Dataset annotation

To identify specific objects or improve the accuracy of object recognition, users need to train the model themselves, and the collection and annotation of data sets are an indispensable part.

1. Dataset collection

Description: For recognition in specific situations, users can use a camera to record videos of objects from various angles, and then capture pictures from the video as training sets

1.1. Extract images

Provide a simple example: extract a frame of image from the video every 15 frames as a data set

```
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

Sample Video：/home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_video

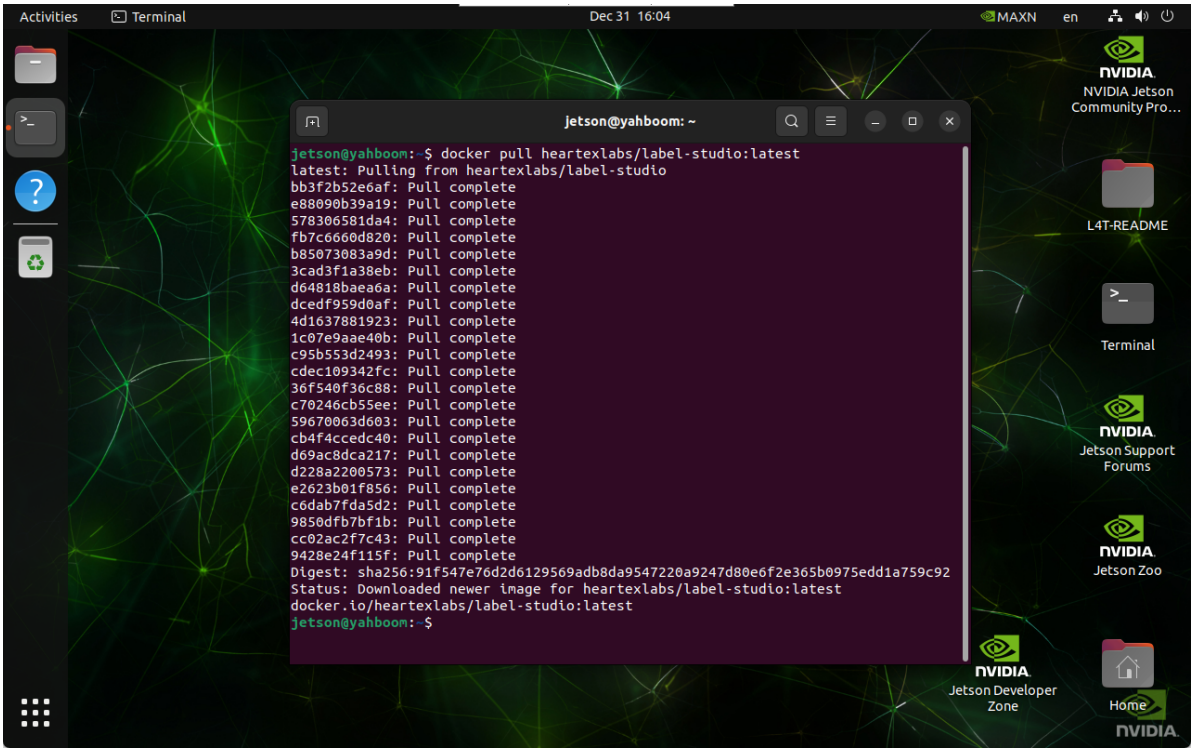
Sample image：/home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_pic

2. Label Studio Usage

Label Studio is an open source data annotation platform used 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. Launch Label Studio

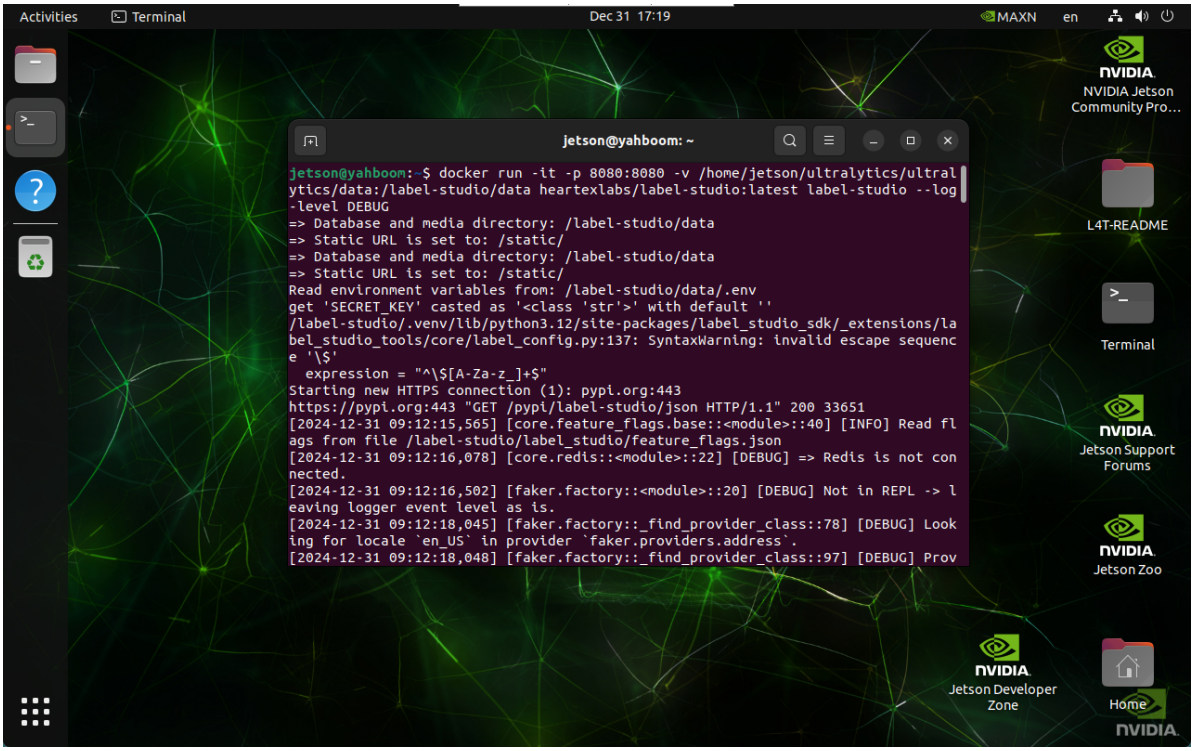
2.2.1. Shared folder permissions

Share the prepared dataset to the folder corresponding to 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
```

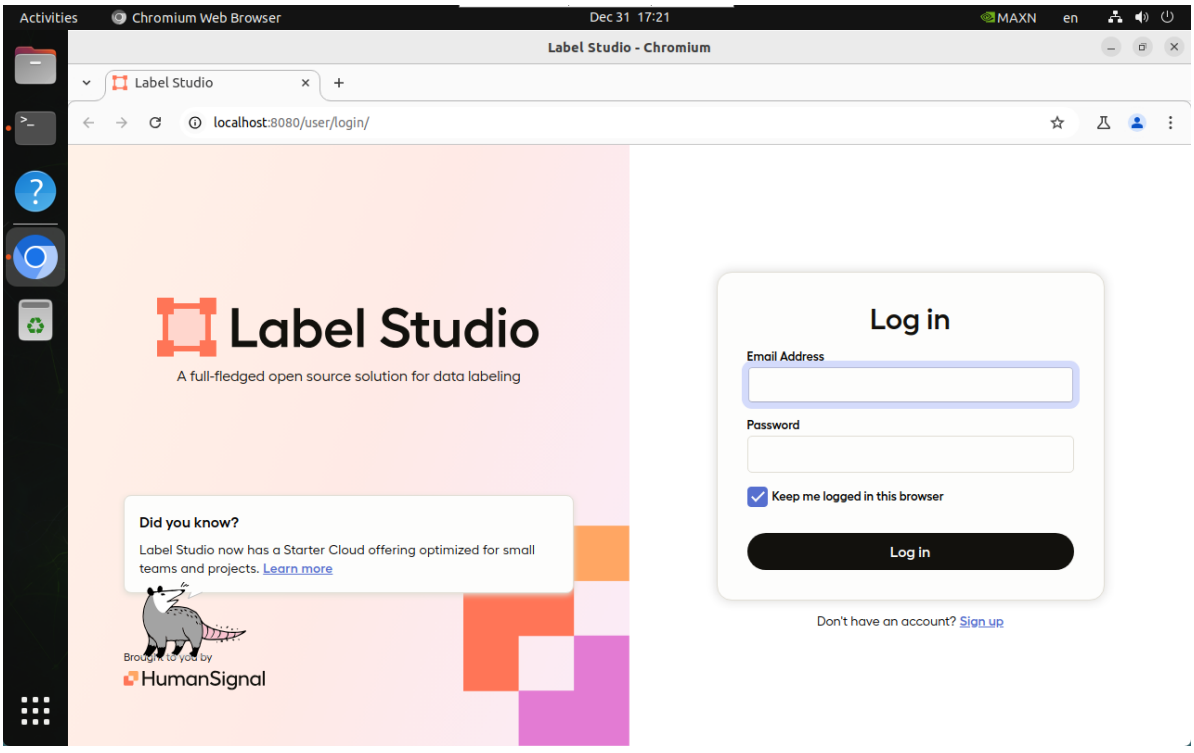


2.2.3. Access Label Studio

Host

Access method for Jetson Orin series development board. You need to enter your username and password for the first time.

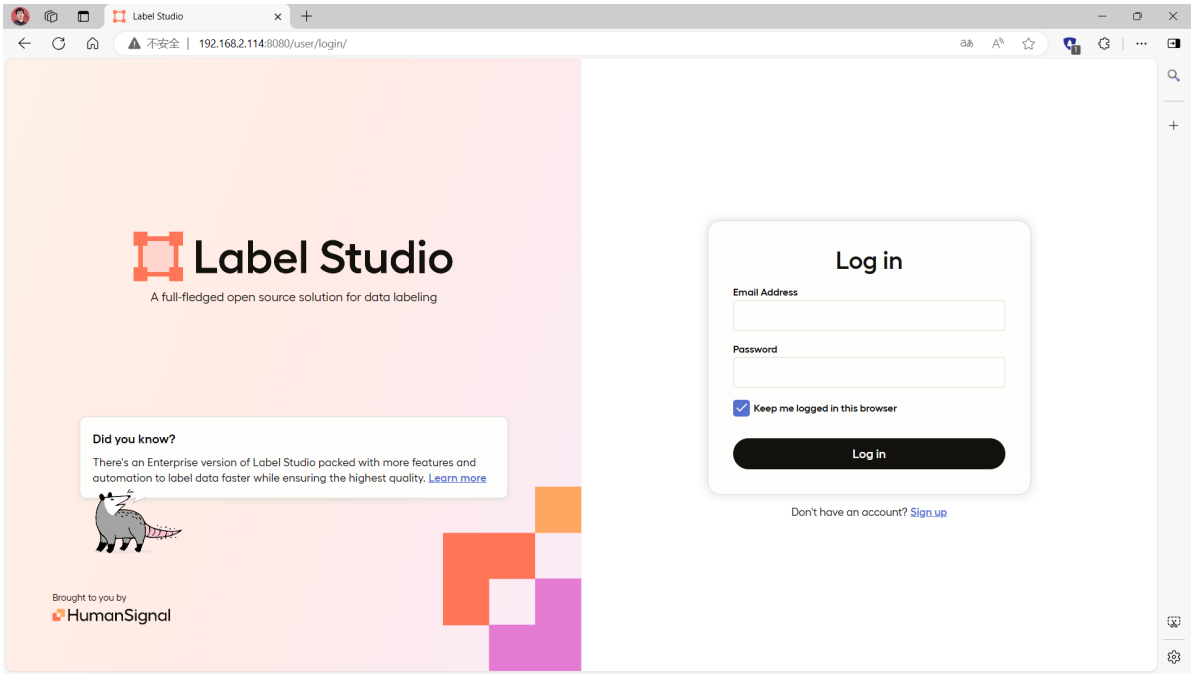
```
http://localhost:8080/
```



Same LAN

Devices in the same LAN can be accessed based on the motherboard IP:8080

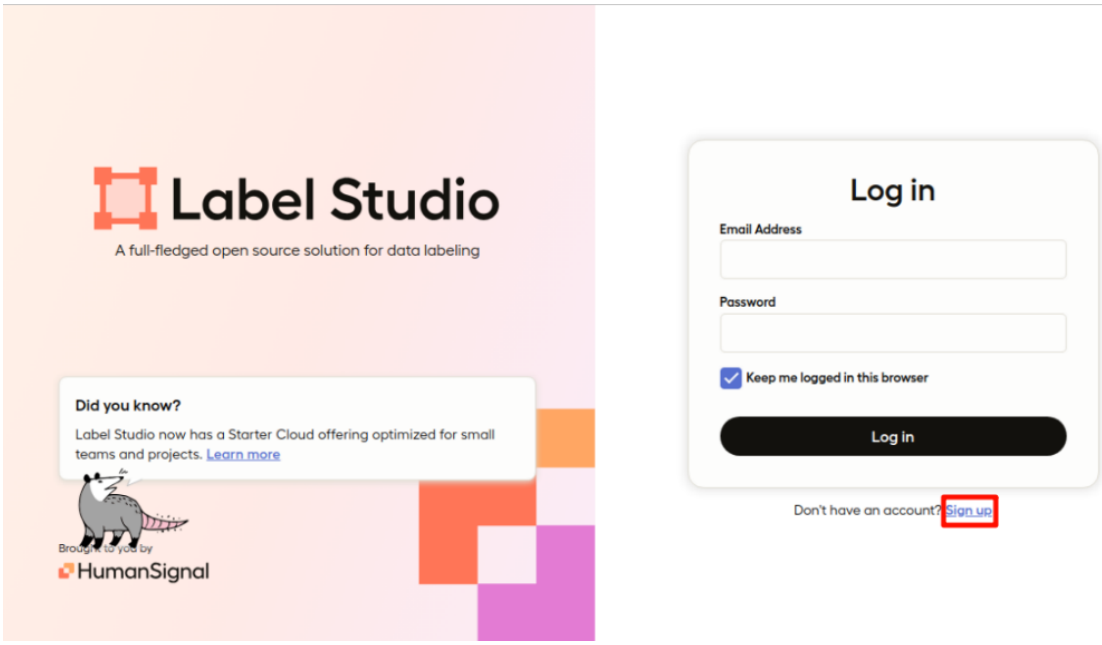
```
http://motherboard IP:8080/ # For example: http://192.168.2.114:8080/
```

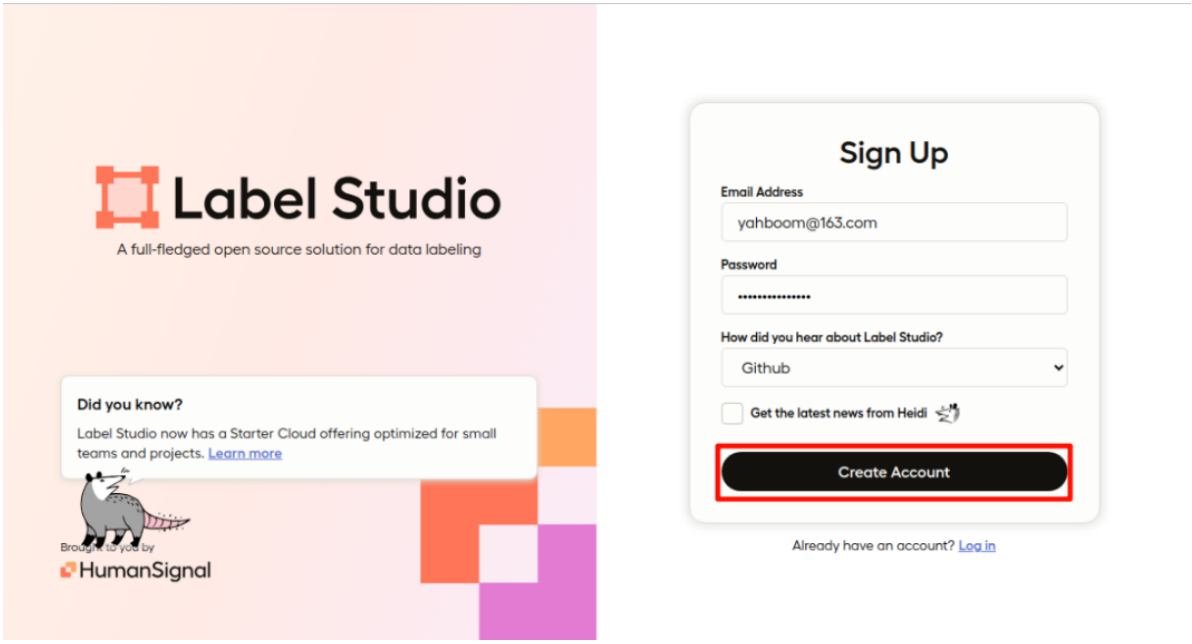


First time use

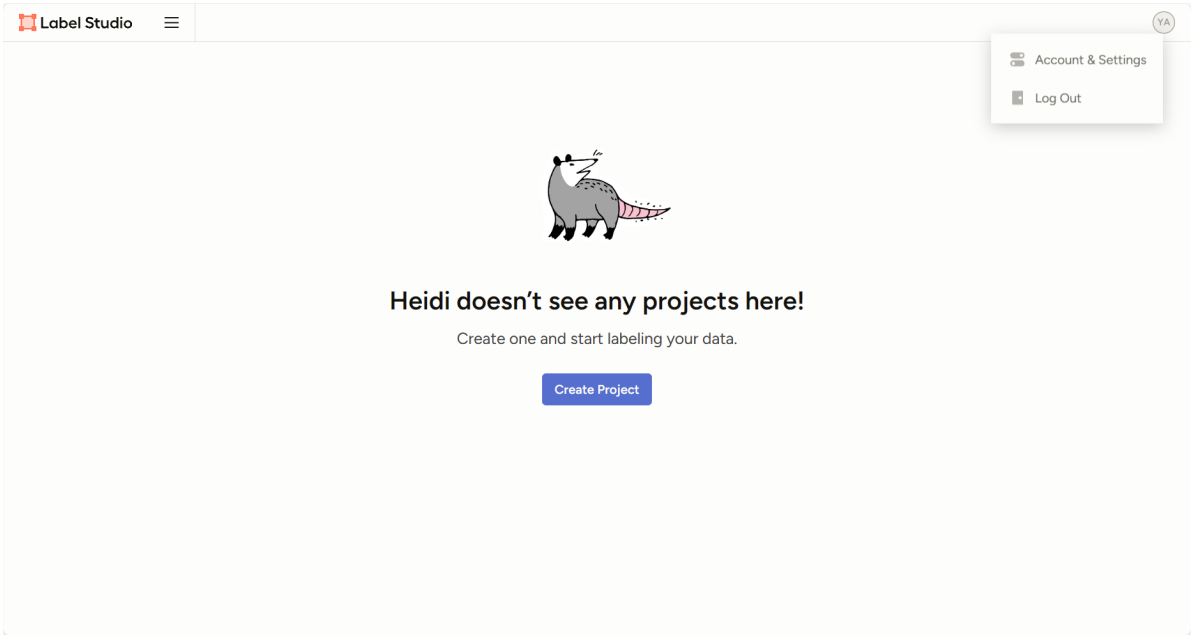
For the first time use, you need to register and log in to your account, and the account information is set by factory settings (the account information is false):

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





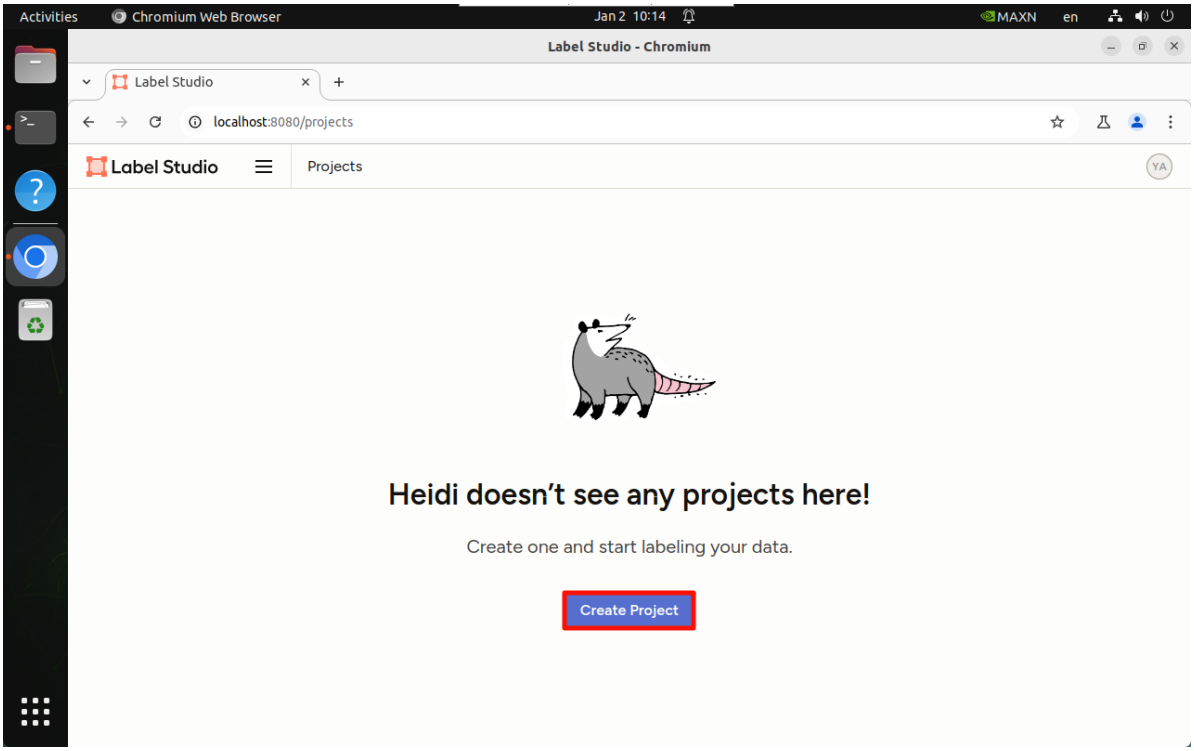
After registration is completed, the website will automatically log in:



3. Dataset labeling

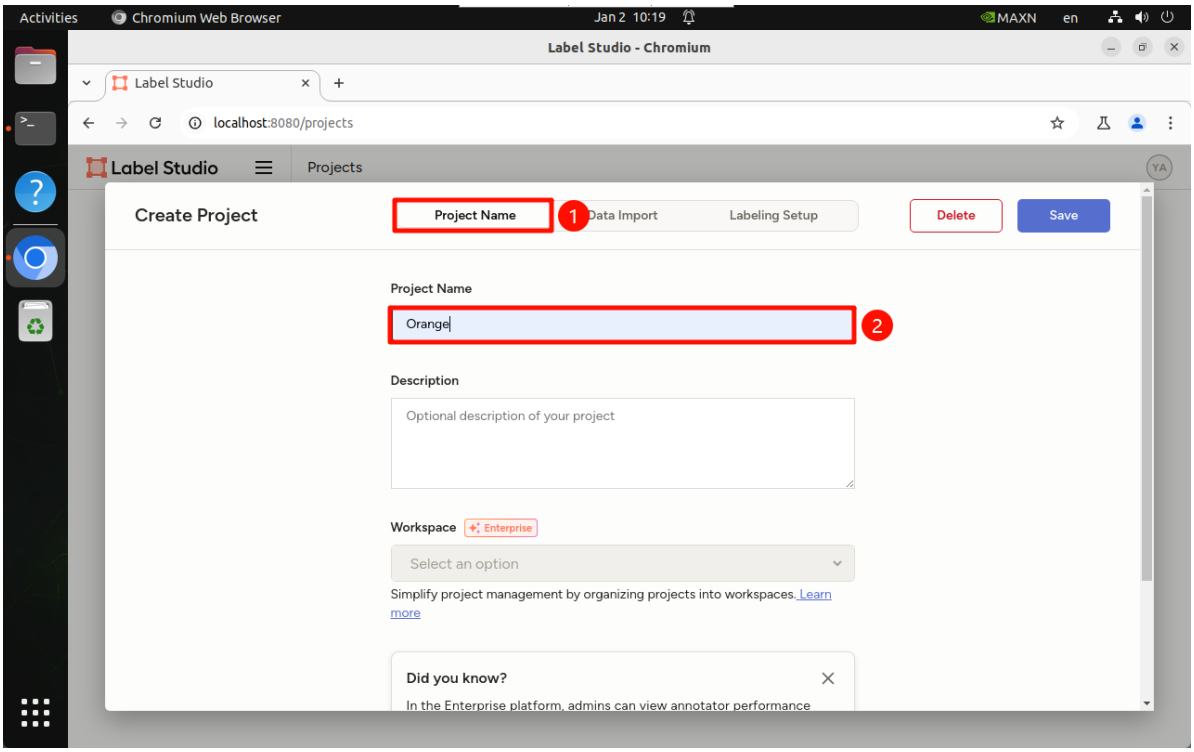
The demonstration dataset is on the main board, so access Label Studio on the main board.

3.1. Create a project



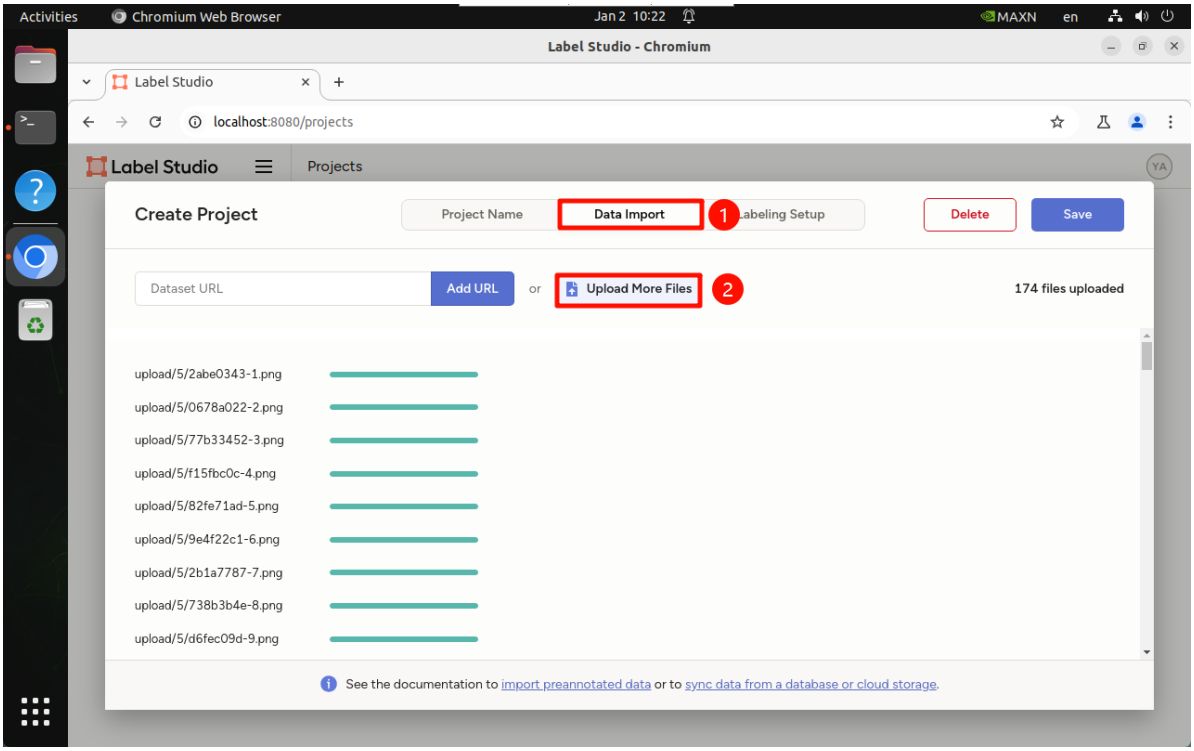
3.2. Project name

You can name the project as you like, based on your own training set:



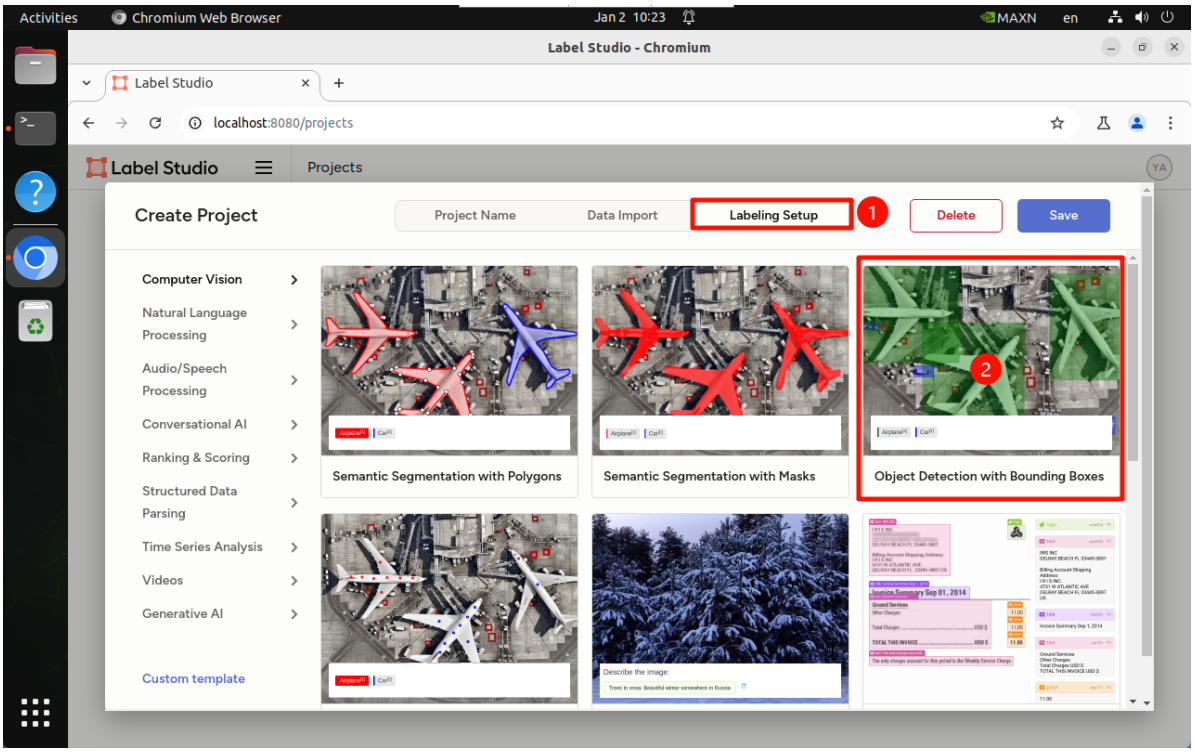
3.3. Import training set

Manually import the prepared dataset: Do not import too many images at a time. It is recommended to select 50-100 images each time and then import them multiple times.

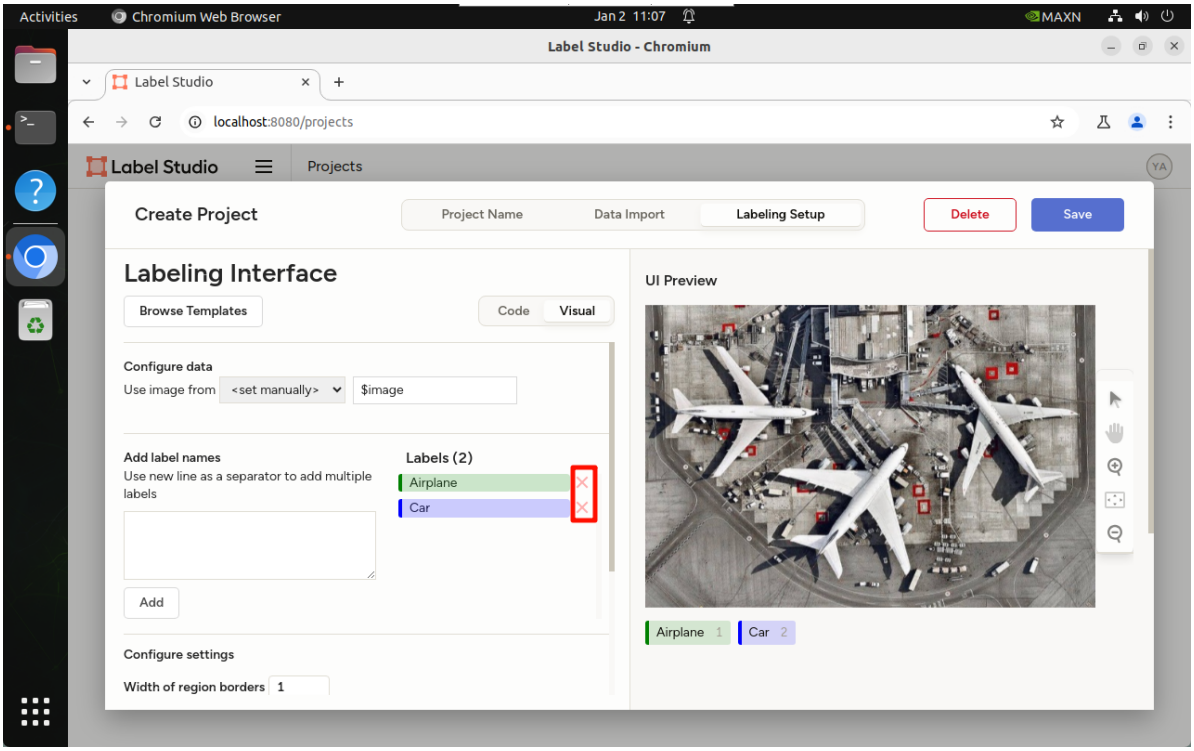


3.4, Label setting

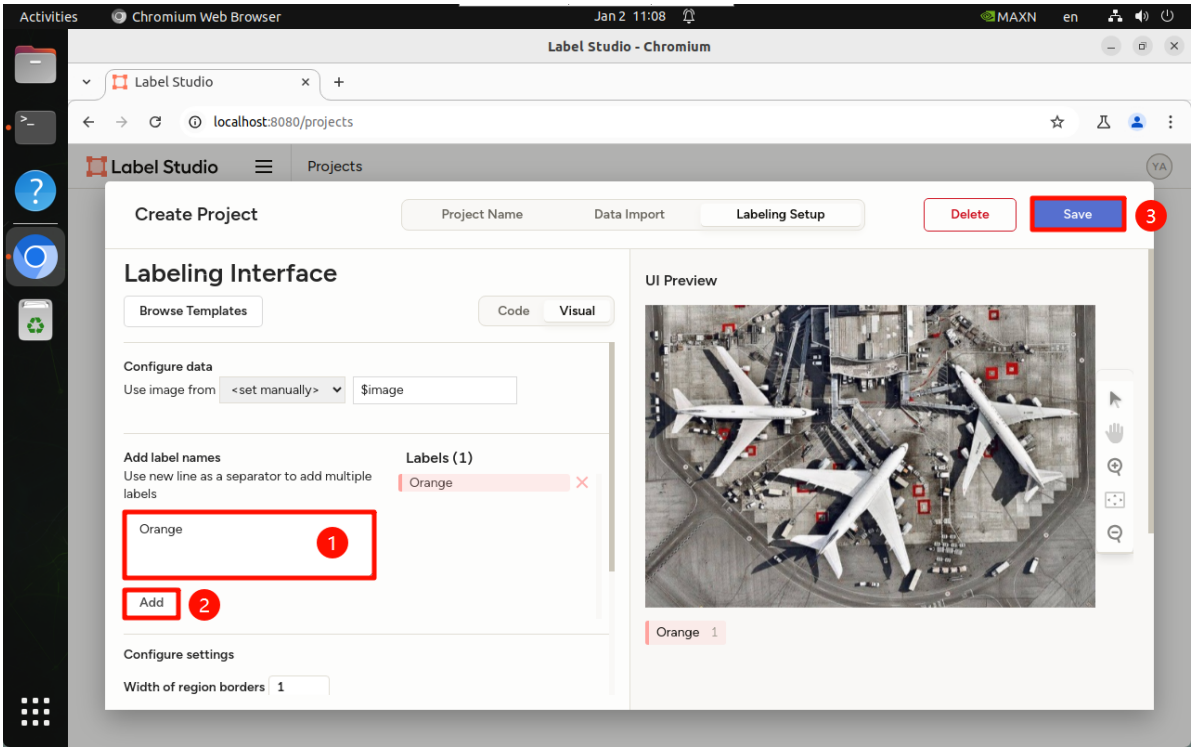
We are demonstrating the recognition of oranges, so select "Object Detection"



To delete a built-in tag:

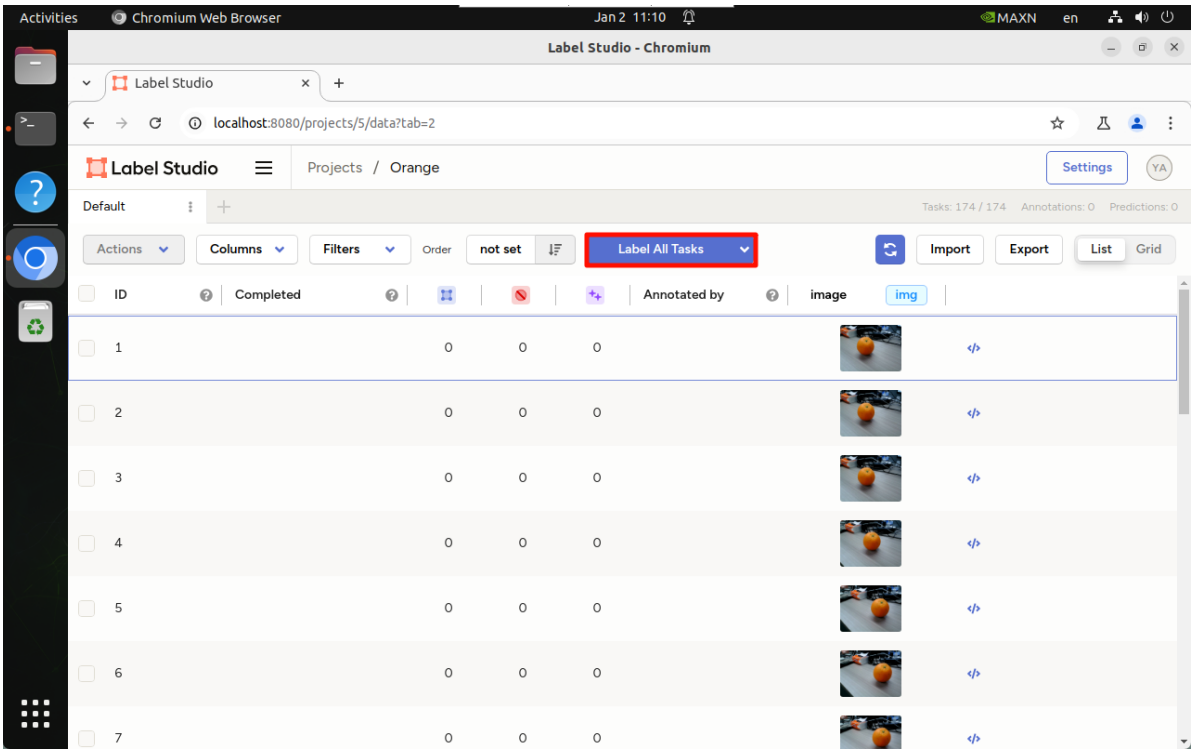


Add new tags:

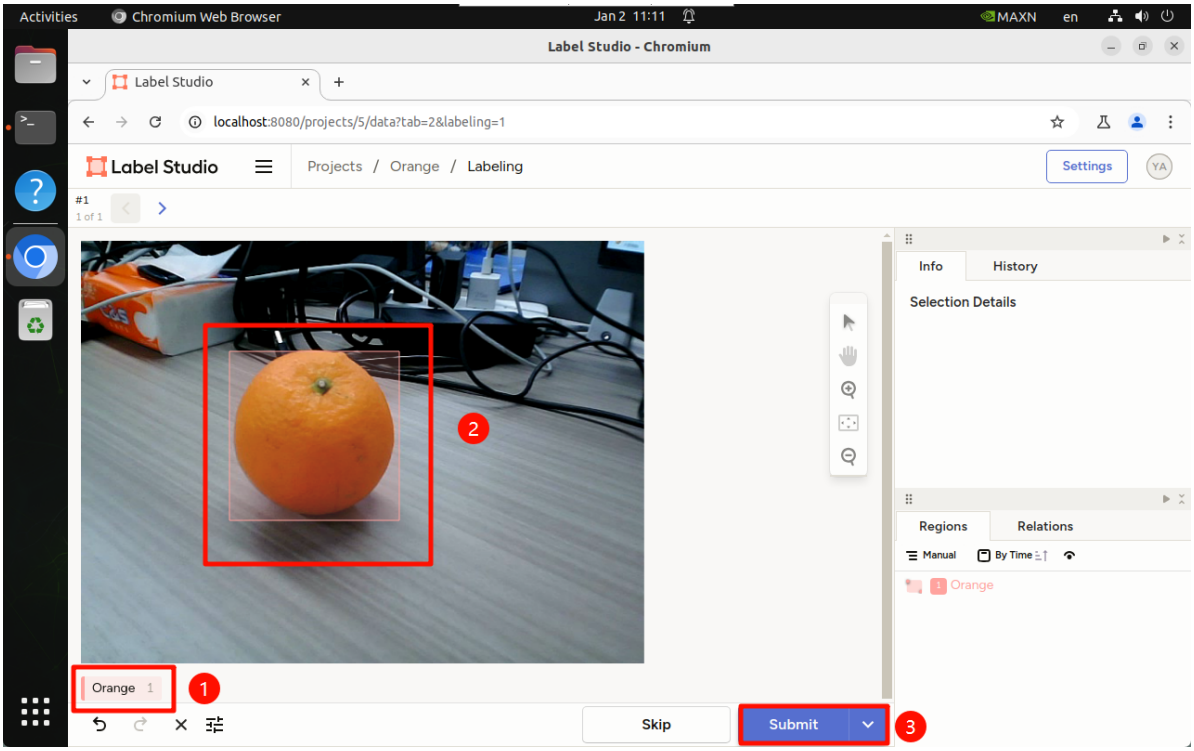


3.5. Dataset annotation

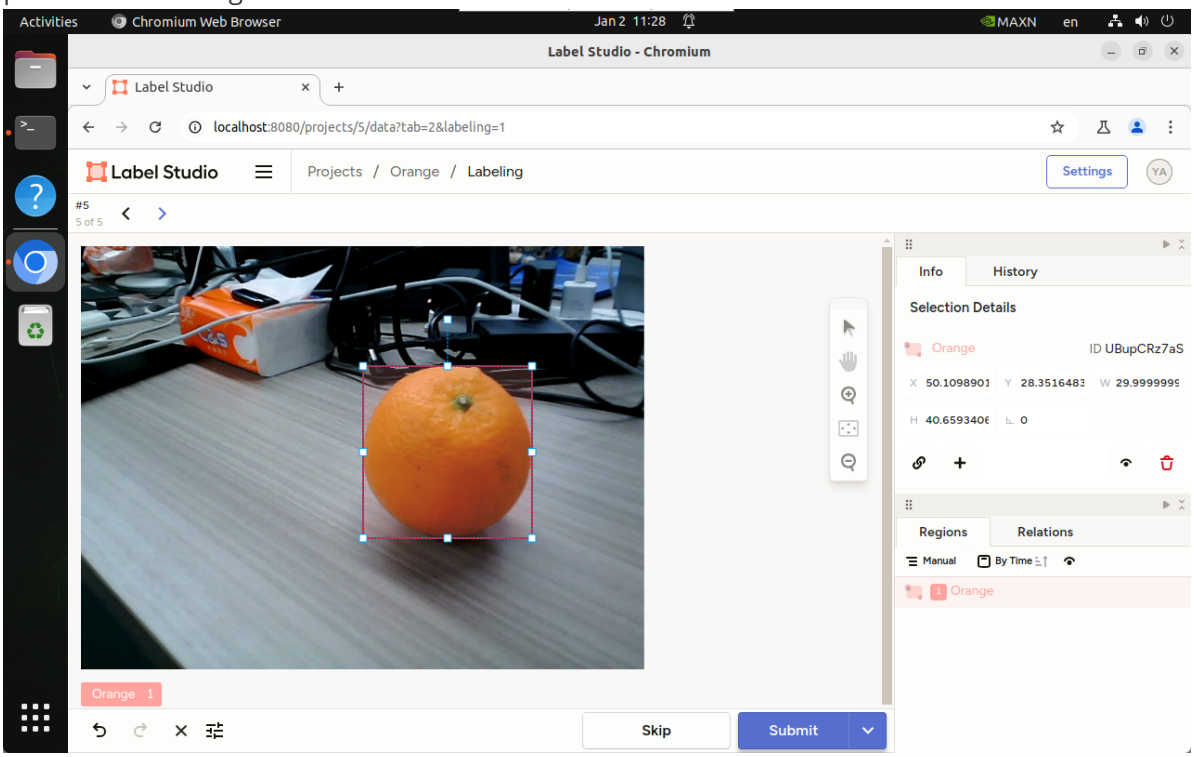
Click to annotate all tasks:



Labeling steps: first click the label "Orange" and then use the mouse to select the corresponding orange in the picture. After completion, click Submit to enter the next labeling

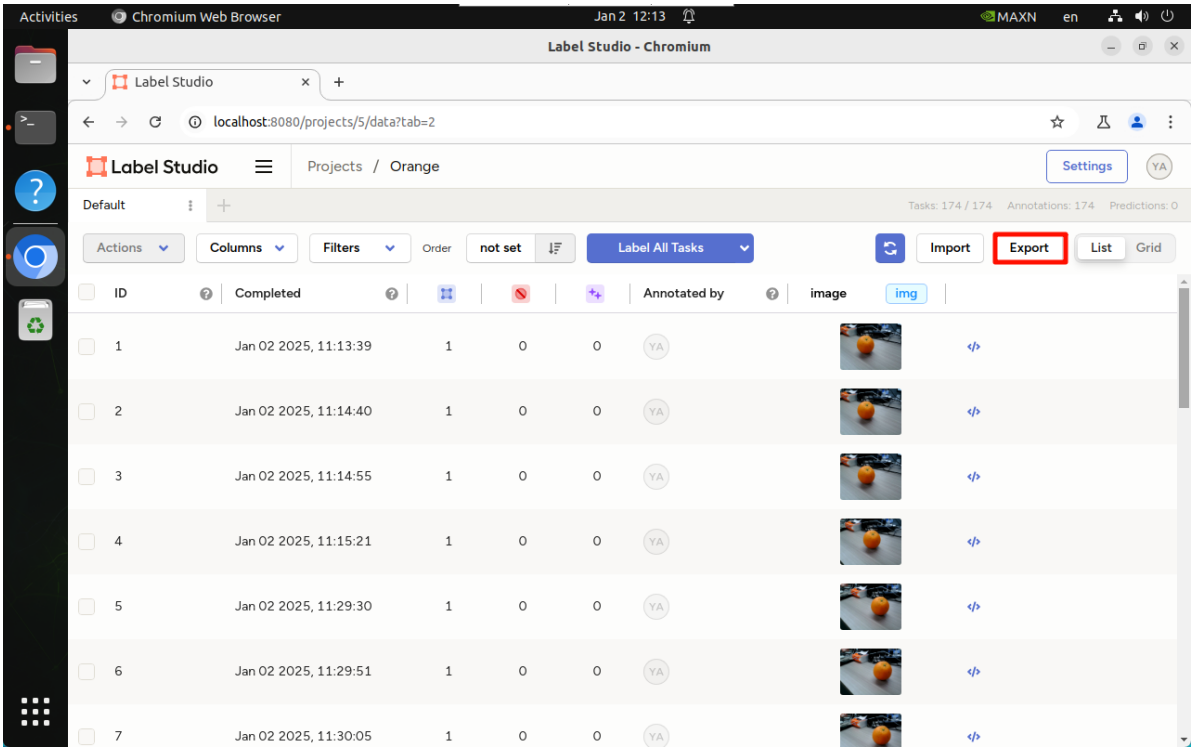


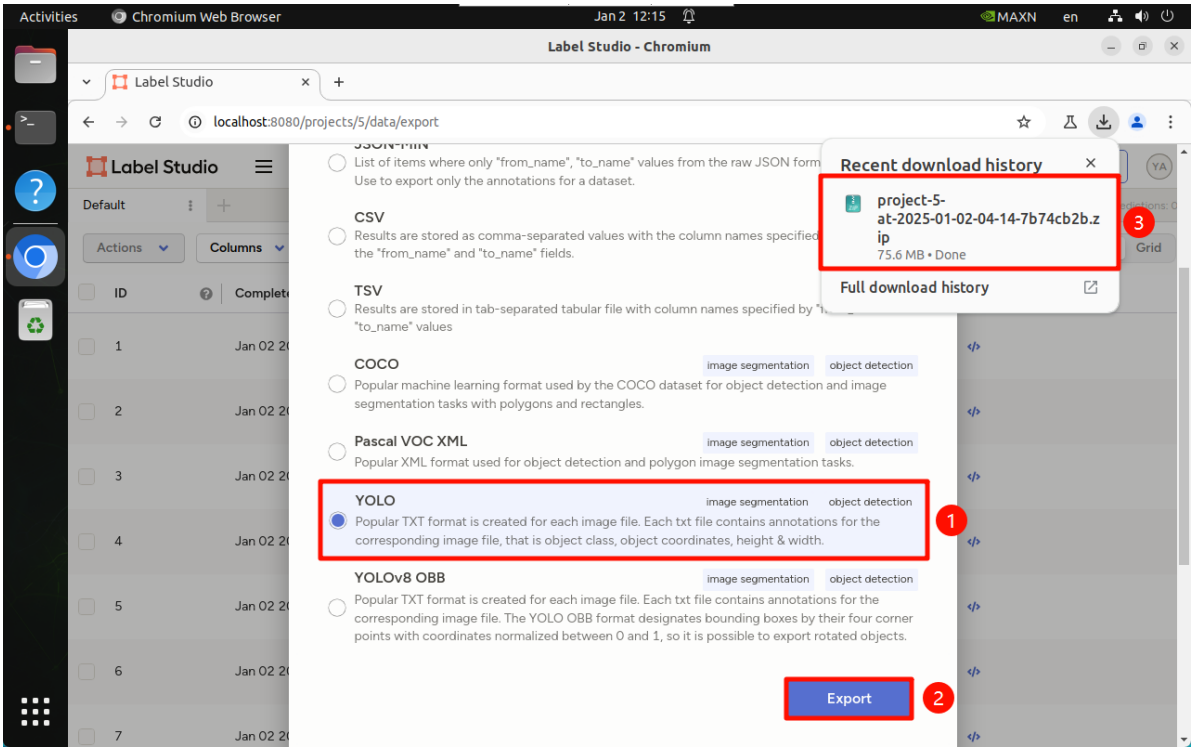
If the selected object is not the finished orange, you can click on the selected box to readjust the positions of the edges:



3.6. Export dataset

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





3.7. Dataset directory framework

Directory framework corresponding to the compressed package exported using Label Studio:

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

We need to export images and labels from Label Studio into train (training set) and val (validation set), with a ratio of about 8:2: the images and labels in train and val need to have one-to-one correspondence in name.

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

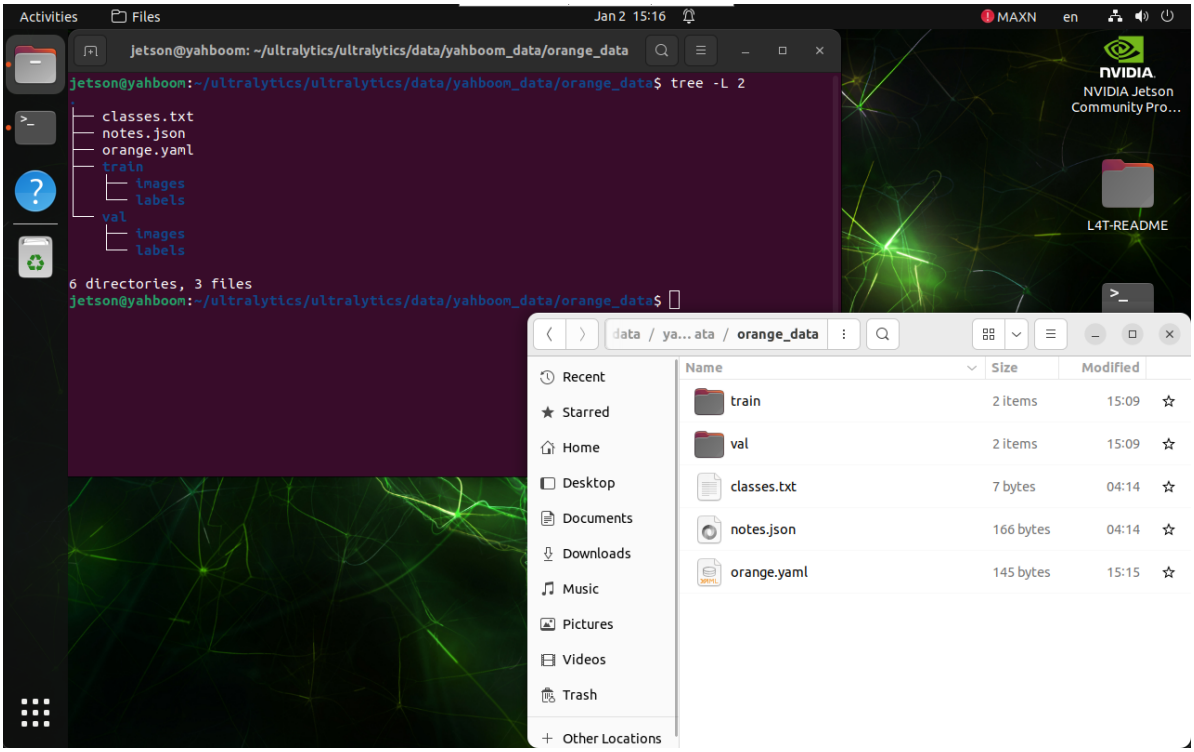
3.8. Dataset configuration file

Contains the paths of the training set and validation set, as well as the number and names of categories.

```
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