

Training and Testing YOLOv8 with Your Data

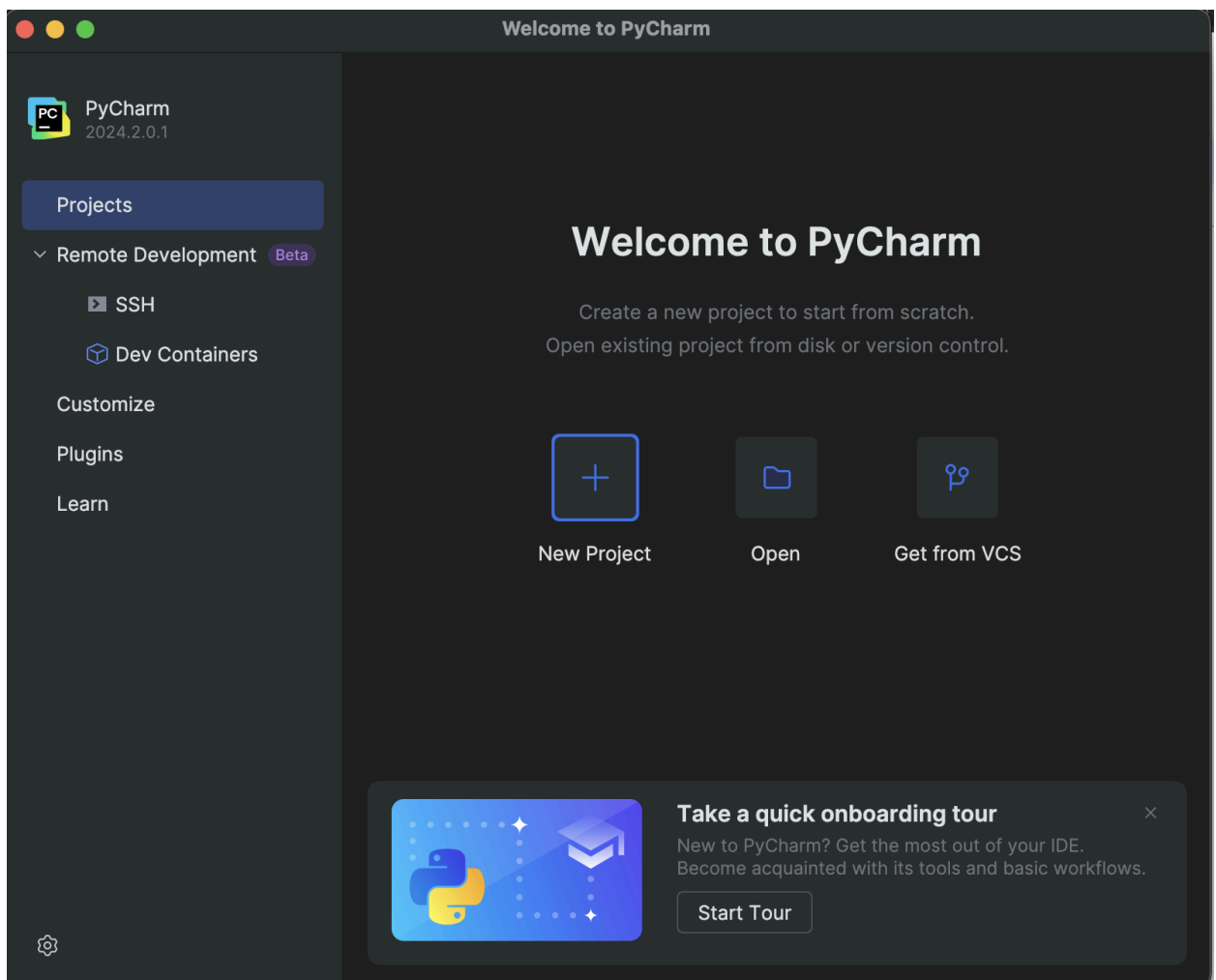
Note: This tutorial provides a quick test to determine if the model functions correctly with your images. For a more comprehensive approach, further steps and refinements will be necessary.

Prerequisites

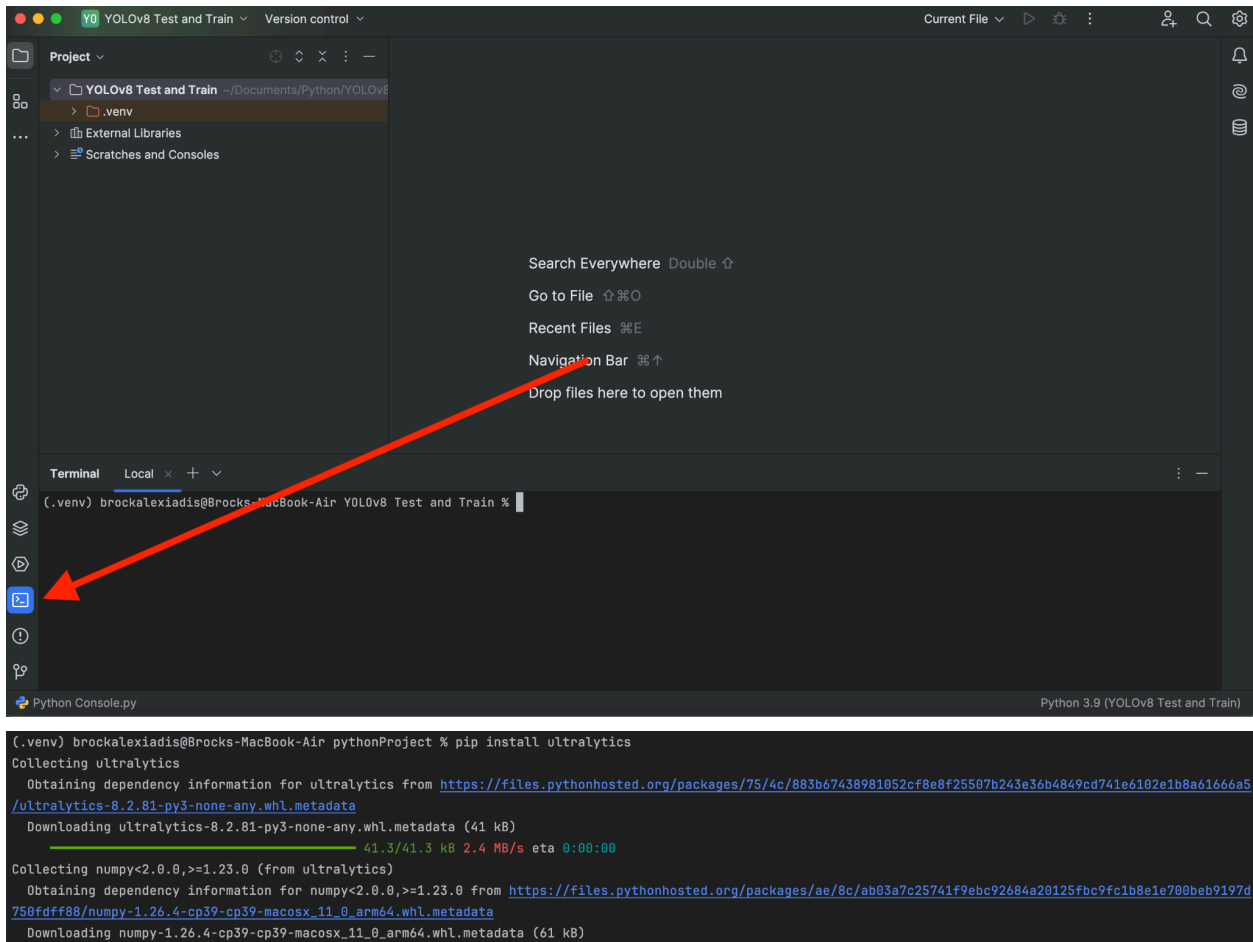
To follow this tutorial, you'll need software capable of running Python code. This example uses PyCharm, but alternatives like Visual Studio Code will also suffice.

Step 1: Set Up Your Project

1. **Create a New Project:** Start by creating a new project in your chosen software and name it as desired.

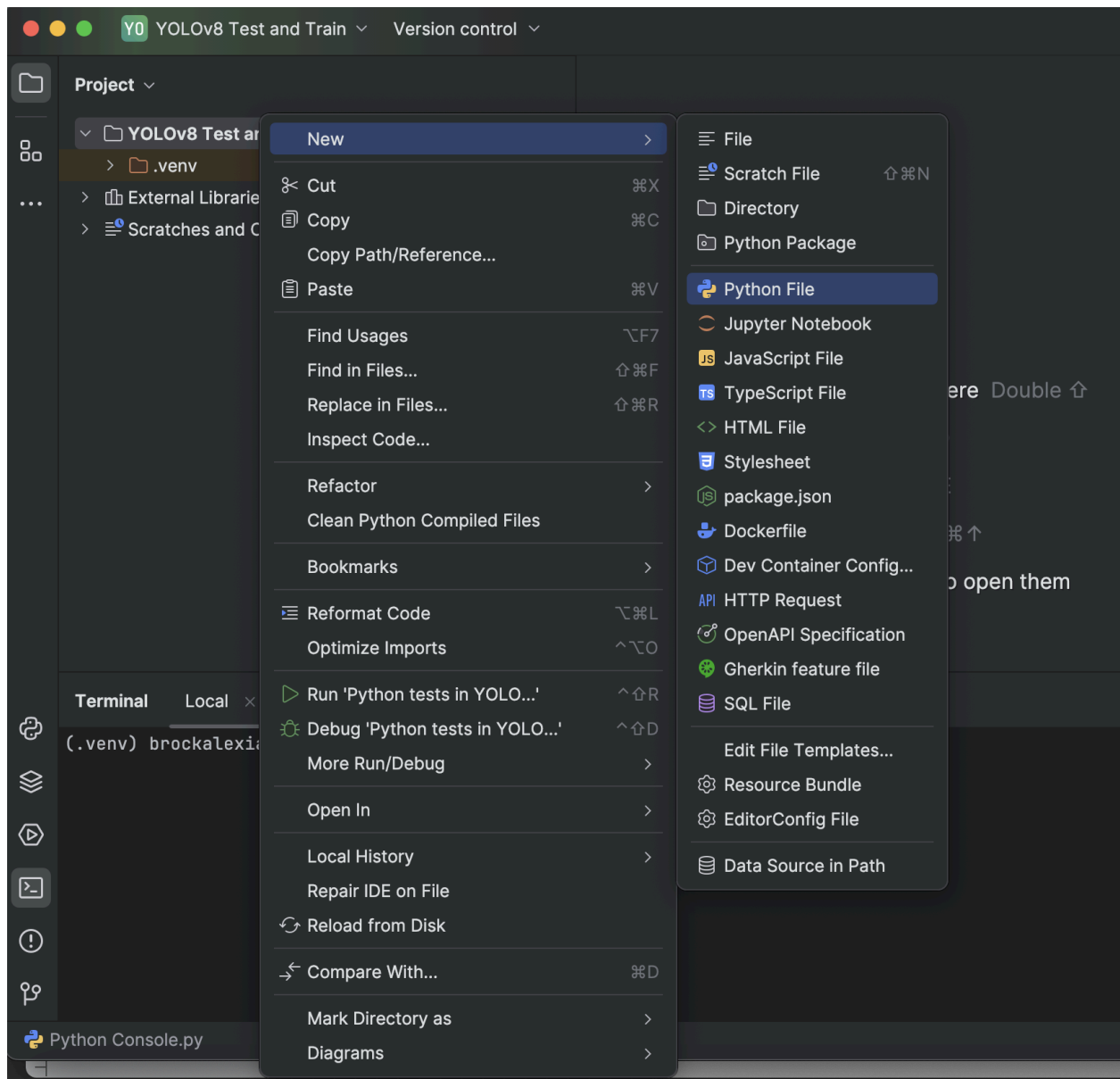


2. **Install Dependencies:** Before proceeding, install the required libraries by opening the terminal and running the following command: `pip install ultralytics`



Step 2: Create and Modify the Python Script

1. **Create a Python File:** In your project folder, create a new Python file by navigating to Right Click on Project Folder > New > Python File, and name it `main.py`.



2. **Obtain the Training Script:** Copy the necessary Python code from the [Ultralytics GitHub repository](#) for training your model. Paste it into the `main.py` file.

```
main.py ×  
1  from ultralytics import YOLO  
2  
3  # Load a model  
4  model = YOLO("yolov8n.yaml") # build a new model from scratch  
5  model = YOLO("yolov8n.pt") # load a pretrained model (recommended for training)  
6  
7  # Use the model  
8  model.train(data="coco8.yaml", epochs=3) # train the model  
9  metrics = model.val() # evaluate model performance on the validation set  
10 results = model("https://ultralytics.com/images/bus.jpg") # predict on an image  
11 path = model.export(format="onnx") # export the model to ONNX format
```

3. **Modify the Script:**

- a. Remove line 5 to ensure you're building a new model.
- b. Delete lines 10 and 11, as line 10's functionality will be addressed later.

```
main.py ×  
1  from ultralytics import YOLO  
2  
3  # Load a model  
4  model = YOLO("yolov8n.yaml") # build a new model from scratch  
5  
6  # Use the model  
7  model.train(data="coco8.yaml", epochs=3) # train the model  
8  metrics = model.val() # evaluate model performance on the validation set
```

Step 3: Configure Your Dataset

1. **Create a Configuration File:** To specify the locations of your images and labels, create a new configuration file by selecting **Right Click on Project Folder > New > File** and name it with a `.yaml` extension, for example, `config.yaml`.

```
les
5
6 # Use the model
7 model.train(data="coco8.yaml", epochs=3) # train the m
8 metrics = model.val() # evaluate model performance on
```

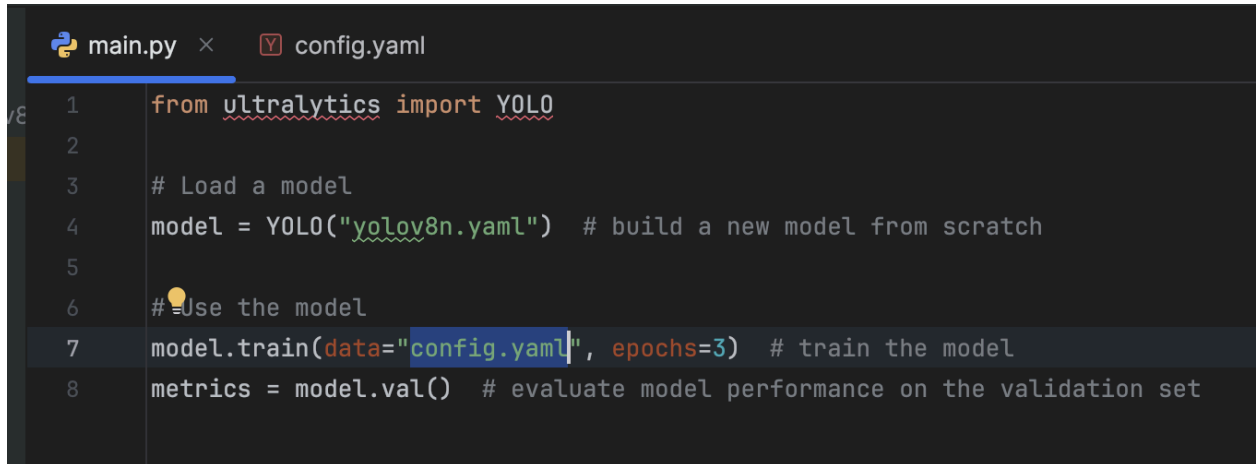
New File

config.yaml|

2. **Define Dataset Parameters:** In the `.yaml` file, specify the paths to your training and validation datasets, as well as the classes used for object detection. The configuration should include paths to your images and labels.

```
main.py  predict.py  config.yaml x  predict_object.py
1
2 path: /Users/brockalexidis/Documents/University/Team Project (B)/Unknown Vehicles #full path to the folder that your train and val folders are within
3 train: train #folder of train images and labels
4 val: val #folder of validation images and labels
5
6 #Different vehicle classes
7 names:
8   0: Sedan
9   1: Wagon
10  2: 4WD
11  3: Utility
12  4: Light_van
13  5: Bicycle
14  6: Motorcycle
15  7: Trailer
16  8: Caravan
17  9: Boat
18  10: Two_Axle_Truck
19  11: Two_Axle_Bus
20  12: Three_Axle_Truck
21  13: Three_Axle_Bus
22  14: Four_Axle_Truck
```

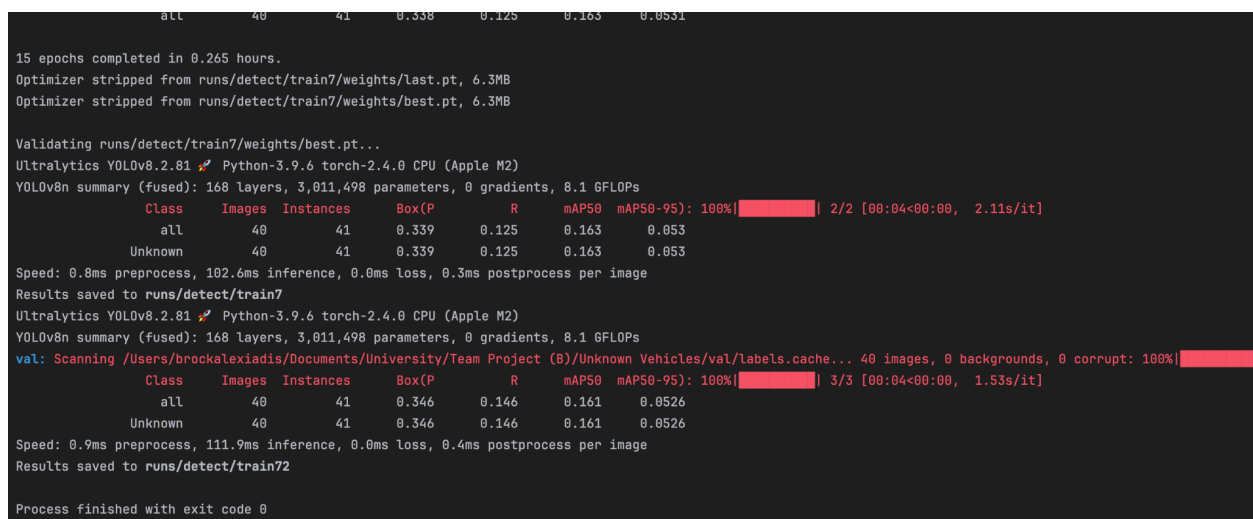
3. **Update the Training Script:** In `main.py`, update the dataset reference by replacing `'coco8.yaml'` with the name of your configuration file, e.g., `'config.yaml'`.



```
main.py x config.yaml
1 from ultralytics import YOLO
2
3 # Load a model
4 model = YOLO("yolov8n.yaml") # build a new model from scratch
5
6 # Use the model
7 model.train(data="config.yaml", epochs=3) # train the model
8 metrics = model.val() # evaluate model performance on the validation set
```

Step 4: Train the Model

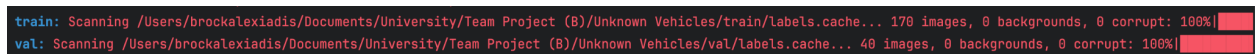
1. **Run the Training Script:** Execute the `main.py` script to start training the model. Pay attention to the output, particularly the line indicating where results are saved, such as `'Results saved to runs/detect/train72'`. This path may vary.



```
all 40 41 0.338 0.125 0.163 0.0531
15 epochs completed in 0.265 hours.
Optimizer stripped from runs/detect/train7/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train7/weights/best.pt, 6.3MB

Validating runs/detect/train7/weights/best.pt...
Ultralytics YOLOv8.2.81 Python-3.9.6 torch-2.4.0 CPU (Apple M2)
YOLOv8n summary (fused): 168 layers, 3,011,498 parameters, 0 gradients, 8.1 GFLOPs
Class Images Instances Box(P) R mAP50 mAP50-95: 100%| 2/2 [00:04<00:00, 2.11s/it]
all 40 41 0.339 0.125 0.163 0.053
Unknown 40 41 0.339 0.125 0.163 0.053
Speed: 0.8ms preprocess, 102.6ms inference, 0.0ms loss, 0.3ms postprocess per image
Results saved to runs/detect/train7
Ultralytics YOLOv8.2.81 Python-3.9.6 torch-2.4.0 CPU (Apple M2)
YOLOv8n summary (fused): 168 layers, 3,011,498 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /Users/brockalexiadis/Documents/University/Team Project (B)/Unknown Vehicles/val/labels.cache... 40 images, 0 backgrounds, 0 corrupt: 100%|
Class Images Instances Box(P) R mAP50 mAP50-95: 100%| 3/3 [00:04<00:00, 1.53s/it]
all 40 41 0.346 0.146 0.161 0.0526
Unknown 40 41 0.346 0.146 0.161 0.0526
Speed: 0.9ms preprocess, 111.9ms inference, 0.0ms loss, 0.4ms postprocess per image
Results saved to runs/detect/train72
Process finished with exit code 0
```

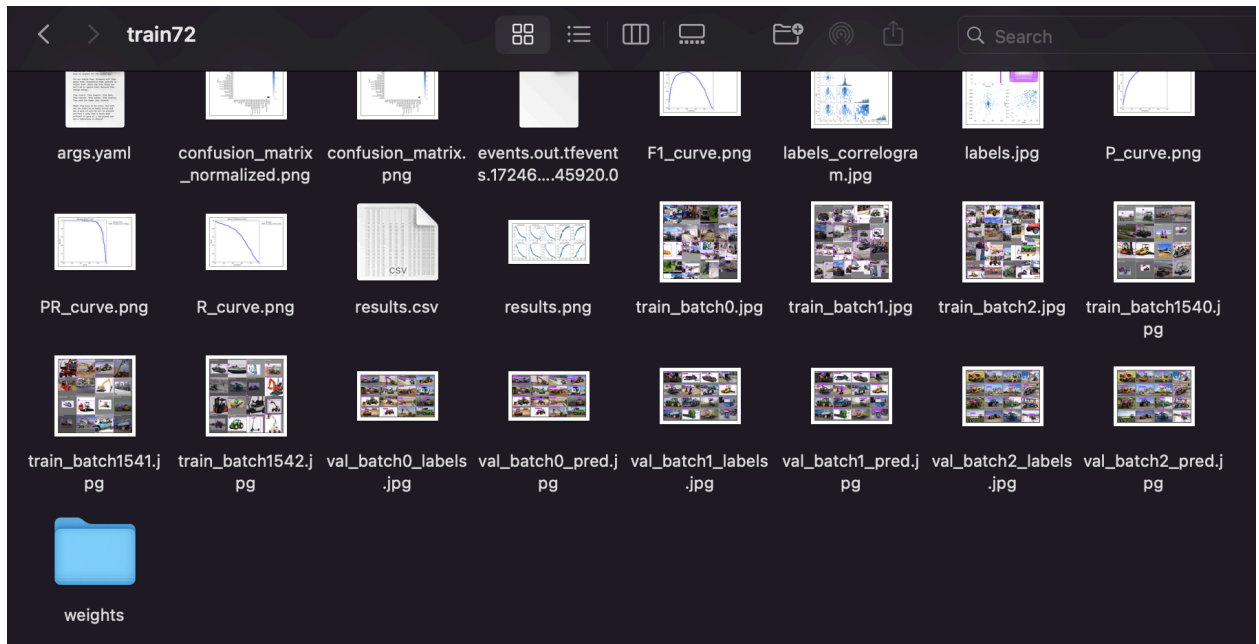
2. **Verify Training Progress:** Optionally, check that all images are being utilized by reviewing the training output for entries showing the count of images in the train and validation sets (e.g., 170 images in train set, 40 in val set).



```
train: Scanning /Users/brockalexiadis/Documents/University/Team Project (B)/Unknown Vehicles/train/labels.cache... 170 images, 0 backgrounds, 0 corrupt: 100%|
val: Scanning /Users/brockalexiadis/Documents/University/Team Project (B)/Unknown Vehicles/val/labels.cache... 40 images, 0 backgrounds, 0 corrupt: 100%|
```

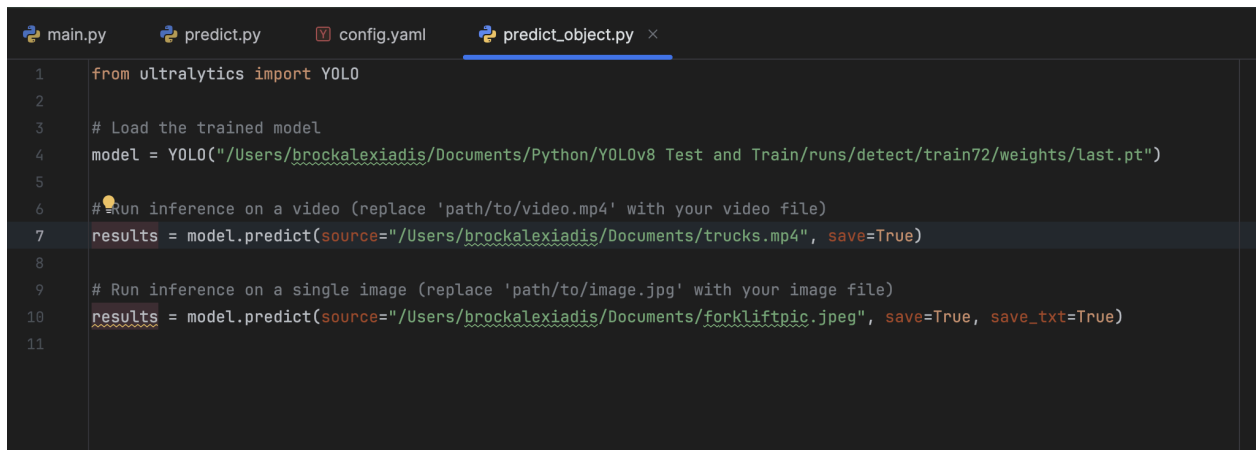
3. You can see all 170 images from the train set have been used and all 40 from the val set

4. **Analyze Results:** The output will include metrics such as confusion matrices and graphs, which can be used to assess the model's performance.



Step 5: Test the Model on Unseen Data

1. **Prepare for Testing:** The trained algorithm is saved in the 'weights' folder, with files named `best.pt` and `last.pt`. You can use either; this tutorial uses `last.pt`.
2. **Create a Prediction Script:** Create another Python file, named `predict_object.py`, to test the algorithm on unseen images.



3. **Configure the Prediction Script:**
 - a. Specify the path to your trained model on line 4, ensuring it points to `'last.pt'`.
 - b. If testing on a video, use line 7. For images, comment out line 7 with a hashtag (`#`) and use line 10.

```

1  from ultralytics import YOLO
2
3  # Load the trained model
4  model = YOLO("/Users/brockalexiadis/Documents/Python/YOLOv8 Test and Train/runs/detect/train72/weights/last.pt")
5
6  # Run inference on a video (replace 'path/to/video.mp4' with your video file)
7  #results = model.predict(source="/Users/brockalexiadis/Documents/trucks.mp4", save=True)
8
9  # Run inference on a single image (replace 'path/to/image.jpg' with your image file)
10 results = model.predict(source="/Users/brockalexiadis/Documents/forkliftpic.jpeg", save=True, save_txt=True)
11

```

4. **Set the Test Image Path:** Update the `source=...` section in the script to point to your test image (e.g., located in the `Documents` folder). This is our test image:



Step 6: Run the Prediction

1. **Execute the Prediction Script:** Run `predict_object.py`. The prediction results will be saved to a path similar to `'runs/detect/predict9'`.

```

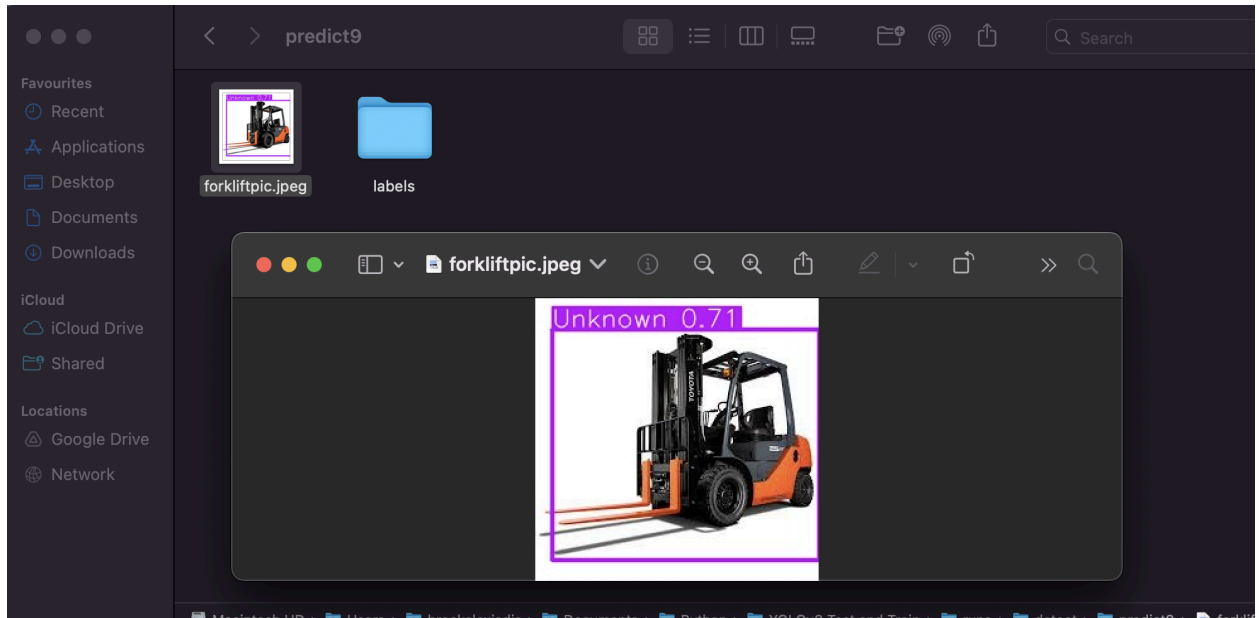
"/Users/brockalexiadis/Documents/Python/YOLOv8 Test and Train/.venv/bin/python" /Users/brockalexiadis/Documents/Python/YOLOv8 Test and Train/predict_object.py

image 1/1 /Users/brockalexiadis/Documents/forkliftpic.jpeg: 640x640 1 Unknown, 66.2ms
Speed: 1.7ms preprocess, 66.2ms inference, 0.6ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict9
1 label saved to runs/detect/predict9/labels

Process finished with exit code 0
|

```


2. **Review Results:** Confirm that the model has correctly labeled the unseen data, such as identifying an object as 'Unknown'.



3. We have successfully tested the algorithm on unseen data