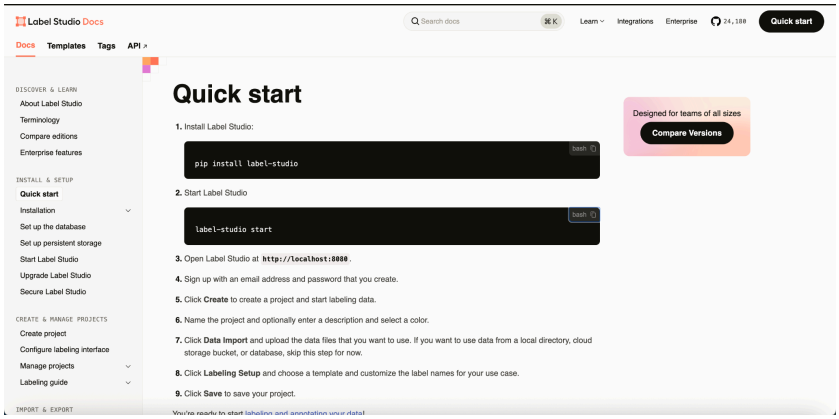
Here is a guide for IRC to use Label Studio and then import the labeled images to Python to run a YOLOv5 Model which then can be run in Addax AI for inference:

Created by: LMU Capstone Team 2025

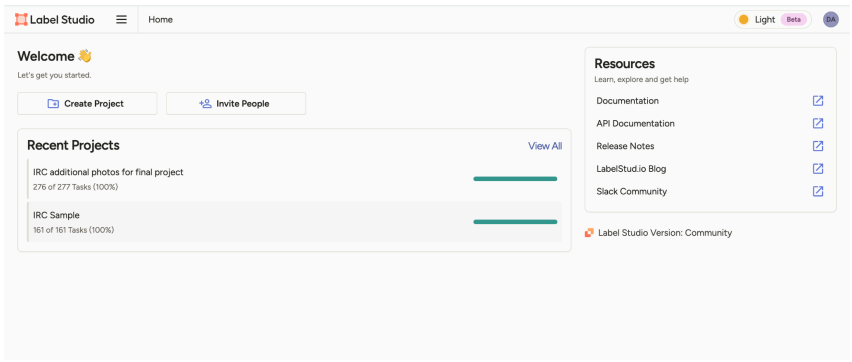
Step 1:

- Go to the following link: https://labelstud.io/guide/quick_start
- Install Label Studio.
- Start Label Studio.
- Open Label Studio at http://localhost:8080
- Sign up with an email and password.

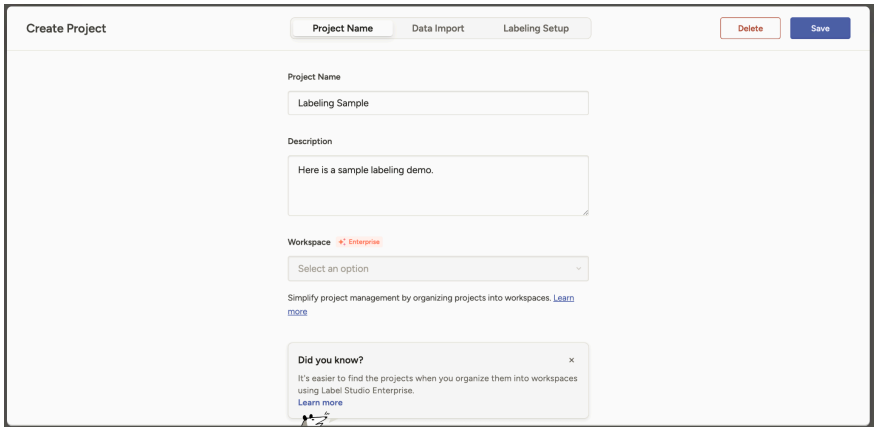


Step 2:

- Click “Create Project.”

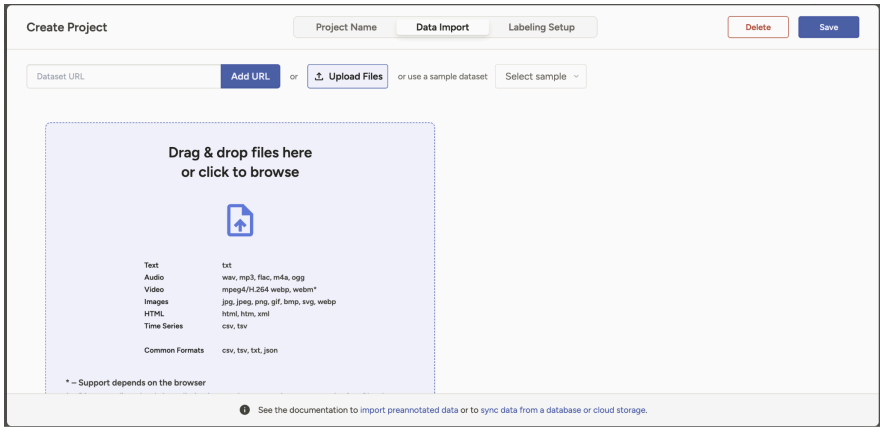


- Give your project a name and description.

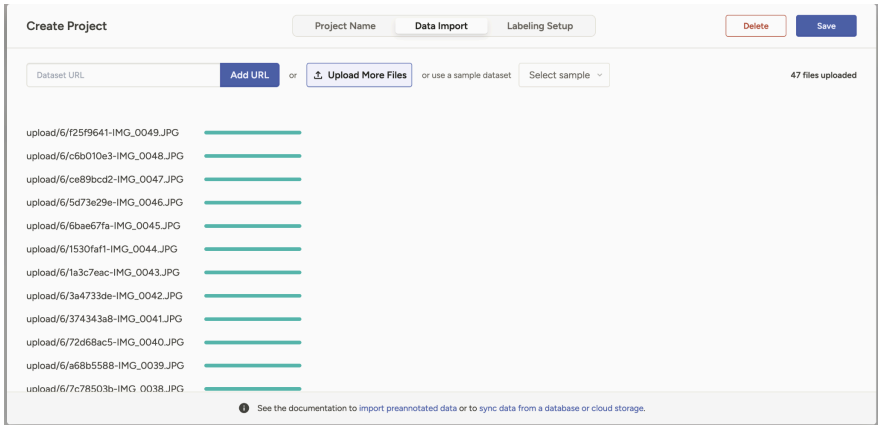


Step 3:

- Click on “Data Import.”
- Click on “Upload Files.” Import your data.
- **Note:** you can only import 100 images at a time.

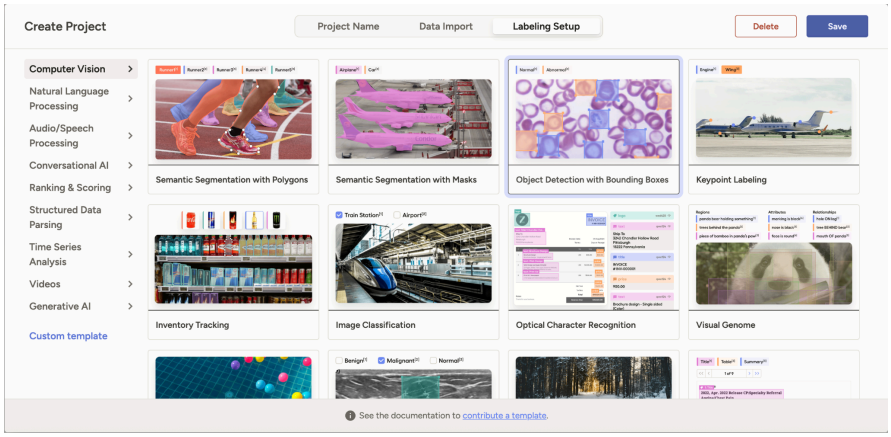


- Once uploaded, it will look like this:

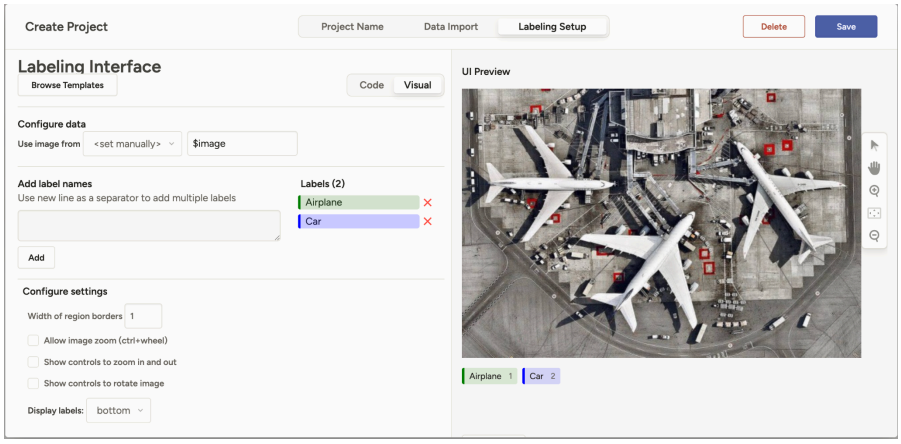


Step 4:

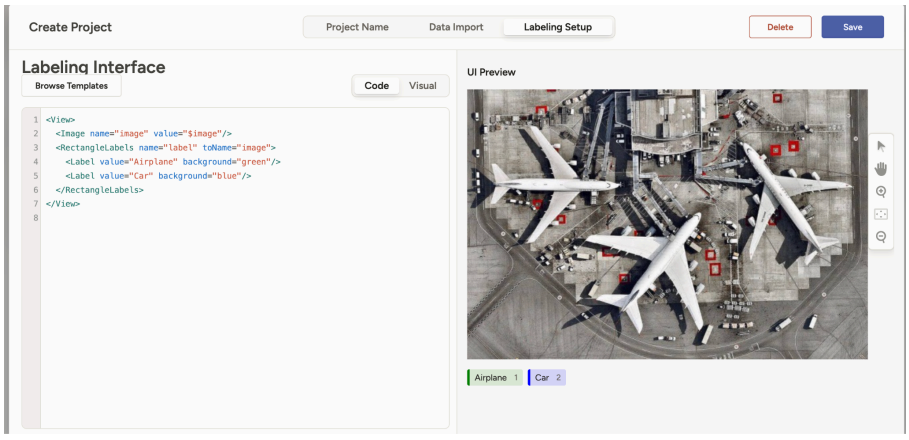
- Click “Labeling Setup.”
- Click “Object Detection with Bounding Boxes.”



- It should now look like this:



- Click on “Code.”

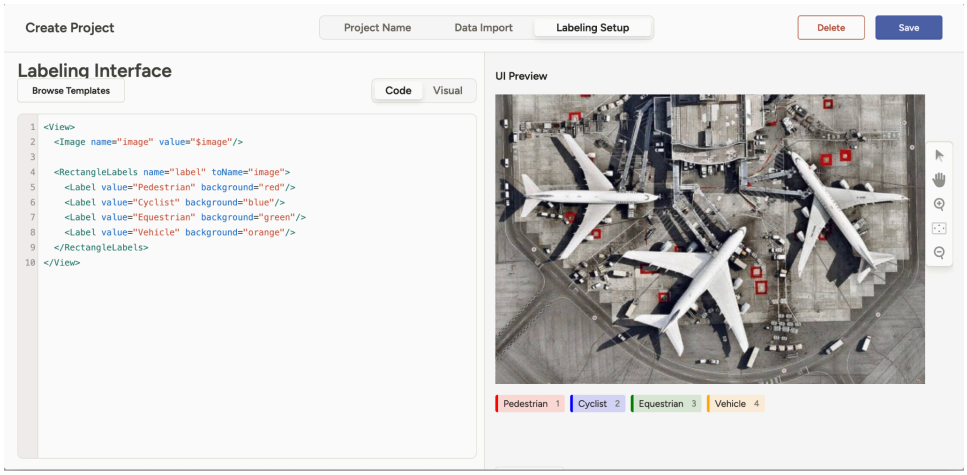


- Delete the existing code and paste the following code:

```
<View>
  <Image name="image" value="$image"/>

  <RectangleLabels name="label" toName="image">
    <Label value="Pedestrian" background="red"/>
    <Label value="Cyclist" background="blue"/>
    <Label value="Equestrian" background="green"/>
    <Label value="Vehicle" background="orange"/>
  </RectangleLabels>
</View>
```

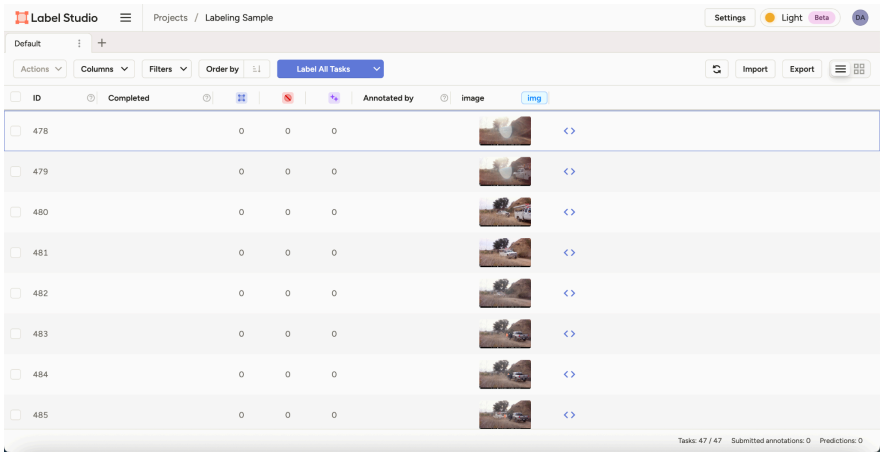
It should now look like this:



- Click “Save.”

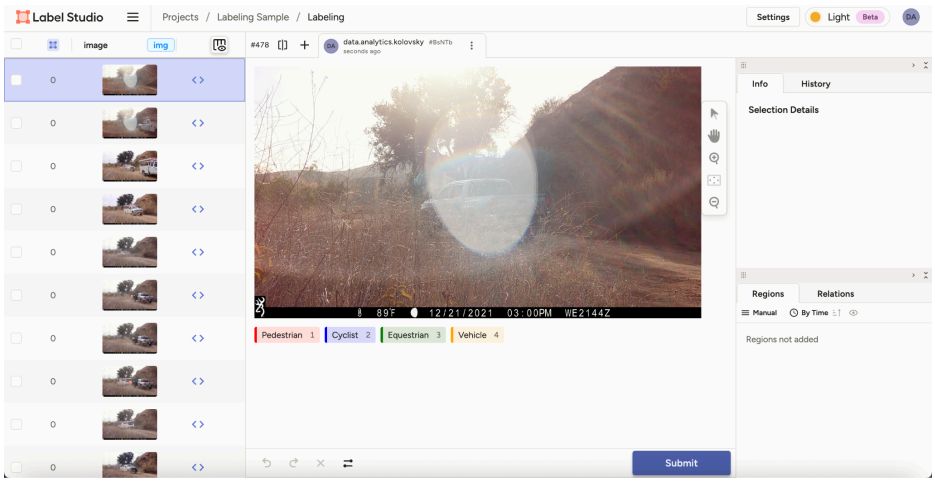
Step 5:

- It should now look like this:

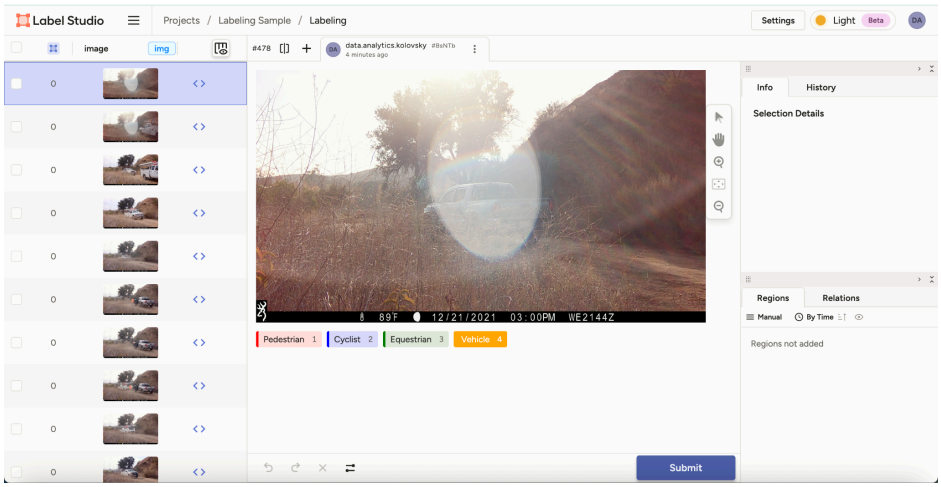


- Click on the first row.

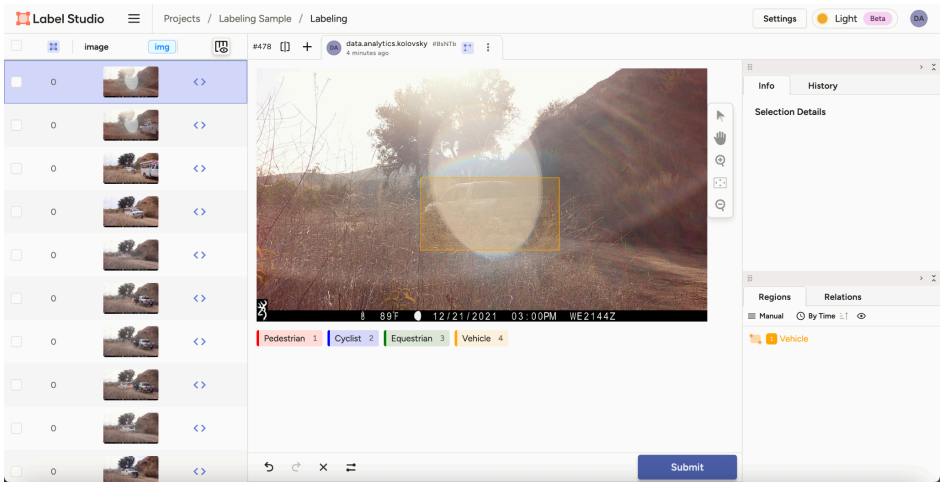
- Your labeling setup should now look like this:



- Click on the appropriate class at the bottom of the screen. In this case, “Vehicle.” The “Vehicle” button should now be highlighted indicating it’s selected.

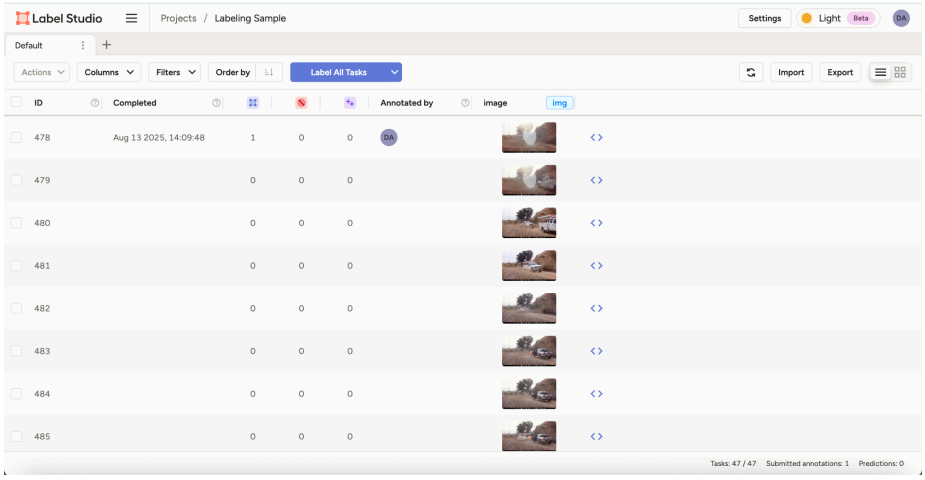


- Draw a bounding box around the object.
- Click “Submit” at the bottom of the screen.

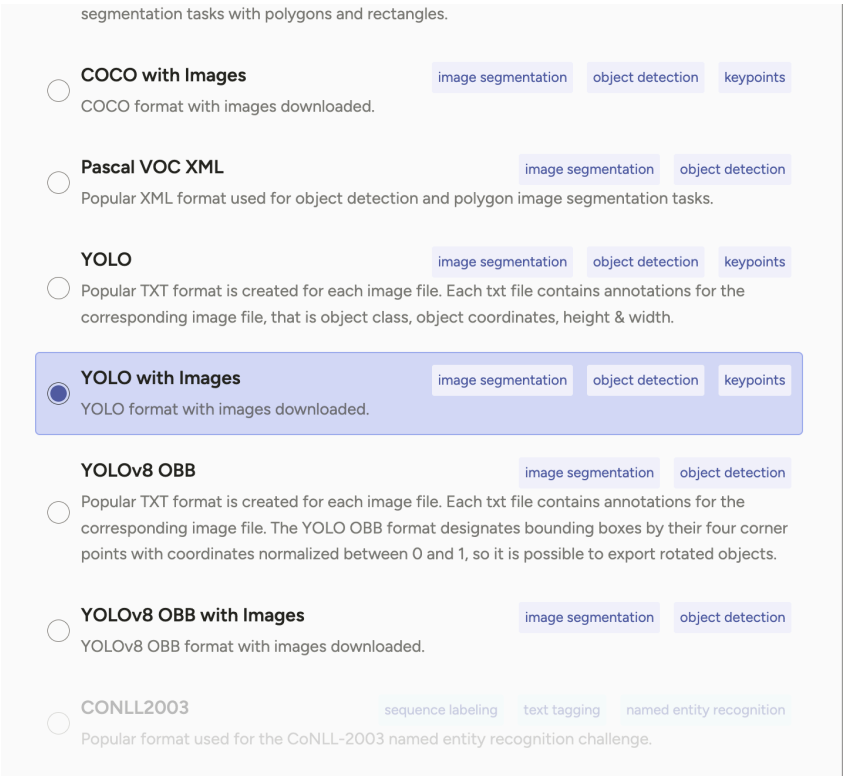


Step 6:

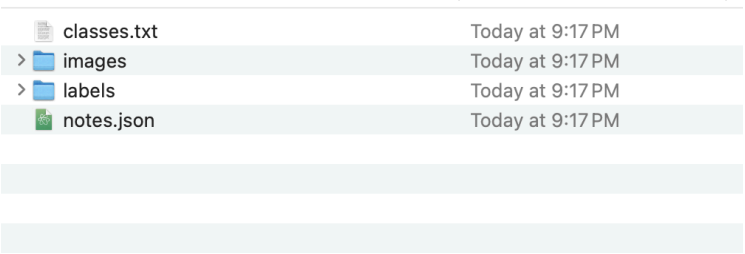
- Once all images are labeled, click on “Export.”



- Scroll down and click on “YOLO with Images.”
- Then scroll down and click “Export.”



- You should end up with a folder with an “Images” folder, a “Labels” folder, a “notes.json” file, and a “classes.txt” file.



Step 7 — Import to Colab, train YOLOv5, and save best.pt

7.1 Create a Colab notebook & set GPU

1. Go to Google Colab → **New Notebook**
2. **Runtime → Change runtime type → GPU (T4/A100 if available)**
3. Run the following cells in order.

 # Verify GPU
!nvidia-smi

7.2 Install YOLOv5

 # Clone YOLOv5 and install deps
!git clone https://github.com/ultralytics/yolov5.git
%cd yolov5
!pip install -r requirements.txt # installs torch, torchvision, etc.

7.3 Bring your labeled export into Colab

You exported **YOLO with Images** in Step 6, which gave you:

Images/ # all images (.jpg/.png)
Labels/ # YOLO .txt files (same basenames as images)
notes.json
classes.txt # class names in order used in Label Studio

Option A — Upload the zip directly

Compress your export into irc_export.zip, then:

from google.colab import files
up = files.upload() # choose irc_export.zip
!mkdir -p /content/irc_raw
!unzip -q irc_export.zip -d /content/irc_raw
!ls -R /content/irc_raw | head -n 200

Option B — Use Google Drive (if your data is large)

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

 # Example: copy from Drive to Colab
!mkdir -p /content/irc_raw
!cp -r "/content/drive/My Drive/IRC/irc_export_folder/." /content/irc_raw/
!ls -R /content/irc_raw | head -n 200

7.4 Split into train/val/test & build data.yaml

Most exports are **not** pre-split. The cell below:

- Reads classes.txt so the class order exactly matches Label Studio
- Splits Images/Labels into train/val/test (80/15/5 by default)
- Writes a YOLOv5-compatible data.yaml

```
import os, shutil, random, glob, textwrap
from pathlib import Path

random.seed(42)

RAW = Path("/content/irc_raw")
IM_DIR = RAW / "Images"
LB_DIR = RAW / "Labels"

assert IM_DIR.exists() and LB_DIR.exists(), "Expected Images/ and Labels/ in /content/irc_raw"

# Read classes in EXACT order from classes.txt
classes_path = RAW / "classes.txt"
assert classes_path.exists(), "classes.txt not found in export root"
with open(classes_path, "r", encoding="utf-8") as f:
    CLASSES = [ln.strip() for ln in f if ln.strip()]

print("Classes:", CLASSES)

# Make YOLOv5 dataset structure
DS = Path("/content/irc_yolov5_dataset")
for split in ["train","val","test"]:
    (DS / f"images/{split}").mkdir(parents=True, exist_ok=True)
    (DS / f"labels/{split}").mkdir(parents=True, exist_ok=True)

# Gather samples (image -> label path)
images = sorted([p for p in IM_DIR.glob("*") if p.suffix.lower() in [".jpg",".jpeg",".png"]])
pairs = []
for img in images:
    lbl = LB_DIR / (img.stem + ".txt")
    if lbl.exists():
        pairs.append((img, lbl))
    else:
        # include unlabeled images if you want; here we skip
        pass
```

```
print(f"Found {len(pairs)} labeled images")

# Split (80/15/5)
n = len(pairs)
n_train = int(n * 0.80)
n_val   = int(n * 0.15)
random.shuffle(pairs)
train = pairs[:n_train]
val   = pairs[n_train:n_train+n_val]
test  = pairs[n_train+n_val:]

def move_pairs(pairs, split):
    for img, lbl in pairs:
        shutil.copyfile(img, DS / f"images/{split}/{img.name}")
        shutil.copyfile(lbl, DS / f"labels/{split}/{lbl.name}")

move_pairs(train, "train")
move_pairs(val, "val")
move_pairs(test, "test")

# Write data.yaml
yaml_text = f"""# IRC YOLOv5 dataset
path: {DS.resolve()}
train: images/train
val: images/val
test: images/test

names:
"""
for i, name in enumerate(CLASSES):
    yaml_text += f"  {i}: {name}\n"

(DS / "data.yaml").write_text(yaml_text, encoding="utf-8")
print("Dataset ready at:", DS.resolve())
print((DS / "data.yaml").read_text())
```

Important: The class order in classes.txt must match your Label Studio labeling config (e.g., Pedestrian, Cyclist, Equestrian, Vehicle). Do **not** change the order later—this same order flows into training and Addax.

7.5 Train YOLOv5

The command below starts with yolov5s.pt (transfer learning on COCO weights). Adjust --batch if you hit GPU memory issues:

```
%cd /content/yolov5
!python train.py \
  --img 640 \
  --batch 16 \
  --epochs 75 \
  --data /content/irc_yolov5_dataset/data.yaml \
  --weights yolov5s.pt \
  --name irc_yolov5_4cls \
  --patience 20
```

Outputs will be under:

```
/content/yolov5/runs/train/irc_yolov5_4cls/
  weights/
    best.pt   <-- this is the file you'll import into Addax AI
    last.pt
  results.png
  PR_curve.png
  confusion_matrix.png
  results.csv
```

7.6 Validate (optional but recommended)

```
!python val.py \
  --weights /content/yolov5/runs/train/irc_yolov5_4cls/weights/best.pt \
  --data /content/irc_yolov5_dataset/data.yaml \
  --img 640
```

This reports **mAP@.5**, **precision**, **recall**, etc. (on your **val** split).

If you created a distinct **test** split, that's used for final reporting—not for training decisions.

7.7 Quick sanity-check predictions

```
# Run predictions on test images; saves boxes to runs/detect/...
!python detect.py \
  --weights /content/yolov5/runs/train/irc_yolov5_4cls/weights/best.pt \
  --source /content/irc_yolov5_dataset/images/test \
```

```
  --img 640 \
  --conf 0.25 \
  --save-txt --save-conf

!ls -R runs/detect | head -n 200
```

Open the images in runs/detect/exp*/ to visually confirm labels look right.

7.8 Download best.pt for Addax AI

```
from google.colab import files
files.download('/content/yolov5/runs/train/irc_yolov5_4cls/weights/best.pt')
```

(Or copy best.pt back to Drive:)

```
!cp /content/yolov5/runs/train/irc_yolov5_4cls/weights/best.pt "/content/drive/My
Drive/IRC/best.pt"
```

(Optional) 7.9 Run locally in Cursor

If training locally with a CUDA GPU:

```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
pip install -r requirements.txt
python train.py --img 640 --batch 16 --epochs 75 \
  --data /absolute/path/to/irc_yolov5_dataset/data.yaml \
  --weights yolov5s.pt --name irc_yolov5_4cls --patience 20
```

Troubleshooting quick hits

- **CUDA OOM:** lower --batch, try --img 512, or use a smaller model (yolov5n.pt).
- **No detections in detect.py:** lower --conf (e.g., 0.20–0.25) or train longer / add data.
- **Wrong classes on predictions:** ensure classes.txt order matched Label Studio and that data.yaml names kept the same order for training.

Now you're all set to run the best.pt file in Addax AI as per our tutorial video.