

# Dataset Annotation

---

## Dataset Annotation

- 1. Dataset Collection
  - 1.1. Image Extraction
  - 1.2. Dataset Path
- 2. Label Studio Usage
  - 2.1. Label Studio Installation
  - 2.2. Label Studio Startup
    - 2.2.1. Shared Folder Permissions
    - 2.2.2. Start Label Studio
    - 2.2.3. Access Label Studio
      - Host Machine
      - Same Local Network
      - First Time Use
- 3. Dataset Annotation
  - 3.1. Create Project
  - 3.2. Project Name
  - 3.3. Import Training Set
  - 3.4. Label Settings
  - 3.5. Dataset Annotation
  - 3.6. Export Dataset
  - 3.7. Dataset Directory Structure
  - 3.8. Dataset Configuration File

## References

For recognizing specific objects or improving object recognition accuracy, users need to train their own models. Dataset collection and annotation are indispensable parts of this process.

## 1. Dataset Collection

---

Description: For recognition in specific scenarios, users can use cameras to record videos of objects from various angles, then extract frames from the videos as training sets.

### 1.1. Image Extraction

Provide a simple example: Extract one frame every 15 frames from video as dataset

```
import cv2
import os

# Path to the input video file
video_file =
'/home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_video/orange.mkv'

# Path to the output folder to save the frames
output_folder =
'/home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_pic'

# Check if the output folder exists, if not, create it
if not os.path.exists(output_folder):
    os.makedirs(output_folder)
```

```

# Open the video file using OpenCV
cap = cv2.VideoCapture(video_file)

# Get the total number of frames in the video
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Get the frames per second (fps) of the video
fps = cap.get(cv2.CAP_PROP_FPS)

# Frame counter to keep track of the current frame number
frame_number = 0

# Define the interval for extracting frames (every 15th frame)
frame_interval = 15

# Initialize a new counter for saved frames (starting from 1)
saved_frame_number = 1

while True:
    ret, frame = cap.read()

    # If the frame is not successfully read, exit the loop
    if not ret:
        break

    # Save every 15th frame
    if frame_number % frame_interval == 0:
        # Format the frame number starting from 1 (e.g., 1.png, 2.png)
        image_name = f'{saved_frame_number}.png'
        image_path = os.path.join(output_folder, image_name)

        # Save the current frame as a PNG image
        cv2.imwrite(image_path, frame)
        print(f'Saving frame {frame_number}/{total_frames} as {image_name}')

    # Increment the saved frame counter
    saved_frame_number += 1

    # Increment the frame counter for the video
    frame_number += 1

# Release the video capture object
cap.release()

# Print message when processing is complete
print("Video processing complete! Every 15th frame saved as an image!")

```

## 1.2. Dataset Path

Example video: /home/jetson/ultralytics/ultralytics/data/yahboom\_data/orange\_video

Example images: /home/jetson/ultralytics/ultralytics/data/yahboom\_data/orange\_pic

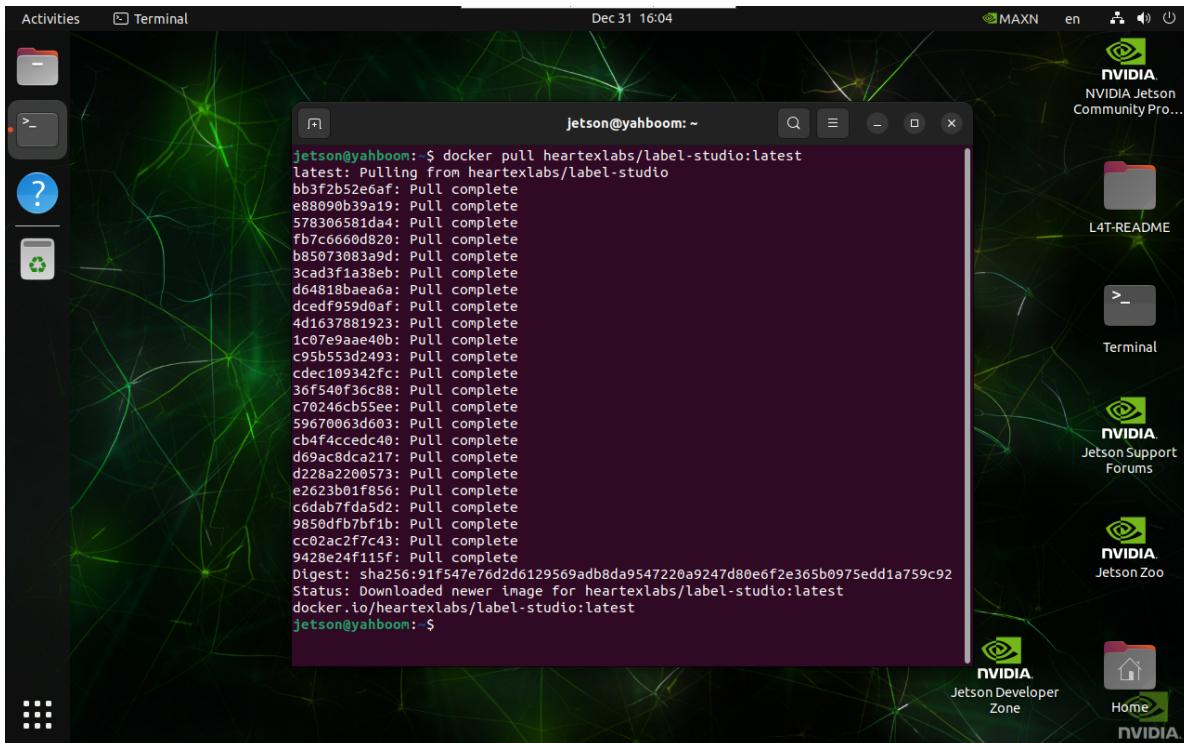
## 2. Label Studio Usage

---

Label Studio is an open-source data annotation platform for data annotation and annotation task management, supporting various types of data input and annotation formats.

## 2.1. Label Studio Installation

```
docker pull heartexlabs/label-studio:latest
```



## 2.2. Label Studio Startup

### 2.2.1. Shared Folder Permissions

Share the prepared dataset to the corresponding folder in Label Studio:

```
sudo chmod 777 /home/jetson/ultralytics/ultralytics/data/
```

### 2.2.2. Start Label Studio

```
sudo docker run -it -p 8080:8080 -v  
/home/jetson/ultralytics/ultralytics/data:/label-studio/data heartexlabs/label-  
studio:latest label-studio --log-level DEBUG
```

A screenshot of a terminal window titled "jetson@yahboom: ~". The window displays the output of a command to run a Docker container. The logs show the container starting up, setting up environment variables, and attempting to connect to Redis. The terminal is part of a desktop interface with a dark background featuring a green neural network pattern.

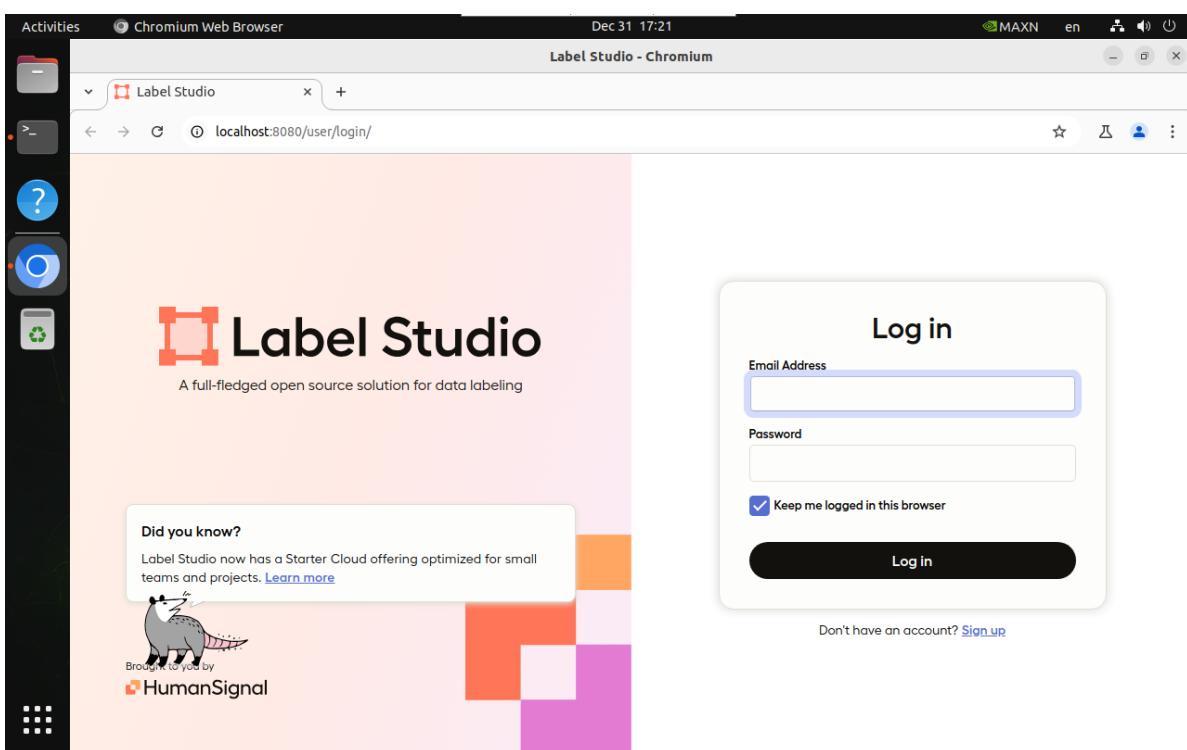
```
jetson@yahboom: $ docker run -it -p 8080:8080 -v /home/jetson/ultralytics/ultralytics/data:/label-studio/data heartexlabs/label-studio:latest label-studio --log -level DEBUG
=> Database and media directory: /label-studio/data
=> Static URL is set to: /static/
=> Database and media directory: /label-studio/data
=> Static URL is set to: /static/
Read environment variables from: /label-studio/data/.env
get 'SECRET_KEY' casted as '<class 'str'>' with default ''
/label-studio/.venv/lib/python3.12/site-packages/label_studio_sdk/_extensions/label_studio_tools/core/label_config.py:137: SyntaxWarning: invalid escape sequence ' \${'
    expression = "\${A-Za-z_}+"
Starting new HTTPS connection (1): pypi.org:443
https://pypi.org:443 "GET /pypi/label-studio/json HTTP/1.1" 200 33651
[2024-12-31 09:12:15,565] [core.feature_flags.base::<module>::40] [INFO] Read flags from file /label-studio/label_studio/feature_flags.json
[2024-12-31 09:12:16,078] [core.redis::<module>::22] [DEBUG] >> Redis is not connected.
[2024-12-31 09:12:16,502] [faker.factory::<module>::20] [DEBUG] Not in REPL -> leaving logger event level as is.
[2024-12-31 09:12:18,045] [faker.factory::_find_provider_class::78] [DEBUG] Looking for locale 'en_US' in provider 'faker.providers.address'.
[2024-12-31 09:12:18,048] [faker.factory::_find_provider_class::97] [DEBUG] Prov
```

## 2.2.3. Access Label Studio

### Host Machine

Jetson Orin series development board access method, first-time access requires entering username and password

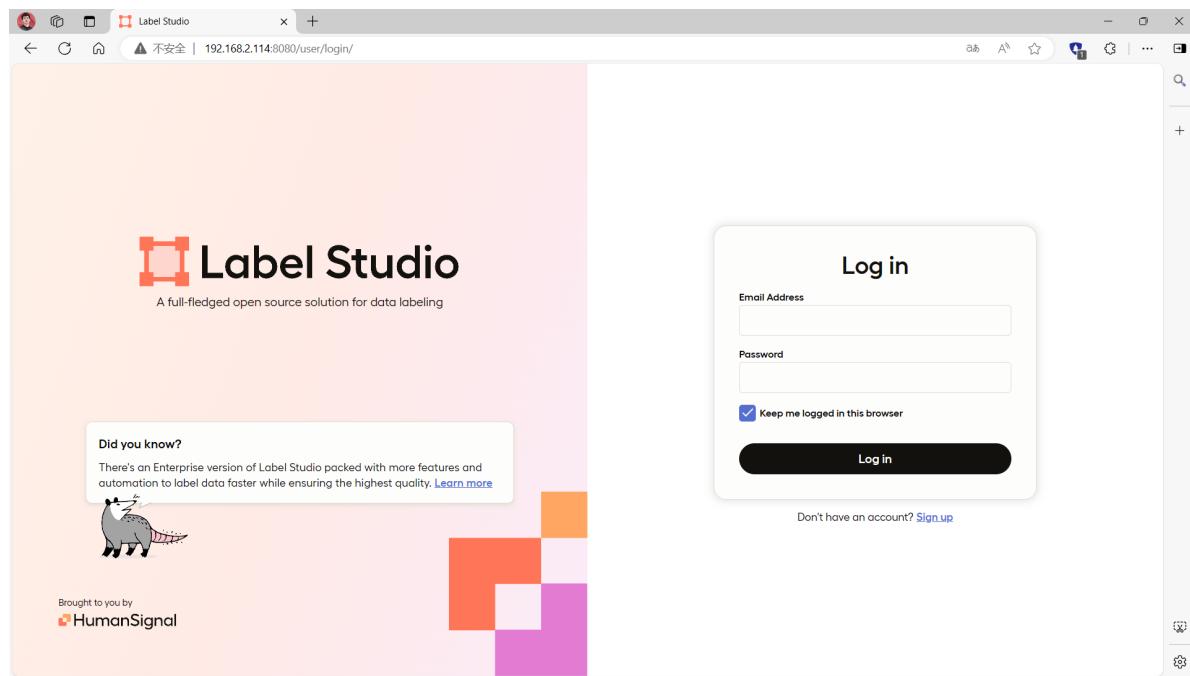
<http://localhost:8080/>



## Same Local Network

Devices on the same local network can access via board IP:8080

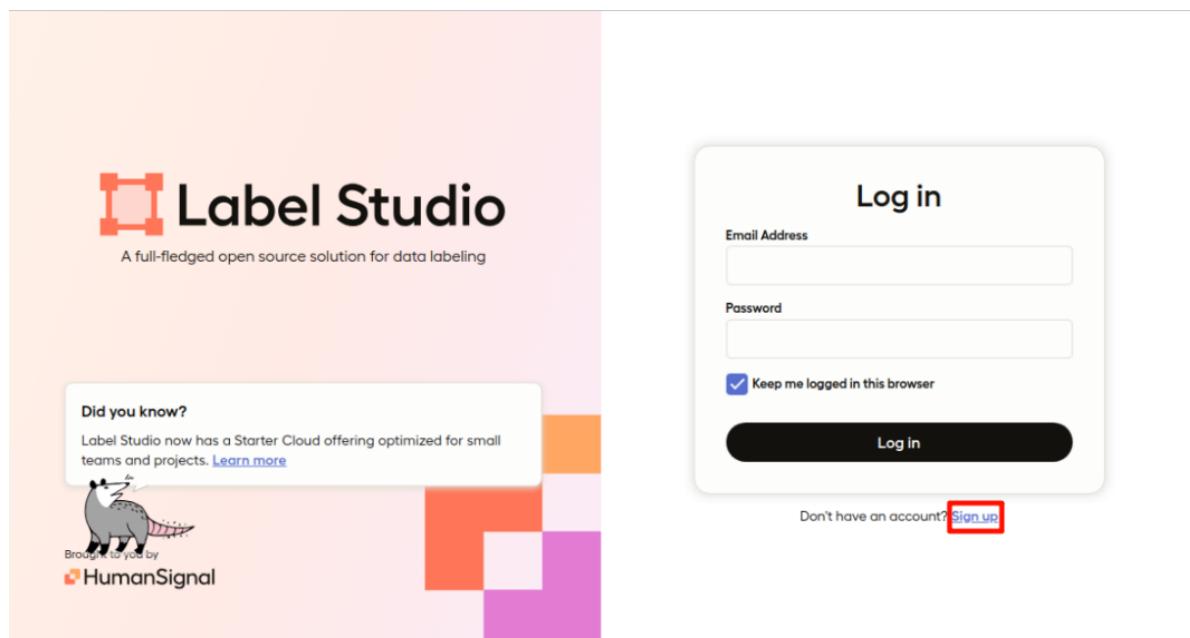
`http://board_IP:8080/ # Example: http://192.168.2.114:8080/`

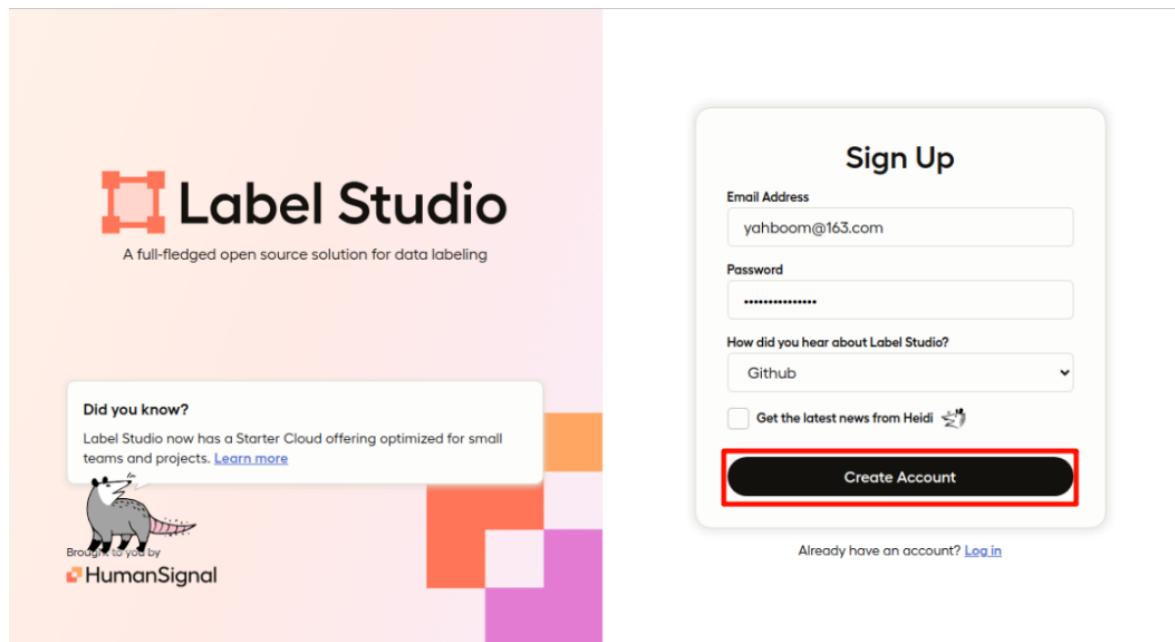


## First Time Use

First-time use requires user registration and login. Factory default account information (account information is fictional):

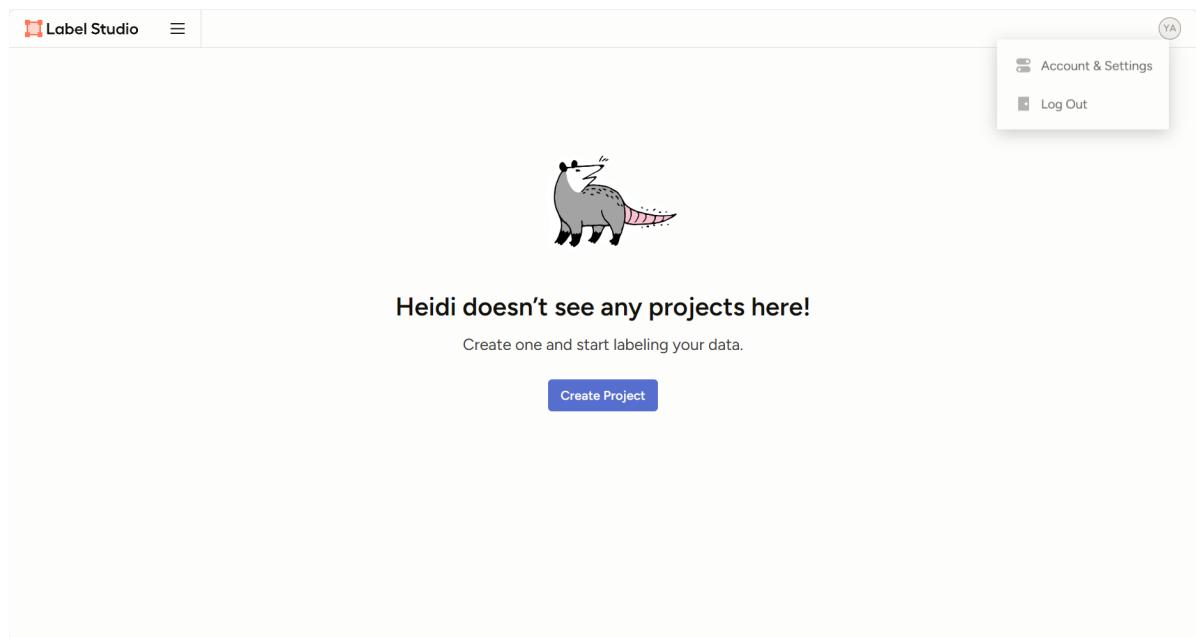
Account: `yahboom@163.com`  
Password: `yahboom@163.com`





The image shows the Label Studio sign-up page. At the top left is the Label Studio logo with the text "Label Studio" and a subtitle "A full-fledged open source solution for data labeling". Below the logo is a "Did you know?" section with a small illustration of a unicorn-like creature. It mentions a Starter Cloud offering for small teams and projects, with a link to "Learn more". To the right is a "Sign Up" form with fields for "Email Address" (yahboom@163.com), "Password" (redacted), "How did you hear about Label Studio?" (Github), and a checkbox for "Get the latest news from Heidi" (unchecked). A red box highlights the "Create Account" button. At the bottom right of the form is a link "Already have an account? Log in".

After registration is complete, the website will automatically log in:

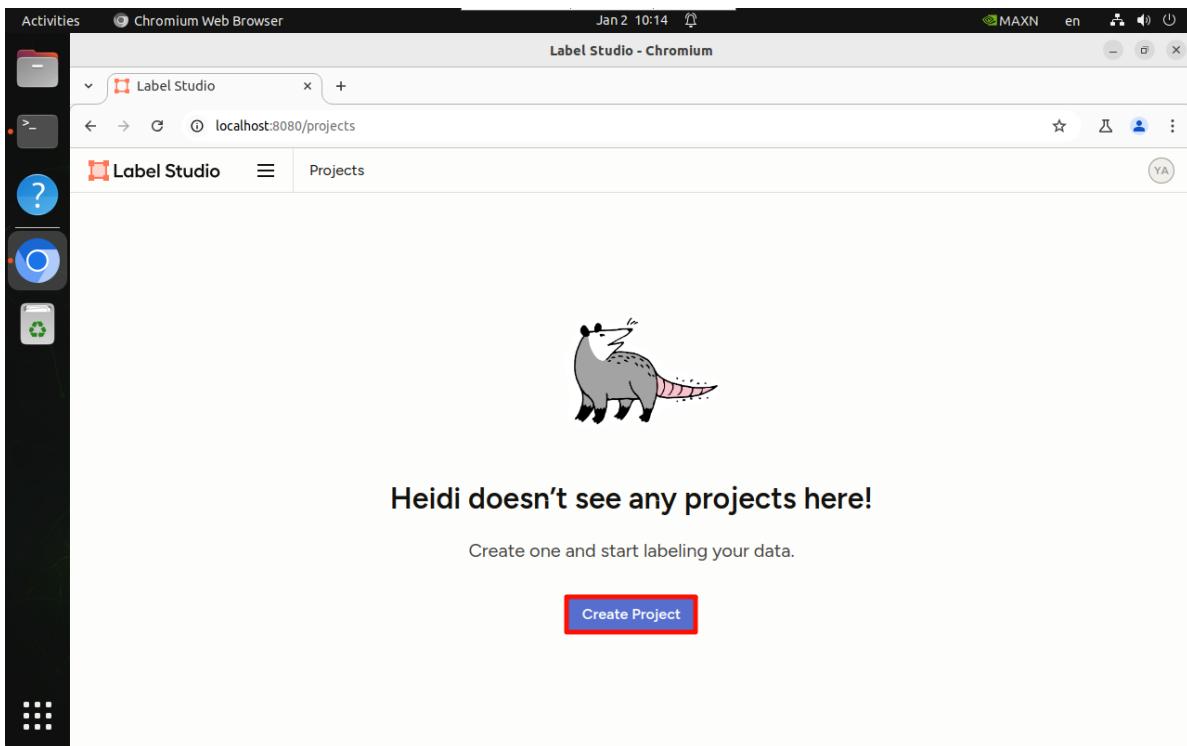


The image shows the Label Studio dashboard after logging in. The top navigation bar includes the Label Studio logo, a menu icon, and a user profile icon with the letters "YA". A dropdown menu from the profile icon shows "Account & Settings" and "Log Out". The main content area features a cartoon unicorn illustration. The text "Heidi doesn't see any projects here!" is displayed, followed by the subtext "Create one and start labeling your data." and a "Create Project" button.

## 3. Dataset Annotation

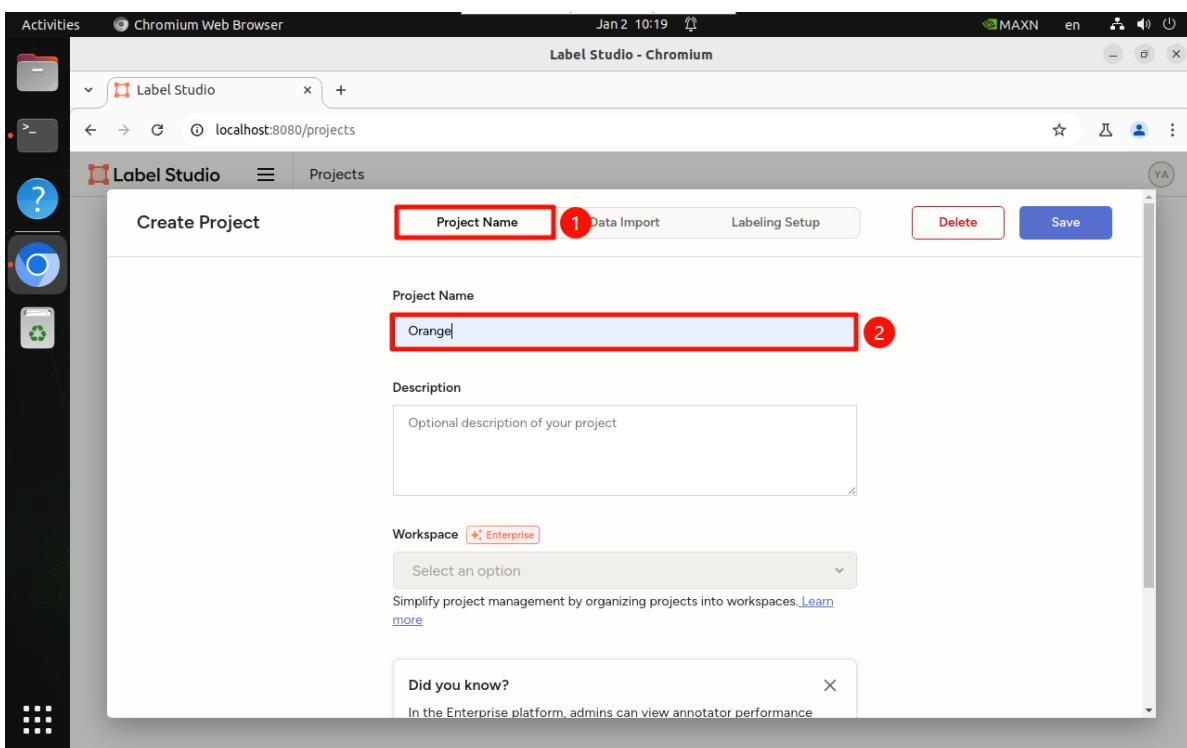
The demonstration dataset is on the board, so access Label Studio on the board.

### 3.1. Create Project



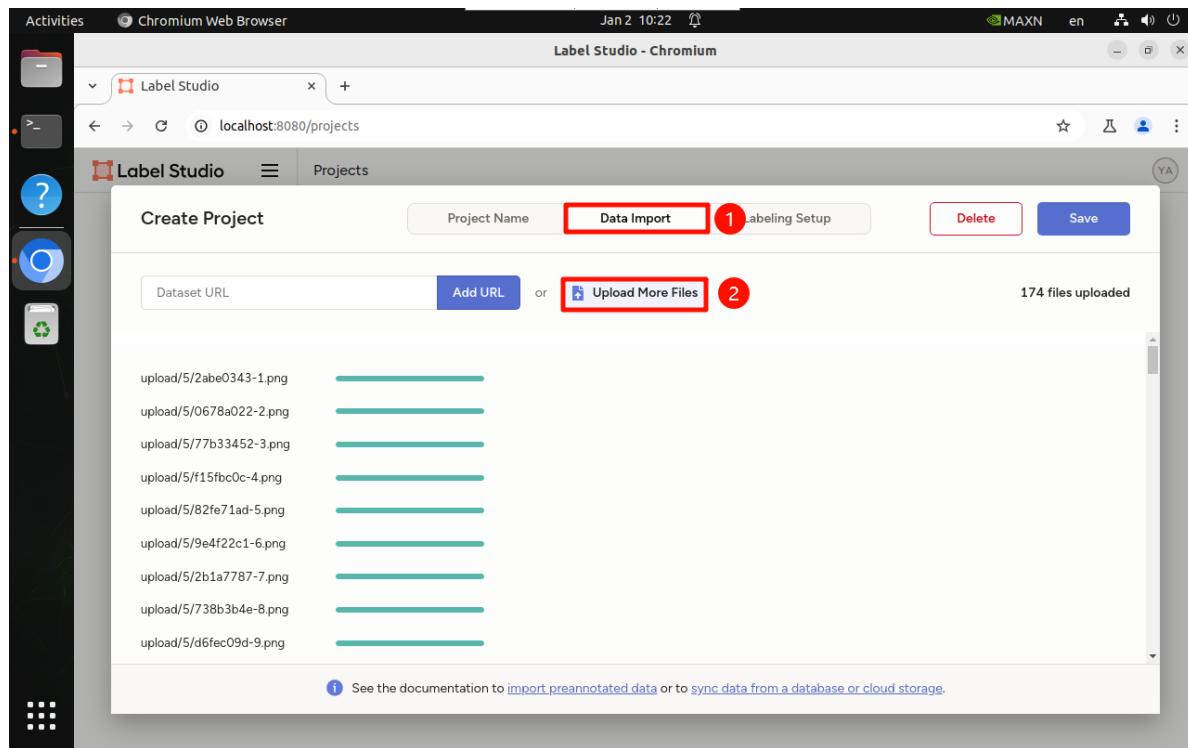
## 3.2. Project Name

The project name can be named arbitrarily, name it according to your training set:



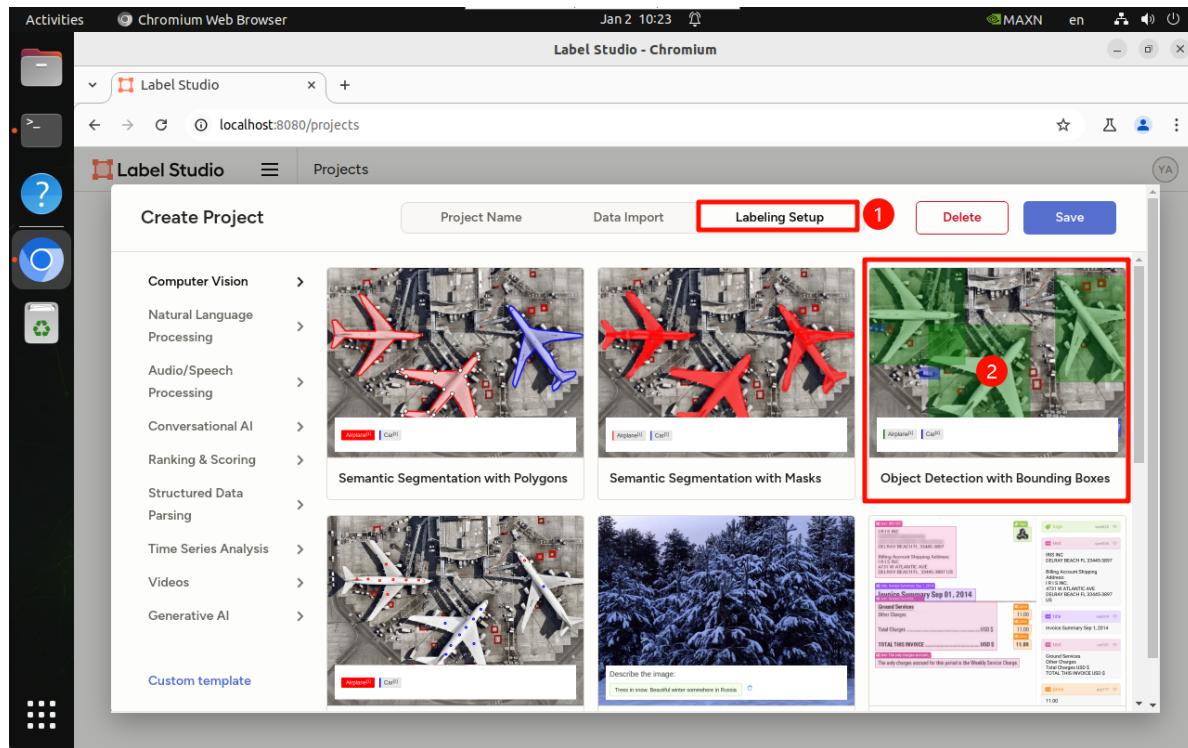
## 3.3. Import Training Set

Manually import the prepared dataset: Cannot import too many images at once, it is recommended to select 50-100 images each time, then import multiple times

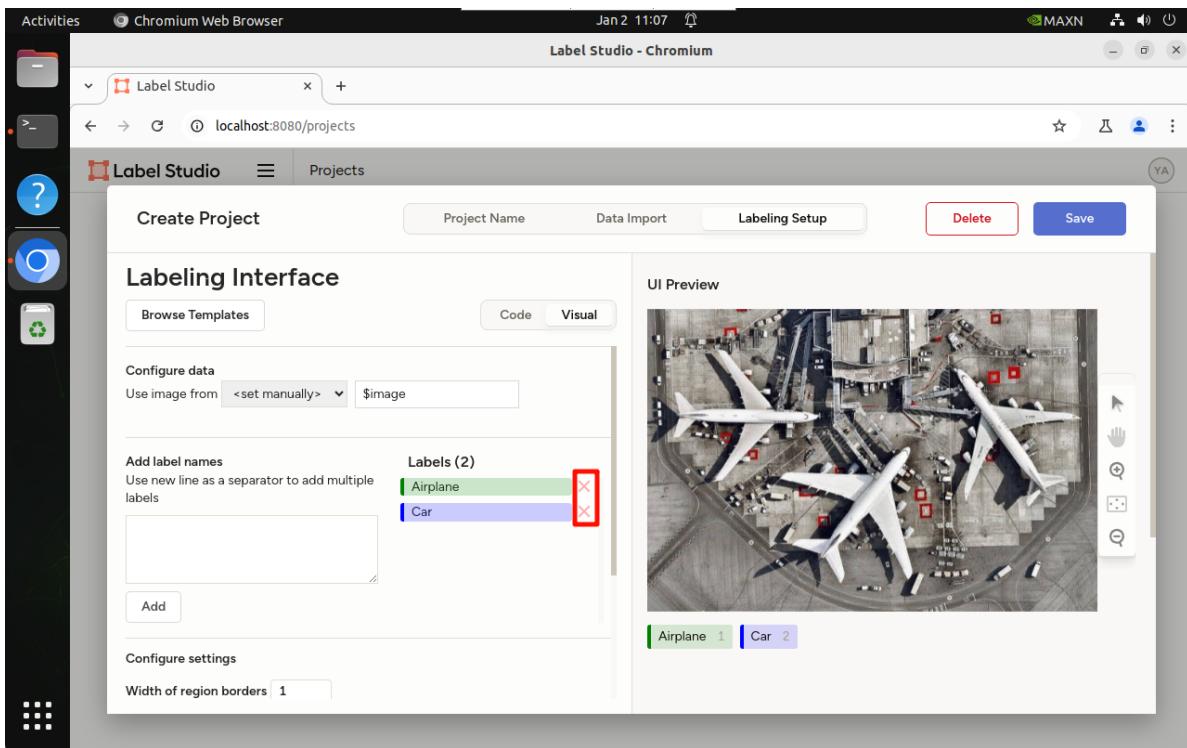


## 3.4. Label Settings

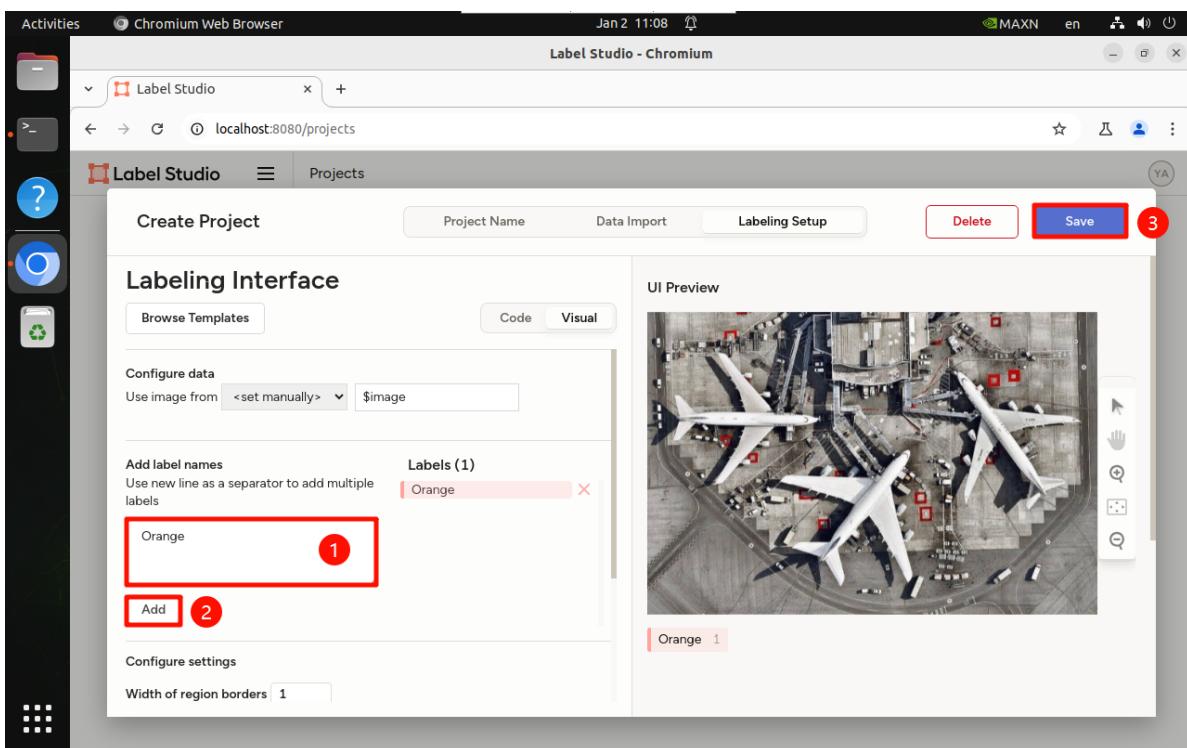
We are demonstrating orange recognition, so select "Object Detection"



Delete built-in labels:



Add new labels:



## 3.5. Dataset Annotation

Click "Annotate All Tasks":

Activities Chromium Web Browser Jan 2 11:10 Label Studio - Chromium

localhost:8080/projects/5/data?tab=2

Label Studio Projects / Orange

Default Actions Columns Filters Order not set Label All Tasks Import Export List Grid

ID	Completed	Annotated by	image
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	

Annotation steps: First click the label "Orange" then use the mouse to select the corresponding orange in the image, after completion click "Submit" to proceed to the next annotation

Activities Chromium Web Browser Jan 2 11:11 Label Studio - Chromium

localhost:8080/projects/5/data?tab=2&labeling=1

Label Studio Projects / Orange / Labeling

#1 1 of 1

Selection Details

Regions Relations

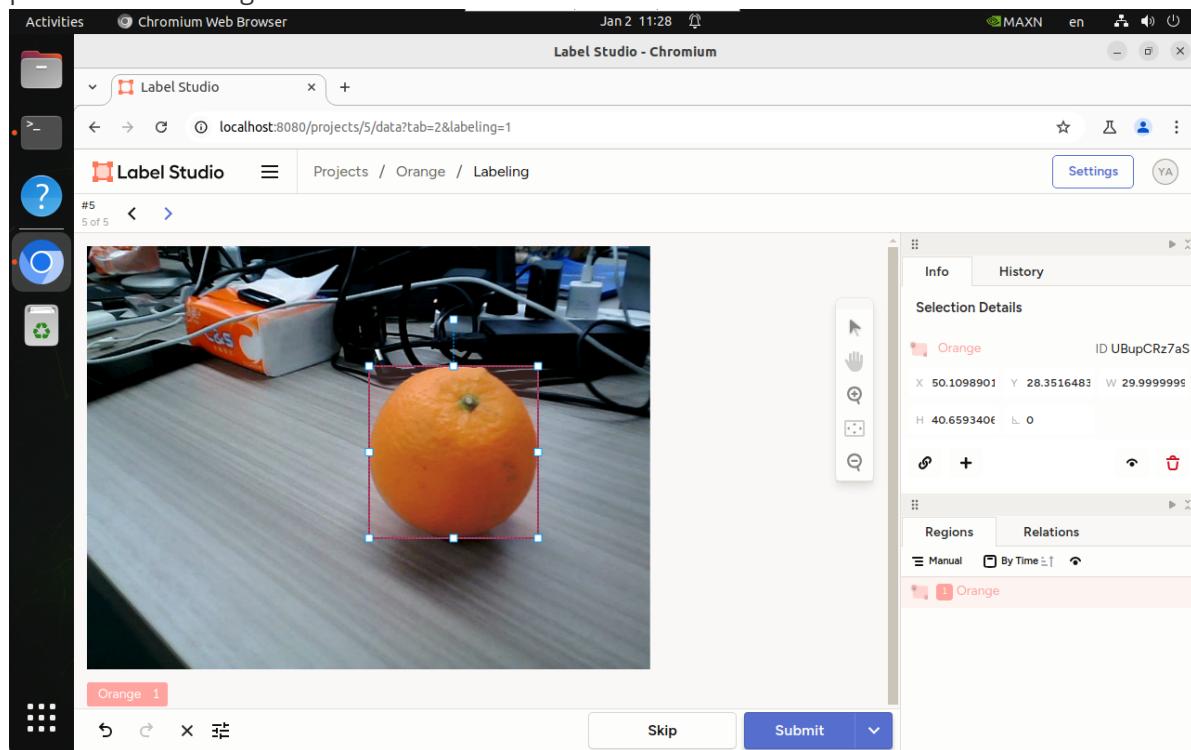
Manual By Time

Orange

Orange 1 1

Skip Submit 3

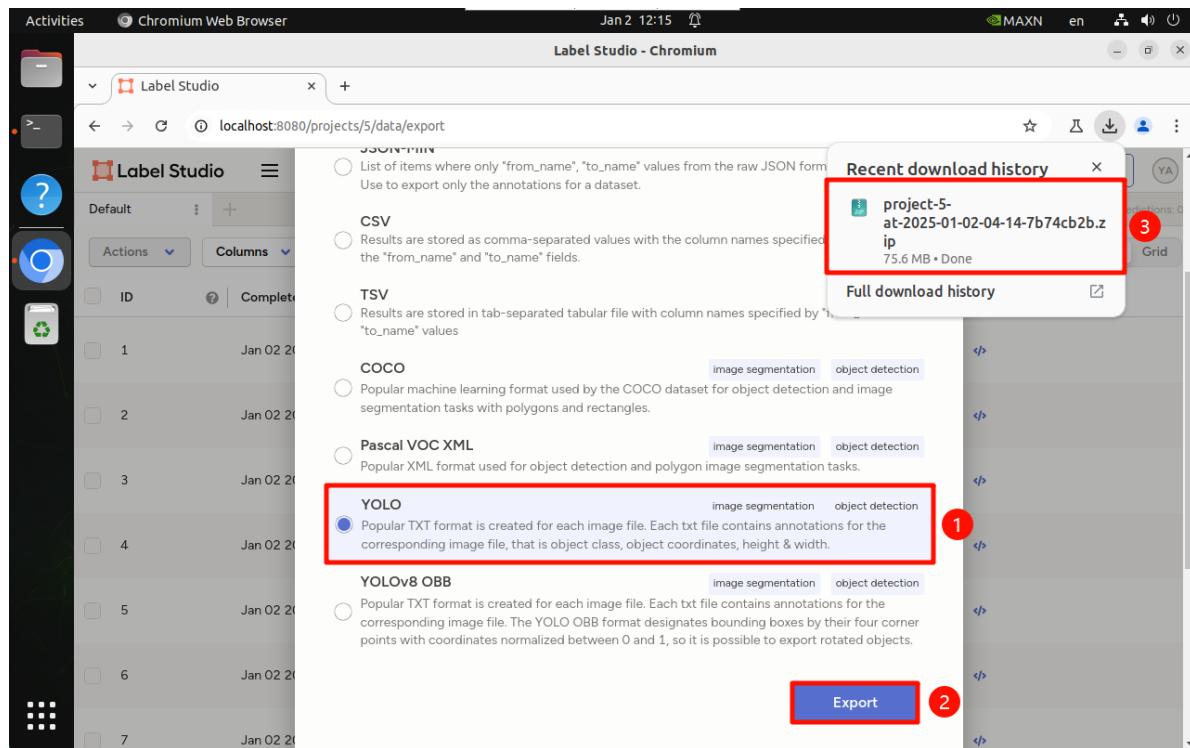
If the selected object is not a complete orange, you can click on the selected box to readjust the position of each edge:



## 3.6. Export Dataset

Select YOLO format for export, the browser will automatically download a compressed package

A screenshot of the Label Studio interface showing a list of annotated tasks. The top navigation bar includes 'Activities', 'Chromium Web Browser', a date/time indicator 'Jan 2 12:13', and system status 'MAXN en'. The main area displays a table of 174 tasks, each with columns for ID, Completed date, Order, not set, Annotated by (labeled 'YA'), image thumbnail, and a download icon. The 'Export' button in the toolbar above the table is highlighted with a red box. The table rows show various annotations for oranges, with thumbnails showing the orange in different contexts (e.g., on a table, in a box). The 'Label All Tasks' dropdown in the toolbar is also highlighted.



## 3.7. Dataset Directory Structure

The directory structure corresponding to the compressed package exported by Label Studio:

```
.
├── images
├── labels
├── classes.txt
└── notes.json
```

We need to separate the images and labels exported by Label Studio into train (training set) and val (validation set), with a ratio of approximately 8:2: The images and labels in both train and val need to have corresponding names.

```
.
├── classes.txt
├── notes.json
├── train
│   ├── images
│   └── labels
└── val
    ├── images
    └── labels
```

## 3.8. Dataset Configuration File

Contains paths to training and validation sets, as well as the number and names of classes.

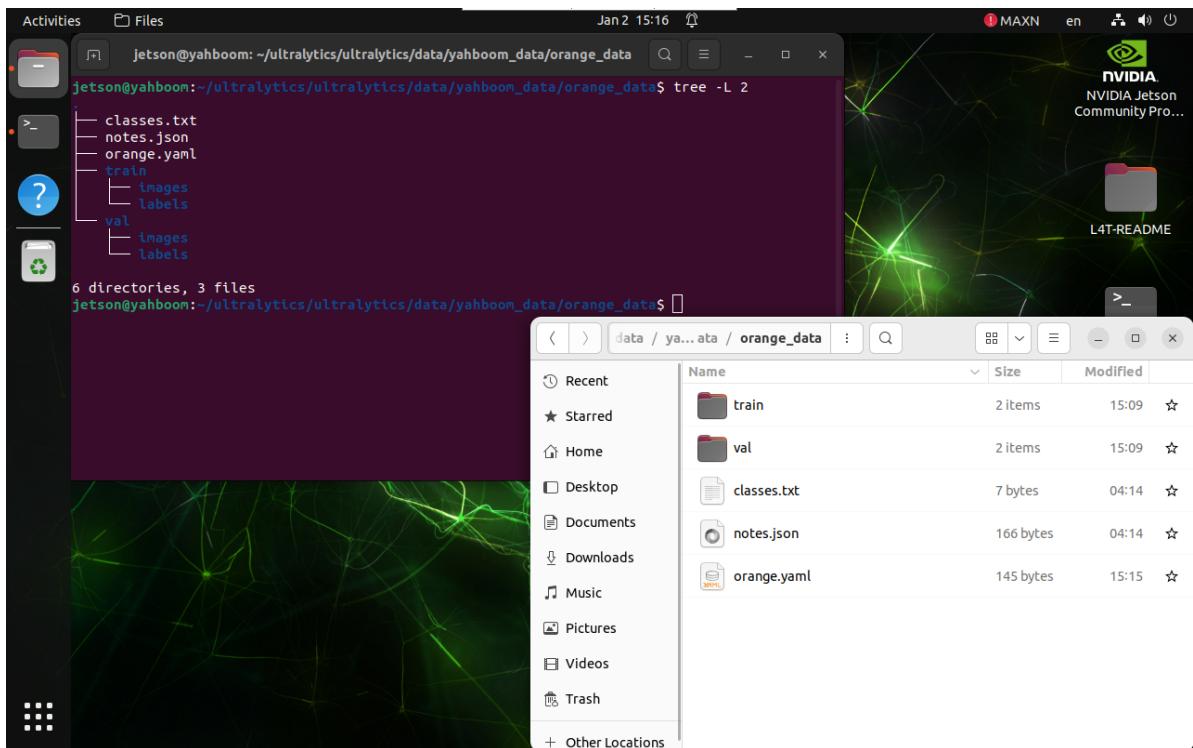
```

path: /home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_data
train: train/images
val: val/images

nc: 1

# classes
names:
0: Orange

```



## References

---

<https://github.com/HumanSignal/label-studio>