Dataset Annotation

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

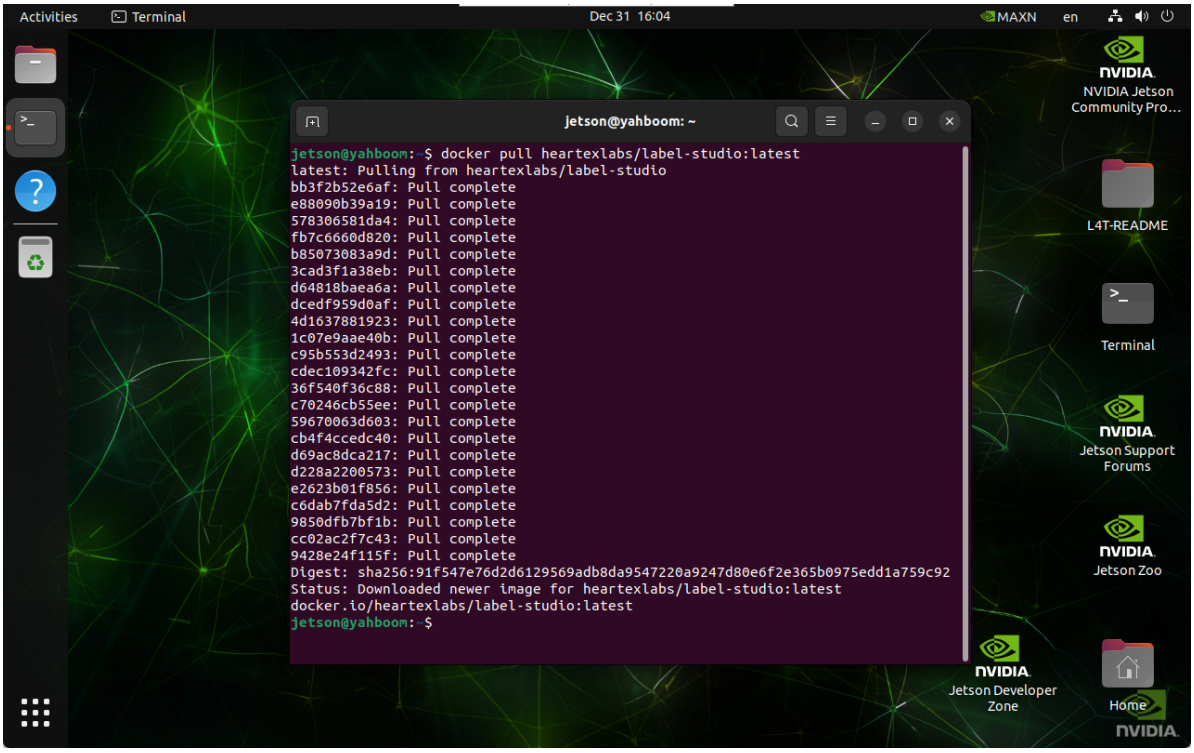
Example images: /home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_pic

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

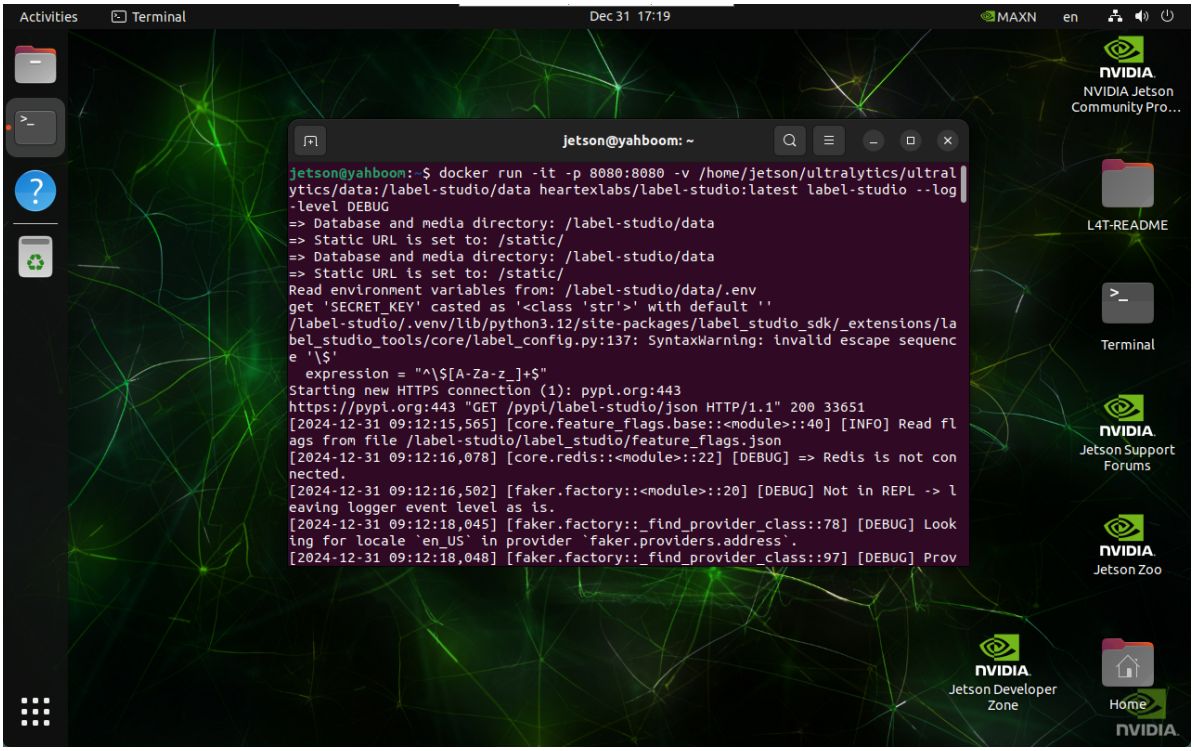
2.2.1. Shared Folder Permissions

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
```

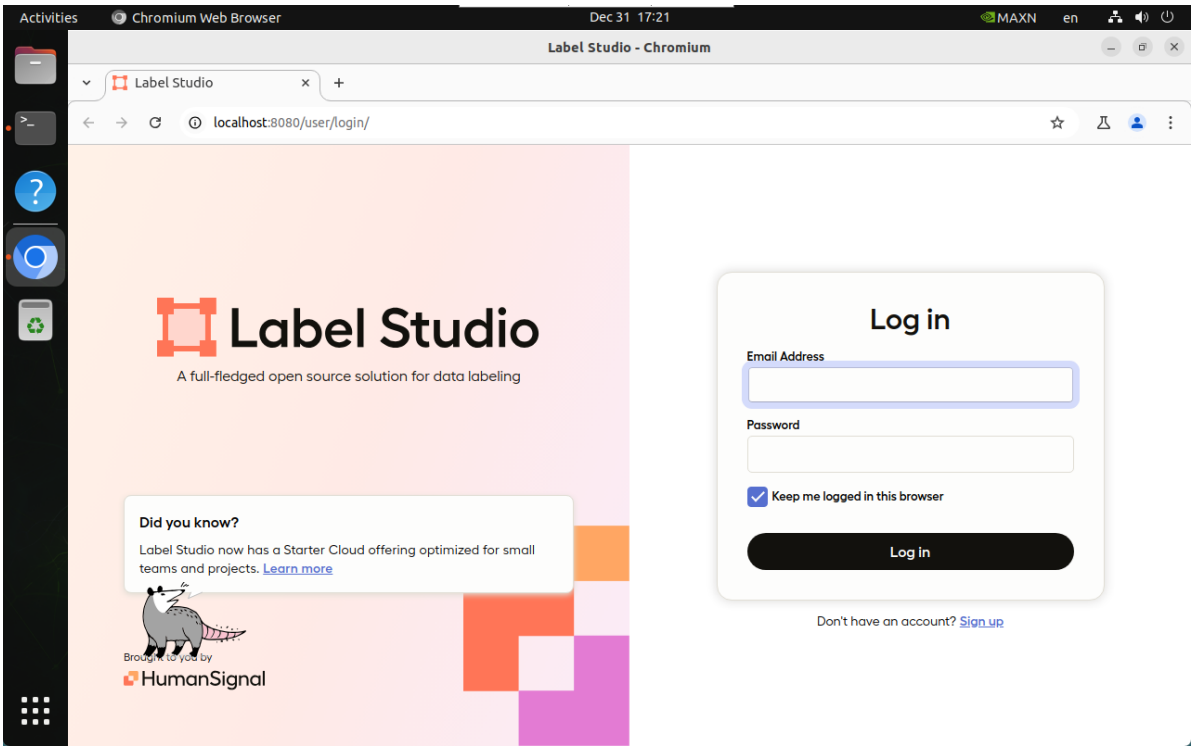


2.2.3. Access Label Studio

Host Machine

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

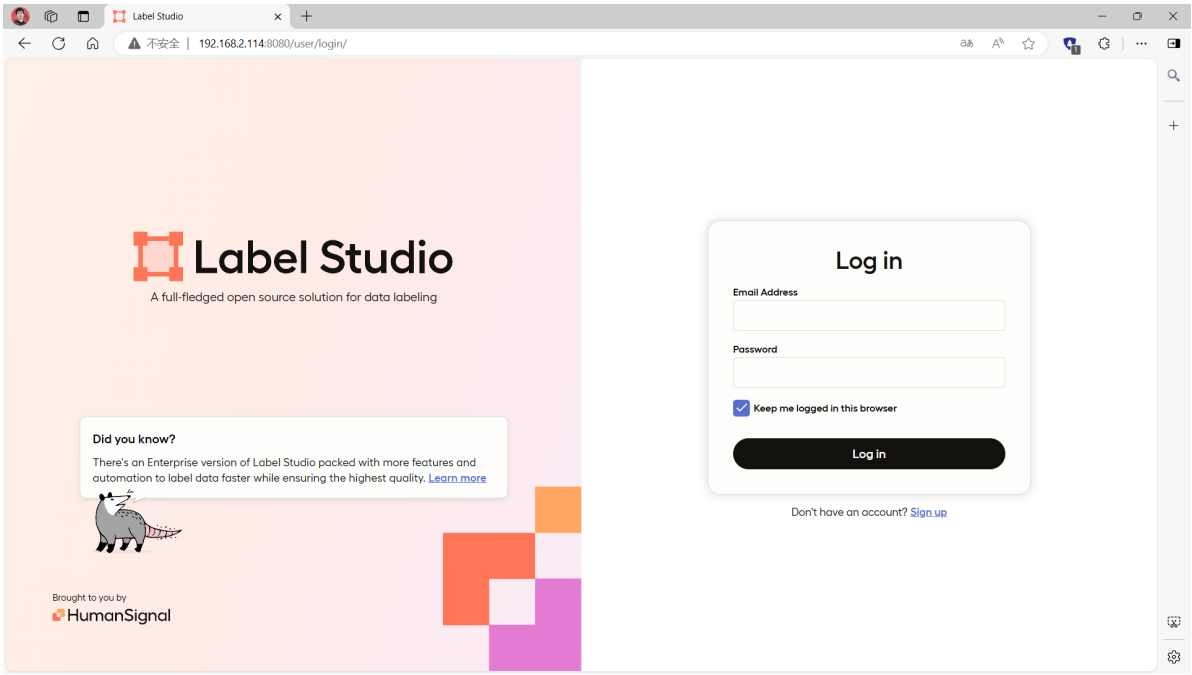
```
http://localhost:8080/
```



Same Local Network

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

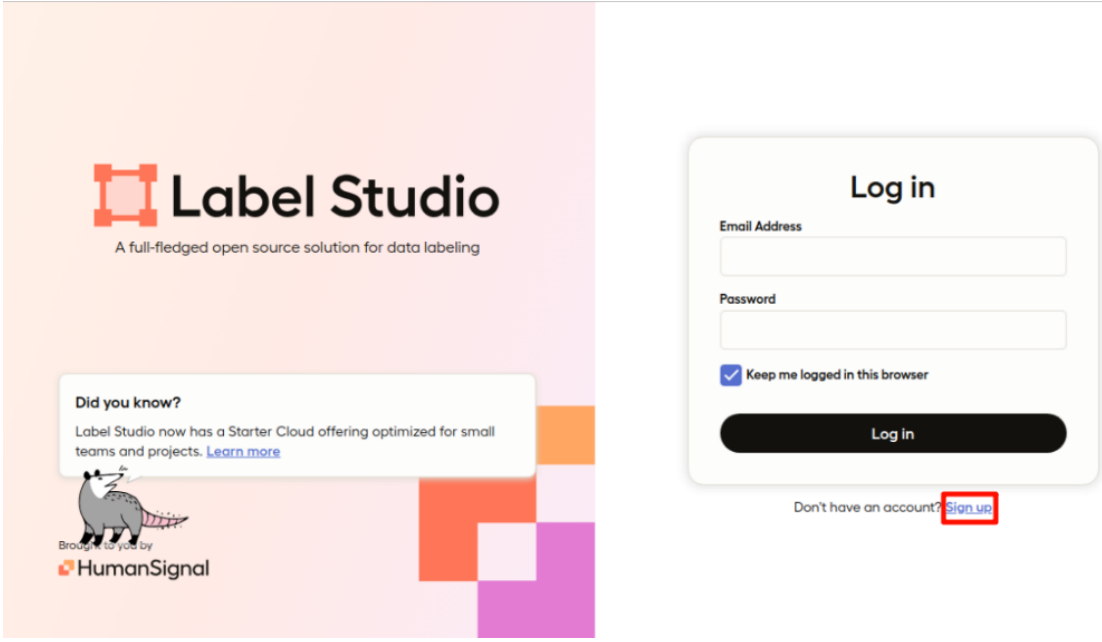
```
http://board_IP:8080/ # Example: http://192.168.2.114:8080/
```

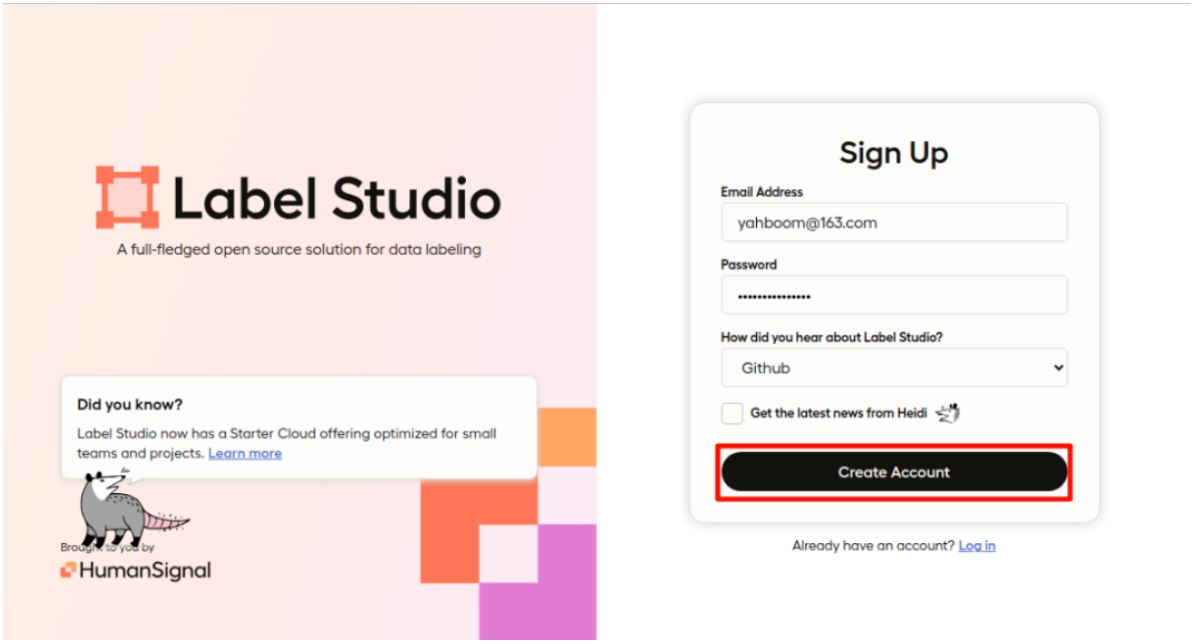


First Time Use

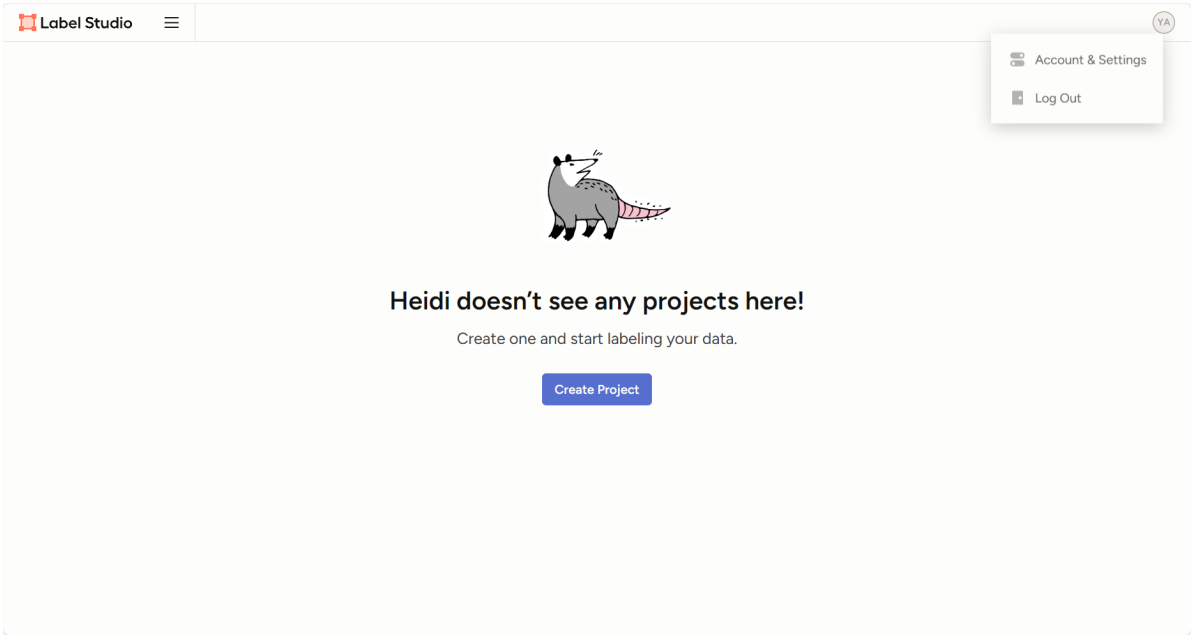
First-time use requires user registration and login. Factory default account information (account information is fictional):

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





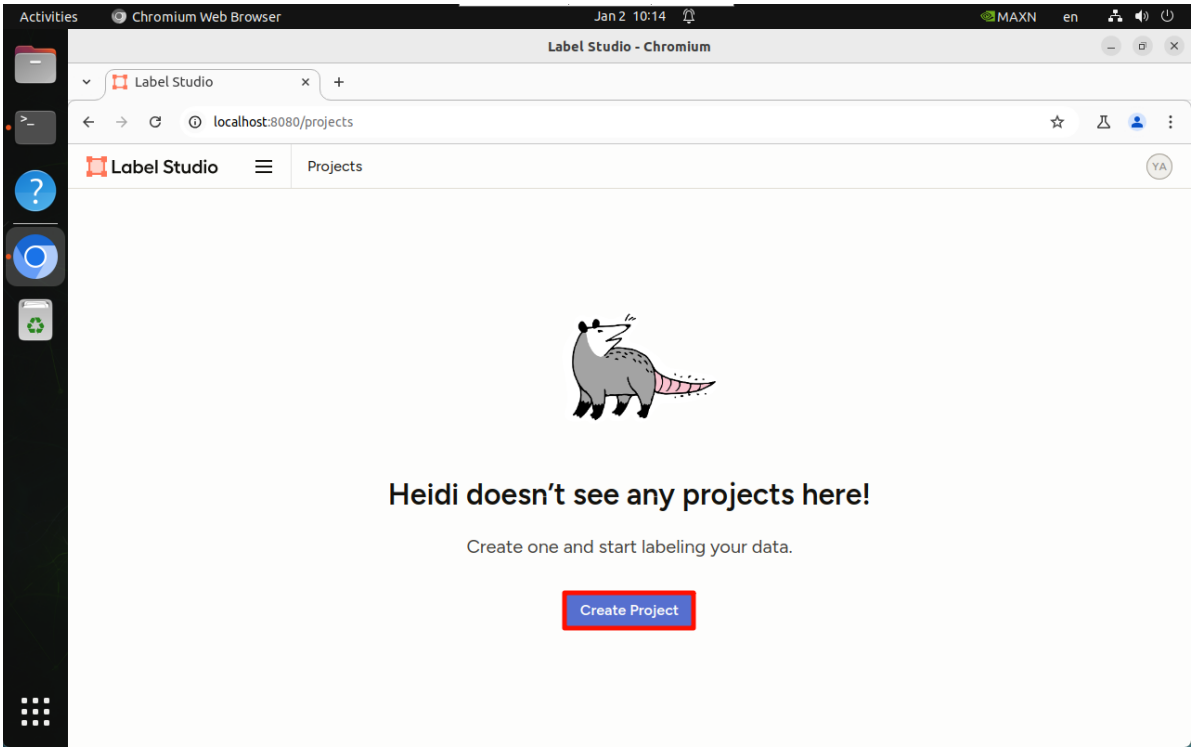
After registration is complete, the website will automatically log in:



3. Dataset Annotation

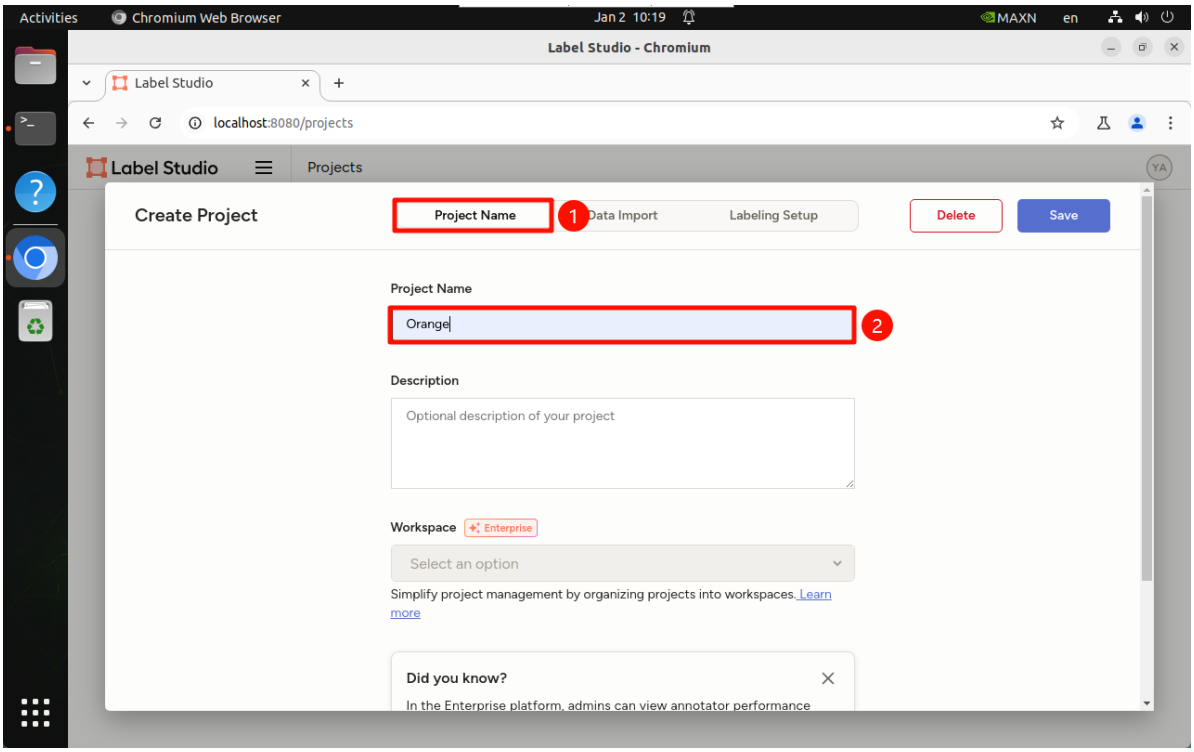
The demonstration dataset is on the board, so access Label Studio on the board.

3.1. Create Project



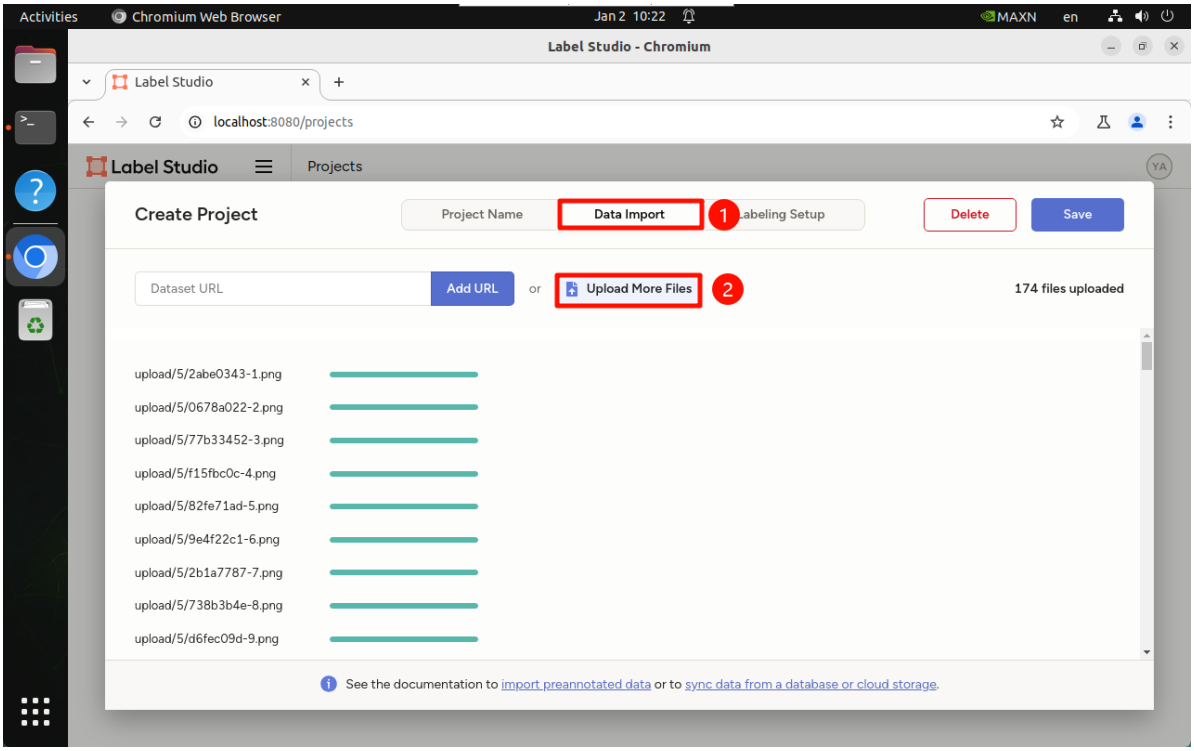
3.2. Project Name

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



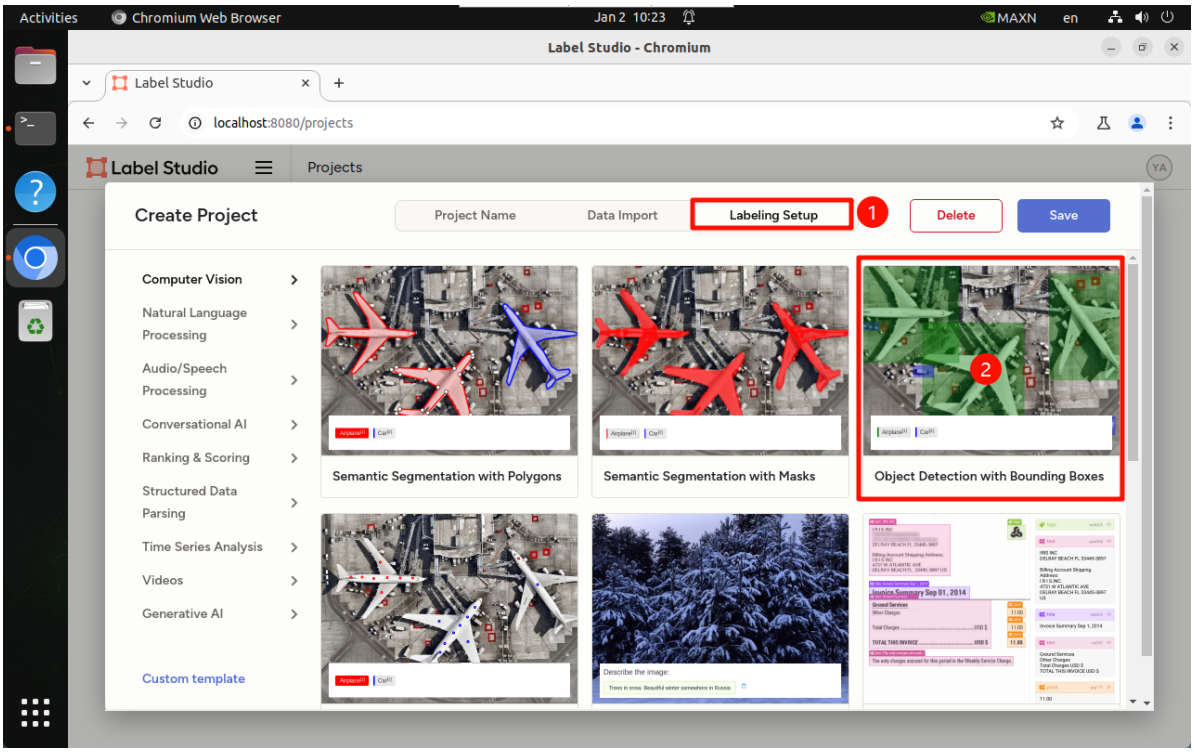
3.3. Import Training Set

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

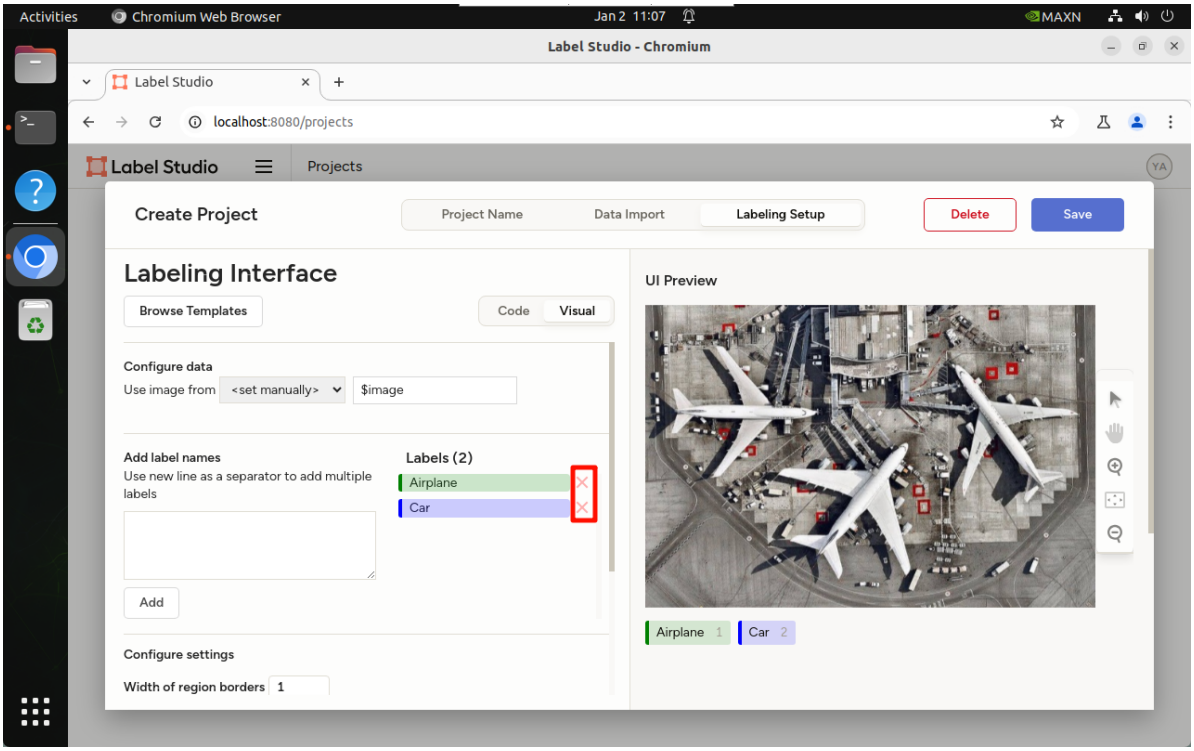


3.4. Label Settings

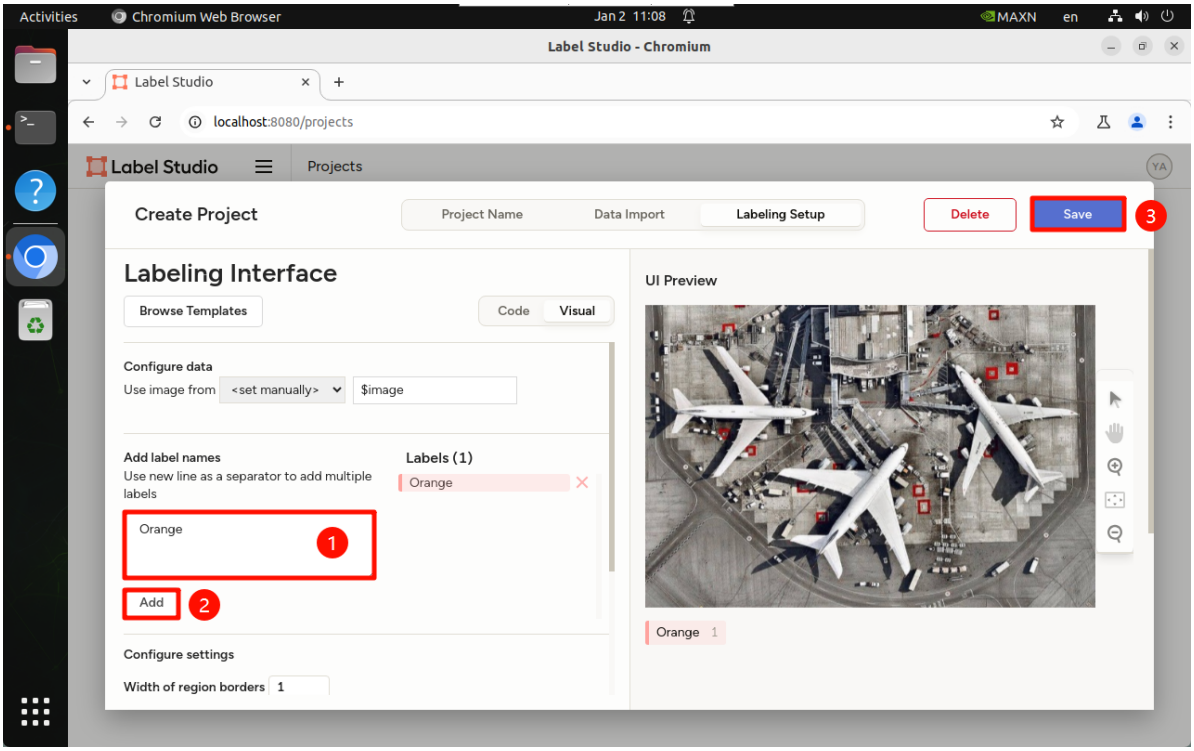
We are demonstrating orange recognition, so select "Object Detection"



Delete built-in labels:

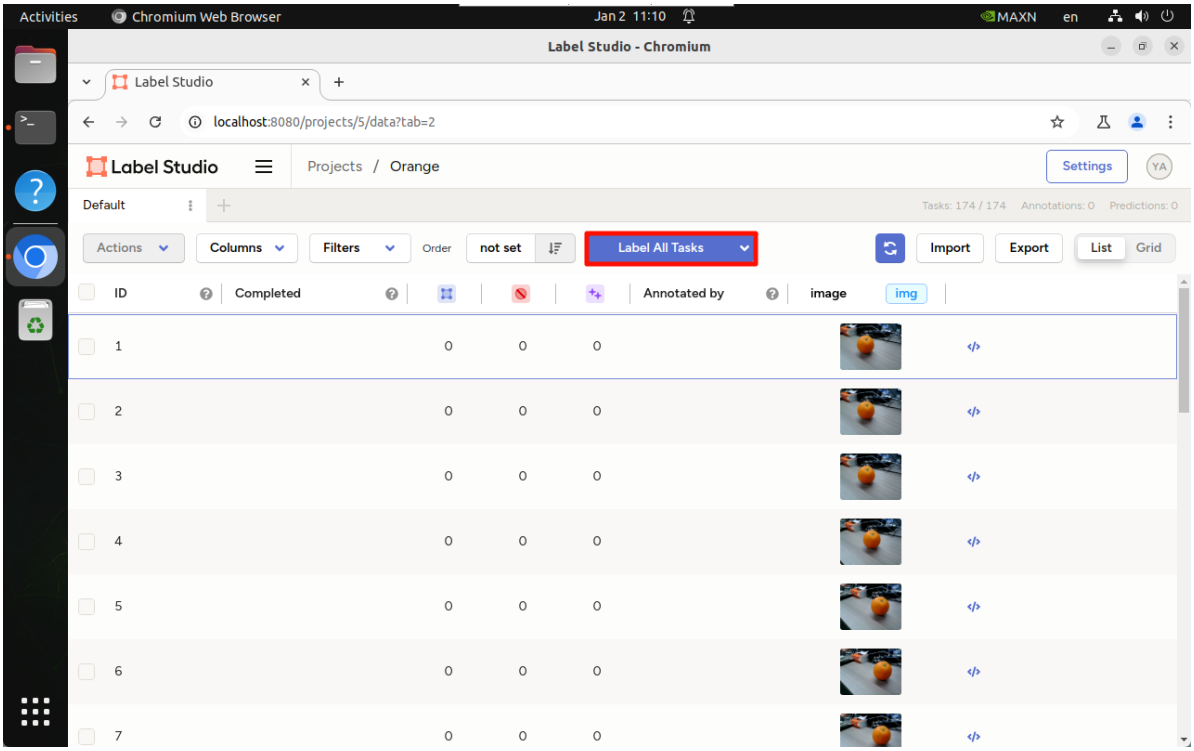


Add new labels:

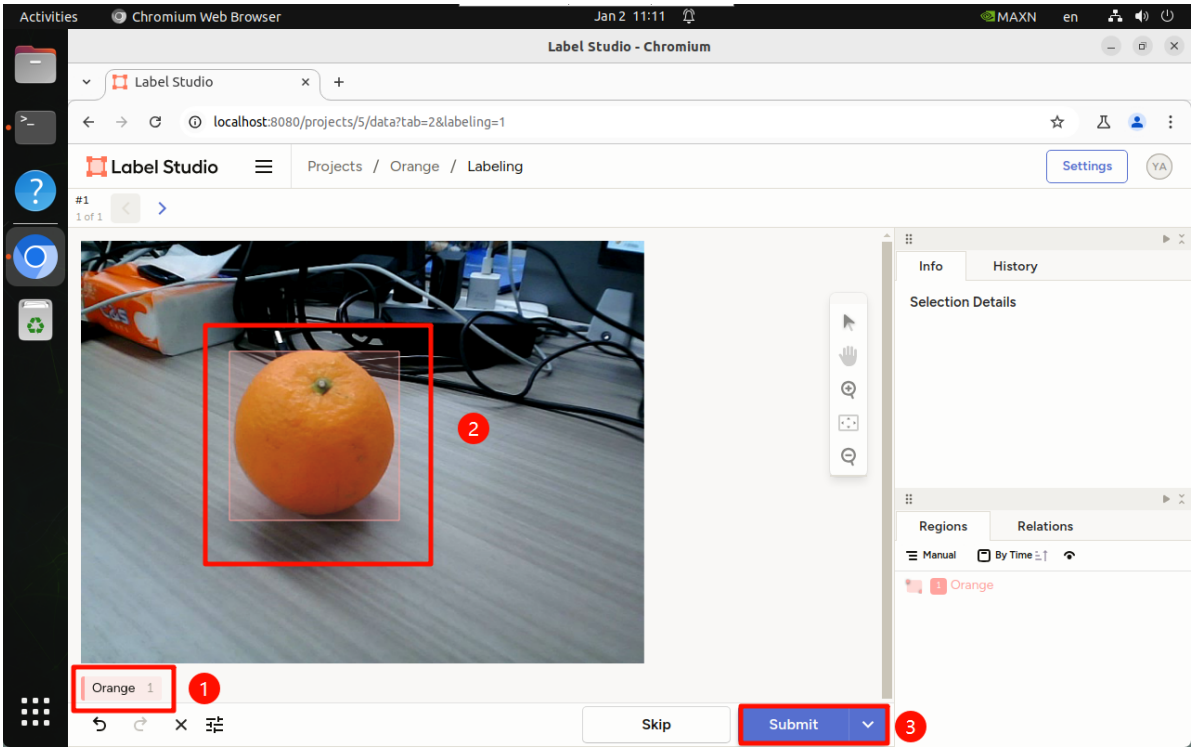


3.5. Dataset Annotation

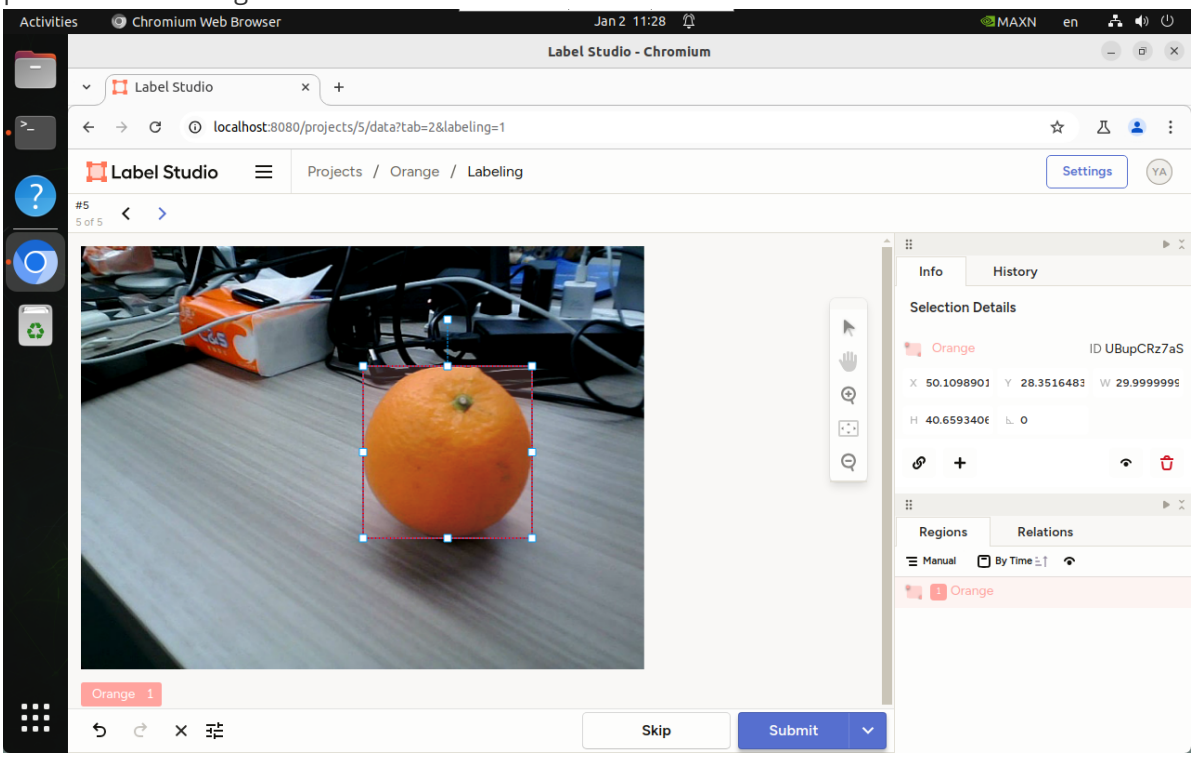
Click "Annotate All Tasks":



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

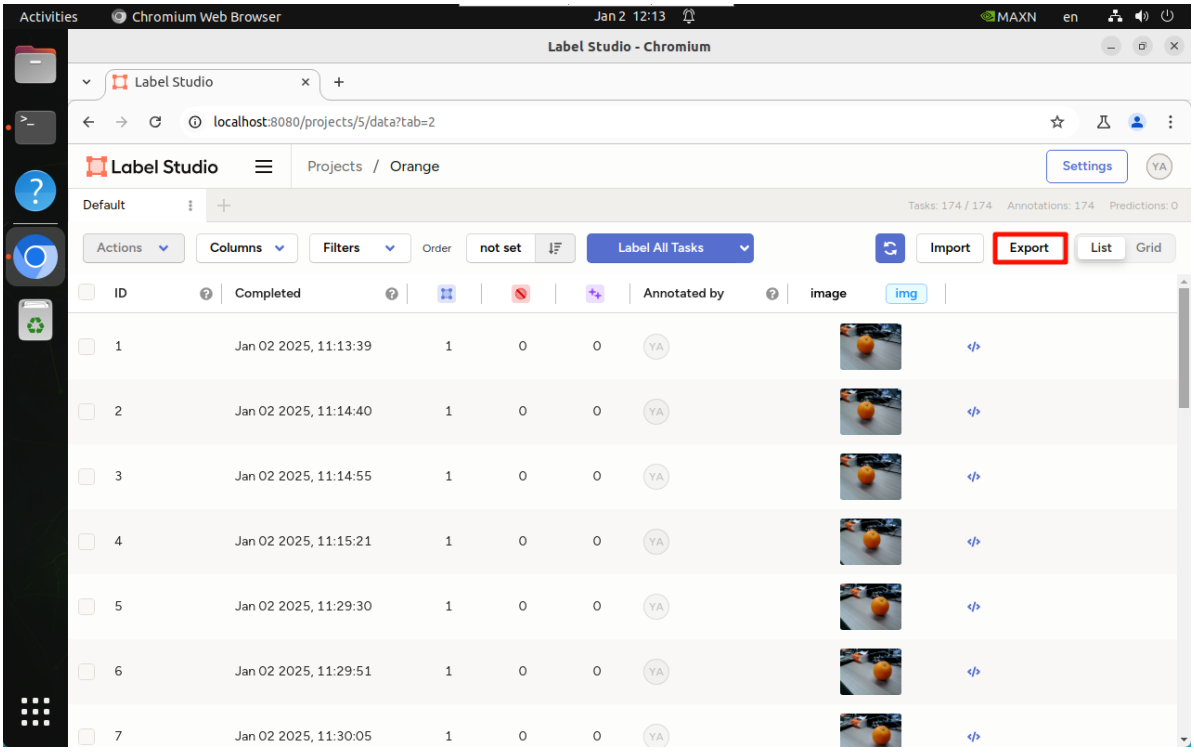


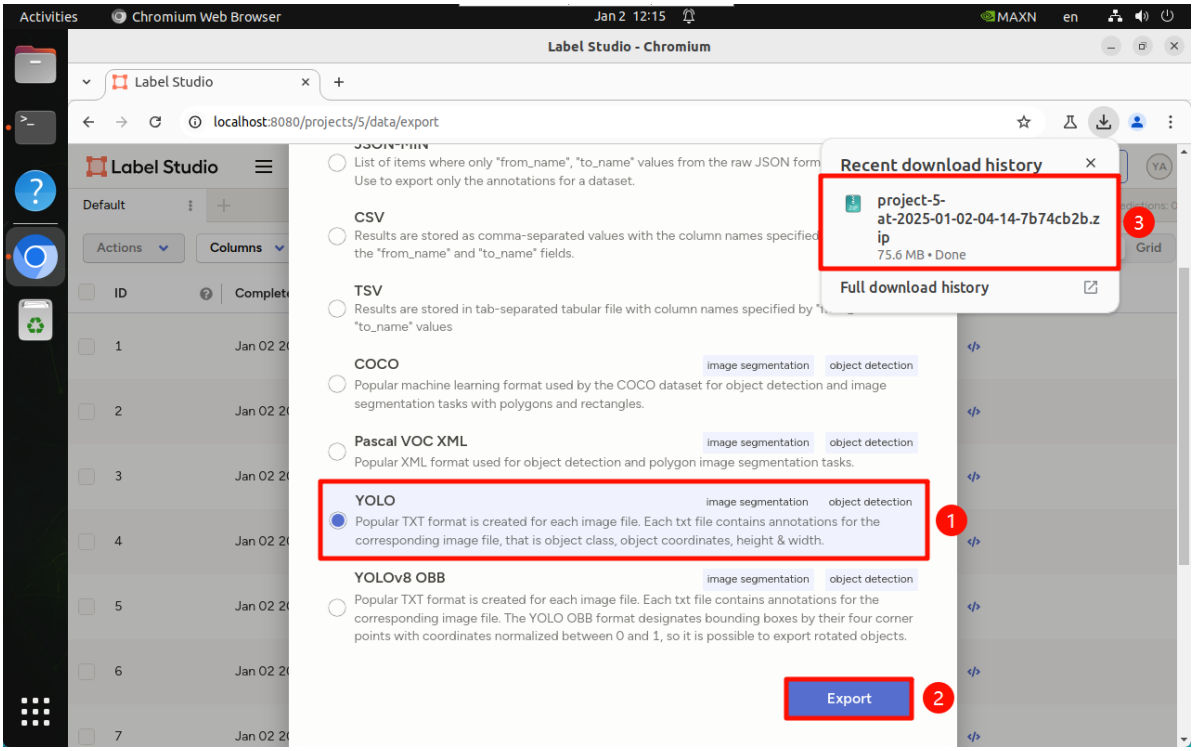
If the selected object is not a complete orange, you can click on the selected box to readjust the position of each edge:



3.6. Export Dataset

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





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
```

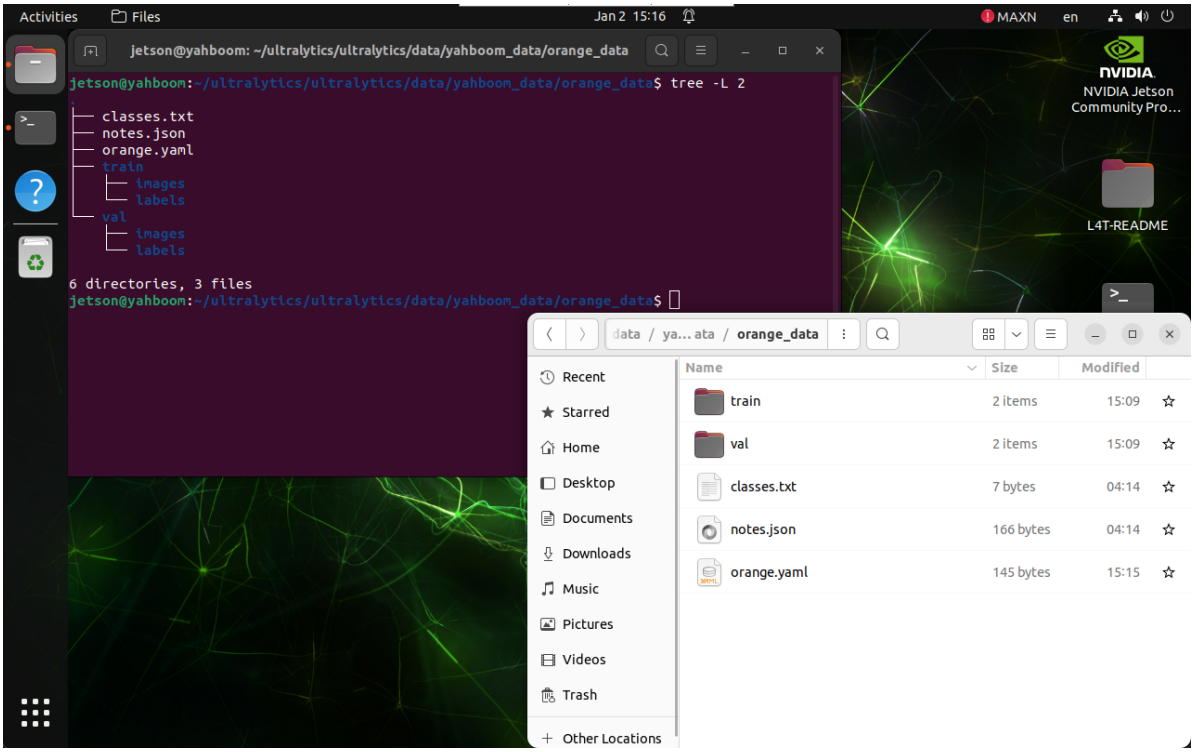
3.8. Dataset Configuration File

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

```yaml
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