Training and Testing YOLOv8 with Your Data

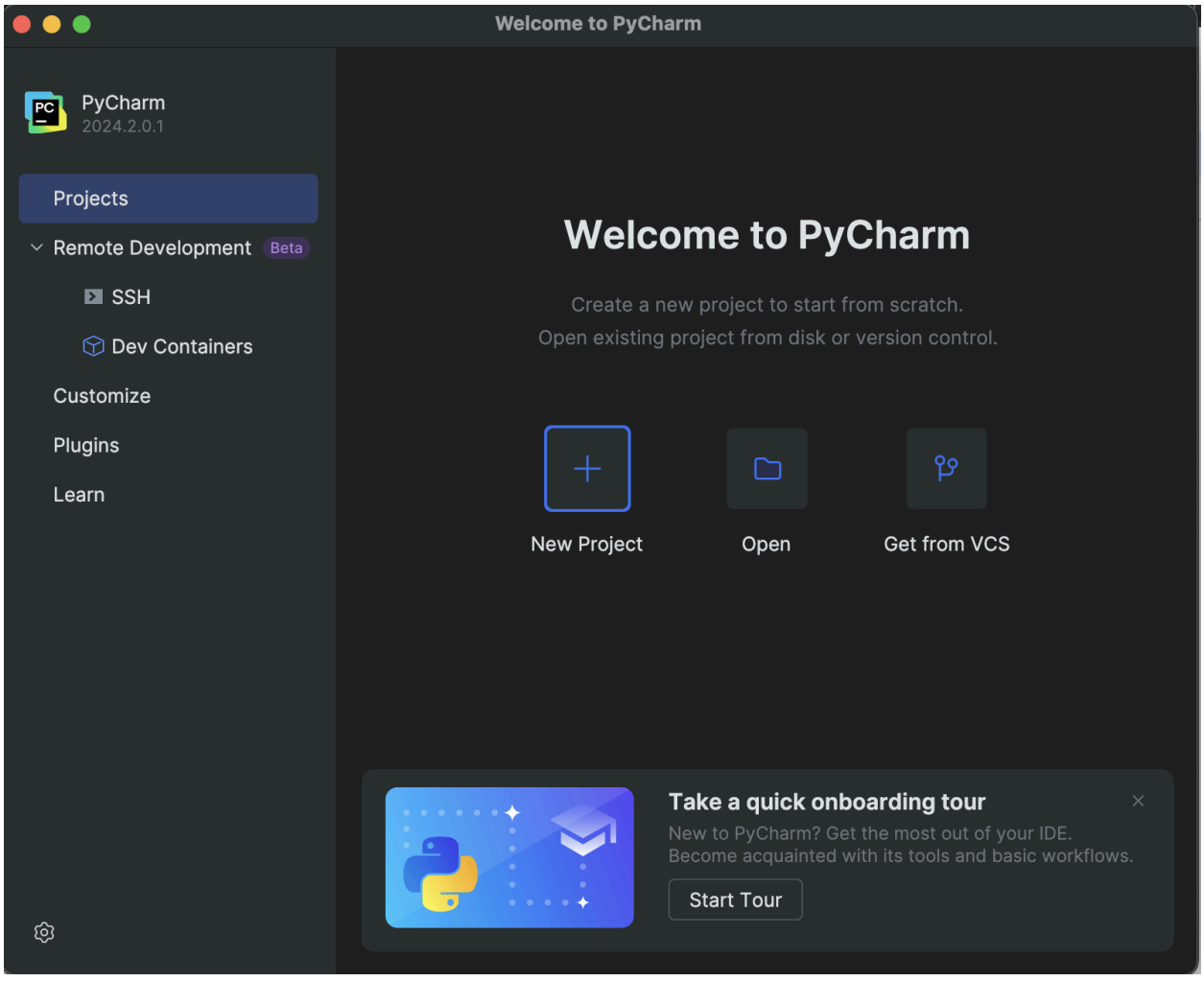
Note: This tutorial provides a quick test to determine if the model functions correctly with your images. For a more comprehensive approach, further steps and refinements will be necessary.

Prerequisites

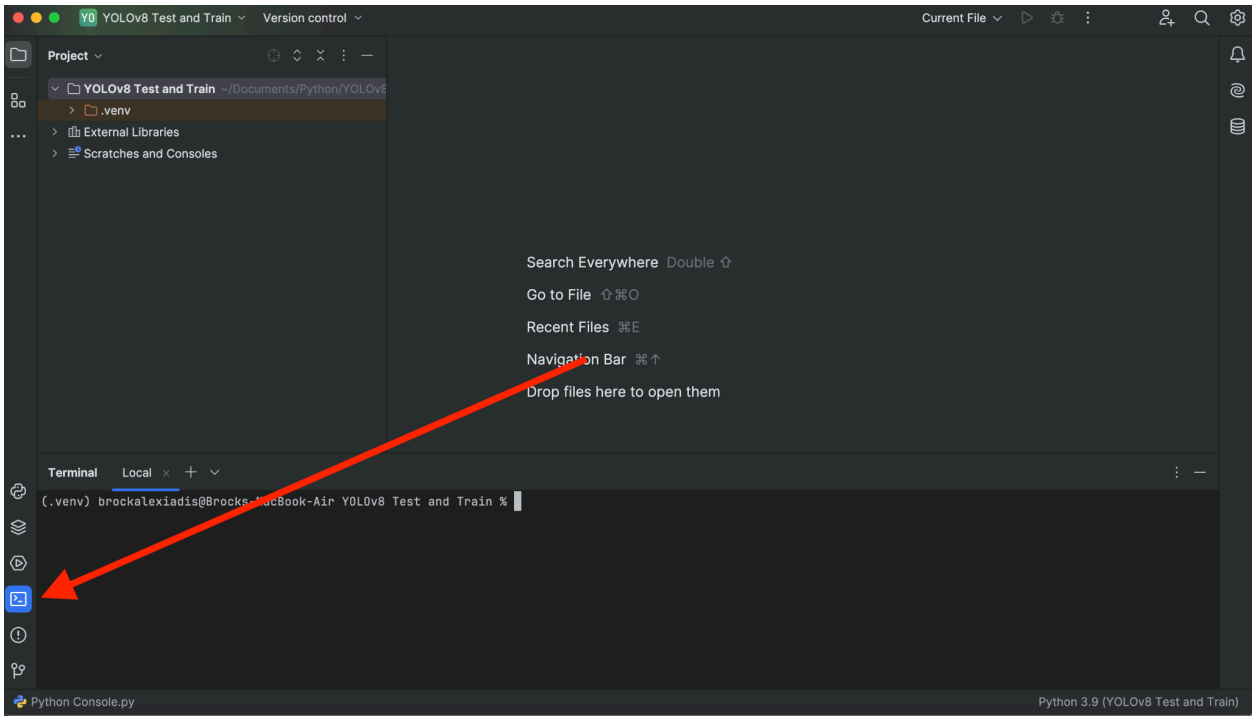
To follow this tutorial, you'll need software capable of running Python code. This example uses PyCharm, but alternatives like Visual Studio Code will also suffice.

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.



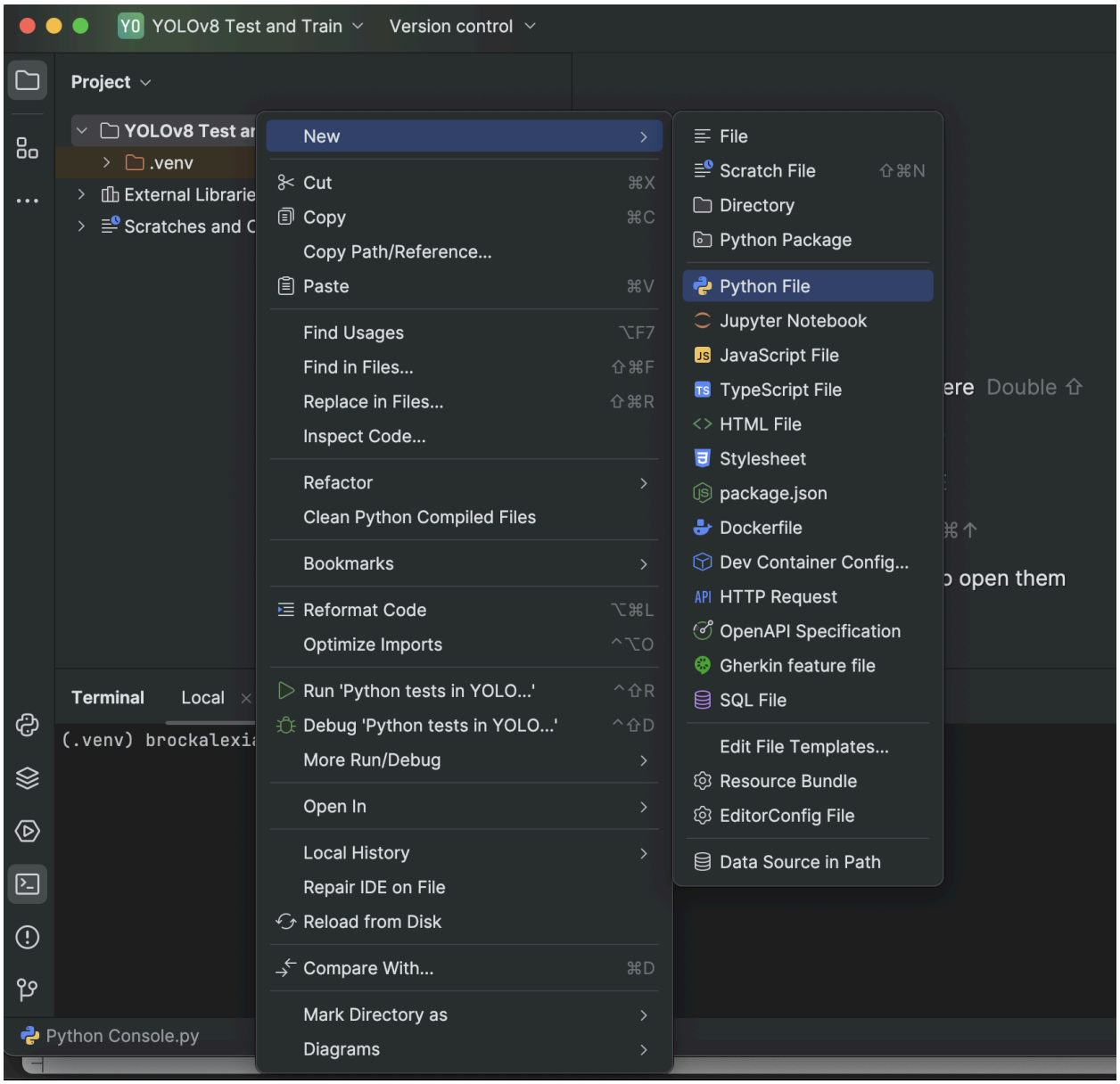
2. **Install Dependencies:** Before proceeding, install the required libraries by opening the terminal and running the following command: `pip install ultralytics`



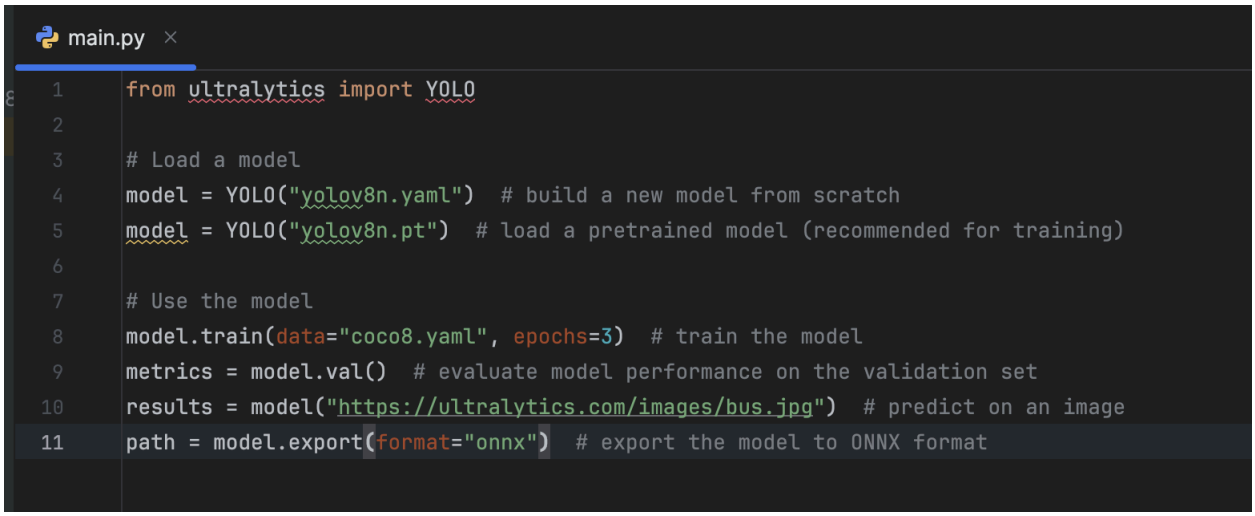

```
(.venv) brockalexiadis@Brocks-MacBook-Air pythonProject % pip install ultralytics
Collecting ultralytics
  Obtaining dependency information for ultralytics from https://files.pythonhosted.org/packages/75/4c/883b6743981052cf8e8f25507b243e36b4849cd741e6102e1b8a61666a5/ultralytics-8.2.81-py3-none-any.whl.metadata
  Downloading ultralytics-8.2.81-py3-none-any.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.3/41.3 kB 2.4 MB/s eta 0:00:00
Collecting numpy<2.0.0,>=1.23.0 (from ultralytics)
  Obtaining dependency information for numpy<2.0.0,>=1.23.0 from https://files.pythonhosted.org/packages/ae/8c/ab03a7c25741f9ebc92684a20125fbc9fc1b8e1e700beb9197d750fdff88/numpy-1.26.4-cp39-cp39-macosx_11_0_arm64.whl.metadata
  Downloading numpy-1.26.4-cp39-cp39-macosx_11_0_arm64.whl.metadata (61 kB)
```

Step 2: Create and Modify the Python Script

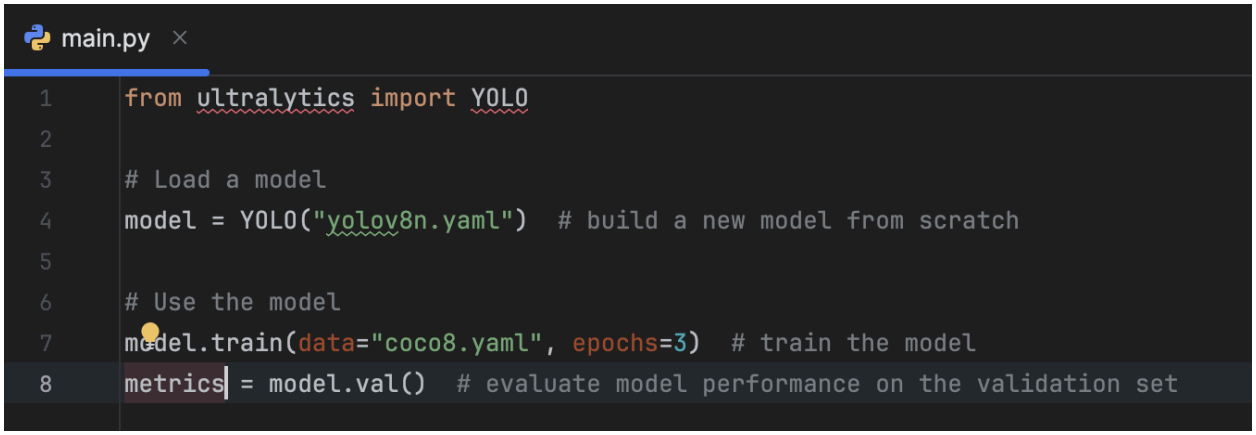
1. **Create a Python File:** In your project folder, create a new Python file by navigating to Right Click on Project Folder > New > Python File, and name it `main.py`.


YOLOv8 Test and Train
Version control
Project
YOLOv8 Test ar
.venv
External Librarie
Scratches and C
New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Inspect Code...
Refactor
Clean Python Compiled Files
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Run 'Python tests in YOLO...' ^⇧R
Debug 'Python tests in YOLO...' ^⇧D
More Run/Debug
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Diagrams
File
Scratch File ⇧⌘N
Directory
Python Package
Python File
Jupyter Notebook
JavaScript File
TypeScript File
HTML File
Stylesheet
package.json
Dockerfile
Dev Container Config...
HTTP Request
OpenAPI Specification
Gherkin feature file
SQL File
Edit File Templates...
Resource Bundle
EditorConfig File
Data Source in Path
Terminal
Local
(.venv) brockalexi
Python Console.py
ere Double ⇧
open them

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.

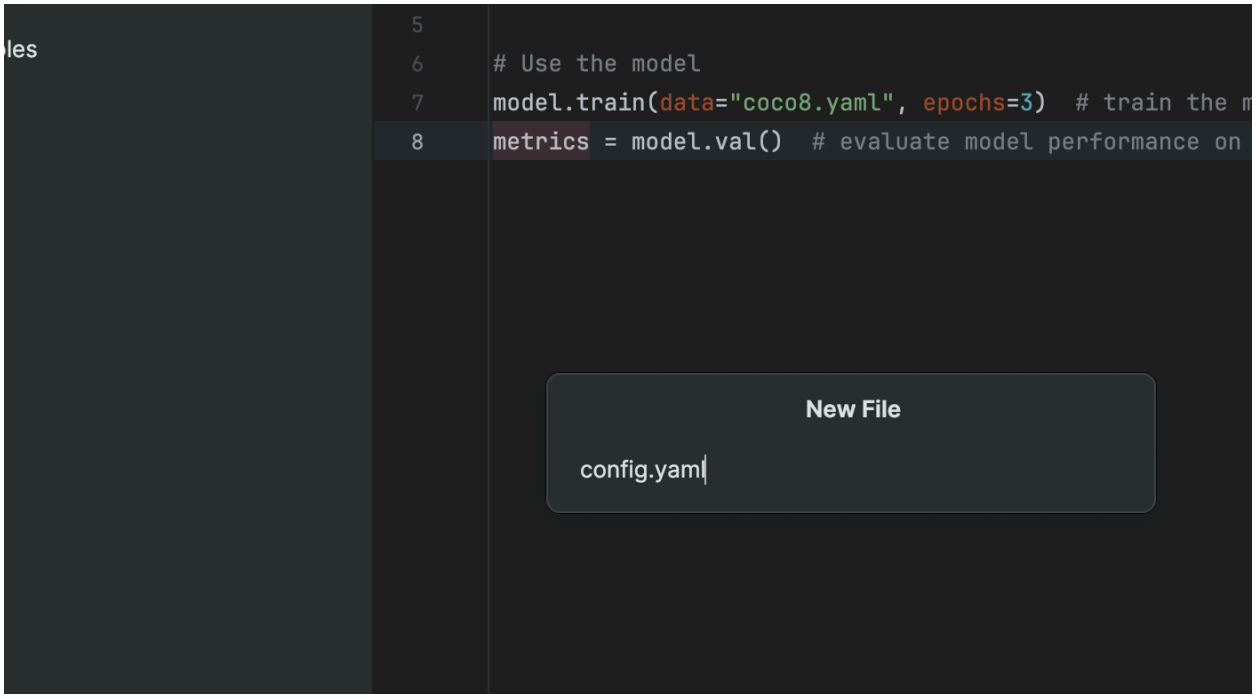


3. **Modify the Script:**
    a. Remove line 5 to ensure you're building a new model.
    b. Delete lines 10 and 11, as line 10's functionality will be addressed later.



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`.

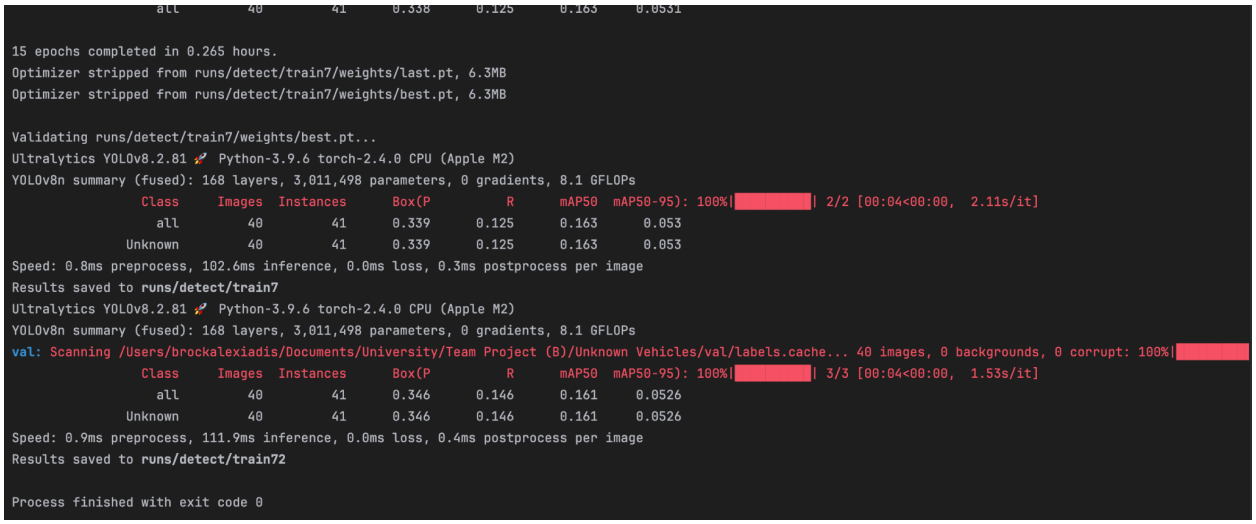


2. **Define Dataset Parameters:** In the `.yaml` file, specify the paths to your training and validation datasets, as well as the classes used for object detection. The configuration should include paths to your images and labels.

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'`.

Step 4: Train the Model

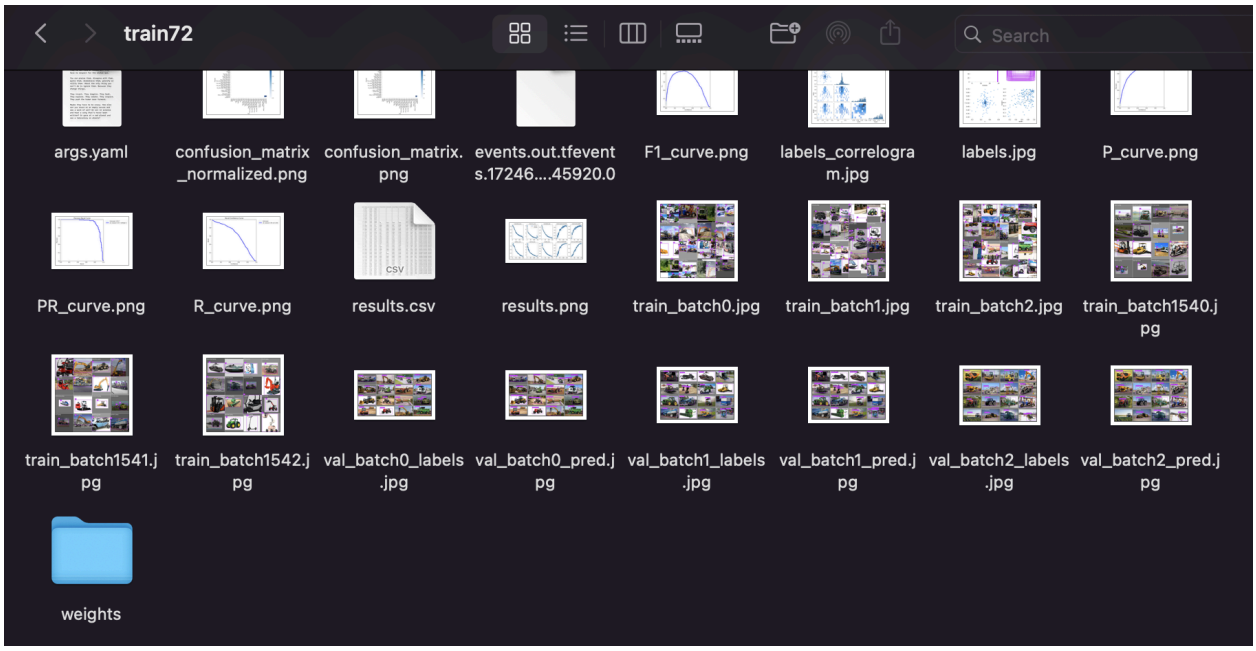
1. **Run the Training Script:** Execute the `main.py` script to start training the model. Pay attention to the output, particularly the line indicating where results are saved, such as `'Results saved to runs/detect/train72'`. This path may vary.



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).

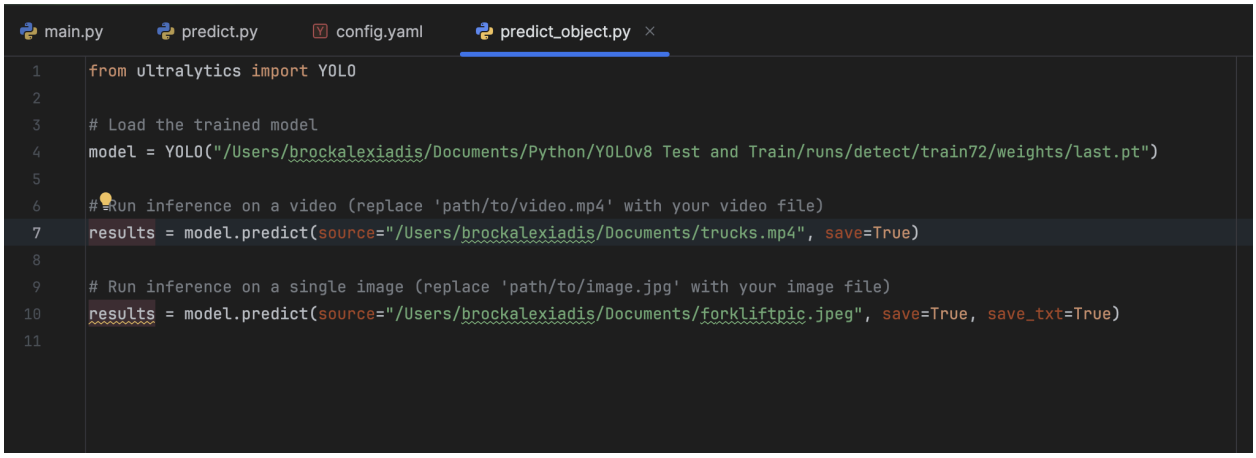
3. You can see all 170 images from the train set have been used and all 40 from the val set

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.

```python
from ultralytics import YOLO

# Load the trained model
model = YOLO("/Users/brockalexiadis/Documents/Python/YOLOv8 Test and Train/runs/detect/train72/weights/last.pt")

# Run inference on a video (replace 'path/to/video.mp4' with your video file)
#results = model.predict(source="/Users/brockalexiadis/Documents/trucks.mp4", save=True)

# Run inference on a single image (replace 'path/to/image.jpg' with your image file)
results = model.predict(source="/Users/brockalexiadis/Documents/forkliftpic.jpeg", save=True, save_txt=True)

```

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

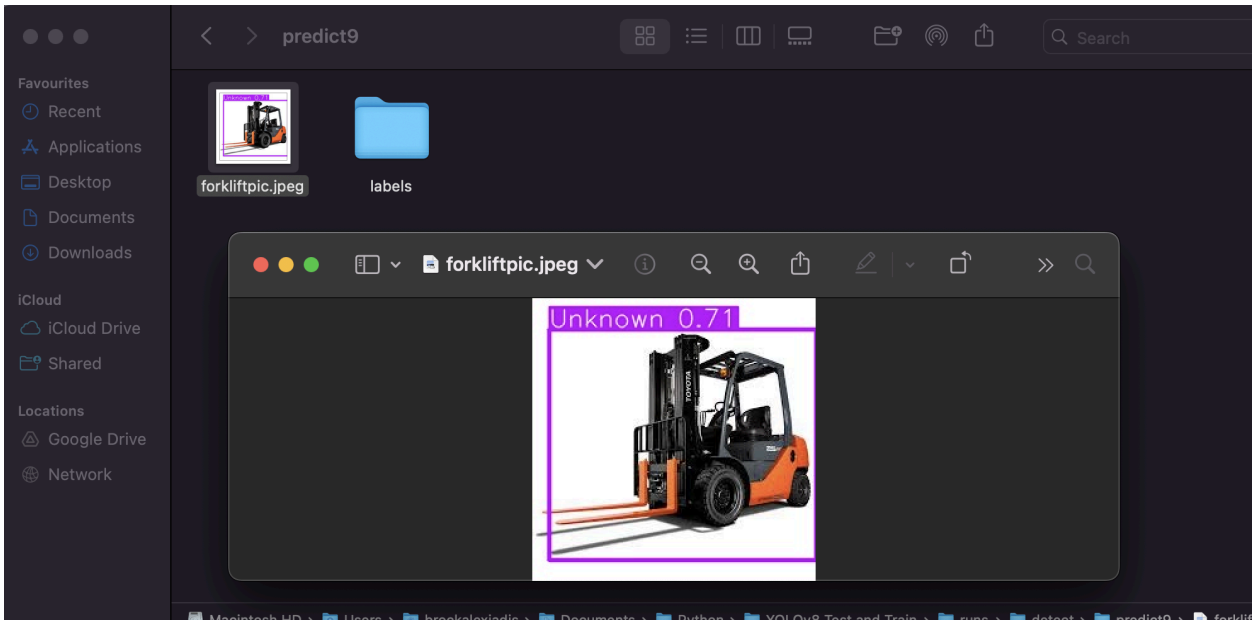
1. **Execute the Prediction Script:** Run `predict_object.py`. The prediction results will be saved to a path similar to `'runs/detect/predict9'`.

```
"/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