HCMUS - VNUHCM / FIT / Computer Vision & Cognitive Cybernetics Department

Digital image & video processing - LQN

Student ID: 21127690

Student name: Ngo Nguyen Thanh Thanh

Report: YOLO AND DARKNET

I. Evaluation summary:

Task	Details of Implementation	Completion Percentage (%)	Notes
YOLOV7	Experiment with the different dataset	100%	Chesspieces (640x640) – 82 images
	Set up Working directory	100%	
	Download YOLOv7- Source Code	100%	
	Prepare the Chess Pieces Dataset	100%	
	Download pre-trained weights	100%	
	Demo	100%	
	Training	100%	
	Test	100%	
YOLOV11	Expeiment with different model, different source code	100%	Yolov11
	Experiment with the same dataset above	100%	Chesspieces (640x640) – 82 images
	Install Yolov11 via Ultralytics	100%	
	Demo with CLI	100%	
	Fine-tune YOLOv11 with custom dataset: Chess Pieces Dataset 640x640	100%	
	Training	100%	
	Test	100%	

II. Dataset: Chess pieces Dataset 640x640

Link: https://universe.roboflow.com/junior-zyvtn/chess-pieces-d0yt3/dataset/12

Total images: 82 :0 missing annotations, 0 null examples

Number Classes: 12 classes

1. black-bishop
2. black-king
3. black-knight
4. black-pawn

5. black-queen
6. black-rook
7. white-bishop
8. white-king
9. white-knight
10. white-pawn
11. white-queen
12. white-rook

Data split:

- **Train set:** 56
- **Valid set:** 18
- **Test set:** 8

Preprocessing: Auto Orient: Applied

Augumentation: No augumentations were applied

Number of Annotations: 1202 (average: 14.7 per image. Across 12classes)

Average image size: 12.19 mp (from 12.19mp to 12.19mp)

Median Image Ratio: 3024x4032 (tall)

Class Balance: Overview of the number of annotations for each class in your dataset. (image below)


Classes
all
train
valid
test
Brown Pawn 266
White Pawn 248
White Rook 94
Brown Bishop 85
Brown Rook 83
White Bishop 70 under represented
White Knight 70 under represented
Brown Knight 60 under represented
White King 59 under represented
Brown King 58 under represented
Brown Queen 58 under represented
White Queen 51 under represented

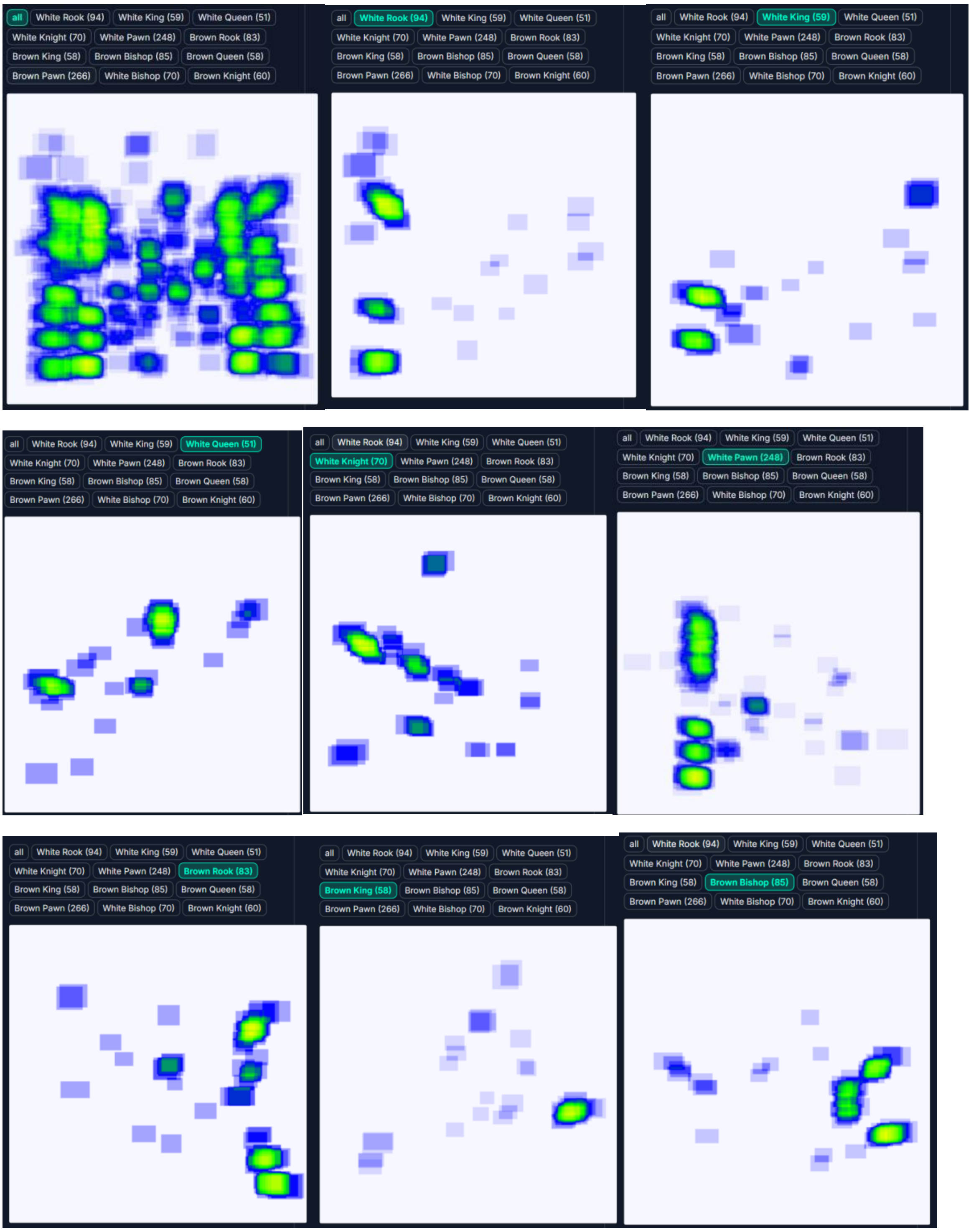

Classes
all
train
valid
test
Brown Pawn 163
White Pawn 154
White Rook 61
Brown Bishop 51
Brown Rook 57
White Bishop 44 under represented
White Knight 47 under represented
Brown Knight 45 under represented
White King 40 under represented
Brown King 39 under represented
Brown Queen 40 under represented
White Queen 33 under represented

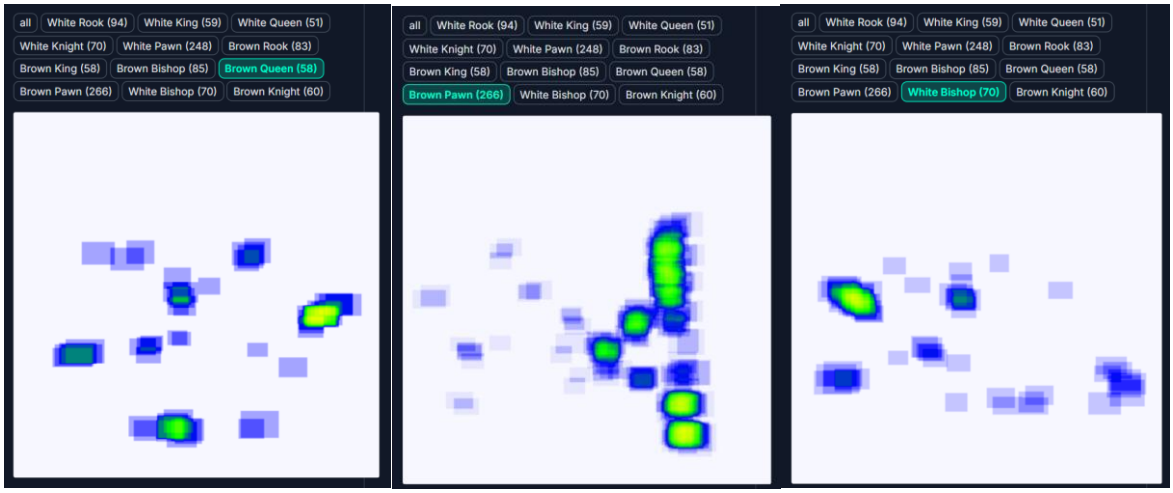
Dimension Insights: Overview of the sizes and aspect ratios of the images in your dataset.

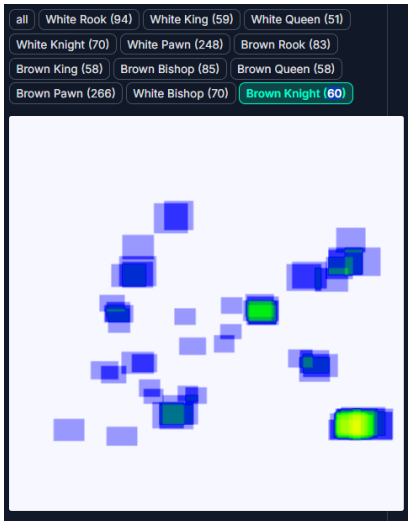
- **Size Distribution**
    - Sizes are based on the number of pixels in your images.
    - All images are the same size, 3024x4032.

- **Aspect Ratio Distribution:** The aspect ratio of your images compares the width vs. the height of your images.

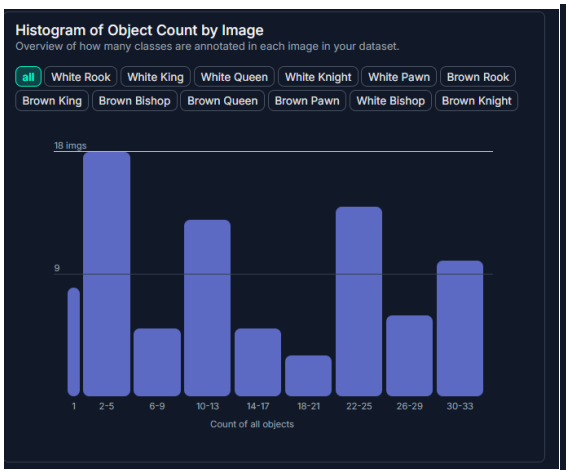
Annotation Heatmap: Overview of where your annotations are located in the images in your dataset. (image below)

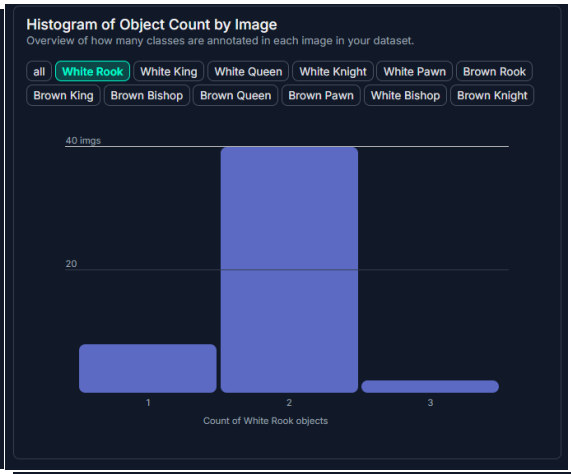




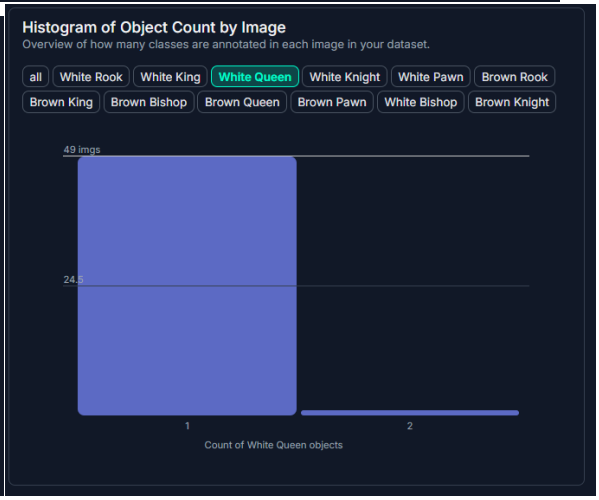


Histogram of Object Count by Image: Overview of how many classes are annotated in each image in your dataset. (image below)

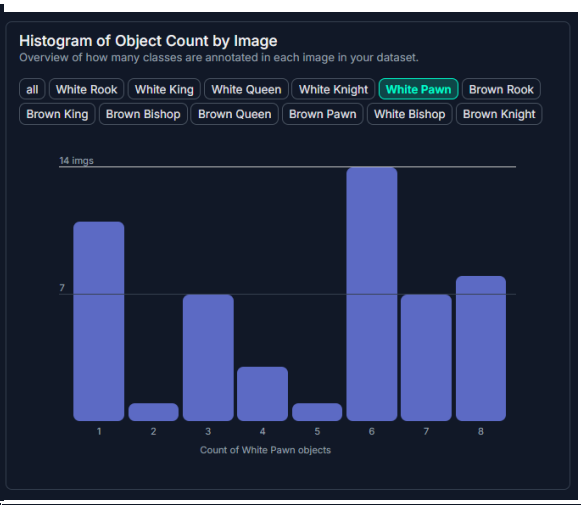

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
18 imgs
9
1
2-5
6-9
10-13
14-17
18-21
22-25
26-29
30-33
Count of all objects

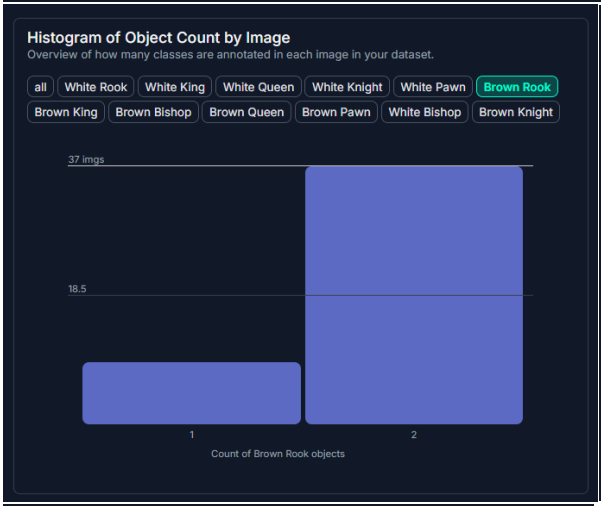

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
40 imgs
20
1
2
3
Count of White Rook objects

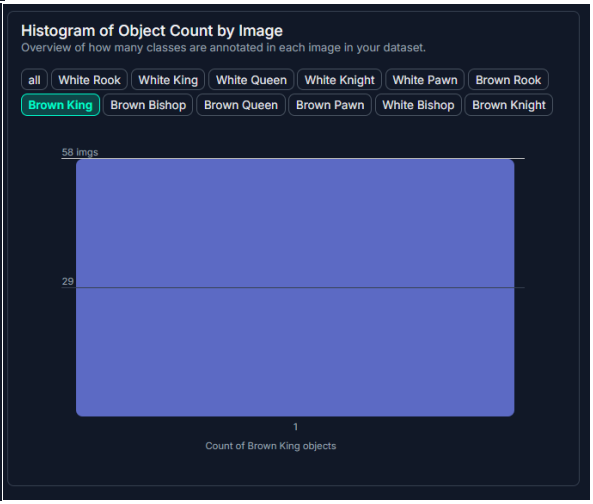

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
59 imgs
29.5
1
Count of White King objects

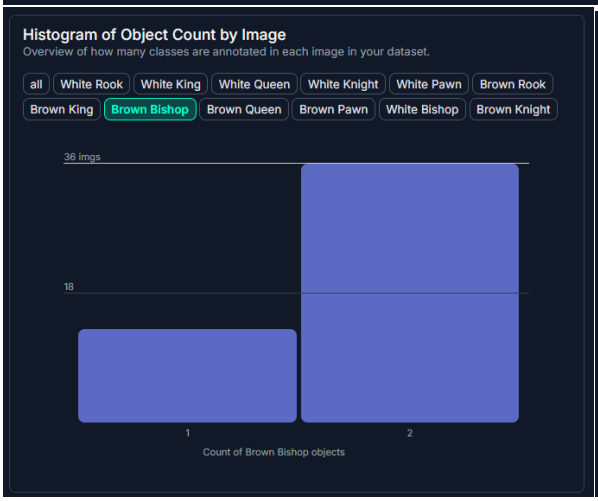

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
49 imgs
24.5
1
2
Count of White Queen objects

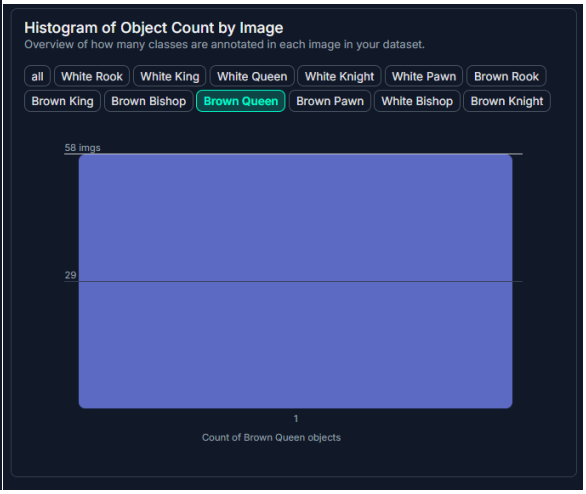

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
42 imgs
21
1
2
Count of White Knight objects

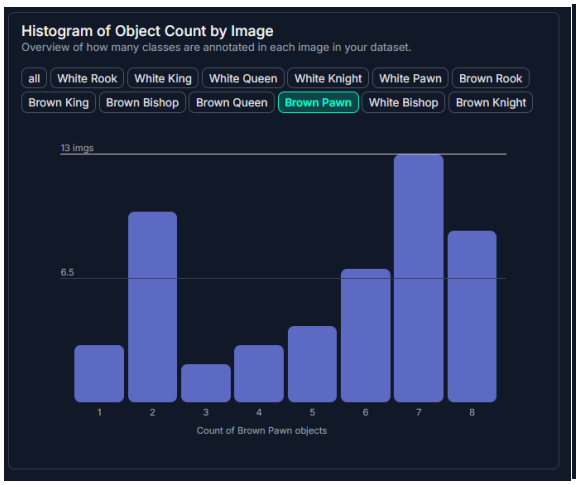

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
14 imgs
7
1
2
3
4
5
6
7
8
Count of White Pawn objects

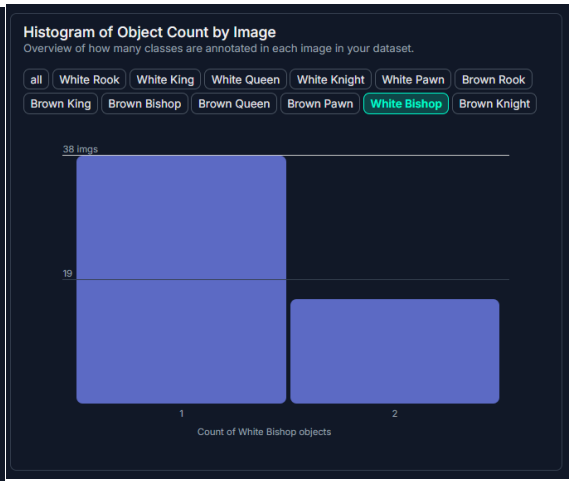

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
37 imgs
18.5
1
2
Count of Brown Rook objects

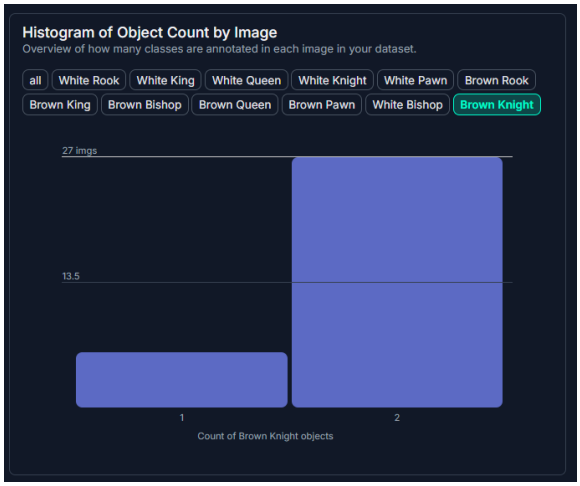

Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
58 imgs
29
1
Count of Brown King objects


Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
36 imgs
18
1
2
Count of Brown Bishop objects


Histogram of Object Count by Image
Overview of how many classes are annotated in each image in your dataset.
all
White Rook
White King
White Queen
White Knight
White Pawn
Brown Rook
Brown King
Brown Bishop
Brown Queen
Brown Pawn
White Bishop
Brown Knight
58 imgs
29
1
Count of Brown Queen objects







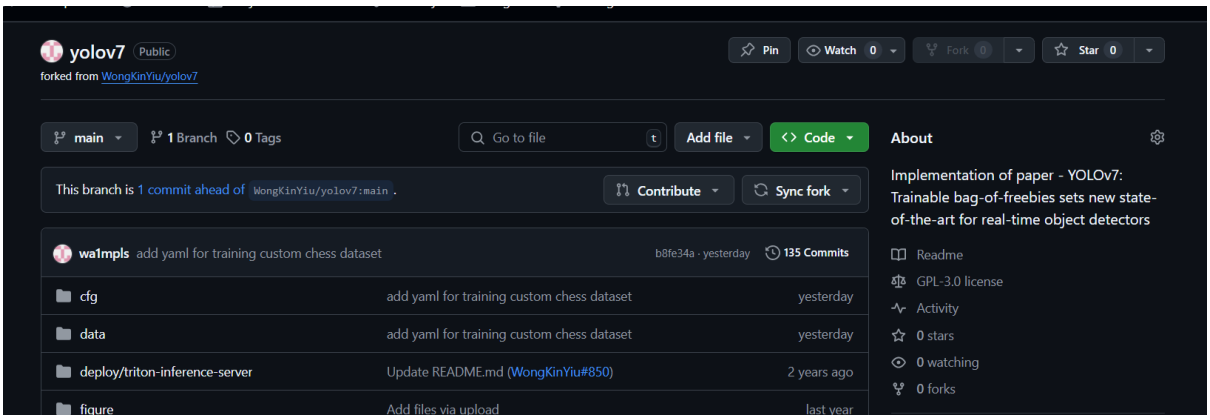
II. Experiment with YOLOV7

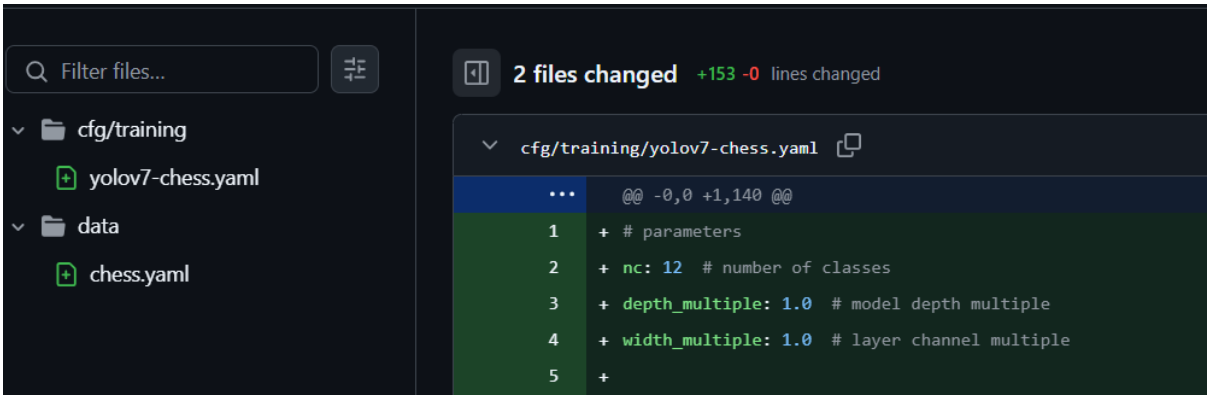
1. List of features and file structure:

A. File structure:

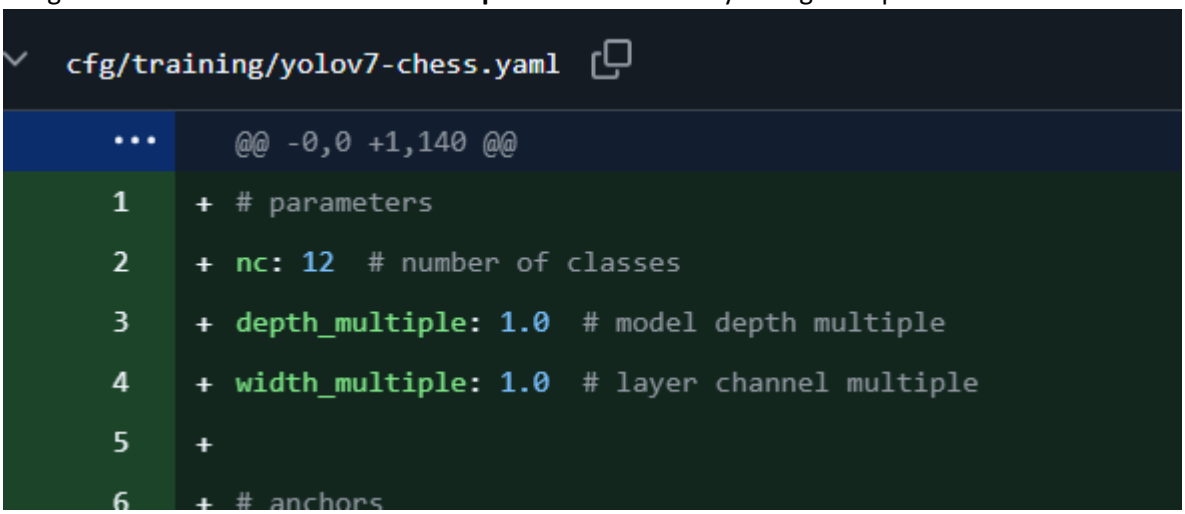
Changes in the source code:

Github link: https://github.com/wa1mpls/yolov7/tree/main

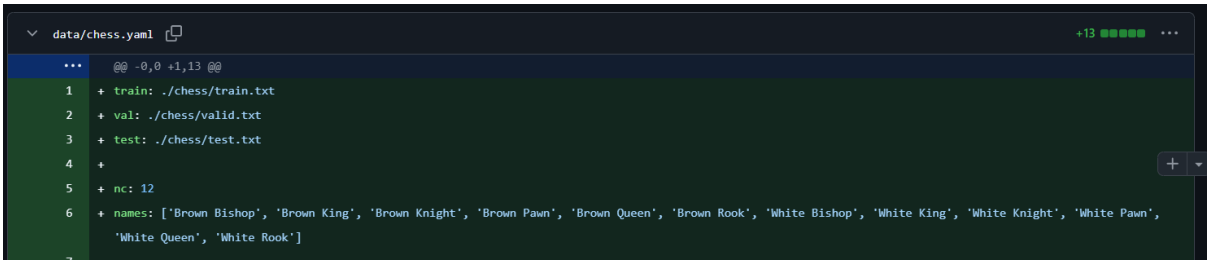




I add two file **in cfg/training folder (yolo7-chess.yaml**) and **data folder (chess.yaml)** as image above to run with the new **Chesspieces dataset.** I only change the part as below



I change number of class into 12 in yolov7-chess.yaml and this part in chess.yaml



I remain the rest as the origional source.

Changes in the chess folder when running colab file

The goal is to organize the Chess folder to follow a specific structure needed for YOLOv7 training (this version required this types of structure – different with yolov11 which I use then). Below is the breakdown of the desired folder structure:

- **images**
  - **train**: Contains training images.
  - **valid**: Contains validation images.
  - **test**: Contains test images.
- **labels**
  - **train**: Contains label files corresponding to the training images.
  - **valid**: Contains label files for the validation images.
  - **test**: Contains label files for the test images.

- **Text Files**
  - **train.txt**: Contains paths to all the training images.
  - **valid.txt**: Contains paths to all the validation images.
  - **test.txt**: Contains paths to all the test images.

```
# Reconstruct the 'chess' folder to
# - images
# ---- train
# ---- valid
# ---- test
# - labels
# ---- train
# ---- valid
# ---- test
# - train.txt (contains all path of the train images)
# - valid.txt (contains all path of the valid images)
# - test.txt (contains all path of the test images)
```

```python
import os
import shutil


os.chdir(CUSTOM_DATA_DIR)

data_names = ['train', 'valid', 'test']

# Function to move and rename folders
def move_and_rename_folders(data_names):
    for name in data_names:
        # Rename and move image folders
        source_image_dir = os.path.join(name, 'images')
        dest_image_dir = os.path.join('images', name)
        os.makedirs(os.path.dirname(dest_image_dir), exist_ok=True)
        os.rename(source_image_dir, dest_image_dir)

        # Rename and move label folders
        source_label_dir = os.path.join(name, 'labels')
        dest_label_dir = os.path.join('labels', name)
        os.makedirs(os.path.dirname(dest_label_dir), exist_ok=True)
        os.rename(source_label_dir, dest_label_dir)

        # Create the txt file
        txt_file = name + '.txt'
        with open(txt_file, 'w') as f:
            for filename in os.listdir(dest_image_dir):
                if filename.endswith(('.jpg', '.jpeg', '.png')):
                    f.write(f"{os.path.join('./images', name, filename)}\n")

move_and_rename_folders(data_names)

os.chdir("..")
```

```python
[ ] # Remove old directories
for dir_name in data_names:
  dir_path = os.path.join(CUSTOM_DATA_DIR, dir_name)
  if os.path.exists(dir_path):
    shutil.rmtree(dir_path)
```

B. List of functions:

Environment and Setup

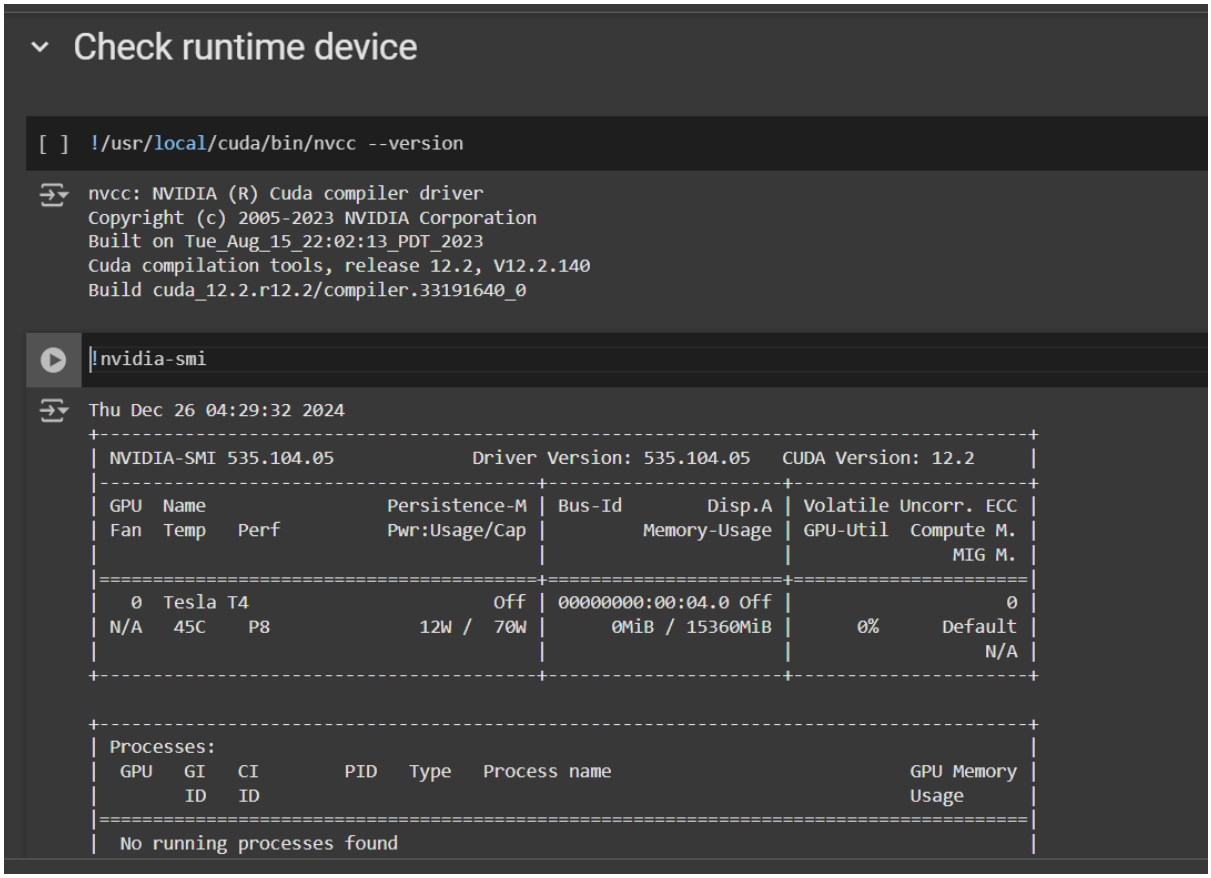
- **Checking GPU Availability**
  - **torch.device('cuda' if torch.cuda.is_available() else 'cpu'):** Checks and sets GPU or CPU as the device for PyTorch.
  - **torch.cuda.is_available**(): Verifies if a CUDA-capable GPU is available.
  - **torch.cuda.get_device_name():** Retrieves the name of the GPU in use.
- **Working Directory & Drive**
  - **drive.mount('/content/drive'):** Mounts Google Drive in Colab.
  - **os.makedirs(...):** Creates a directory (if it does not exist).
  - **os.path.exists(...):** Checks if a path exists.

- **%cd $WORKING_DIR:** Changes the current working directory to a specified path.

Experiment with YOLOv7

- **Cloning & Downloading**
  - **!git clone <repo>:** Clones the specified GitHub repository (YOLOv7 code).
  - **!gdown <file_id>:** Downloads files from Google Drive using their file IDs.
  - !**unzip <zipfile>:** Extracts the contents of a ZIP archive.
- **Data Preparation and Organization**
  - **os.rename(...):** Renames (and effectively moves) directories or files.
  - **os.makedirs(...):** Creates directories for images and labels if they don't exist.
  - **shutil.rmtree(...):** Recursively deletes a directory.
  - **open(...):** Creates and writes to the text files (train.txt, valid.txt, test.txt).
- **Downloading Pre-trained Weights**
  - **!curl -L <url> -o <outputfile>:** Downloads pre-trained weights from the given URL.
- **Demo, Training, and Testing**
  - **!python detect.py --weights ... --conf ... --source ...:** Runs the YOLOv7 detection script on a specified image.
  - **from PIL import Image:** Used to open and display images after detection.
  - **!python train.py --workers ... --device ... --epochs ... --data ... --cfg ... --weights ...:** Trains the YOLOv7 model with custom data.
  - **!python test.py --data ... --weights ...:** Tests/evaluates the trained YOLOv7 model.

C. Image proof:



```
[ ] import torch

    # setting device on GPU if available, else CPU
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print('Using device:', device)

    # additional info when using cuda
    if device.type == 'cuda':
        print('Device name: ', torch.cuda.get_device_name(0))
        # print(torch.cuda.memory_summary(device, abbreviated=False))
```

```
Using device: cuda
Device name:  Tesla T4
```

Set up Working directory

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] WORKING_DIR = '/content/drive/MyDrive/LAB05-xlav'
```

```
[ ] import os

    if not os.path.exists(WORKING_DIR):
        os.makedirs(WORKING_DIR)

    %cd $WORKING_DIR
```

```
/content/drive/MyDrive/LAB05-xlav
```

1. Experiment with YOLOv7

Clone Yolov7 code

```
[ ] !rm -rf 'yolov7'
```

```
[ ] !git clone https://github.com/wa1mpls/yolov7.git
```

```
Cloning into 'yolov7'...
remote: Enumerating objects: 626, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 626 (delta 0), reused 2 (delta 0), pack-reused 620 (from 1)
Receiving objects: 100% (626/626), 42.35 MiB | 15.24 MiB/s, done.
Resolving deltas: 100% (305/305), done.
Updating files: 100% (110/110), done.
```

```
[ ] %cd 'yolov7'
```

```
/content/drive/MyDrive/LAB05-xlav/yolov7
```

Prepare the Chess Pieces Dataset - 640x640

Original link: https://universe.roboflow.com/junior-zyvtn/chess-pieces-d0yt3/dataset/12

```
[ ] # Download the dataset from gdrive
    !gdown '1oQZelBeMPrhXRil1kBy5CLR_Dqy92zJB'
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1oQZelBeMPrhXRil1kBy5CLR_Dqy92zJB
From (redirected): https://drive.google.com/uc?id=1oQZelBeMPrhXRil1kBy5CLR_Dqy92zJB&confirm=t&uuid=c47f47f6-86f2-498e-b7a4-c57ab2695d59
To: /content/drive/MyDrive/LAB05-xlav/yolov7/ChessPieces_yolov7pytorch.zip
100% 104M/104M [00:00<00:00, 161MB/s]
```

```
[ ] CUSTOM_DATA_DIR = 'chess'
```

```
!rm -rf $CUSTOM_DATA_DIR
!unzip 'ChessPieces_yolov7pytorch.zip' -d $CUSTOM_DATA_DIR
!rm 'ChessPieces_yolov7pytorch.zip'
```

```
Archive:  ChessPieces_yolov7pytorch.zip
  inflating: chess/data.yaml
  inflating: chess/__MACOSX/._data.yaml
  inflating: chess/README.dataset.txt
  inflating: chess/README.roboflow.txt
   creating: chess/test/
   creating: chess/test/images/
   creating: chess/test/labels/
  inflating: chess/test/images/unnamed-91-_jpg.rf.3313095e1471c20f5b380342f0b10d40.jpg
  inflating: chess/test/images/unnamed-2024-09-20T162950-004_jpg.rf.c1e63175908d482e9f03811a9e08c0f5.jpg
  inflating: chess/test/images/unnamed-94-_jpg.rf.9be30b3168aeeadaf86aff20ca689d25.jpg
```

```
# Reconstruct the 'chess' folder to
# - images
# ---- train
# ---- valid
# ---- test
# - labels
# ---- train
# ---- valid
# ---- test
# - train.txt (contains all path of the train images)
# - valid.txt (contains all path of the valid images)
# - test.txt (contains all path of the test images)

import os
import shutil


os.chdir(CUSTOM_DATA_DIR)

data_names = ['train', 'valid', 'test']

# Function to move and rename folders
def move_and_rename_folders(data_names):
    for name in data_names:
        # Rename and move image folders
        source_image_dir = os.path.join(name, 'images')
        dest_image_dir = os.path.join('images', name)
        os.makedirs(os.path.dirname(dest_image_dir), exist_ok=True)
        os.rename(source_image_dir, dest_image_dir)

        # Rename and move label folders
        source_label_dir = os.path.join(name, 'labels')
```

lab5.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 18:37

+ Code + Text

```python
# Function to move and rename folders
def move_and_rename_folders(data_names):
    for name in data_names:
        # Rename and move image folders
        source_image_dir = os.path.join(name, 'images')
        dest_image_dir = os.path.join('images', name)
        os.makedirs(os.path.dirname(dest_image_dir), exist_ok=True)
        os.rename(source_image_dir, dest_image_dir)

        # Rename and move label folders
        source_label_dir = os.path.join(name, 'labels')
        dest_label_dir = os.path.join('labels', name)
        os.makedirs(os.path.dirname(dest_label_dir), exist_ok=True)
        os.rename(source_label_dir, dest_label_dir)

        # Create the txt file
        txt_file = name + '.txt'
        with open(txt_file, 'w') as f:
            for filename in os.listdir(dest_image_dir):
                if filename.endswith(('.jpg', '.jpeg', '.png')):
                    f.write(f"{os.path.join('./images', name, filename)}\n")

move_and_rename_folders(data_names)

os.chdir("..")
```

```python
[ ]  # Remove old directories
     for dir_name in data_names:
       dir_path = os.path.join(CUSTOM_DATA_DIR, dir_name)
       if os.path.exists(dir_path):
         shutil.rmtree(dir_path)
```

Download pre-trained weights

```
%mkdir weights
!curl -L https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7_training.pt -o weights/yolov7_training.pt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 72.1M  100 72.1M    0     0  47.2M      0  0:00:01  0:00:01 --:--:-- 82.9M
```

```
from PIL import Image
Image.open('runs/detect/exp/horses.jpg')
```

Demo

```
[ ] !rm -rf runs/detect/
```

```
[ ] !python detect.py --weights weights/yolov7_training.pt --conf 0.25 --img-size 640 --source inference/images/horses.jpg
```

```
Namespace(weights=['weights/yolov7_training.pt'], source='inference/images/horses.jpg', img_size=640, conf_thres=0.25, iou_thres=0.45, device='', view_img=False, save_txt=False, save_
YOLOR 🚀 b8fe34a torch 2.5.1+cu121 CUDA:0 (Tesla T4, 15102.0625MB)

/content/drive/MyDrive/LAB05-xlav/yolov7/models/experimental.py:252: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the de
  ckpt = torch.load(w, map_location=map_location)  # load
Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
Model Summary: 314 layers, 36907898 parameters, 6194944 gradients
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

/usr/local/lib/python3.10/dist-packages/torch/functional.py:534: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered int
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
5 horses, Done. (17.1ms) Inference, (586.6ms) NMS
 The image with the result is saved in: runs/detect/exp/horses.jpg
Done. (0.966s)
```

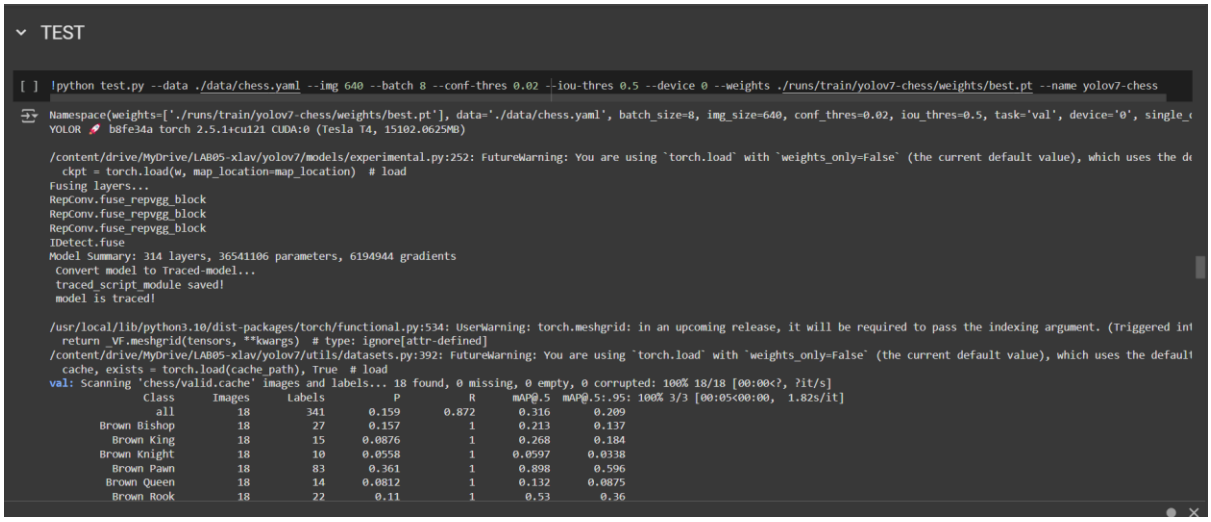
Training

+ Code + Text

```
[ ] !rm -rf wandb runs/train runs/test
```

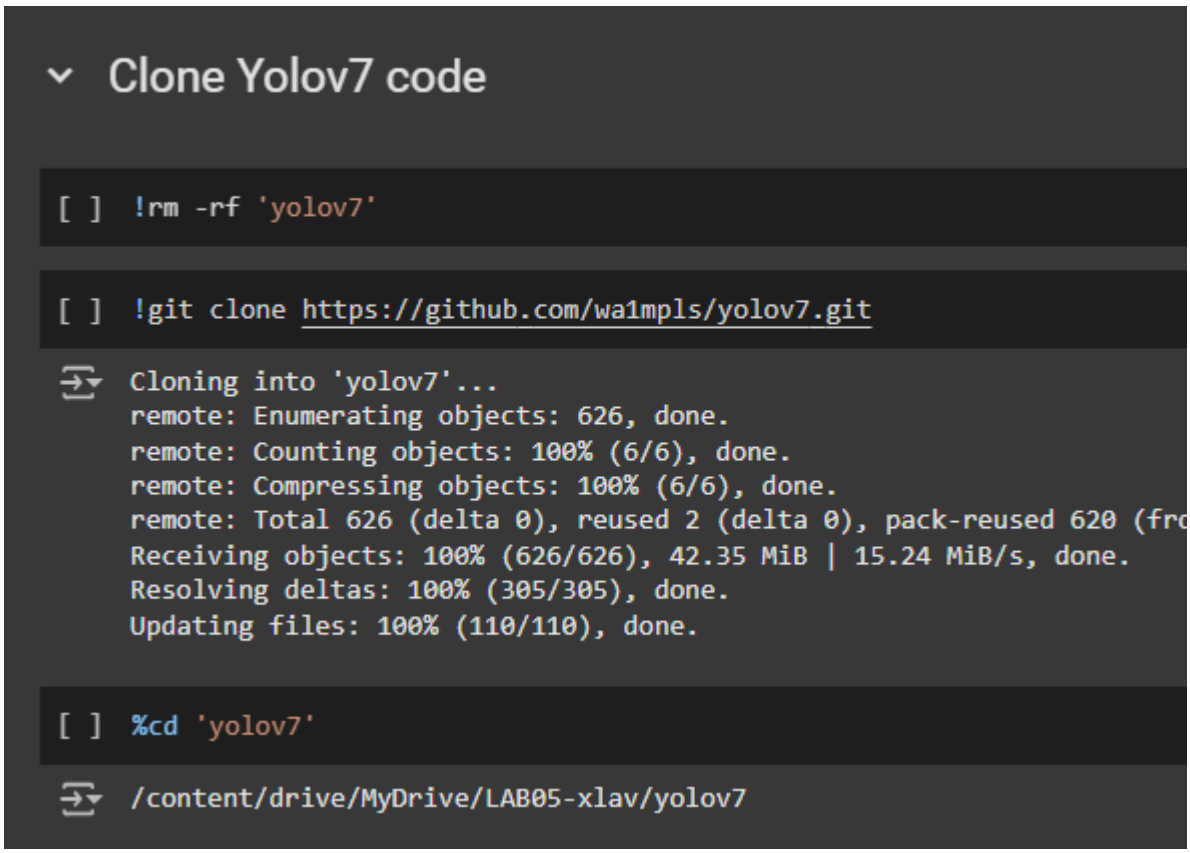
```
[ ] !python train.py --workers 8 --device 0 --batch-size 8 --epochs 200 --data data/chess.yaml --img 640 640 --cfg cfg/training/yolov7-chess.yaml --weights 'weights/yolov7_training.pt'
```

```
     199/199     8.27G   0.01835   0.02284   0.01934   0.06053       142       640: 100% 7/7 [00:02<00:00,  2.36it/s]
                 Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:02<00:00,  1.40s/it]
                   all          18         341       0.165       0.908       0.305       0.206
          Brown Bishop          18          27       0.157           1       0.224       0.149
            Brown King          18          15      0.0887           1       0.286       0.207
          Brown Knight          18          10      0.0583           1      0.0604      0.0397
            Brown Pawn          18          83       0.411           1       0.839       0.552
           Brown Queen          18          14      0.0832           1       0.114      0.0813
            Brown Rook          18          22       0.127           1       0.318       0.211
          White Bishop          18          19       0.137       0.842       0.156      0.0882
            White King          18          15      0.0927       0.933       0.243       0.169
          White Knight          18          19       0.142       0.947       0.183       0.121
            White Pawn          18          76       0.445           1       0.586        0.38
           White Queen          18          14      0.0825       0.214       0.111      0.0668
            White Rook          18          27       0.155       0.963       0.544       0.409
200 epochs completed in 0.374 hours.

/content/drive/MyDrive/LAB05-xlav/yolov7/utils/general.py:802: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the defau
  x = torch.load(f, map_location=torch.device('cpu'))
Optimizer stripped from runs/train/yolov7-chess/weights/last.pt, 74.9MB
Optimizer stripped from runs/train/yolov7-chess/weights/best.pt, 74.9MB
Images sizes do not match. This will causes images to be display incorrectly in the UI.
wandb: ⢿ uploading artifact run_9n3xp1yy_model (1.5s)
wandb:   ↳ ⢿ best.pt 1.1MB/71.4MB (0.5s)
wandb: ⢿ uploading output.log 48.0KB/96.5KB (0.4s)
wandb: ⢿ uploading config.yaml 5.6KB/5.6KB (0.4s)
```



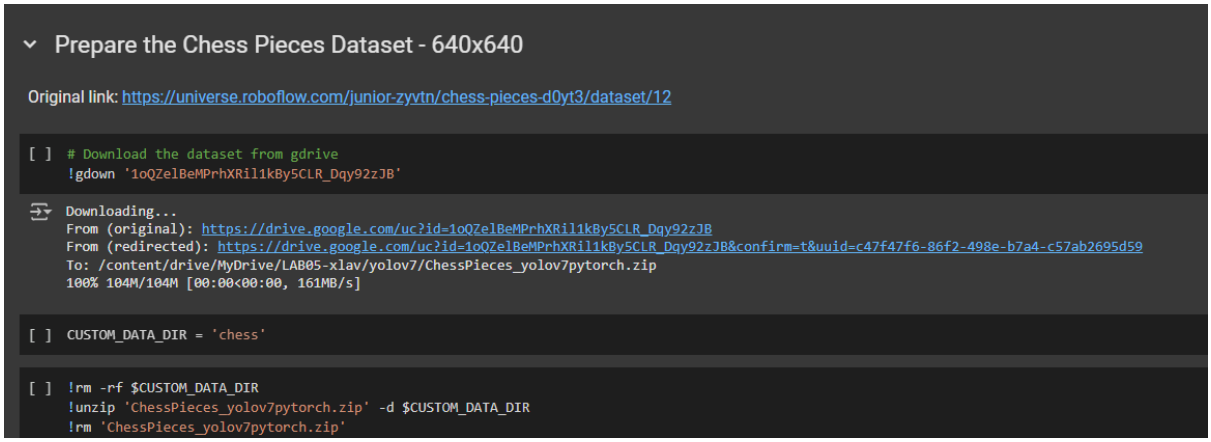
2. Summary of Function Usage and Explanation

1. Clone YOLOv7 Code

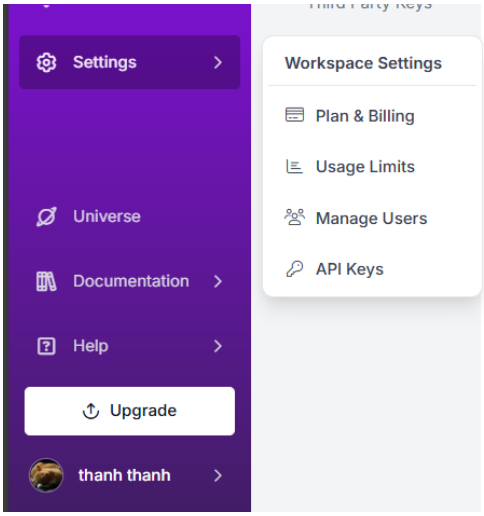


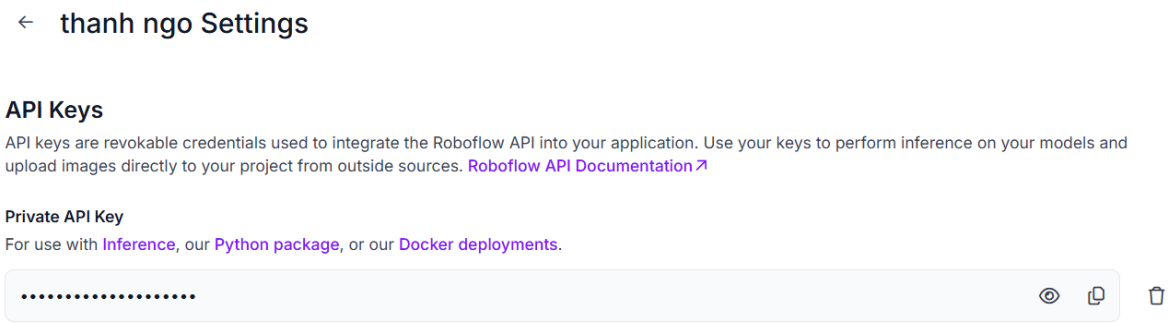
- **!rm -rf 'yolov7'**: This command removes any existing folder named yolov7 to ensure a clean setup.
- **!git clone https://github.com/wa1mpls/yolov7.git**: This command clones the YOLOv7 repository from GitHub into the current directory.
- **%cd 'yolov7'**: This changes the working directory to the yolov7 folder that was just cloned. It ensures that subsequent commands are executed within the YOLOv7 code base.

2. Prepare the Chess Pieces Dataset



- **!gdown '1oQZelBeMPrhXRil1kBy5CLR_Dqy92zJB'**: This command uses gdown to download a dataset from Google Drive using the provided file ID. The dataset contains images and annotations for training YOLO.
- **CUSTOM_DATA_DIR = 'chess'**: Sets a variable for the directory where the dataset will be unzipped.
- **!rm -rf $CUSTOM_DATA_DIR**: Removes any previous versions of the chess folder, ensuring that the new dataset is extracted cleanly.
- **!unzip 'ChessPieces_yolov7pytorch.zip' -d $CUSTOM_DATA_DIR**: Unzips the downloaded dataset into the specified directory (chess).
- **!rm 'ChessPieces_yolov7pytorch.zip'**: Deletes the zip file after extraction.
- **You need an account to download dataset. It required private API key from Robflow website.**

- 



3. Reorganizing Dataset Folders

```
import os
import shutil


os.chdir(CUSTOM_DATA_DIR)

data_names = ['train', 'valid', 'test']

# Function to move and rename folders
def move_and_rename_folders(data_names):
    for name in data_names:
        # Rename and move image folders
        source_image_dir = os.path.join(name, 'images')
        dest_image_dir = os.path.join('images', name)
        os.makedirs(os.path.dirname(dest_image_dir), exist_ok=True)
        os.rename(source_image_dir, dest_image_dir)

        # Rename and move label folders
        source_label_dir = os.path.join(name, 'labels')
        dest_label_dir = os.path.join('labels', name)
        os.makedirs(os.path.dirname(dest_label_dir), exist_ok=True)
        os.rename(source_label_dir, dest_label_dir)

        # Create the txt file
        txt_file = name + '.txt'
        with open(txt_file, 'w') as f:
            for filename in os.listdir(dest_image_dir):
                if filename.endswith(('.jpg', '.jpeg', '.png')):
                    f.write(f"{os.path.join('./images', name, filename)}\n")

move_and_rename_folders(data_names)

os.chdir("..")
```

```
[ ] # Remove old directories
    for dir_name in data_names:
      dir_path = os.path.join(CUSTOM_DATA_DIR, dir_name)
      if os.path.exists(dir_path):
        shutil.rmtree(dir_path)
```

- **os.chdir(CUSTOM_DATA_DIR)**: Changes the current working directory to the folder containing the dataset.
- **data_names = ['train', 'valid', 'test']**: Specifies the three main directories of the dataset: train, valid, and test, which contain images and labels for training, validation, and testing.
- **move_and_rename_folders(data_names)**: This function does the following for each dataset split:
    - **source_image_dir & dest_image_dir**: Moves the images from their original directory to a new directory structure (images/train, images/valid, images/test).
    - **source_label_dir & dest_label_dir**: Moves the label files to corresponding directories (labels/train, labels/valid, labels/test).

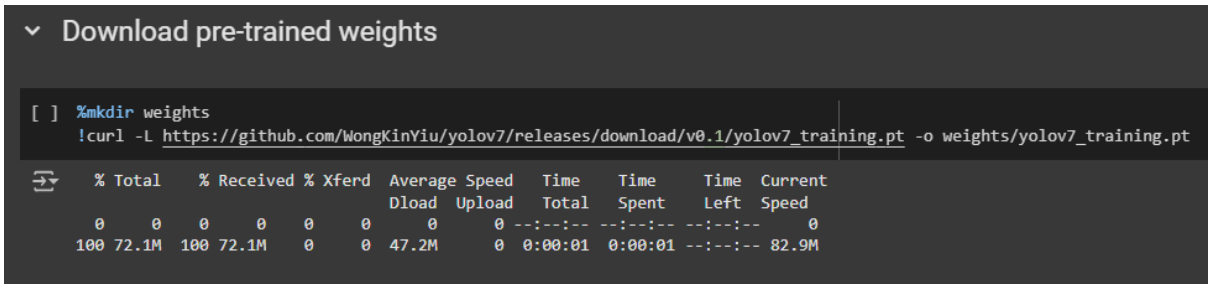
- **txt_file**: For each dataset split (train, valid, test), it creates a .txt file that lists all image file paths, which is required by YOLO for training.

4. Remove Old Dataset Folders

```
# Remove old directories
for dir_name in data_names:
  dir_path = os.path.join(CUSTOM_DATA_DIR, dir_name)
  if os.path.exists(dir_path):
    shutil.rmtree(dir_path)
```

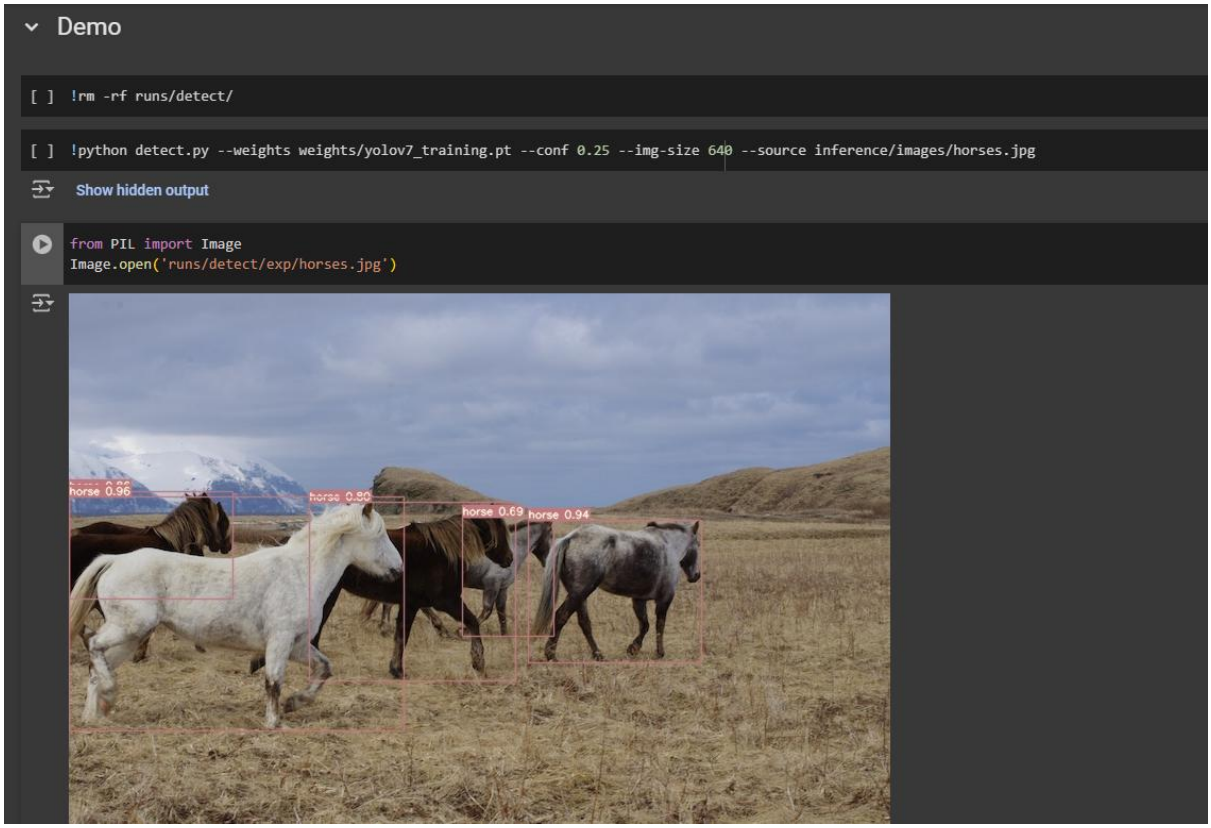
- **os.chdir("..")**: Changes the working directory back to the parent directory.
- **shutil.rmtree(dir_path)**: Deletes any old folders (train, valid, test) that might have existed before the reorganization.

5. Download Pre-trained Weights



- **%mkdir weights**: Creates a new directory called weights to store the pre-trained weights.
- **!curl -L https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7_training.pt -o weights/yolov7_training.pt**: Uses curl to download the pre-trained weights for YOLOv7 from a GitHub release. These weights help improve the training process by providing a starting point.

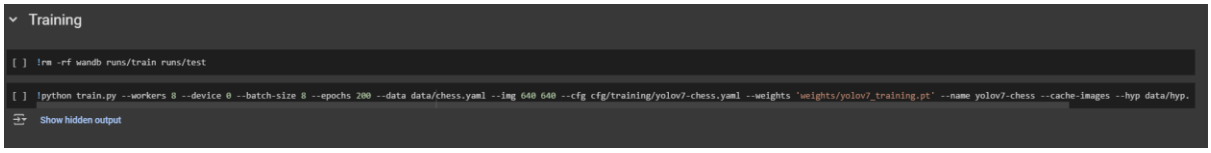
6. Run YOLOv7 Detection



!rm -rf runs/detect/: Removes any previous detection results stored in the runs/detect/ directory.

- **!python detect.py**: Runs the detection script using YOLOv7 with the following parameters:
    - **--weights weights/yolov7_training.pt**: Specifies the path to the pre-trained YOLOv7 weights.
    - **--conf 0.25**: Sets the confidence threshold for detection. Objects detected with a confidence score below this value will not be considered.
    - **--img-size 640**: Specifies the image size (640x640) for the input images.
    - **--source inference/images/horses.jpg**: Specifies the input image (horses.jpg) for the detection.
- **Image.open('runs/detect/exp/horses.jpg')**: Opens and displays the detection result using the PIL library.

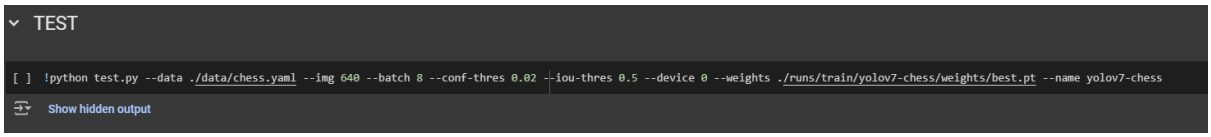
7. Train YOLOv7



- **!rm -rf wandb runs/train runs/test**: Removes previous training results and logs.
- **!python train.py**: Starts the training process with YOLOv7, using the following parameters:
    - **--workers 8**: Specifies the number of data loading workers to speed up data loading.
    - **--device 0**: Specifies which GPU to use (device 0, the first GPU).

- **--batch-size 8**: Sets the batch size to 8 images per training iteration.
- **--epochs 200**: Specifies the number of epochs to train the model (200).
- **--data data/chess.yaml**: Specifies the path to the dataset configuration file (chess.yaml), which contains details about the dataset (number of classes, paths, etc.).
- **--img 640 640**: Specifies the input image size (640x640).
- **--cfg cfg/training/yolov7-chess.yaml**: Specifies the YOLOv7 model configuration for this task.
- **--weights 'weights/yolov7_training.pt'**: Specifies the pre-trained weights for fine-tuning.
- **--name yolov7-chess**: Defines a name for this training session.
- **--cache-images**: Caches the images to speed up training.
- **--hyp data/hyp.scratch.custom.yaml**: Specifies a custom hyperparameter configuration file for training. (if you want to adjust other hyparameter like learning rates, batch size,… Do it in this file)

8. Test YOLOv7



!python test.py: Tests the trained YOLOv7 model with the following parameters:

- **--data ./data/chess.yaml**: Specifies the dataset configuration.
- **--img 640**: Specifies the input image size for testing.
- **--batch 8**: Specifies the batch size during testing.
- **--conf-thres 0.02**: Sets the confidence threshold for detections.
- **--iou-thres 0.5**: Sets the intersection-over-union (IoU) threshold for valid detections.
- **--device 0**: Specifies which GPU to use.
- **--weights ./runs/train/yolov7-chess/weights/best.pt**: Specifies the best weights from the training session.
- **--name yolov7-chess**: Defines the name for this test session.

3. Results and Evaluations

A. Results:

a) Demo:

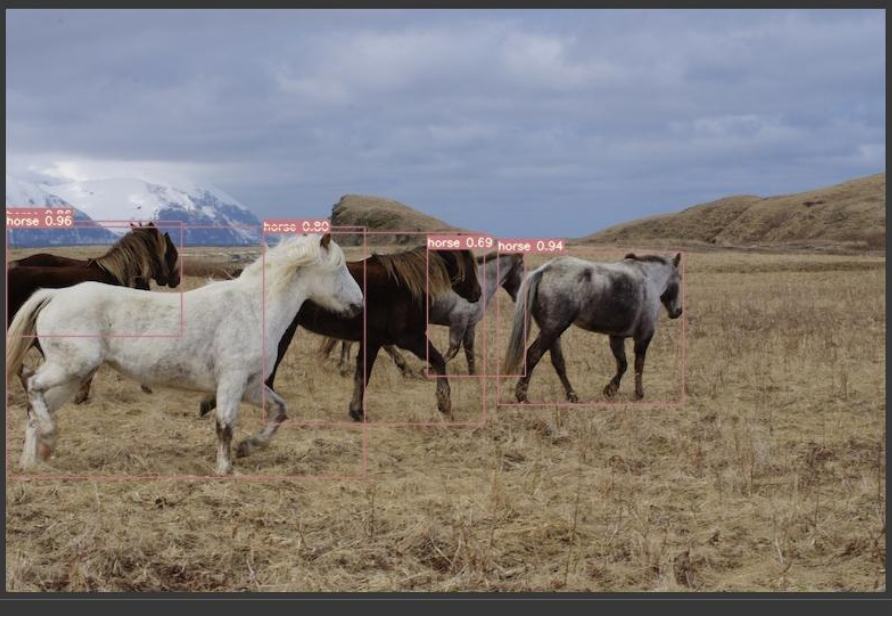
Demo

```
[ ] !rm -rf runs/detect/
```

```
!python detect.py --weights weights/yolov7_training.pt --conf 0.25 --img-size 640 --source inference/images/horses.jpg
```

Show hidden output

```
from PIL import Image
Image.open('runs/detect/exp/horses.jpg')
```

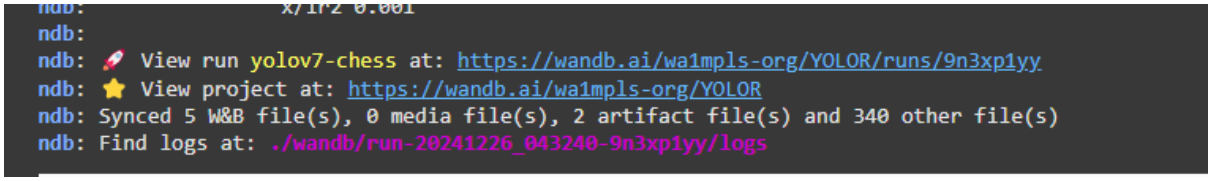


b) Training: (epochs = 200)

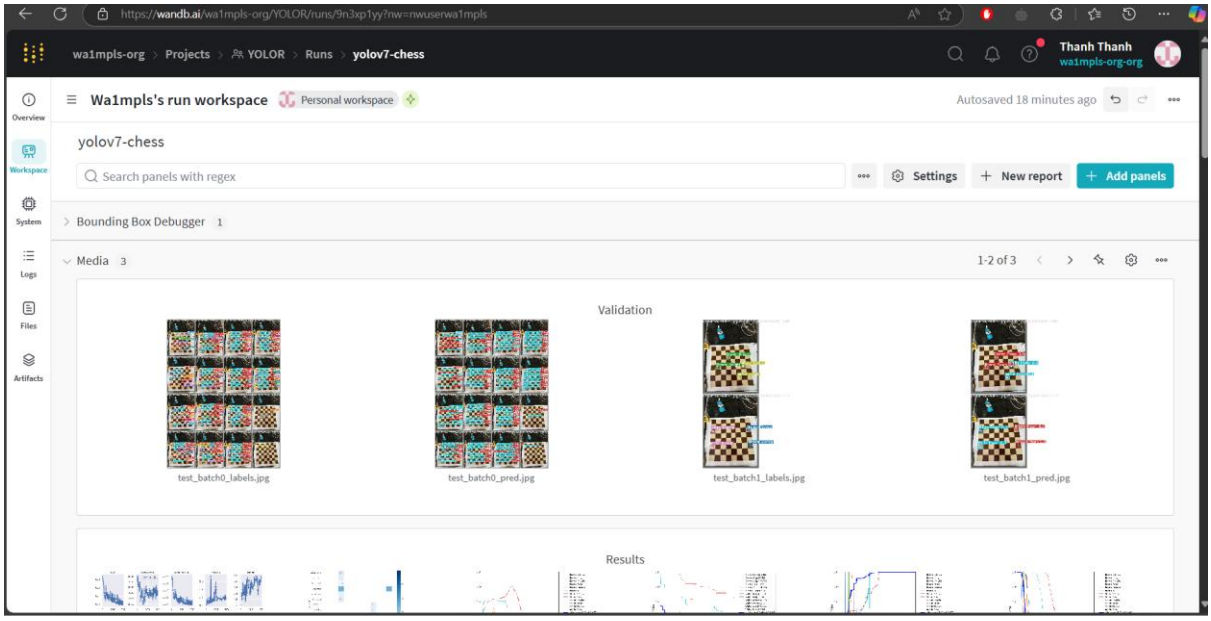
```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 195/199     8.27G   0.01764   0.02583   0.01928   0.06275       111       640: 100% 7/7 [00:03<00:00,  2.10it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:00<00:00,  4.13it/s]
                 all          18         341       0.163       0.837       0.299       0.199

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 196/199     8.27G   0.01752   0.02725    0.0191   0.06387       121       640: 100% 7/7 [00:03<00:00,  2.33it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:00<00:00,  4.87it/s]
                 all          18         341        0.16       0.979       0.302       0.203

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 197/199     8.27G   0.03394   0.02687   0.02261   0.08342       134       640: 100% 7/7 [00:03<00:00,  2.30it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:00<00:00,  4.95it/s]
                 all          18         341       0.161       0.934       0.303       0.205

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 198/199     8.27G   0.01714   0.02436   0.01905   0.06055        78       640: 100% 7/7 [00:03<00:00,  1.93it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:00<00:00,  3.81it/s]
                 all          18         341       0.165       0.908       0.305       0.206

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 199/199     8.27G   0.01835   0.02284   0.01934   0.06053       142       640: 100% 7/7 [00:02<00:00,  2.36it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:02<00:00,  1.40s/it]
                 all          18         341       0.165       0.908       0.305       0.206
        Brown Bishop          18          27       0.157           1       0.224       0.149
          Brown King          18          15      0.0887           1       0.286       0.207
        Brown Knight          18          10      0.0583           1      0.0604      0.0397
          Brown Pawn          18          83       0.411           1       0.839       0.552
         Brown Queen          18          14      0.0832           1       0.114      0.0813
          Brown Rook          18          22       0.127           1       0.318       0.211
        White Bishop          18          19       0.137       0.842       0.156      0.0882
          White King          18          15      0.0927       0.933       0.243       0.169
        White Knight          18          19       0.142       0.947       0.183       0.121
          White Pawn          18          76       0.445           1       0.586        0.38
         White Queen          18          14      0.0825       0.214       0.111      0.0668
          White Rook          18          27       0.155       0.963       0.544       0.409
0 epochs completed in 0.374 hours.
```

View train detail in : https://wandb.ai/wa1mpls-org/YOLOR/runs/9n3xp1yy?nw=nwuserwa1mpls

This mean you need an account to login

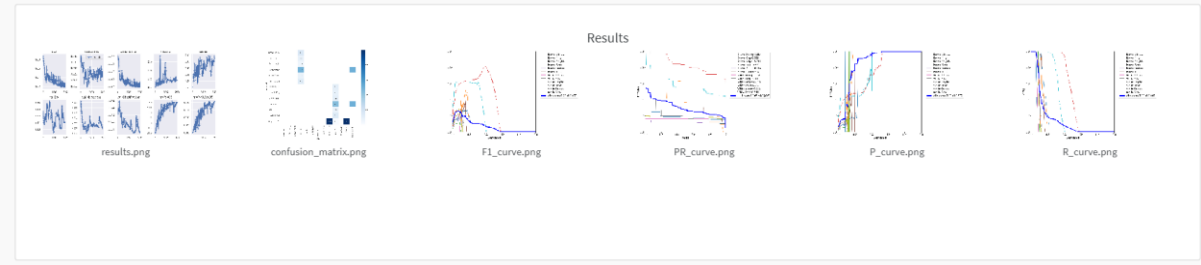


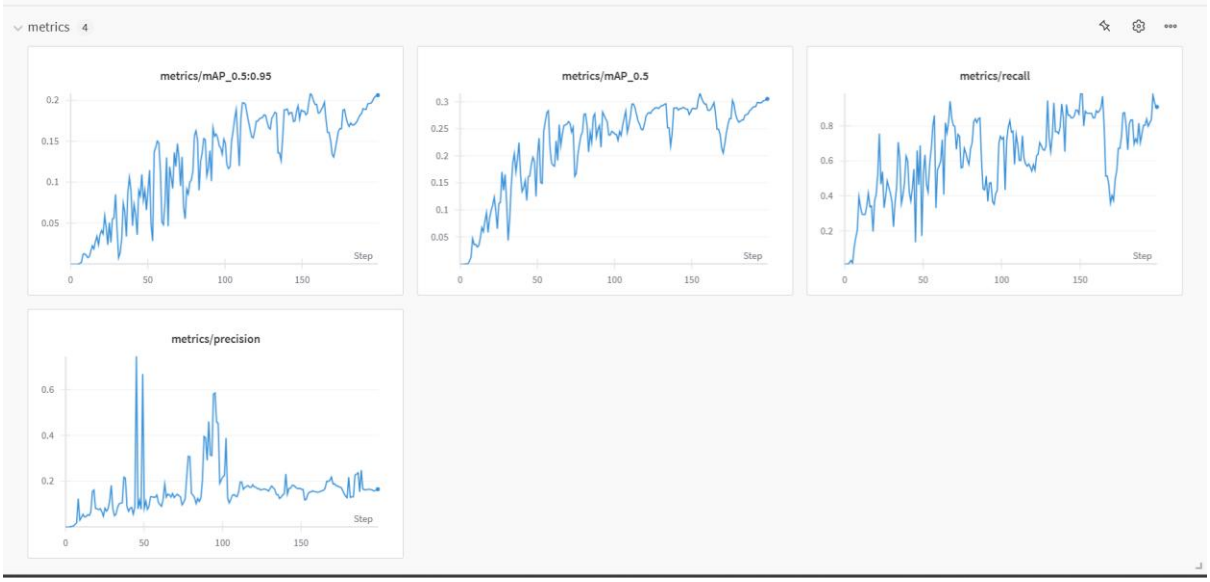
In that link I have some pictures as below:

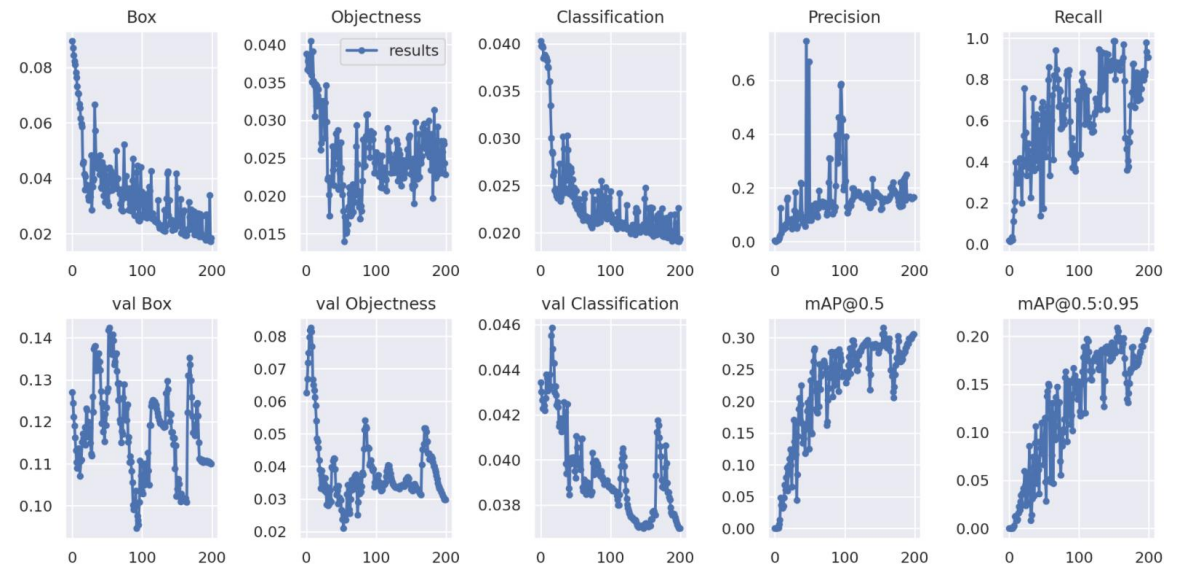

https://wandb.ai/wa1mpls-org/YOLOR/runs/9n3xp1yy?nw=nwuserwa1mpls
wa1mpls-org
Projects
YOLOR
Runs
yolov7-chess
Thanh Thanh
wa1mpls-org-org
Wa1mpls's run workspace
Personal workspace
Autosaved 18 minutes ago
Overview
Workspace
System
Logs
Files
Artifacts
yolov7-chess
Search panels with regex
Settings
New report
Add panels
Bounding Box Debugger 1
Media 3
1-2 of 3
Validation
test_batch0_labels.jpg
test_batch0_pred.jpg
test_batch1_labels.jpg
test_batch1_pred.jpg
Results

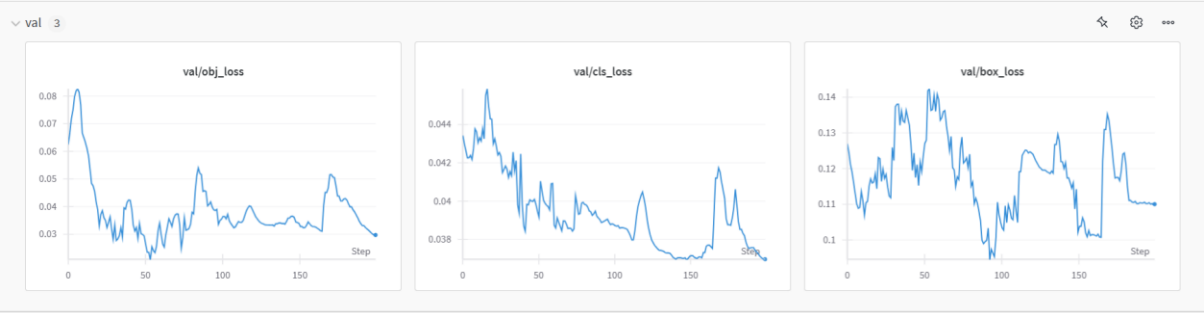

Validation
test_batch0_labels.jpg
test_batch0_pred.jpg
test_batch1_labels.jpg
test_batch1_pred.jpg


Mosaics
train_batch0.jpg
train_batch1.jpg
train_batch2.jpg
train_batch3.jpg
train_batch4.jpg
train_batch5.jpg
train_batch6.jpg
train_batch7.jpg


Results
results.png
confusion_matrix.png
F1_curve.png
PR_curve.png
P_curve.png
R_curve.png


metrics 4
metrics/mAP_0.5:0.95
metrics/mAP_0.5
metrics/recall
metrics/precision
Step


Box
Objectness
Classification
Precision
Recall
results
val Box
val Objectness
val Classification
mAP@0.5
mAP@0.5:0.95



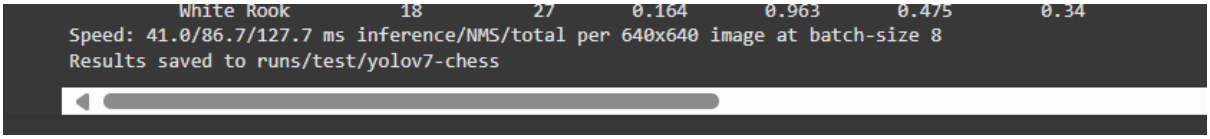
c) Testing

TEST

```
!python test.py --data ./data/chess.yaml --img 640 --batch 8 --conf-thres 0.02 --iou-thres 0.5 --device 0 --weights ./runs/train/yolov7-chess/weights/best.pt --name yolov7-chess

Namespace(weights=['./runs/train/yolov7-chess/weights/best.pt'], data='./data/chess.yaml', batch_size=8, img_size=640, conf_thres=0.02, iou_thres=0.5, task='val', device='0', single_
YOLOR 🚀 b8fe34a torch 2.5.1+cu121 CUDA:0 (Tesla T4, 15102.0625MB)

/content/drive/MyDrive/LAB05-xlav/yolov7/models/experimental.py:252: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the d
  ckpt = torch.load(w, map_location=map_location)  # load
Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
Model Summary: 314 layers, 36541106 parameters, 6194944 gradients
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

/usr/local/lib/python3.10/dist-packages/torch/functional.py:534: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered in
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
/content/drive/MyDrive/LAB05-xlav/yolov7/utils/datasets.py:392: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the defaul
  cache, exists = torch.load(cache_path), True  # load
val: Scanning 'chess/valid.cache' images and labels... 18 found, 0 missing, 0 empty, 0 corrupted: 100% 18/18 [00:00<?, ?it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 3/3 [00:05<00:00,  1.82s/it]
                 all          18         341       0.159       0.872       0.316       0.209
         Brown Bishop          18          27       0.157           1       0.213       0.137
           Brown King          18          15      0.0876           1       0.268       0.184
         Brown Knight          18          10      0.0558           1      0.0597      0.0338
           Brown Pawn          18          83       0.361           1       0.898       0.596
          Brown Queen          18          14      0.0812           1       0.132      0.0875
           Brown Rook          18          22        0.11           1        0.53        0.36
         White Bishop          18          19        0.14       0.895       0.154      0.0972
           White King          18          15      0.0205       0.172      0.0682      0.0396
         White Knight          18          19       0.146           1       0.176       0.107
           White Pawn          18          76       0.445           1        0.69       0.443
          White Queen          18          14       0.136       0.429       0.132       0.083
           White Rook          18          27       0.164       0.963       0.475        0.34
Speed: 41.0/86.7/127.7 ms inference/NMS/total per 640x640 image at batch-size 8
Results saved to runs/test/yolov7-chess
```

Open folder below to see results: **run/test/yolov7-chess.**

In the yolov11 I did this part. If you want to know. Read the yolov11 part.



B. Evaluation

After running for 200 epochs, the results from YOLOv7 were not satisfactory, as shown below:

YOLOv7 Run Summary:

wandb: metrics/mAP_0.5 0.30535

wandb: metrics/mAP_0.5:0.95 0.20625

wandb: metrics/precision 0.16492

wandb: metrics/recall 0.90834

wandb: train/box_loss 0.01835

wandb: train/cls_loss 0.01934

wandb: train/obj_loss 0.02284

wandb: val/box_loss 0.11004

wandb: val/cls_loss 0.03696

wandb: val/obj_loss 0.02974

wandb: x/lr0 0.001

wandb: x/lr1 0.001

wandb: x/lr2 0.001

The results show that after 200 epochs, YOLOv7 still struggled with relatively low performance, especially with precision (0.16492) and mAP (mean Average Precision) of 0.30535 at IoU 0.5. Additionally, the training took a long time.

III. EXPERIMENT WITH YOLOV11

1. List of features and file structure:

A. File structure:

I use the library version of yolov11. So I don't have specific file structures here.

B. List of functions:

Experiment with YOLOv11 (Ultralytics)

- **Installation & Checks**
  - **%pip install <packages>:** Installs the required packages (ultralytics, supervision, roboflow).
  - **ultralytics.checks():** Verifies that the Ultralytics environment is correctly set up.
- **Roboflow Integration**
  - **from roboflow import Roboflow:** Imports the Roboflow client library.
  - **rf = Roboflow(api_key=...):** Creates a new Roboflow instance using an API key.
  - **project = rf.workspace("...").project("..."): Retrieves** a specific project.
  - **version = project.version(12):** Gets a particular version of the project.
  - **version.download("yolov11"):** Downloads the dataset in YOLOv11 format.
- **YOLOv11 CLI Commands**
  - **!yolo task=detect mode=predict model=... source**=...: Runs inference with YOLOv11 on an image.
  - **!yolo task=detect mode=train model=... data=... epochs=...:** Trains a YOLOv11 model using the specified data and parameters.
  - **!yolo task=detect mode=val model=... data=...:** Validates the YOLOv11 model on the test set.
  - **IPyImage(filename=...):** Displays images directly in the Colab notebook.

C. Image proof:

2. Experiment with YOLOv11

```
[ ] %cd $WORKING_DIR
```

```
/content/drive/MyDrive/LAB05-xlav
```

Install YOLOv11 via Ultralytics

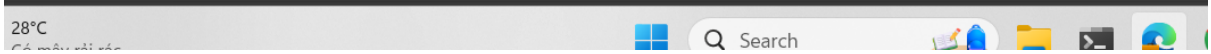
```
[ ] !rm -rf yolov11
```

```
[ ] import os

    os.makedirs('yolov11', exist_ok=True)
    HOME = os.path.join(os.getcwd(), 'yolov11')
    os.chdir(HOME)
    os.getcwd()
```

```
'/content/drive/MyDrive/LAB05-xlav/yolov11'
```

```
[ ] %pip install "ultralytics<=8.3.40" supervision roboflow
    import ultralytics
    ultralytics.checks()
```

```
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 37.8/112.6 GB disk)
```

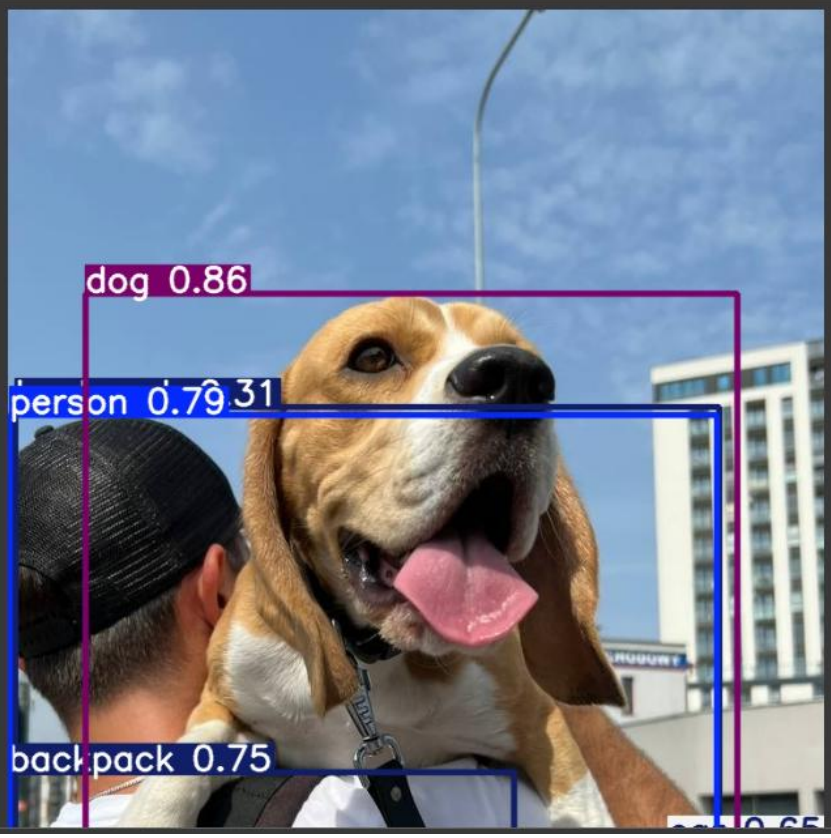


Demo with CLI

```
[ ] !yolo task=detect mode=predict model=yolo11m.pt conf=0.25 source='https://media.roboflow.com/notebooks/examples/dog.jpeg' save=True
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11m.pt to 'yolo11m.pt'...
100% 38.8M/38.8M [00:01<00:00, 39.0MB/s]
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11m summary (fused): 303 layers, 20,091,712 parameters, 0 gradients, 68.0 GFLOPs

Downloading https://media.roboflow.com/notebooks/examples/dog.jpeg to 'dog.jpeg'...
100% 104k/104k [00:00<00:00, 36.5MB/s]
image 1/1 /content/drive/MyDrive/LAB05-xlav/yolov11/dog.jpeg: 640x384 1 person, 1 car, 1 dog, 2 backpacks, 97.3ms
Speed: 5.5ms preprocess, 97.3ms inference, 1123.8ms postprocess per image at shape (1, 3, 640, 384)
Results saved to runs/detect/predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

```
from IPython.display import Image as IPyImage

IPyImage(filename=f'{HOME}/runs/detect/predict/dog.jpg', width=500)
```



Fine-tune YOLOv11 with custom dataset: Chess Pieces Dataset - 640x640

```
[ ] !mkdir {HOME}/datasets
    %cd {HOME}/datasets

    from roboflow import Roboflow
    import getpass

    ROBOFLOW_API_KEY = getpass.getpass("Enter your Roboflow API key: ")
    rf = Roboflow(api_key=ROBOFLOW_API_KEY)
    project = rf.workspace("junior-zyvtn").project("chess-pieces-d0yt3")
    version = project.version(12)
    dataset = version.download("yolov11")
```

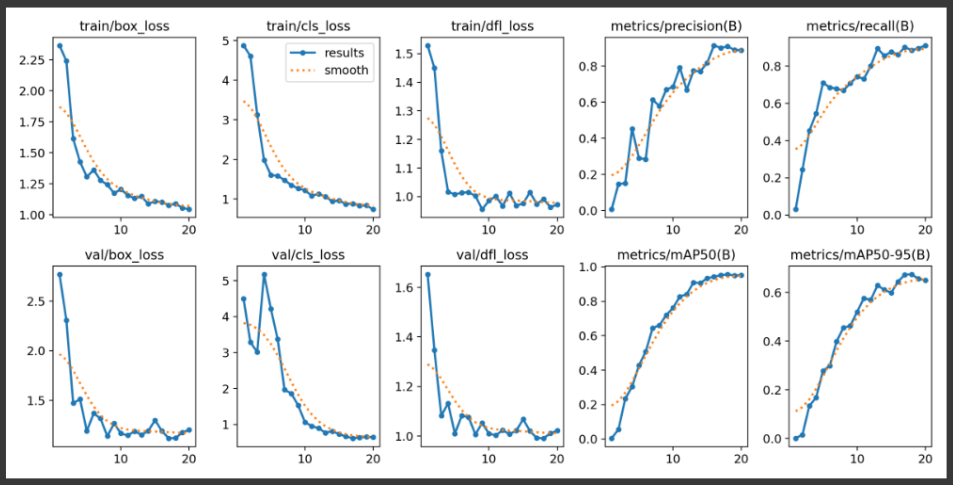
```
/content/yolov11/datasets
Enter your Roboflow API key: ··········
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Chess-Pieces-12 to yolov11:: 100%|██████████| 106205/106205 [00:02<00:00, 42148.10it/s]

Extracting Dataset Version Zip to Chess-Pieces-12 in yolov11:: 100%|██████████| 176/176 [00:00<00:00, 518.80it/s]
```

Training

```
%cd {HOME}

!yolo task=detect mode=train model=yolo11m.pt data={dataset.location}/data.yaml epochs=20 imgsz=640 plots=True
```

```
/content/yolov11
New https://pypi.org/project/ultralytics/8.3.54 available 😃 Update with 'pip install -U ultralytics'
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolo11m.pt, data=/content/yolov11/datasets/Chess-Pieces-12/data.yaml, epochs=20, time=None, patience=100, batch=16, imgsz=640, sa
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 21.2MB/s]
Overriding model.yaml nc=80 with nc=12

                   from  n    params  module                                       arguments
  0                  -1  1      1856  ultralytics.nn.modules.conv.Conv             [3, 64, 3, 2]
  1                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  2                  -1  1    111872  ultralytics.nn.modules.block.C3k2            [128, 256, 1, True, 0.25]
  3                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
  4                  -1  1    444928  ultralytics.nn.modules.block.C3k2            [256, 512, 1, True, 0.25]
  5                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
  6                  -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  7                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
  8                  -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1    990976  ultralytics.nn.modules.block.C2PSA           [512, 512, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 12            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 13                  -1  1   1642496  ultralytics.nn.modules.block.C3k2            [1024, 512, 1, True]
 14                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 15            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 16                  -1  1    542720  ultralytics.nn.modules.block.C3k2            [1024, 256, 1, True]
```

```
from IPython.display import Image as IPyImage

IPyImage(filename=f'{HOME}/runs/detect/train/results.png', width=800)
```



TEST

```
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

```
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11m summary (fused): 303 layers, 20,039,284 parameters, 0 gradients, 67.7 GFLOPs
val: Scanning /content/yolov11/datasets/Chess-Pieces-12/valid/labels.cache... 18 images, 0 backgrounds, 0 corrupt: 100% 18/18 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:02<00:00,  1.39s/it]
                   all         18        341      0.909      0.885      0.956      0.673
          Brown Bishop         15         27      0.837      0.852      0.895      0.587
            Brown King         15         15      0.874        0.8      0.914      0.656
          Brown Knight          5         10          1      0.668      0.995      0.705
            Brown Pawn         14         83          1      0.911      0.995      0.712
           Brown Queen         14         14      0.821          1      0.958      0.677
            Brown Rook         11         22      0.975          1      0.995      0.716
          White Bishop         14         19      0.813      0.688      0.859      0.568
            White King         15         15       0.76          1      0.948      0.689
          White Knight         15         19      0.949      0.987      0.993       0.67
            White Pawn         14         76          1      0.974      0.995      0.673
           White Queen         14         14       0.95      0.857      0.978      0.753
            White Rook         13         27      0.926      0.889      0.954      0.669
Speed: 6.7ms preprocess, 37.2ms inference, 0.0ms loss, 56.1ms postprocess per image
Results saved to runs/detect/val2
💡 Learn more at https://docs.ultralytics.com/modes/val
```

2. Summary of Function Usage and Explanation

1. Install YOLOv11 via Ultralytics

2. Experiment with YOLOv11

```
[ ] %cd $WORKING_DIR
```

```
/content/drive/MyDrive/LAB05-xlav
```

Install YOLOv11 via Ultralytics

```
[ ] !rm -rf yolov11
```

```
[ ] import os

    os.makedirs('yolov11', exist_ok=True)
    HOME = os.path.join(os.getcwd(), 'yolov11')
    os.chdir(HOME)
    os.getcwd()
```

```
'/content/drive/MyDrive/LAB05-xlav/yolov11'
```

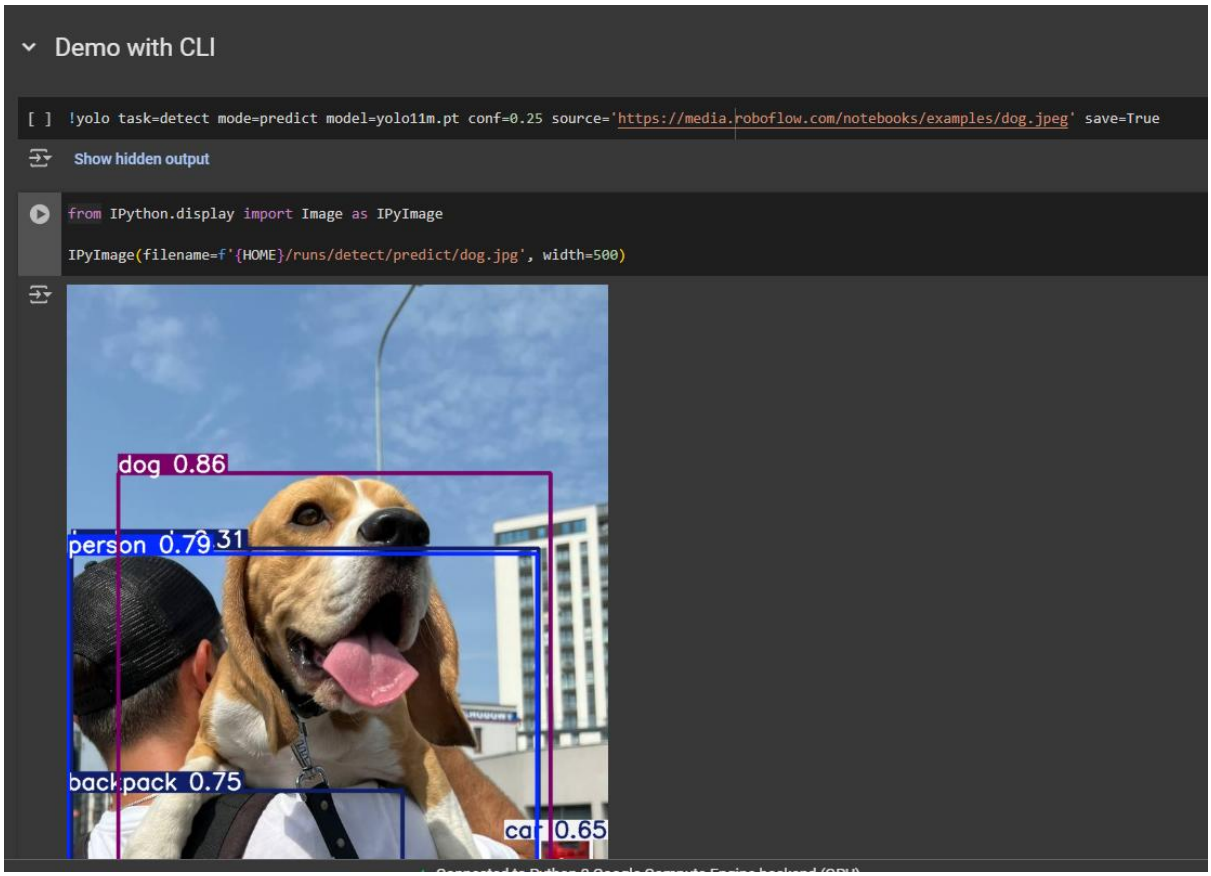
```
%pip install "ultralytics<=8.3.40" supervision roboflow
import ultralytics
ultralytics.checks()
```

```
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 37.8/112.6 GB disk)
```

- **%cd $WORKING_DIR**: Changes the working directory to the specified $WORKING_DIR where the YOLOv11 setup will occur.
- **!rm -rf yolov11**: Removes any existing folder named yolov11 to ensure a clean setup.
- **os.makedirs('yolov11', exist_ok=True)**: Creates a directory named yolov11 if it doesn't already exist.
- **HOME = os.path.join(os.getcwd(), 'yolov11')**: Sets the HOME variable to the path of the yolov11 directory.
- **os.chdir(HOME)**: Changes the working directory to the yolov11 folder.
- **%pip install "ultralytics<=8.3.40" supervision roboflow**: Installs the ultralytics package (which contains YOLOv11), along with the supervision and roboflow libraries. These are necessary for YOLOv11 model tasks and interacting with Roboflow for dataset management.
    - ultralytics<=8.3.40: Specifies the version of ultralytics to install (ensures compatibility with YOLOv11).
    - supervision: A library for managing object detection tasks.
    - roboflow: A library to access datasets from Roboflow, which is useful for managing datasets and labeling.
- **import ultralytics**: Imports the YOLOv11 package from the ultralytics library.

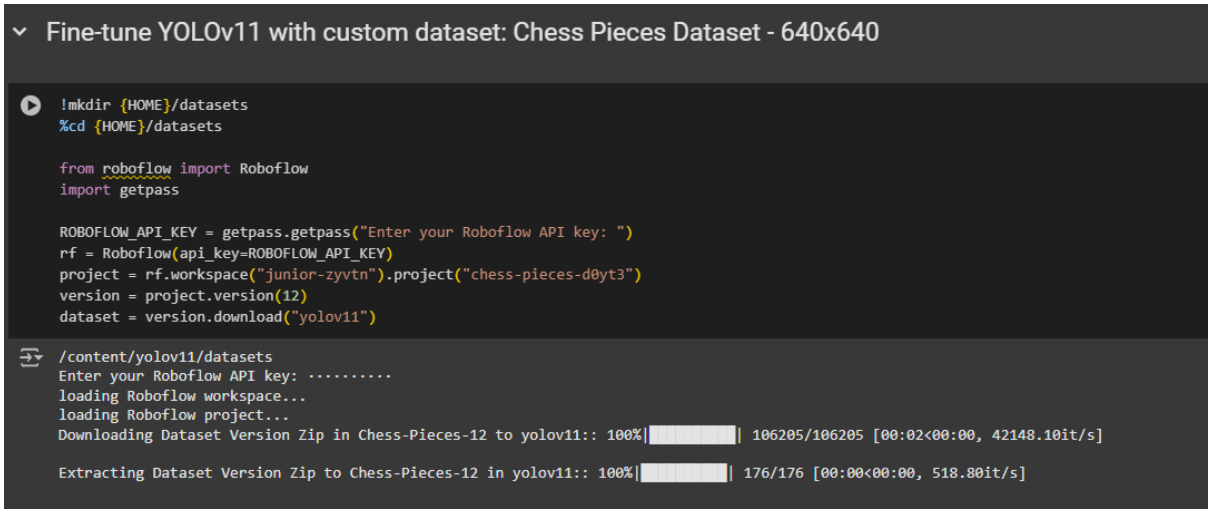
- **ultralytics.checks()**: Checks the installation and configuration of the YOLOv11 package to ensure it's correctly set up.

2. Demo with Command-Line Interface (CLI)



- **!yolo task=detect mode=predict**: Runs the YOLOv11 model in prediction mode for object detection on input images.
    - **task=detect**: Specifies that the task is object detection.
    - **mode=predict**: Indicates that the model will be used to predict objects in images.
    - **model=yolo11m.pt**: Specifies the pre-trained YOLOv11 model file (yolo11m.pt), which is used for predictions.
    - **conf=0.25**: Sets the confidence threshold for detections. Only detections with confidence greater than 0.25 will be considered valid.
    - **source='https://media.roboflow.com/notebooks/examples/dog.jpeg'**: Provides the source image for prediction. This image is from a URL.
    - **save=True**: Saves the resulting image with detected objects to the output directory.
- **from IPython.display import Image as IPyImage**: Imports the Image function from the IPython.display module for displaying images inline within the Jupyter notebook.
- **IPyImage(filename=f'{HOME}/runs/detect/predict/dog.jpg', width=500)**: Displays the resulting image (dog.jpg) from the prediction in the notebook, showing the detected objects.

3. Fine-Tune YOLOv11 with Custom Dataset (Chess Pieces Dataset - 640x640)

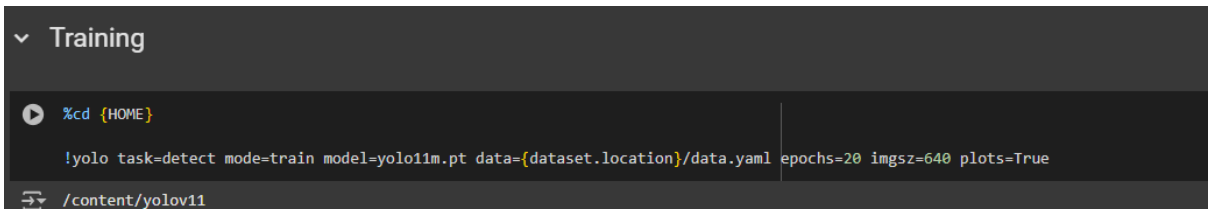


- **!mkdir {HOME}/datasets**: Creates a new directory called datasets within the yolov11 folder to store the dataset.
- **%cd {HOME}/datasets**: Changes the working directory to the newly created datasets folder.

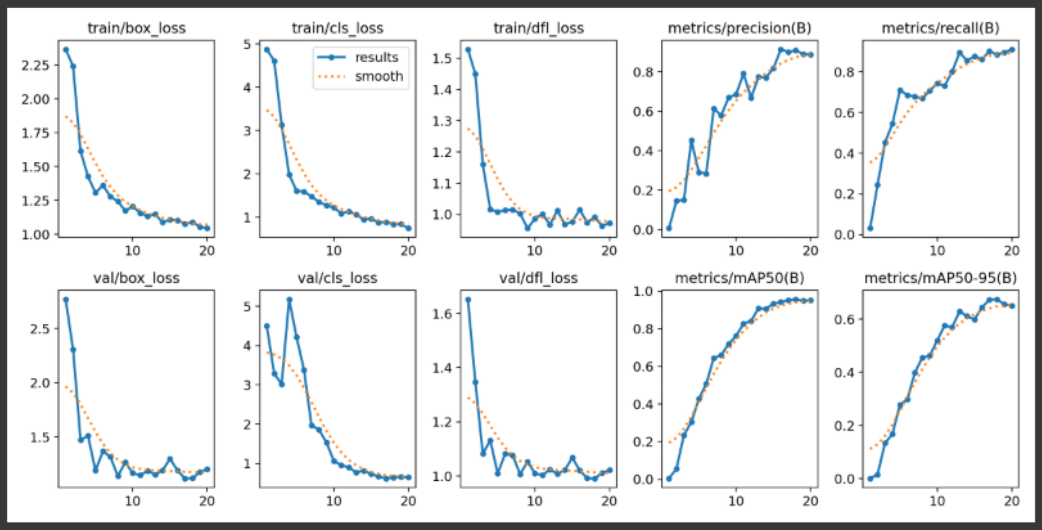
4. Download Dataset from Roboflow

- **from roboflow import Roboflow**: Imports the Roboflow class from the roboflow package to interact with the Roboflow API.
- **import getpass**: Imports the getpass module, which securely prompts for the Roboflow API key.
- **ROBOFLOW_API_KEY = getpass.getpass("Enter your Roboflow API key: ")**: Prompts the user to input their Roboflow API key securely.
- **rf = Roboflow(api_key=ROBOFLOW_API_KEY)**: Initializes the Roboflow API client with the provided API key.
- **project = rf.workspace("junior-zyvtn").project("chess-pieces-d0yt3")**: Accesses the specific project (chess-pieces-d0yt3) within the workspace (junior-zyvtn) on Roboflow.
- **version = project.version(12)**: Downloads version 12 of the dataset from the project.
- **dataset = version.download("yolov11")**: Downloads the dataset in a format compatible with YOLOv11.

5. Train YOLOv11 with Custom Dataset

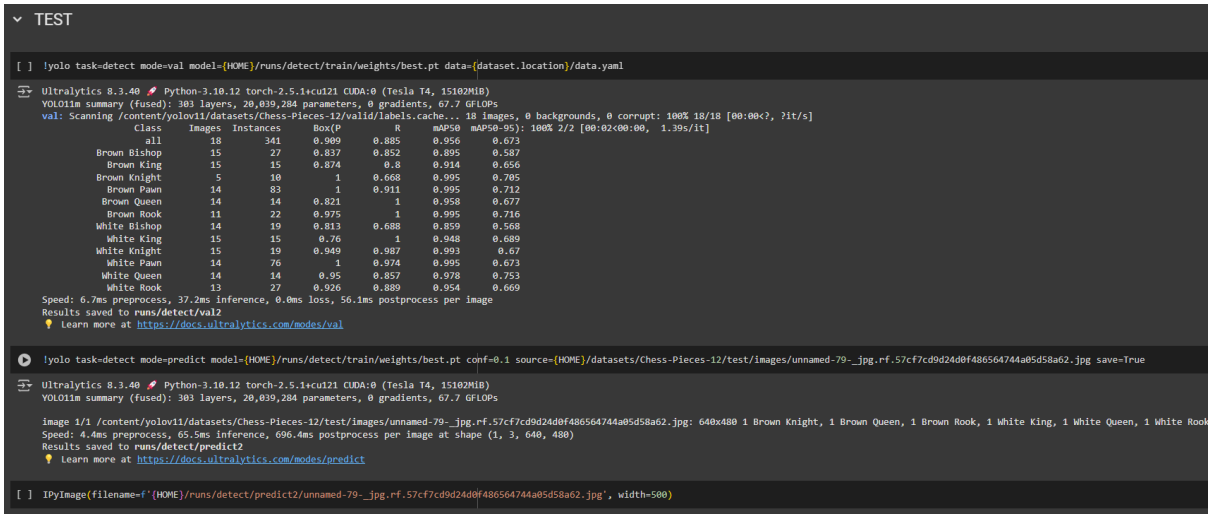


```
from IPython.display import Image as IPyImage

IPyImage(filename=f'{HOME}/runs/detect/train/results.png', width=800)
```



- **%cd {HOME}**: Changes the working directory back to the yolov11 folder.
- **!yolo task=detect mode=train**: Starts the training process for YOLOv11 with the following parameters:
    - **task=detect**: Specifies that the task is object detection.
    - **mode=train**: Runs the model in training mode.
    - **model=yolo11m.pt**: Specifies the pre-trained YOLOv11 model file (yolo11m.pt), which will be fine-tuned with the custom dataset.
    - **data={dataset.location}/data.yaml**: Specifies the path to the data.yaml file, which contains the dataset configuration (e.g., class names and paths).
    - **epochs=20**: Sets the number of epochs for training (20 epochs).
    - **imgsz=640**: Specifies the input image size (640x640).
    - **plots=True**: Enables plotting of training results such as loss curves.
- **IPyImage(filename=f'{HOME}/runs/detect/train/results.png', width=800)**: Displays the training results in the notebook (e.g., loss curve) after training.

6. Test YOLOv11



- **!yolo task=detect mode=val**: Runs validation on the trained model:
    - **task=detect**: Object detection task.
    - **mode=val**: Validation mode.
    - **model={HOME}/runs/detect/train/weights/best.pt**: Specifies the path to the best weights after training.
    - **data={dataset.location}/data.yaml**: Specifies the dataset configuration file for validation.
- **!yolo task=detect mode=predict**: Runs prediction on a test image:
    - **task=detect**: Object detection task.
    - **mode=predict**: Prediction mode.
    - **model={HOME}/runs/detect/train/weights/best.pt**: Uses the best weights from training.
    - **conf=0.1**: Sets the confidence threshold for detection.
    - **source={HOME}/datasets/Chess-Pieces-12/test/images/unnamed-79-_jpg.rf.57cf7cd9d24d0f486564744a05d58a62.jpg**: Specifies the source image for prediction.
    - **save=True**: Saves the resulting predicted image.
- **IPyImage(filename=f'{HOME}/runs/detect/predict2/unnamed-79-_jpg.rf.57cf7cd9d24d0f486564744a05d58a62.jpg', width=500)**: Displays the predicted image with detected objects.

3. Results and evaluations

A. Results:

a) Demo:

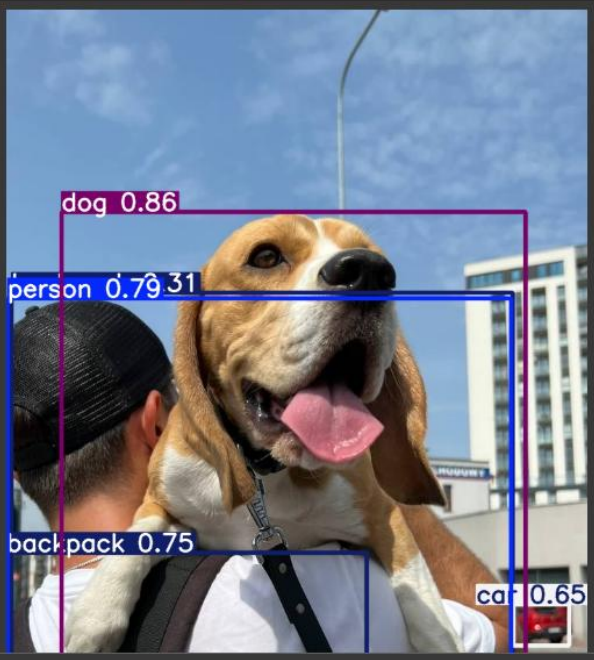
+ Code + Text

Demo with CLI

```
!yolo task=detect mode=predict model=yolo11m.pt conf=0.25 source='https://media.roboflow.com/notebooks/examples/dog.jpeg' save=Tr
```

Show hidden output

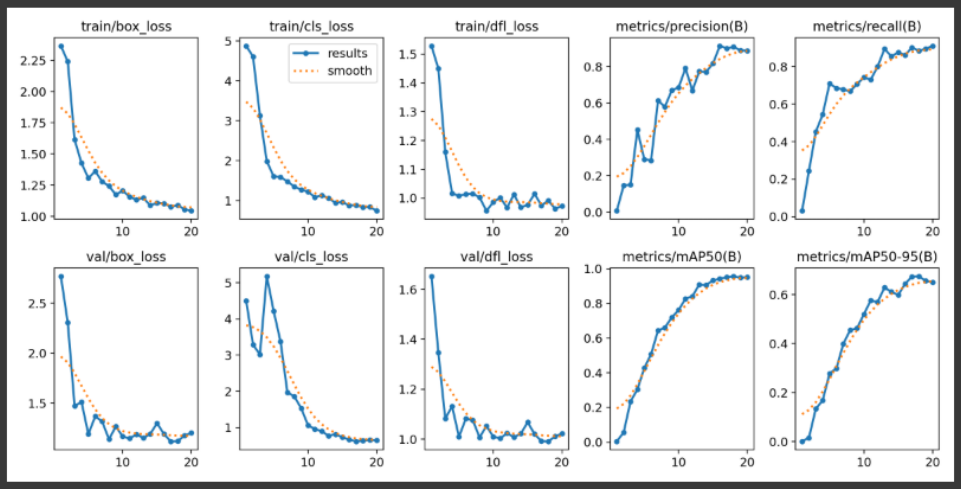
```
from IPython.display import Image as IPyImage

IPyImage(filename=f'{HOME}/runs/detect/predict/dog.jpg', width=500)
```



b) Training (epochs = 20)

```
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.73it/s]
                   all         18        341      0.901      0.903      0.952      0.674

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/20      8.68G      1.091     0.8354     0.9911        138        640: 100% 4/4 [00:02<00:00,  1.97it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.13it/s]
                   all         18        341      0.909      0.885      0.956      0.674

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      8.61G      1.057     0.8402     0.9631        116        640: 100% 4/4 [00:02<00:00,  1.96it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.07it/s]
                   all         18        341      0.889      0.897       0.95      0.656

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      8.71G      1.045     0.7434     0.9722        108        640: 100% 4/4 [00:02<00:00,  1.88it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.57it/s]
                   all         18        341      0.888      0.912      0.952       0.65

20 epochs completed in 0.032 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 40.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 40.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11m summary (fused): 303 layers, 20,039,284 parameters, 0 gradients, 67.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.92it/s]
                   all         18        341      0.909      0.886      0.956      0.673
           Brown Bishop         15         27      0.838      0.852      0.894      0.573
             Brown King         15         15      0.875        0.8      0.914      0.639
           Brown Knight          5         10          1      0.668      0.995       0.72
             Brown Pawn         14         83          1      0.911      0.995      0.717
            Brown Queen         14         14      0.821          1      0.958      0.694
             Brown Rook         11         22      0.976          1      0.995      0.705
           White Bishop         14         19      0.813      0.688      0.859      0.568
             White King         15         15      0.759          1      0.948      0.695
           White Knight         15         19      0.949      0.987      0.993       0.67
             White Pawn         14         76      0.999      0.974      0.995      0.674
            White Queen         14         14       0.95      0.857      0.978      0.756
             White Rook         13         27      0.926      0.889      0.953      0.669
Speed: 0.2ms preprocess, 10.0ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

```
from IPython.display import Image as IPyImage

IPyImage(filename=f'{HOME}/runs/detect/train/results.png', width=800)
```



Training

```
%cd {HOME}

!yolo task=detect mode=train model=yolo11m.pt data={dataset.location}/data.yaml epochs=20 imgsz=640 plots=True
```

```
/content/yolov11
New https://pypi.org/project/ultralytics/8.3.54 available 😃 Update with 'pip install -U ultralytics'
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolo11m.pt, data=/content/yolov11/datasets/Chess-Pieces-12/data.yaml, epochs=20, time=None, patience=100, ba
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 21.2MB/s]
Overriding model.yaml nc=80 with nc=12

                   from  n    params  module                                       arguments
  0                  -1  1      1856  ultralytics.nn.modules.conv.Conv             [3, 64, 3, 2]
  1                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  2                  -1  1    111872  ultralytics.nn.modules.block.C3k2            [128, 256, 1, True, 0.25]
  3                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
  4                  -1  1    444928  ultralytics.nn.modules.block.C3k2            [256, 512, 1, True, 0.25]
  5                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
  6                  -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  7                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
  8                  -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1    990976  ultralytics.nn.modules.block.C2PSA           [512, 512, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
```

c) Testing

I choose 1 random picture in **run/detect/val2 folder** as below to do testing

TEST

```
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

```
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11m summary (fused): 303 layers, 20,039,284 parameters, 0 gradients, 67.7 GFLOPs
val: Scanning /content/yolov11/datasets/Chess-Pieces-12/valid/labels.cache... 18 images, 0 backgrounds, 0 corrupt: 100% 18/18 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:02<00:00,  1.39s/it]
                   all         18        341      0.909      0.885      0.956      0.673
           Brown Bishop         15         27      0.837      0.852      0.895      0.587
             Brown King         15         15      0.874        0.8      0.914      0.656
           Brown Knight          5         10          1      0.668      0.995      0.705
             Brown Pawn         14         83          1      0.911      0.995      0.712
            Brown Queen         14         14      0.821          1      0.958      0.677
             Brown Rook         11         22      0.975          1      0.995      0.716
           White Bishop         14         19      0.813      0.688      0.859      0.568
             White King         15         15       0.76          1      0.948      0.689
           White Knight         15         19      0.949      0.987      0.993       0.67
             White Pawn         14         76          1      0.974      0.995      0.673
            White Queen         14         14       0.95      0.857      0.978      0.753
             White Rook         13         27      0.926      0.889      0.954      0.669
Speed: 6.7ms preprocess, 37.2ms inference, 0.0ms loss, 56.1ms postprocess per image
Results saved to runs/detect/val2
💡 Learn more at https://docs.ultralytics.com/modes/val
```

I did testing in the valid folder because in the yolov11 library it is the default mode. Firstly It will run validation on the valid folder image and if this folder doesn't exists, it will run on the test images folder. If you want to run on the test folder, you can delete the valid folder.

C. Evaluations

YOLOv11 Results:

Validating runs/detect/train/weights/best.pt...

Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

YOLO11m summary (fused): 303 layers, 20,039,284 parameters, 0 gradients, 67.7 GFLOPs

Class Images Instances Box(P R mAP50 mAP50-95): 100% 1/1 [00:00<00:00, 2.92it/s]

all	18	341	0.909	0.886	0.956	0.673
Brown Bishop	15	27	0.838	0.852	0.894	0.573
Brown King	15	15	0.875	0.8	0.914	0.639
Brown Knight	5	10	1	0.668	0.995	0.72
Brown Pawn	14	83	1	0.911	0.995	0.717
Brown Queen	14	14	0.821	1	0.958	0.694

Brown Rook	11	22	0.976	1	0.995	0.705
White Bishop	14	19	0.813	0.688	0.859	0.568
White King	15	15	0.759	1	0.948	0.695
White Knight	15	19	0.949	0.987	0.993	0.67
White Pawn	14	76	0.999	0.974	0.995	0.674
White Queen	14	14	0.95	0.857	0.978	0.756
White Rook	13	27	0.926	0.889	0.953	0.669

Speed: 0.2ms preprocess, 10.0ms inference, 0.0ms loss, 1.4ms postprocess per image

Results saved to runs/detect/train

The results from YOLOv11 show much better performance, with mAP50 at 0.956 and recall values higher than 0.8 for most classes. Additionally, YOLOv11 demonstrated fast processing times (10ms for inference per image), and the results were achieved quickly, making it a more efficient model for this dataset.