cell like this:



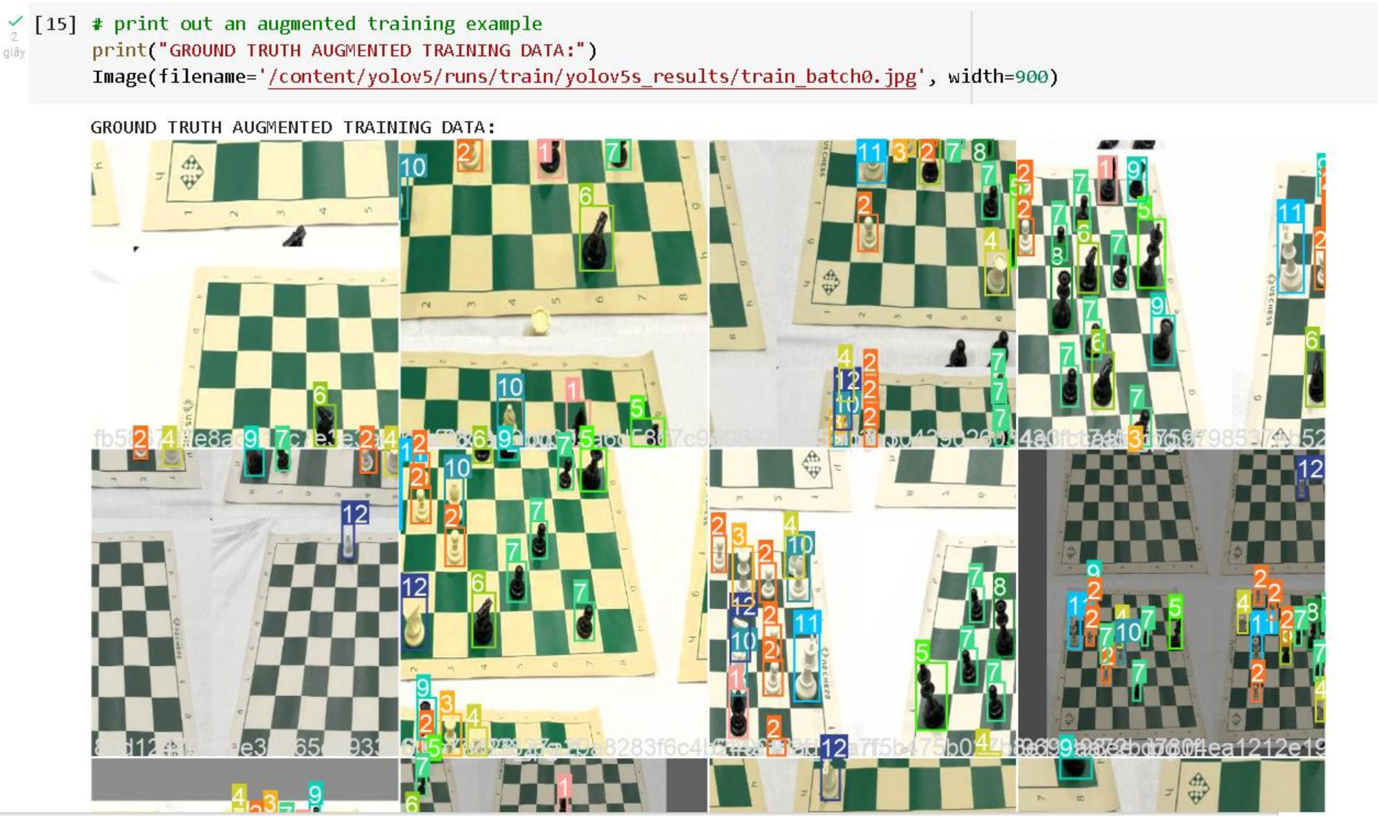
```
%cd /content/yolov5
#after following the link above, recieve python code with these fields filled in
from roboflow import Roboflow
rf = Roboflow(api_key="W2VnYMzHFwI8UDBv4U0l")
project = rf.workspace().project("chess-pieces-ohyza")
dataset = project.version(1).download("yolov5")
```

- Run it, we will get the dataset in my training:



```
..
sample_data
yolov5
  Chess-Pieces-1
    test
    train
    valid
    README.dataset.txt
    README.roboflow.txt
```

- Running step by step we will get the following result:

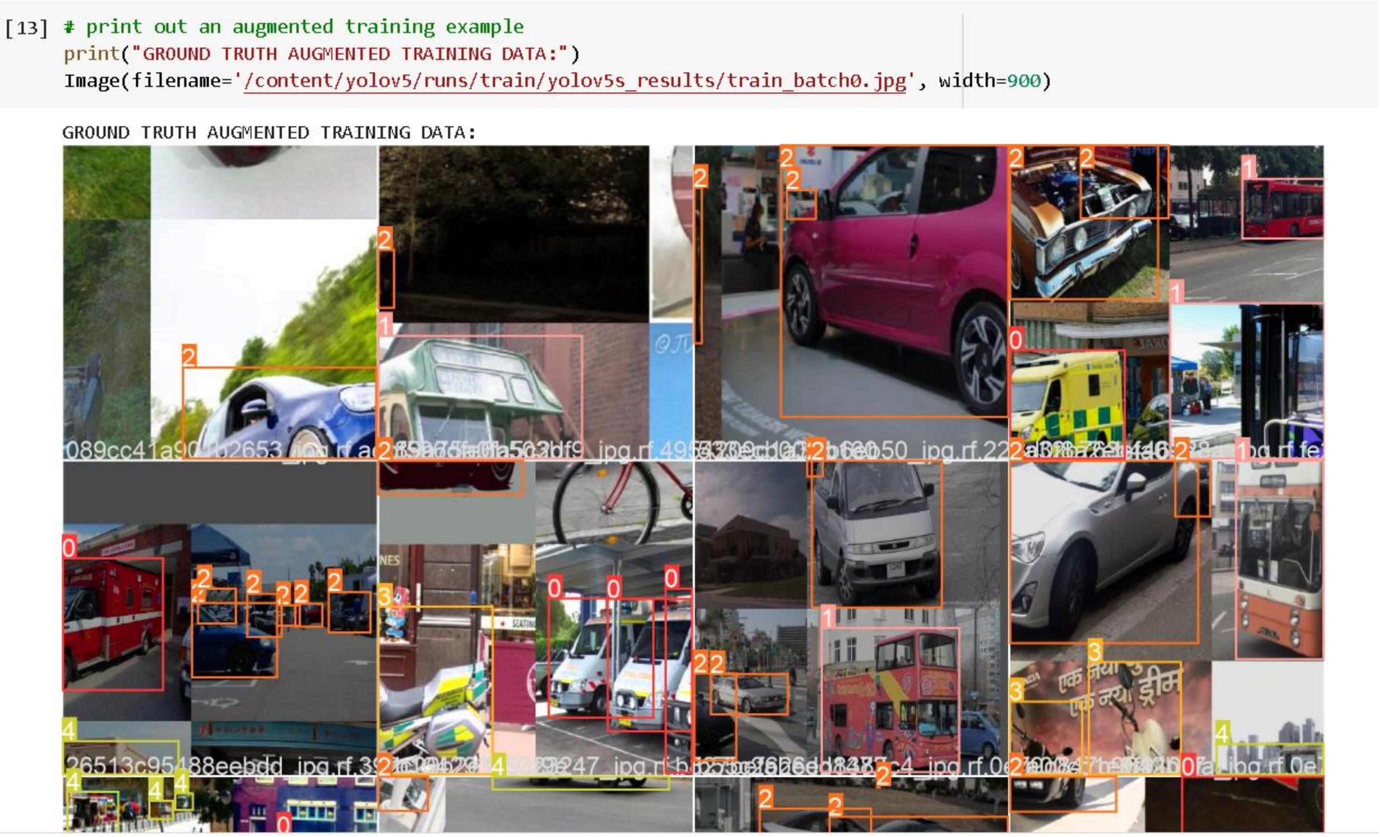


```
[15] # print out an augmented training example
     print("GROUND TRUTH AUGMENTED TRAINING DATA:")
     Image(filename='/content/yolov5/runs/train/yolov5s_results/train_batch0.jpg', width=900)
```

```
GROUND TRUTH AUGMENTED TRAINING DATA:
```

2. Using new dataset (Vehicles Dataset)

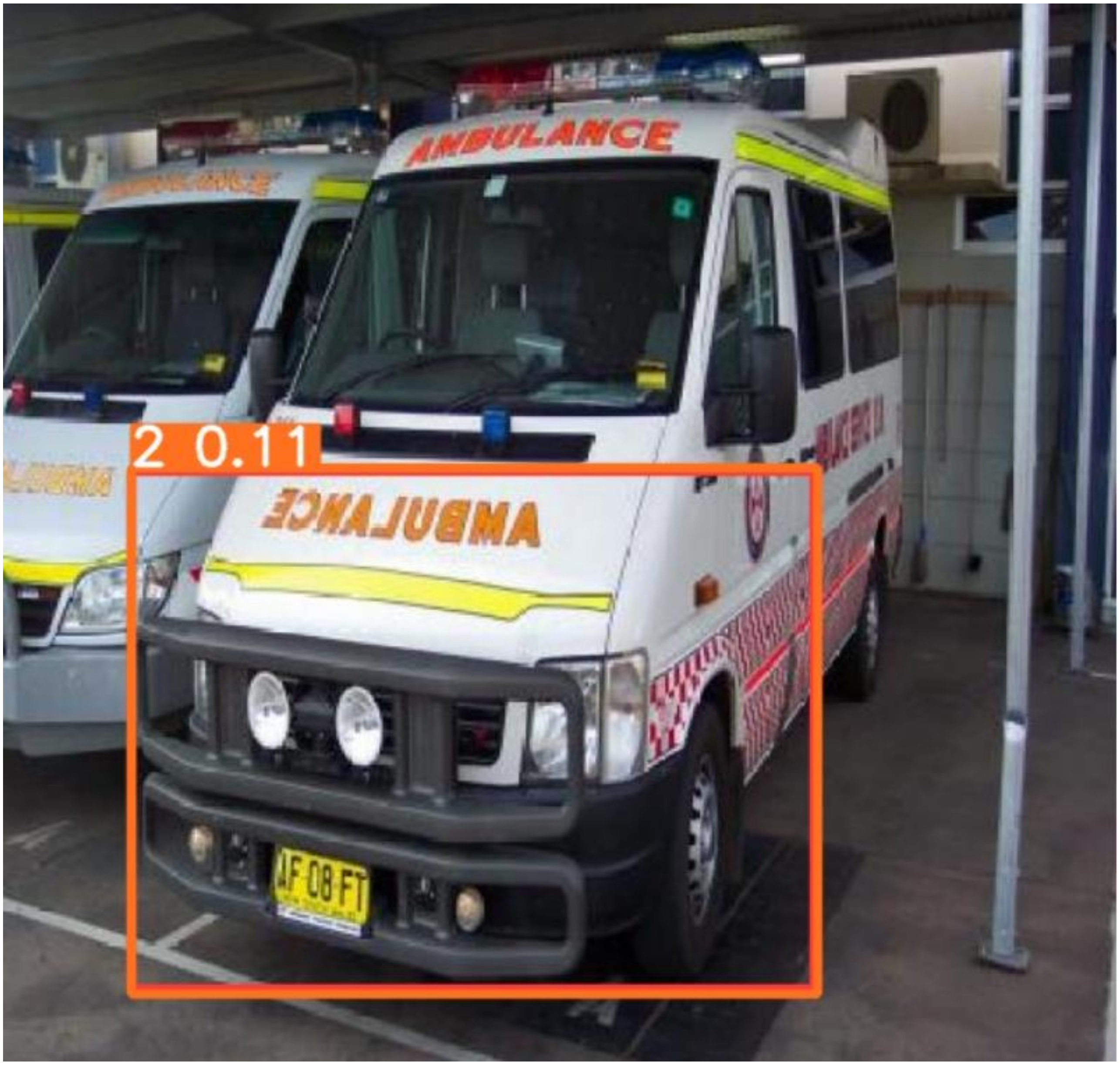
- Implement the same steps as the old data set, we also obtain the following results:

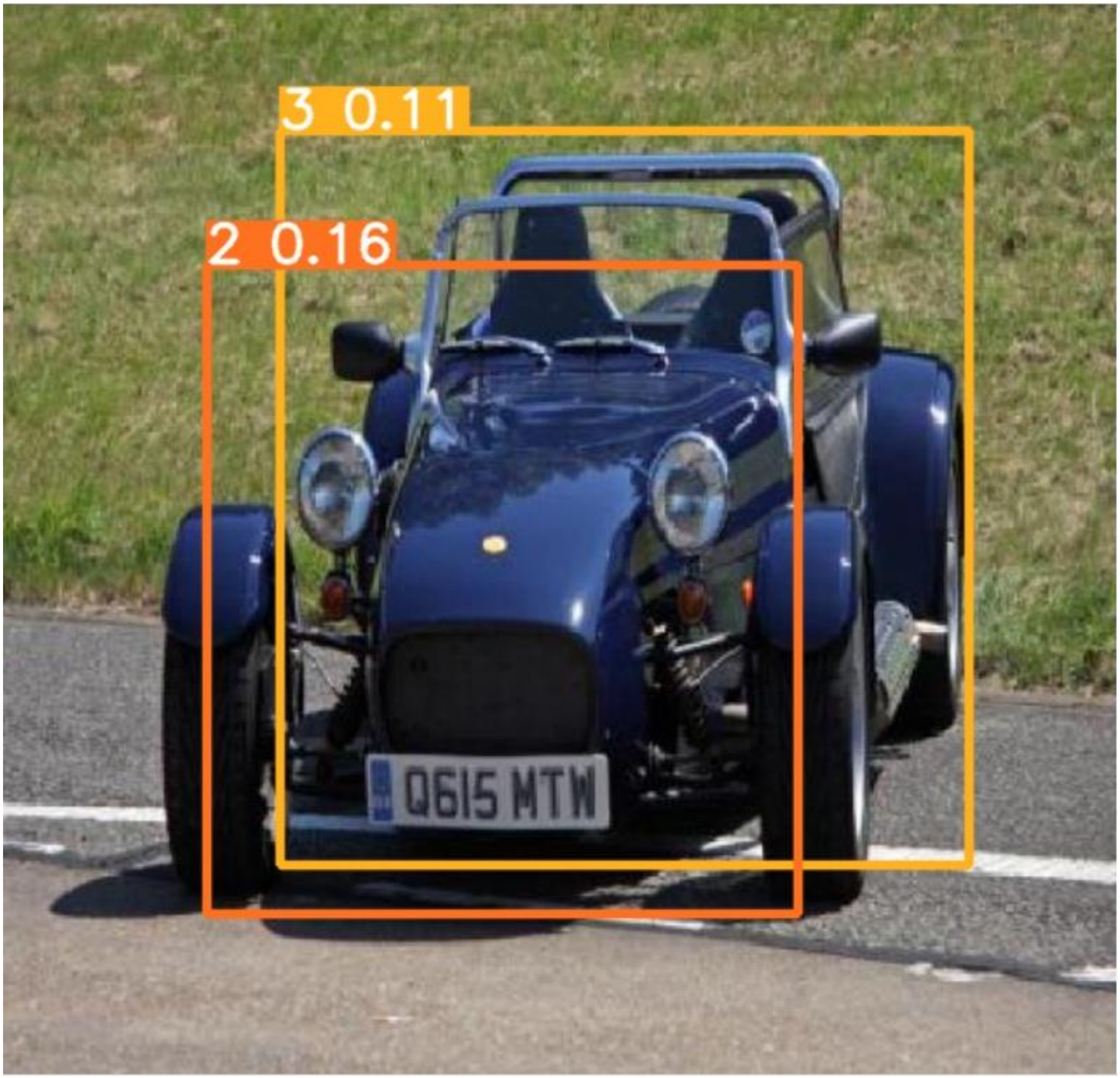


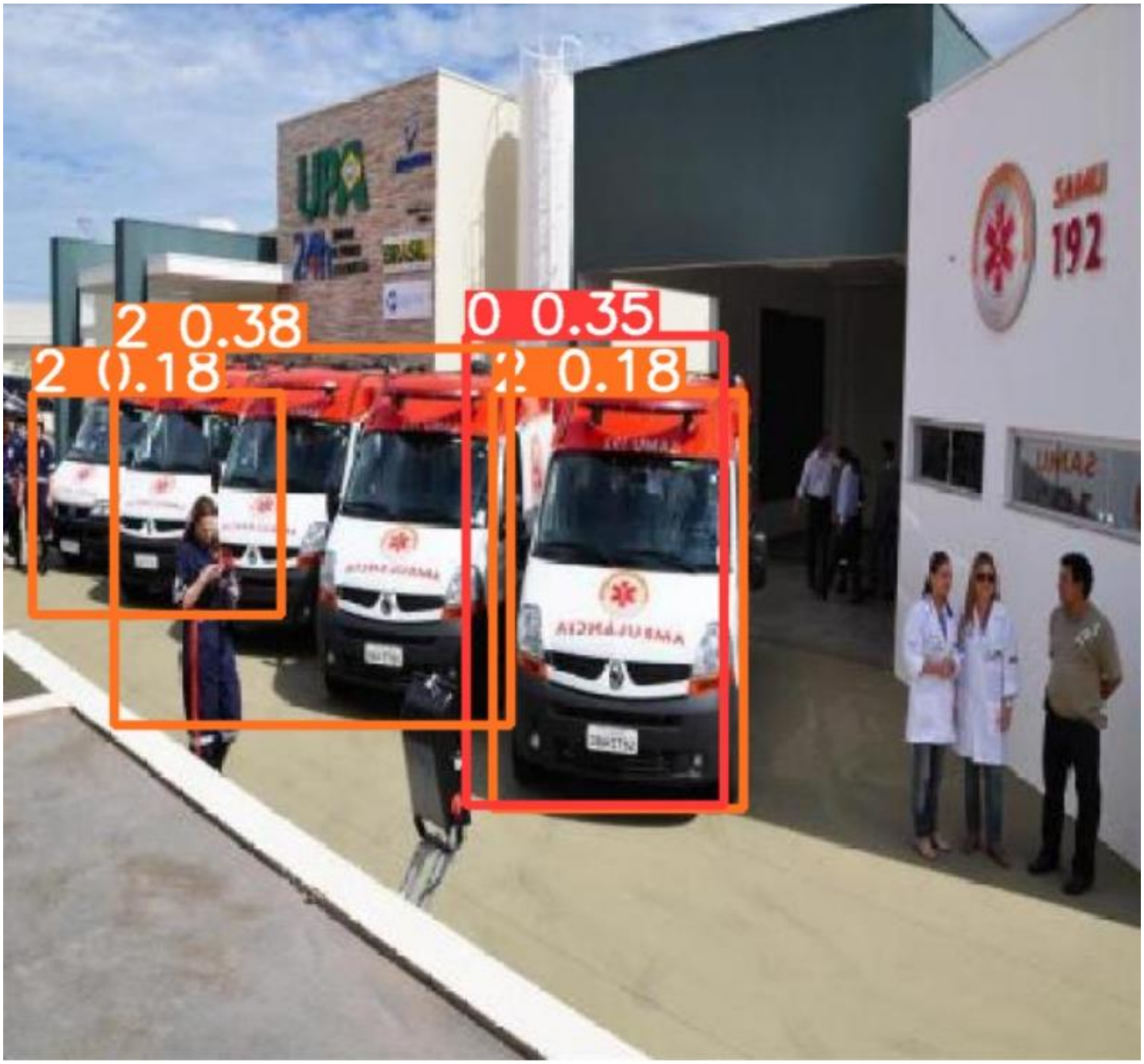
```
[13] # print out an augmented training example
     print("GROUND TRUTH AUGMENTED TRAINING DATA:")
     Image(filename='/content/yolov5/runs/train/yolov5s_results/train_batch0.jpg', width=900)
```

```
GROUND TRUTH AUGMENTED TRAINING DATA:
```

- Run Inference With Trained Weights:




3 0.11
2 0.16
Q615 MTW



III. Conclusion

- Train YOLOv5-Pytorch process is faster than YOLOv4-Pytorch. However, the train speed depends on the GPU that we use.
- To distinguish the difference of these two models easily, you can follow the following link: https://blog.roboflow.com/yolov4-versus-yolov5/
- As a result, in both YOLOv4-Pytorch and YOLOv5-Pytorch models, it was only acceptable, and some models were not detected. This limitation may be because the data is too little or the super parameters we use is not the best because it takes a lot of time to test.