SO-ARM101开源6轴机械臂使用文档

1.LeRobot简介

LeRobot是一个开源的机器人学习框架，由**Hugging Face**开发，专门用于机器人行为克隆和强化学习。该框架为研究人员和开发者提供了一个统一的平台，用于训练、部署和评估机器人策略。

1.1 核心特性

1. **多模态数据支持**：支持视觉、触觉、音频等多种传感器数据
2. **灵活的策略架构**：支持行为克隆、强化学习等多种学习范式
3. **丰富的机器人支持**：兼容多种机器人平台和硬件配置
4. **云端训练支持**：支持分布式训练和云端部署
5. **易于扩展**：模块化设计，便于添加新的机器人类型和算法

1.2 技术架构

LeRobot基于**PyTorch**构建，采用现代深度学习技术栈：

1. **数据处理**：高效的数据加载和预处理管道
2. **模型训练**：支持多种神经网络架构和训练策略
3. **实时推理**：优化的推理引擎，支持实时机器人控制
4. **可视化工具**：丰富的数据可视化和训练监控工具

1.3 应用场景

1. **工业自动化**：装配、分拣、焊接等工业任务
2. **服务机器人**：家庭服务、医疗辅助、教育机器人
3. **研究开发**：机器人学习算法研究和原型验证
4. **教育培训**：机器人学习和人工智能教学

1.4 生态系统

LeRobot与**Hugging Face**生态系统深度集成：

1. **模型共享**：通过**Hugging Face Hub**分享训练好的模型
2. **数据集管理**：统一的数据集存储和版本控制
3. **社区支持**：活跃的开源社区和丰富的文档资源
4. **持续更新**：定期发布新功能和性能优化

1.5 发展前景

SO-ARM101开源6轴机械臂代表了机器人学习领域的重要发展方向，通过降低机器人学习的门槛，加速了机器人技术的普及和应用。随着人工智能技术的不断发展，**SO-ARM101开源6轴机械臂**将继续推动机器人学习技术的创新和进步。

2.硬件与环境准备

2.1 硬件清单

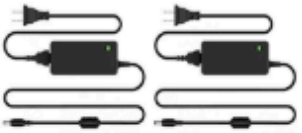
DIY版（散件）

用户需自备3D打印结构件，课程资料里面附带三维图纸，可自行3D打印。

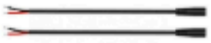
HX-30HM总线舵机×6

HX-10HM总线舵机×6

多功能驱动模块×2



12V 5A适配器×2
(DC4.0*1.7)



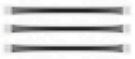
电源线 ×2
(DC4.0*1.7母，
长200mm)

USB集线器
(4孔，长1000mm)

3寸G字夹×4

Type-C数据线×2
(长1000mm)

拍摄支架



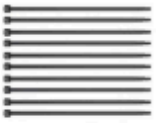
舵机线×3
(长100mm)

舵机线×5
(长160mm)

舵机线×7
(长200mm)

金属主舵盘×14

金属副舵盘×12



扎带×10 (3*150mm)

螺丝刀×2

M2.5*10
双通尼龙柱×10

M3*10沉头螺丝×4

M3*5*7带垫
自攻螺丝×14

M3*6螺丝×120

M2*6自攻螺丝×40

M2.5*4螺丝×20

M2*28半牙自攻螺丝×20

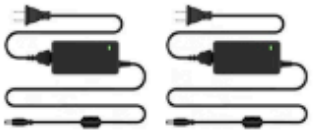
M3螺母×4

入门版（成品）

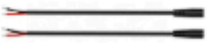
组装调试好发货，到手即用。

主机械臂

从机械臂



12V 5A适配器×2 (DC4.0*1.7)



电源线×2
(DC4.0*1.7母，长200mm)

USB集线器 (4孔，长1000mm)

3寸G字夹×4

Type-C数据线×2 (长1000mm)

配件包

舵机线 (长100mm)

舵机线 (长160mm)

舵机线 (长200mm)

金属主舵盘×2+副舵盘×2

扎带 (3*150mm)

螺丝刀×2

螺丝螺母配件包

2.2 环境配置

2.2.1安装Miniconda

- **windows系统安装**

①下载**Miniconda**包

Miniconda官网安装包

找到**Miniconda3-py311_25.7.0-2-Windows-x86_64.exe**将包下载到电脑上。（或者直接使用：**软件工具&源码程序\软件工具**）

```
Miniconda3-py311_25.7.0-2-Windows-x86_64.exe
```

> **Note**
>
> **因为官网是国外网站，这里推荐使用国内清华大学的镜像源，对于国内的网络友好，下载速度更快!**

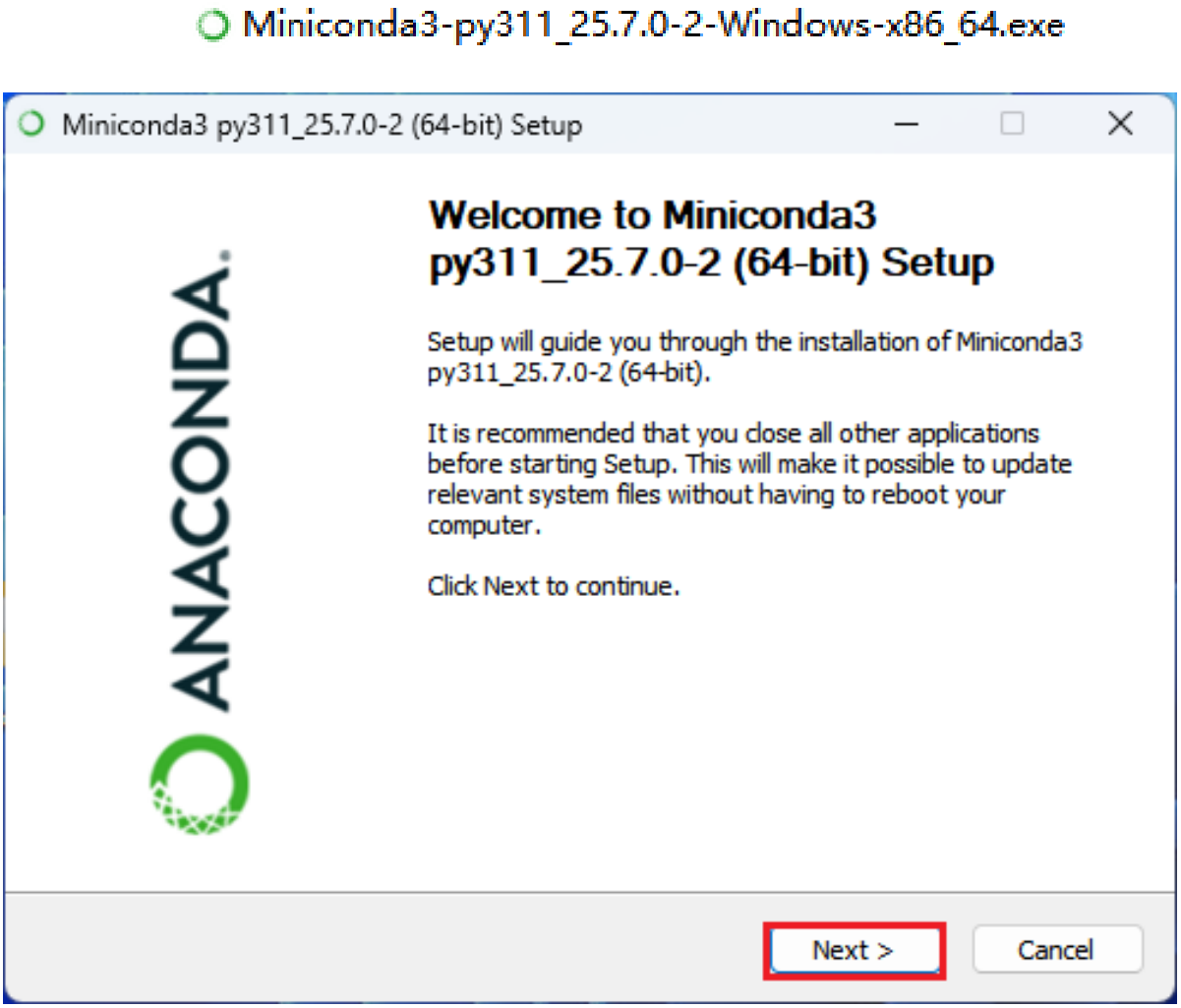
Index of /anaconda/miniconda/ | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror

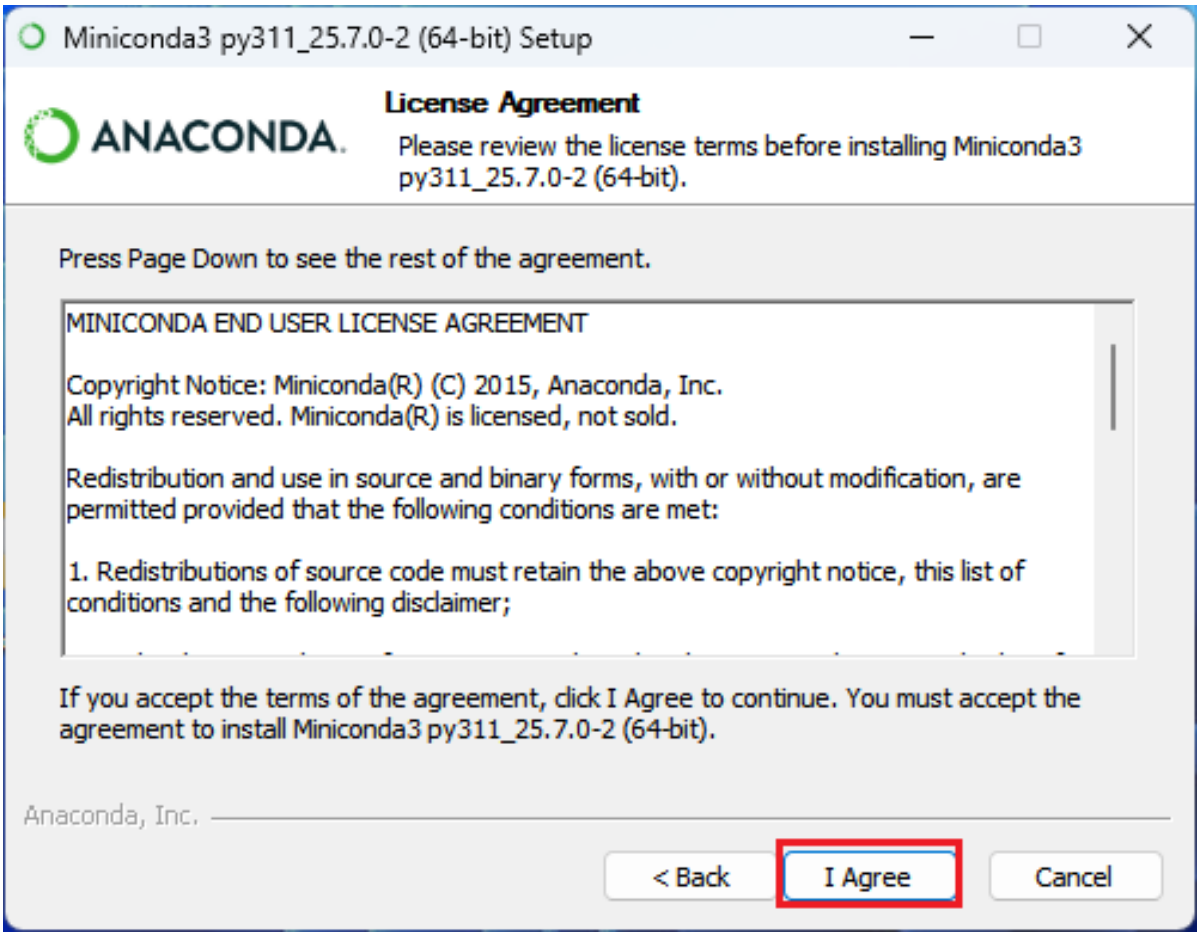
点击上面链接进行下载，这里选择的同样也是**Miniconda3-py310_25.7.0-2-Windows-x86_64.exe**。

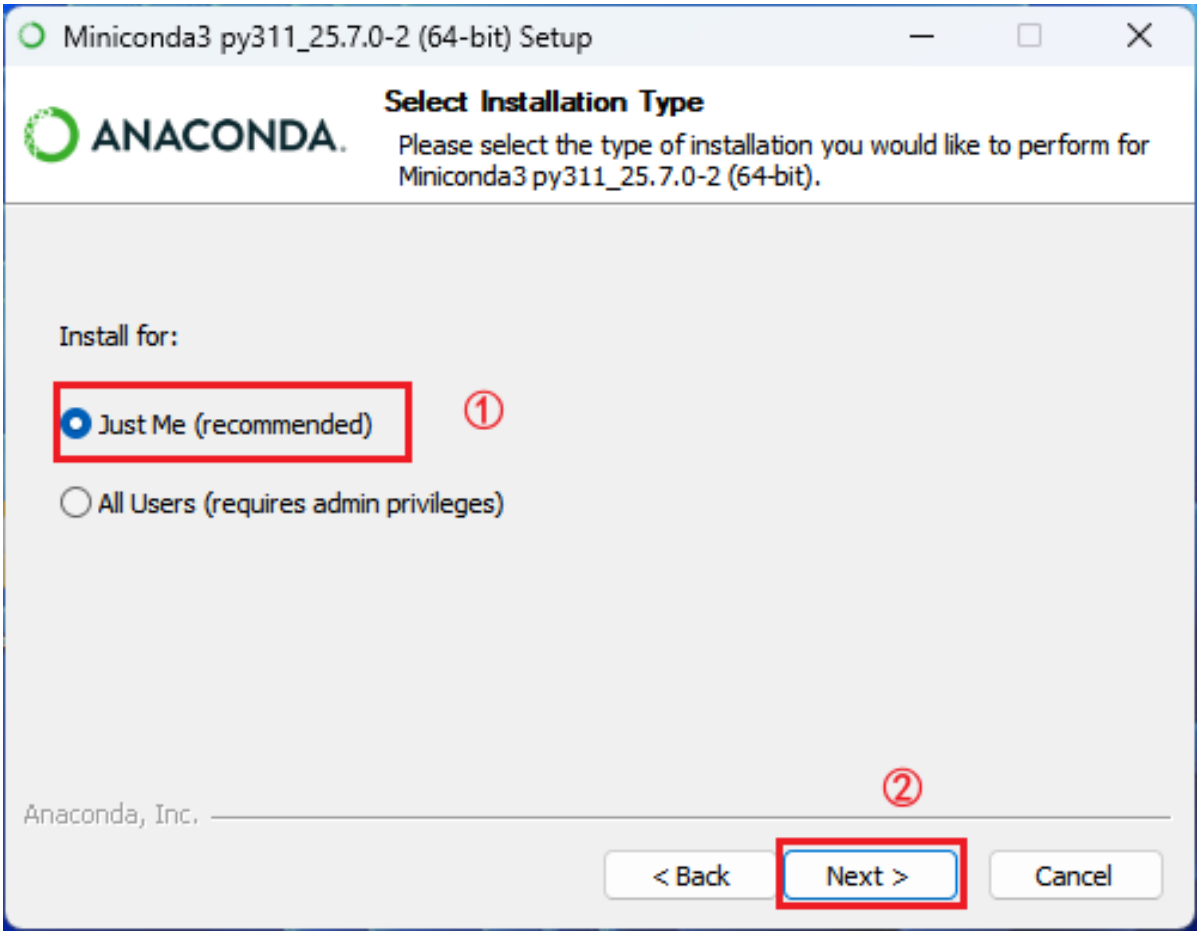
```
Miniconda3-py311_25.7.0-2-Windows-x86_64.exe
```

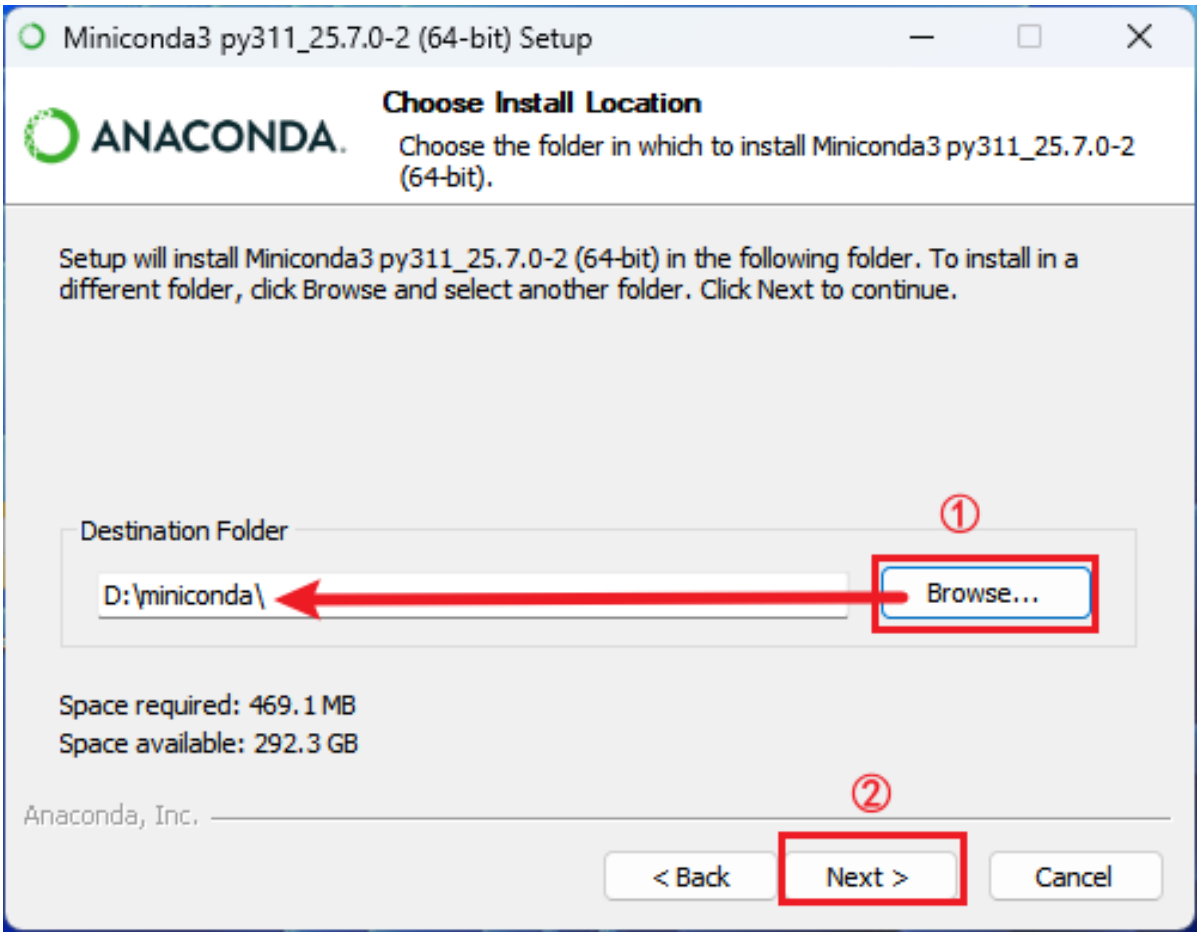
②安装**Miniconda**

找到你所下载的**Miniconda**安装包，双击进行安装。

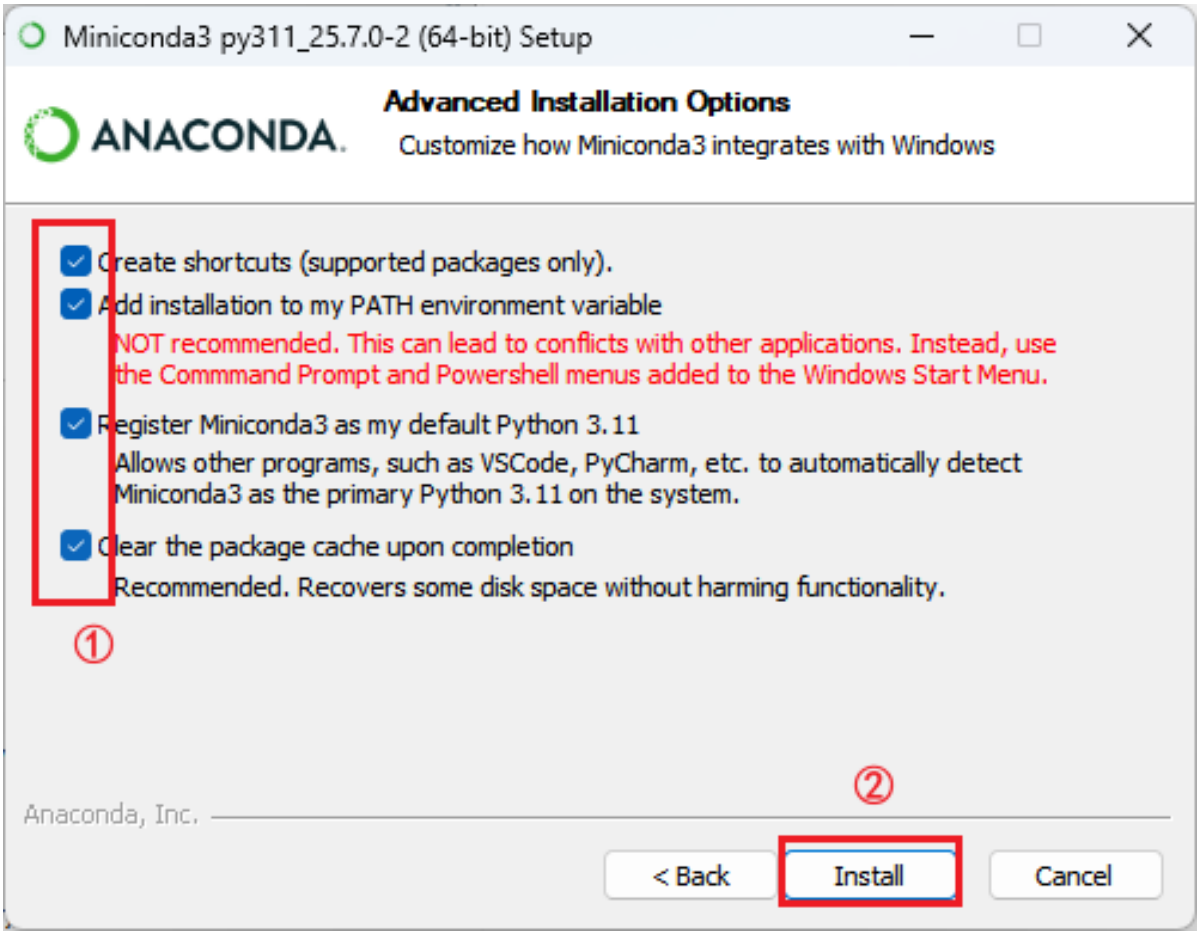


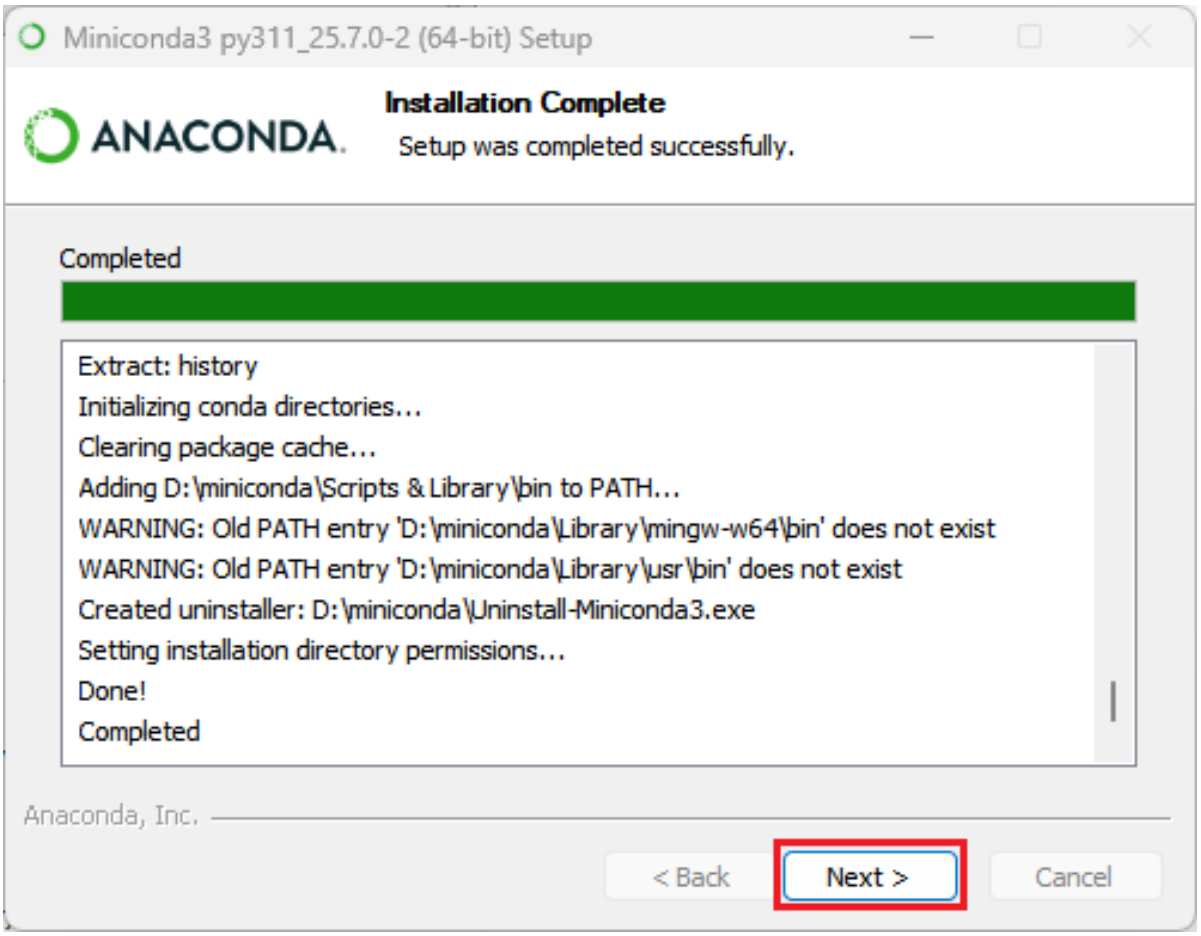

Miniconda3 py311_25.7.0-2 (64-bit) Setup
ANACONDA.
License Agreement
Please review the license terms before installing Miniconda3 py311_25.7.0-2 (64-bit).
Press Page Down to see the rest of the agreement.
MINICONDA END USER LICENSE AGREEMENT
Copyright Notice: Miniconda(R) (C) 2015, Anaconda, Inc.
All rights reserved. Miniconda(R) is licensed, not sold.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer;
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Miniconda3 py311_25.7.0-2 (64-bit).
Anaconda, Inc.
< Back
I Agree
Cancel

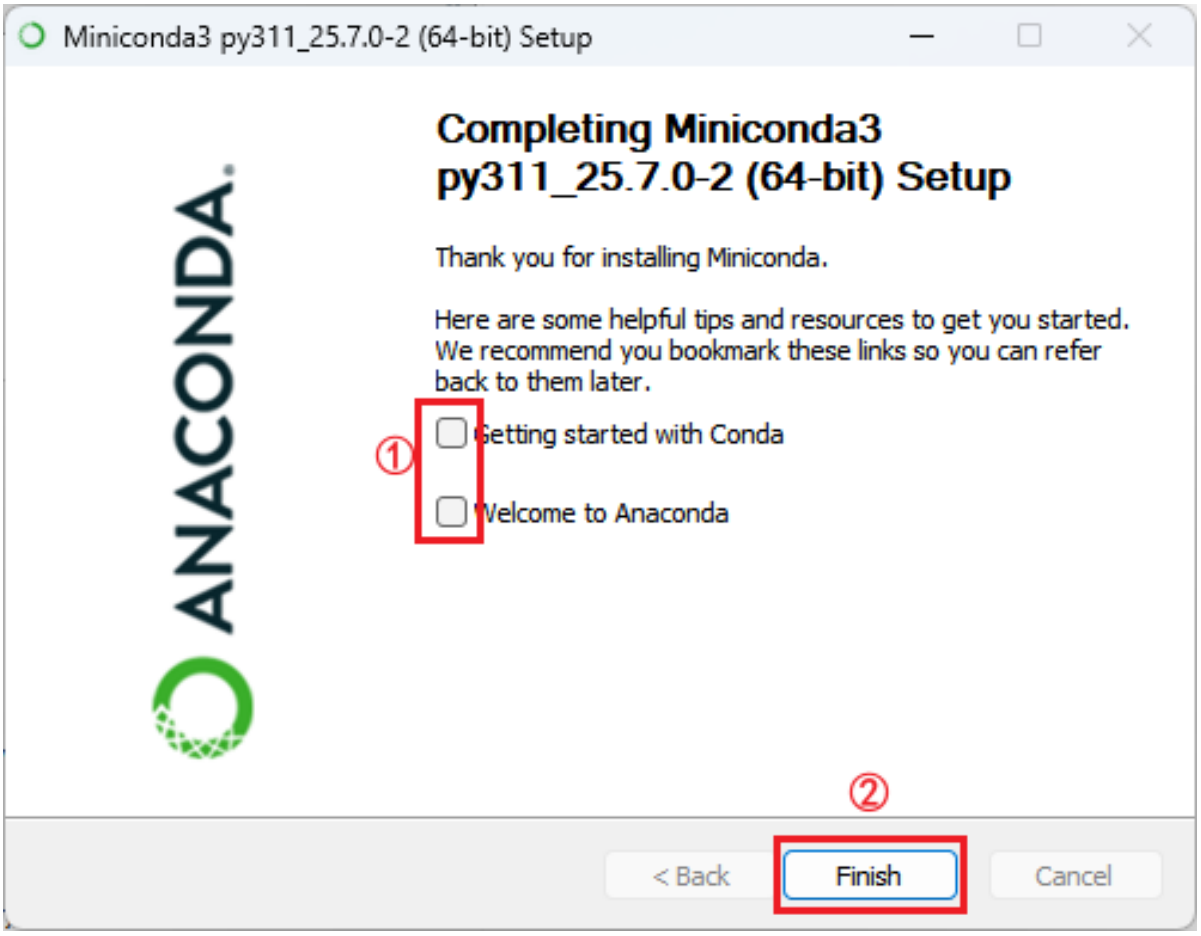

Miniconda3 py311_25.7.0-2 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Miniconda3 py311_25.7.0-2 (64-bit).
Install for:
Just Me (recommended)
①
All Users (requires admin privileges)
②
Anaconda, Inc.
< Back
Next >
Cancel



下方一定要全部勾选，否则环境配置会出现问题！







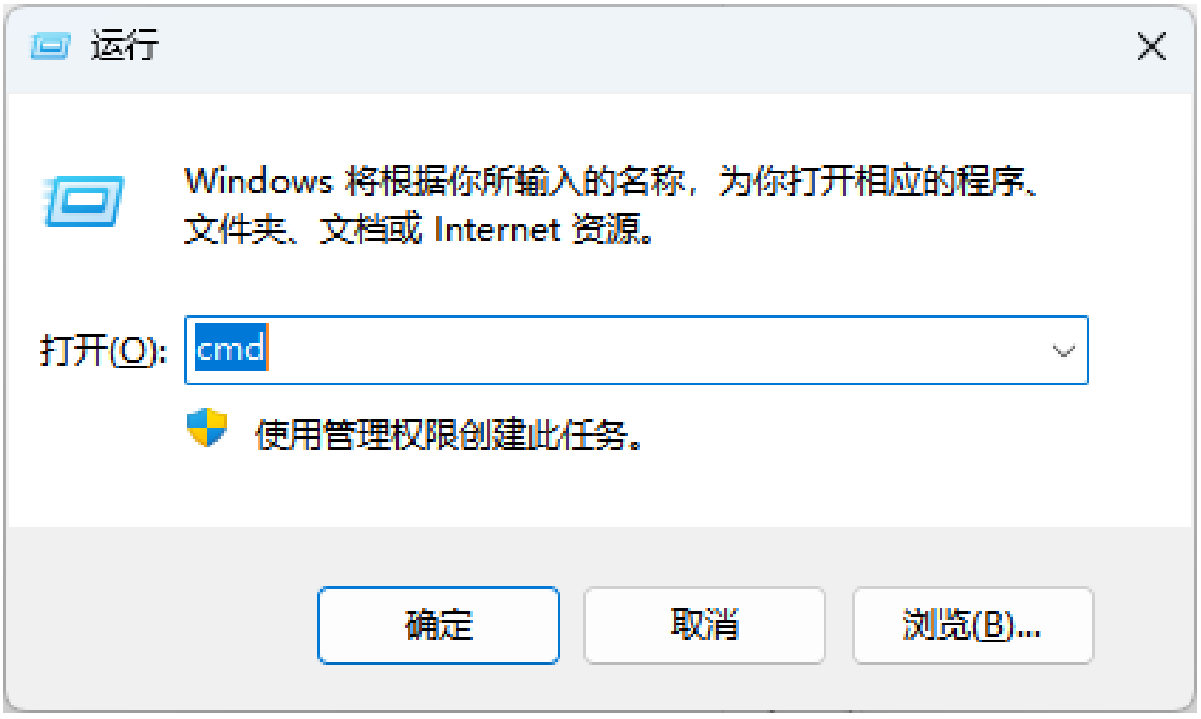
③换源

[anaconda | 镜像站使用帮助 | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror]

点击链接，进入**Miniconda**软件仓库，找到下图所框选的第三方源。


```
channels:
  - defaults
show_channel_urls: true
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

win+R打开控制面板，输入**cmd**，打开终端。

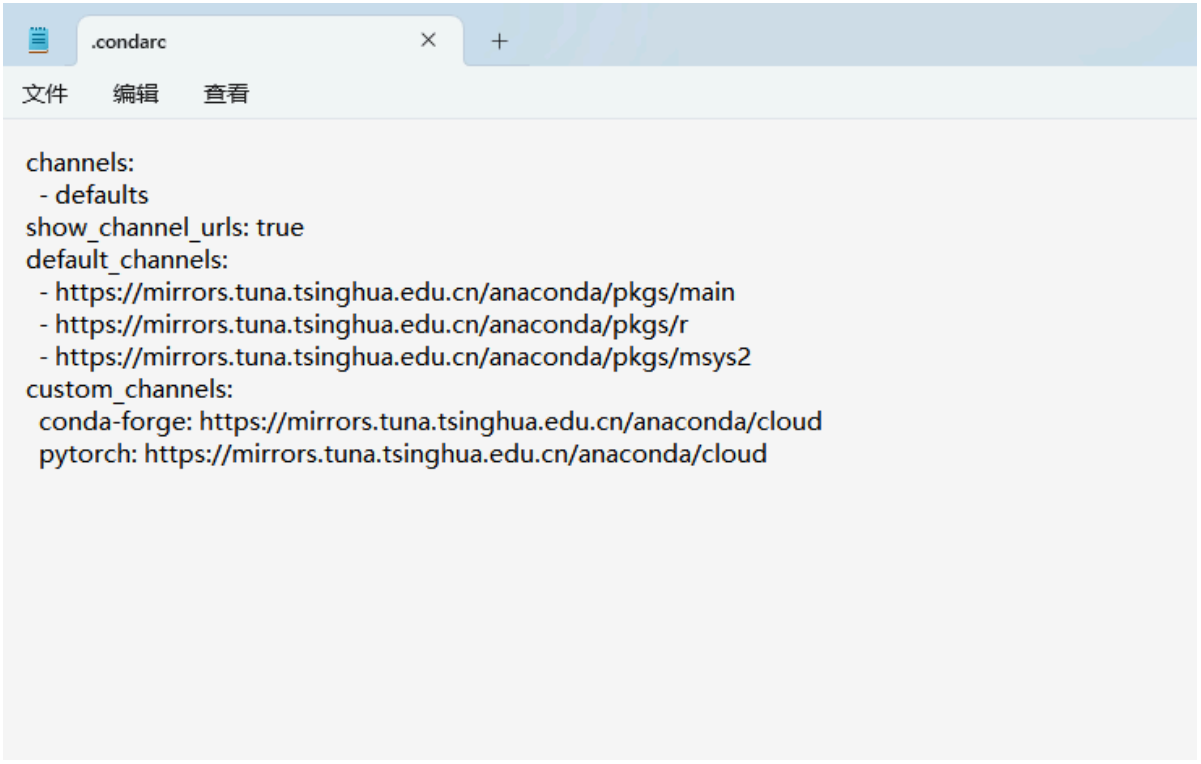


终端输入指令，生成`.condarc`。

```
conda config --set show_channel_urls yes
```

```
C:\Users\Admin>conda config --set show_channel_urls yes
```

`.condarc`文件一般放在用户文件下，例如`C:\Users\Admin`，打开`.condarc`，将内容进行替换，如下所示：



运行指令，清理索引缓存。

```
conda clean -i
```

```
C:\Users\Admin>conda clean -i
```

- **ubuntu系统安装**

①下载**Miniconda**包

Ctrl+Alt+T打开终端，输入指令，下载**Miniconda**包。

```
wget https://repo.anaconda.com/miniconda/Miniconda3-py311_25.7.0-2-Linux-
x86_64.sh
```

```
ubuntu@ubuntu-virtual-machine:~$ wget https://repo.anaconda.com/miniconda/Minico
nda3-py311_25.7.0-2-Linux-x86_64.sh
```

②安装**Miniconda**

输入指令，安装**Miniconda**。

```
sh Miniconda3-py311_25.7.0-2-Linux-x86_64.sh
```

```
ubuntu@ubuntu-virtual-machine:~$ sh Miniconda3-py311_25.7.0-2-Linux-x86_64.sh
```

点击**Enter**继续。

```
Welcome to Miniconda3 py311_25.7.0-2

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

输入**yes**，点击**Enter**继续。

```
Please answer 'yes' or 'no':'
>>> yes
```

这里默认安装到`/home/ubuntu/miniconda3`，如果不需要更改安装目录直接点击**Enter**继续即可，需要更改安装目录则输入需要安装的目录，再点击**Enter**。

```
[/home/ubuntu/miniconda3] >>>
```

输入**yes**，点击**Enter**就安装完成。

```
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
[no] >>> yes
```

现在把终端关掉，重新打开终端就会出现下图界面。

```
(base) ubuntu@ubuntu-virtual-machine:~$
```

如果不想默认使用的是**Miniconda**的**python**环境，需要在`.bashrc`文件中添加一行指令。

输入指令，打开`.bashrc`文件。

```
gedit .bashrc
```

```
(base) ubuntu@ubuntu-virtual-machine:~$ gedit .bashrc
```

找到conda initialize的代码，在下面添加一行指令，添加完**Ctrl+s**保存即可叉掉，重新打开终端即可。

```
conda config --set auto_activate false
```

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/ubuntu/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/ubuntu/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/ubuntu/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/ubuntu/miniconda3/bin:$PATH"
    fi
fi

unset __conda_setup
# <<< conda initialize <<<

conda config --set auto_activate false
```

③换源

anaconda | 镜像站使用帮助 | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror

点击链接，进入**Miniconda**软件仓库，找到下图所框选的第三方源。


```
channels:
  - defaults
show_channel_urls: true
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

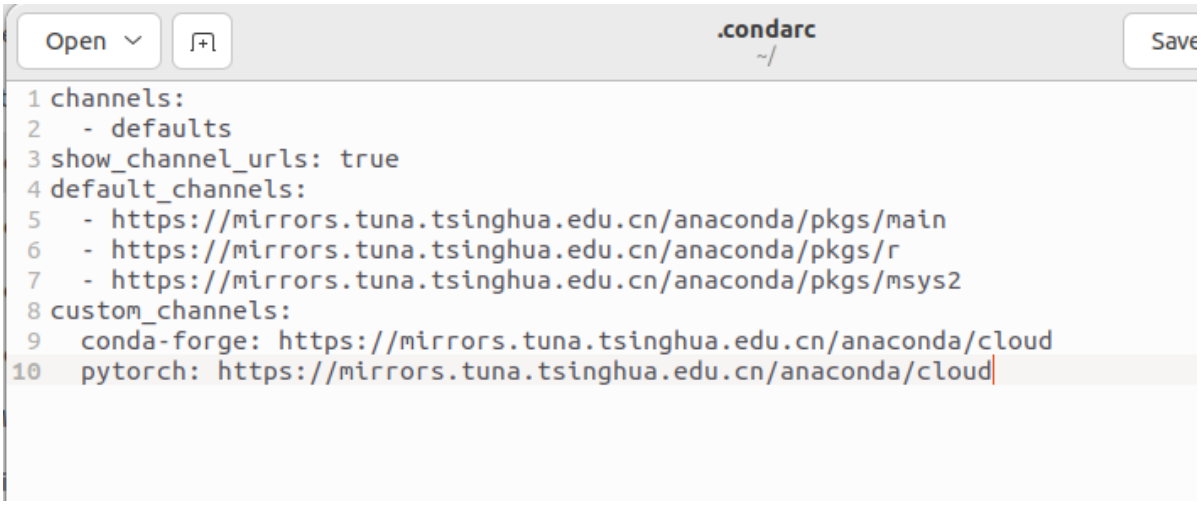
终端输入指令，生成`.condarc`。

```
conda config --set show_channel_urls yes
```

```
ubuntu@ubuntu-virtual-machine:~$ conda config --set show_channel_urls yes
```

`.condarc`文件一般放在用户文件下，例如`/home/ubuntu`，使用`gedit`工具打开`.condarc`，输入指令，将内容进行替换，如下所示。

```
gedit .condarc
```



运行指令，清除索引缓存，输入**y**，点击**Enter**即可。

```
conda clean -i
```

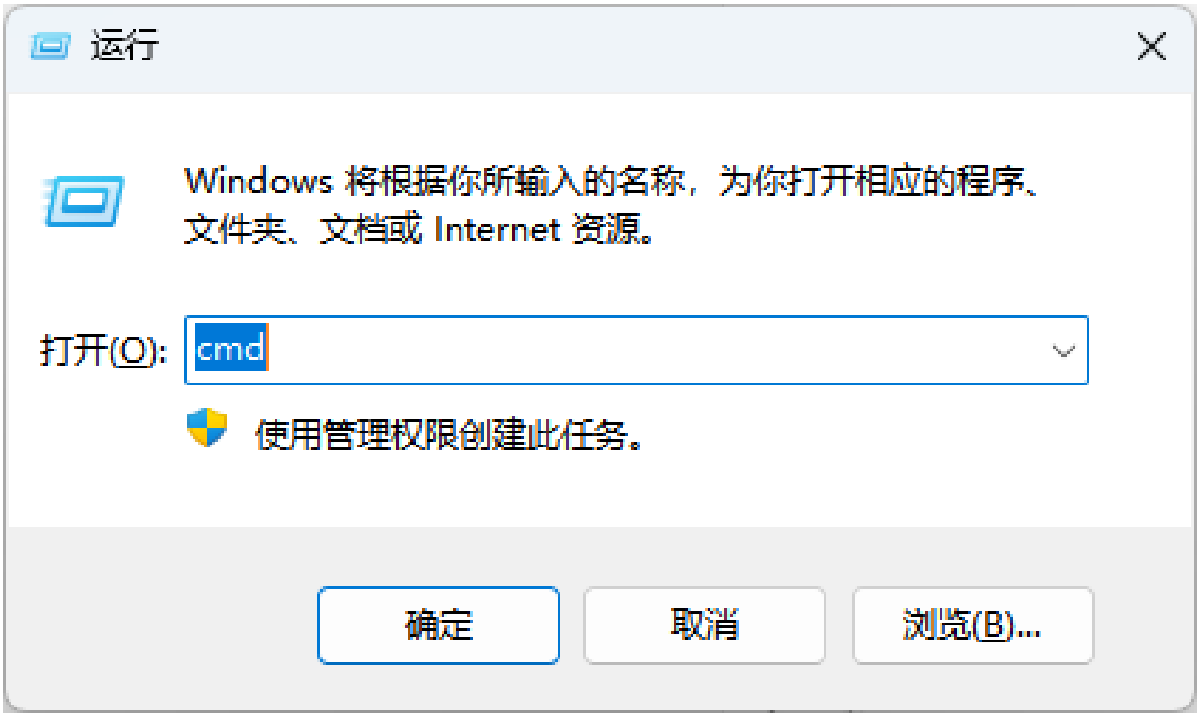
```
ubuntu@ubuntu-virtual-machine:~$ conda clean -i
Will remove 1 index cache(s).
Proceed ([y]/n)? y
```

2.2.2配置虚拟环境

- **windows配置虚拟环境**

①打开终端

win+R打开控制面板，输入**cmd**，打开终端。



②创建虚拟环境

输入指令，按下**Enter**，创建虚拟环境并安装**ffmpeg包**（建议7.1.1，支持 **libsvtav1编码**）。

```
conda create -n lerobot python=3.10.18 ffmpeg=7.1.1 -c conda-forge
```

```
C:\Users\Admin>conda create -n lerobot python=3.10.18 ffmpeg=7.1.1 -c conda-forge
```

出现下图标志，按**y**，再按**Enter**继续。

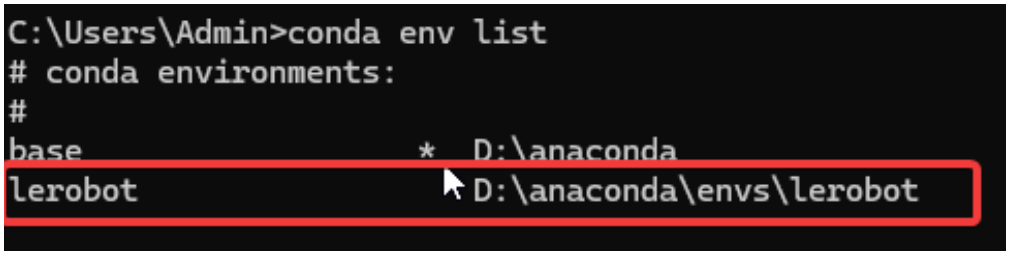
```
The following NEW packages will be INSTALLED:

    bzip2:           1.0.8-h2bbff1b_6        defaults
    ca-certificates: 2025.7.15-haa95532_0    defaults
    expat:           2.7.1-h8ddb27b_0        defaults
    libffi:          3.4.4-hd77b12b_1        defaults
    openssl:         3.0.17-h35632f6_0       defaults
    pip:             25.2-pyhc872135_0       defaults
    python:          3.10.18-h981015d_0      defaults
    setuptools:      78.1.1-py310haa95532_0  defaults
    sqlite:          3.50.2-hda9a48d_1       defaults
    tk:              8.6.15-hf199647_0       defaults
    tzdata:          2025b-h04d1e81_0        defaults
    ucrt:            10.0.22621.0-haa95532_0 defaults
    vc:              14.3-h2df5915_10        defaults
    vc14_runtime:    14.44.35208-h4927774_10 defaults
    vs2015_runtime:  14.44.35208-ha6b5a95_10 defaults
    wheel:           0.45.1-py310haa95532_0  defaults
    xz:              5.6.4-h4754444_1        defaults
    zlib:            1.2.13-h8cc25b3_1       defaults

Proceed ([y]/n)? y
```

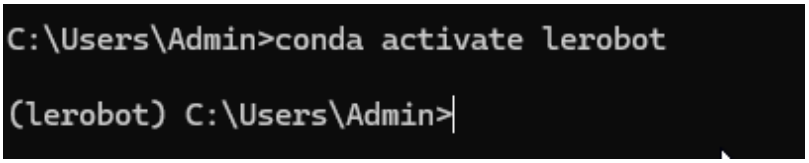
创建完成之后，输入指令，即可查到所创建的虚拟环境。

```
conda env list
```



③进入虚拟环境

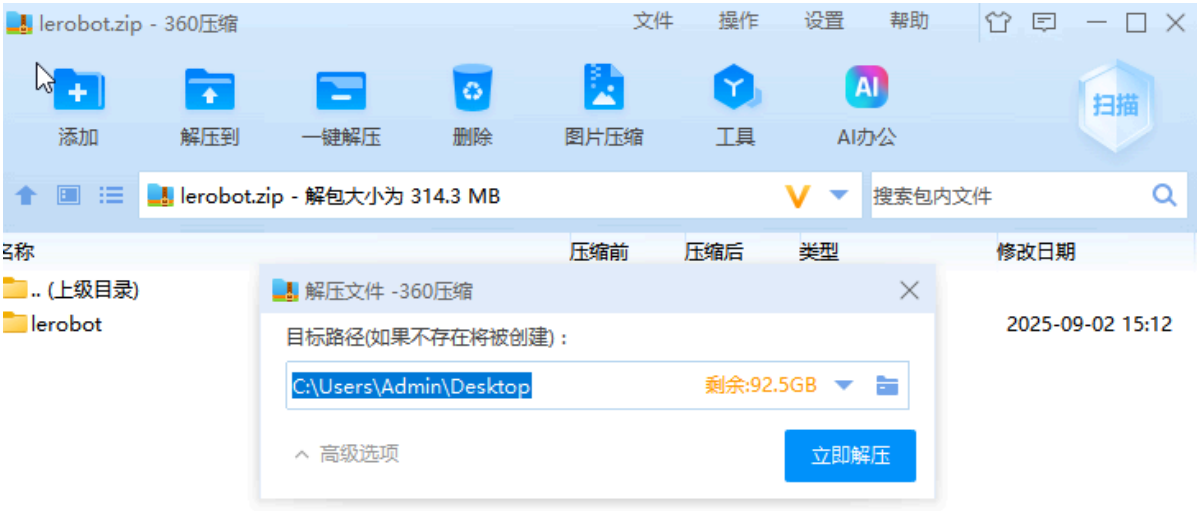
```
conda activate lerobot
```



④下载代码库

SO-ARM101开源6轴机械臂工程包路径：**软件工具&源码程序\源码程序\lerobot.zip**

打开**SO-ARM101开源6轴机械臂**工程包，将工程文件，解压到桌面。



⑤安装依赖包

进入到**lerobot**目录，安装**lerobot**的依赖包并且指定添加飞特舵机相关的驱动。

```
cd Desktop\lerobot
pip install -e ".[feetech]" -i https://pypi.tuna.tsinghua.edu.cn/simple
```

```
(lerobot) C:\Users\Admin>cd Desktop\lerobot

(lerobot) C:\Users\Admin\Desktop\lerobot>pip install -e ".[feetech]" -i https://pypi.tuna.tsinghua.edu.cn/simple
```

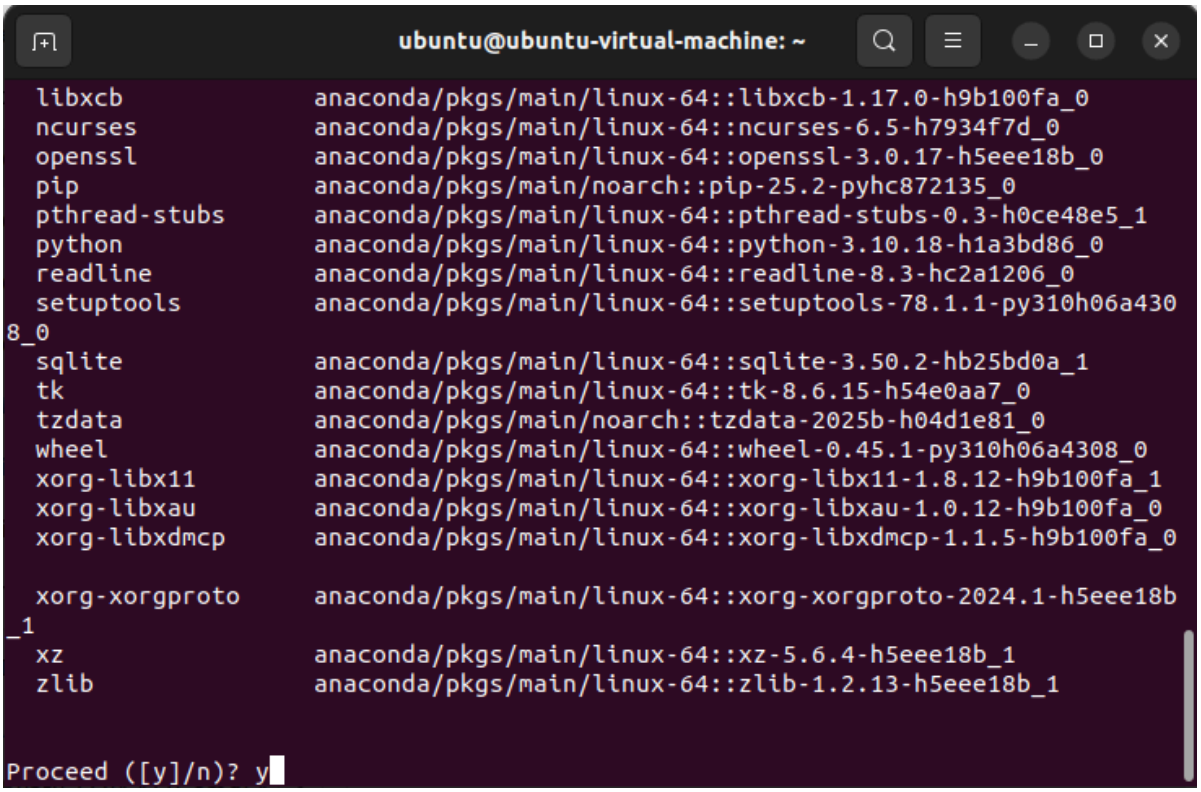
- **ubuntu配置虚拟环境**

①创建虚拟环境

Ctrl+Alt+T打开终端，输入指令，按下**Enter**，创建虚拟环境并安装**ffmpeg包**（建议7.1.1，支持**libsvtav1编码**）。

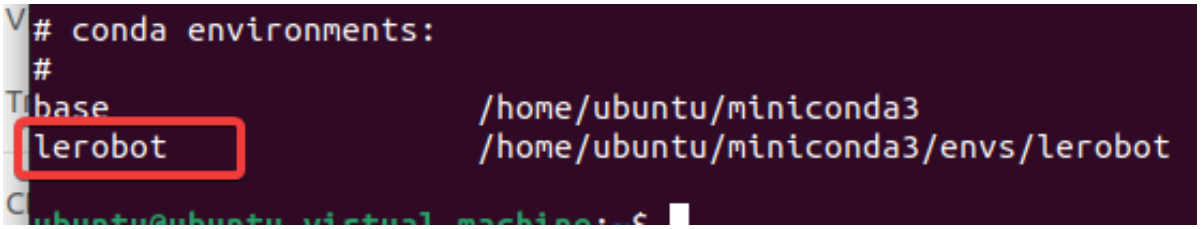
```
conda create -n lerobot python=3.10.18 ffmpeg=7.1.1 -c conda-forge
```

出现下图标志，按**y**，再按**Enter**继续。



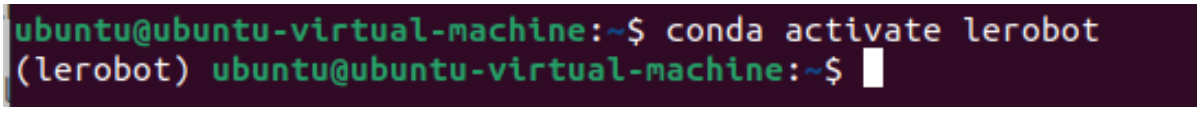
创建完成之后，输入指令，即可查到所创建的虚拟环境。

```
conda env list
```



②进入虚拟环境

```
conda activate lerobot
```



③下载代码库

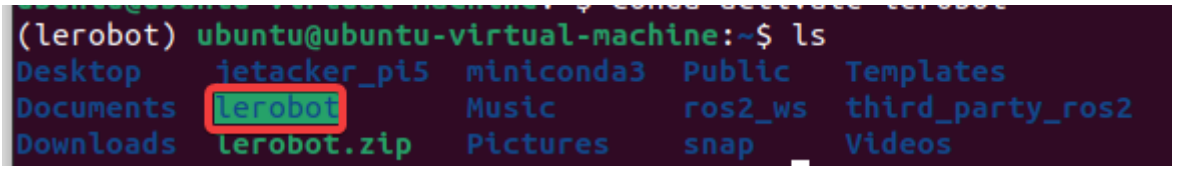
LeRobot SO-101机械臂工程包路径：**软件工具&源码程序\源码程序\lerobot.zip**

将**LeRobot SO-101机械臂**工程包拷到**ubuntu**系统上放到`/home/ubuntu`下，使用`unzip`工具进行解压，输入指令，进行解压。

```
unzip lerobot.zip
```

终端输入指令查看文件。

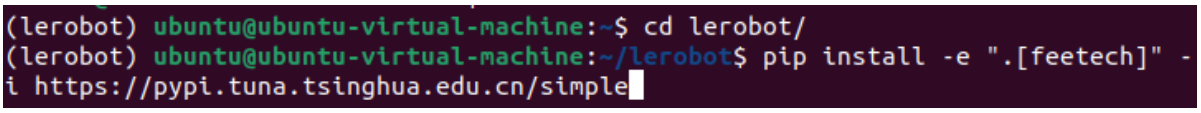
```
ls
```



④安装依赖包

进入到**lerobot**目录，安装**lerobot**的依赖包并且指定添加飞特舵机相关的驱动。

```
cd lerobot
pip install -e ".[feetech]" -i https://pypi.tuna.tsinghua.edu.cn/simple
```



3.机械臂组装

后续步骤**ubuntu**系统和**windows**系统操作相同，这里以**windows**为例。

> **Note**
>
> **在ubuntu系统 上，可能需要通过运行以下命令来授予对 USB 端口的访问权限：**
>
> **sudo chmod 666 /dev/ttyACM0**
> **sudo chmod 666 /dev/ttyACM1**

3.1 硬件组装

查看相关视频，路径：1.课程资料\视频教程\3.1硬件组装

3.2 相机安装

查看相关视频，路径：1.课程资料\视频教程\3.2 相机安装教程

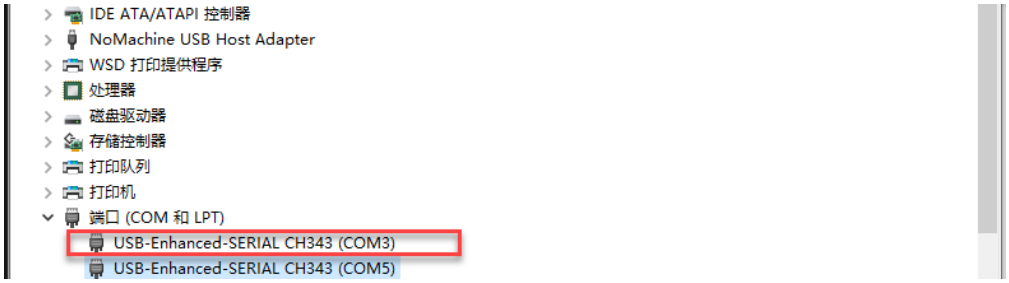
3.3 电路连接

查看相关视频，路径：1.课程资料\视频教程\3.3 电路连接教程

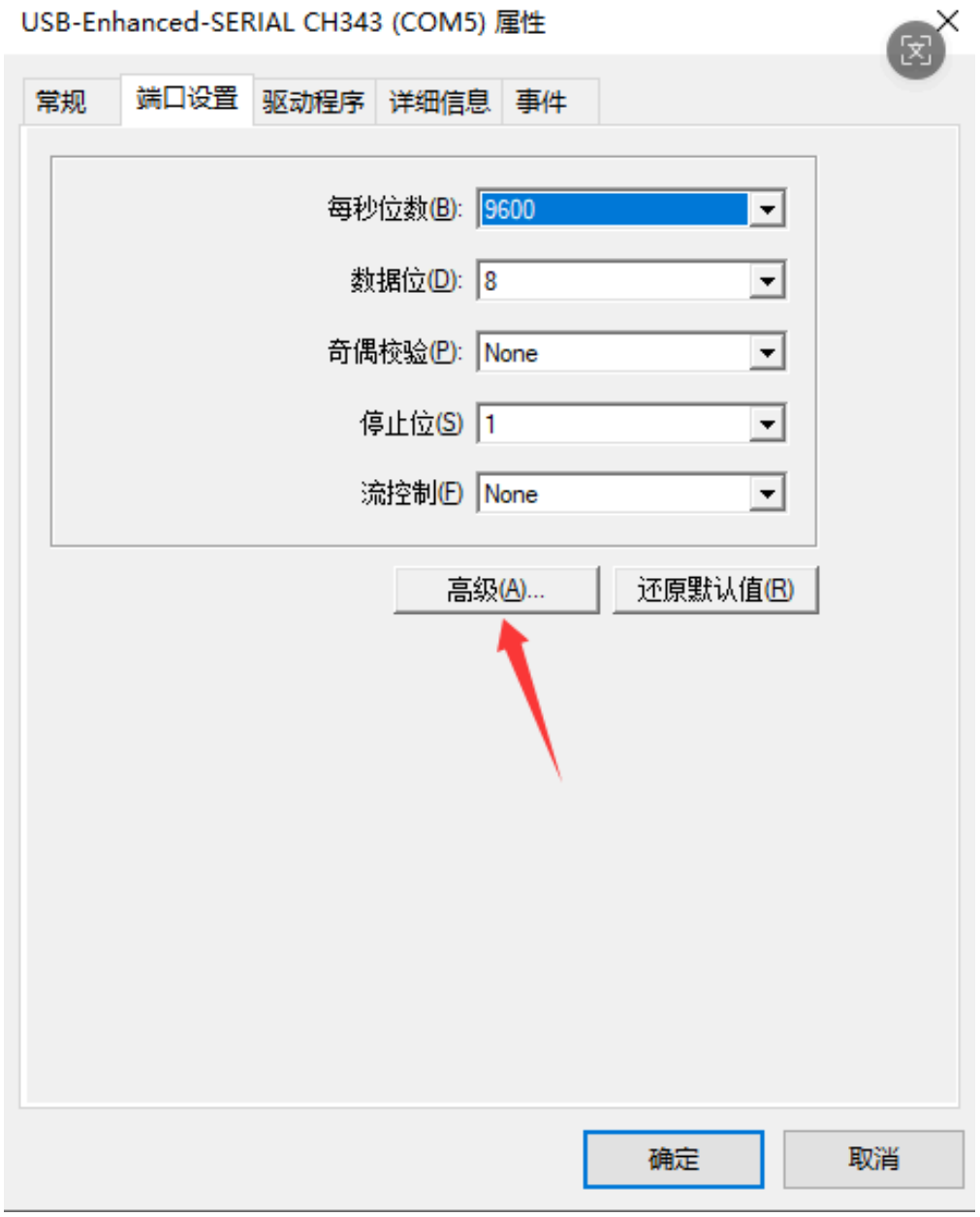
3.4 查找端口号

1. 可先插入从臂，打开设备管理器，查看端口号。

2. 再插入主臂，查看新增的端口号。

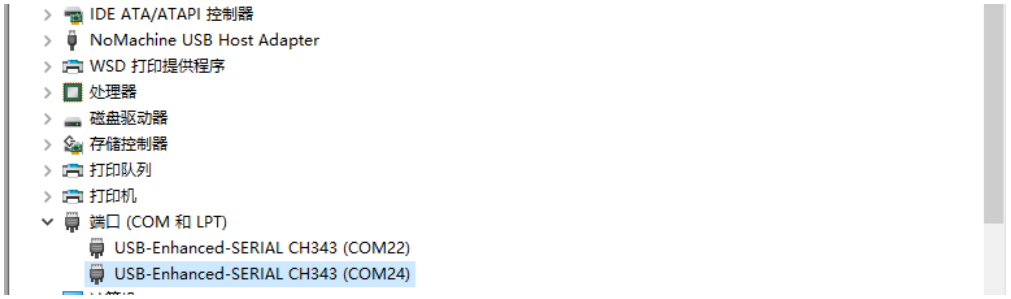


3. 为了后续更方便使用，可以设置固定端口号。右键COM图标，点属性，点击端口设置->高级。



4. 点击端口号选择主臂端口号设置为COM22，从臂端口号设置为COM24，由于我之前已经设置过，COM22有使用中的标志，则说明端口被占用。

5. 主臂和从臂端口号分别设置为22和24，设置好后，如下图所示即为成功。



3.5 舵机ID设置（选看）

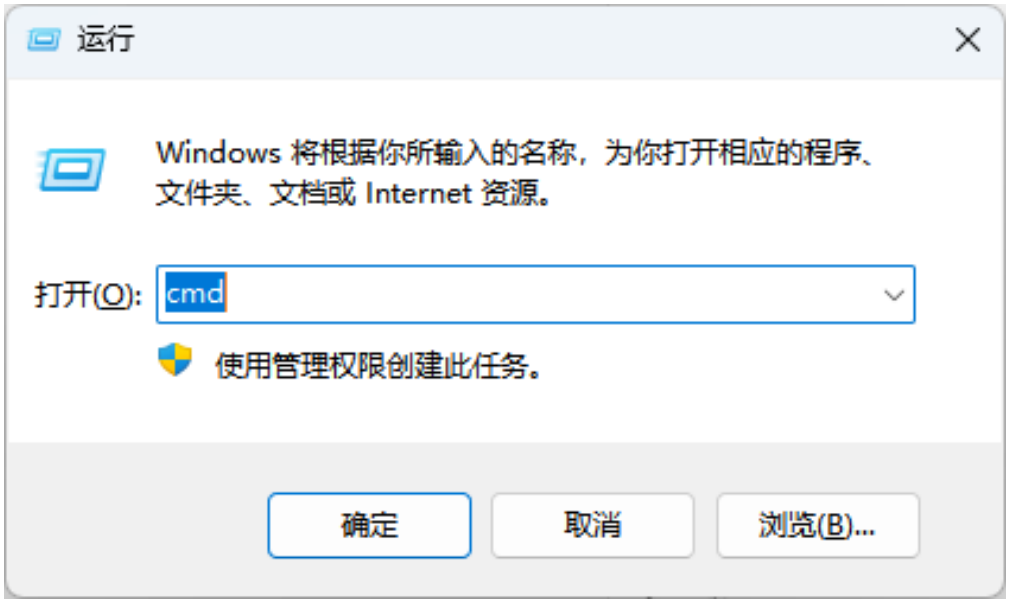
本小节适用于DIY散件用户，成品用户只需了解即可。

机械臂从上往下的每个舵机的命名依次为：`gripper`、`wrist_flex`、`wrist_flex`、`elbow_flex`、`shoulder_lift`和`shoulder_pan`，分别对应着`ID6`到`ID1`。

> **Note**
>
> **在设置舵机ID时，只能有一个舵机连接主控板，其它串联的信号线都需要拔掉。**

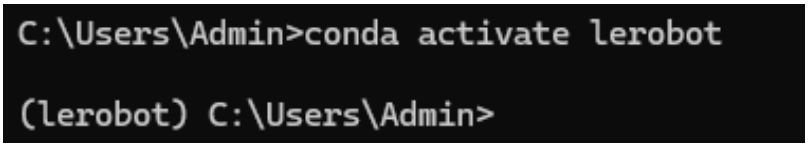
3.5.1从机械臂（跟随者）

win+R打开控制面板，输入**cmd**，打开终端。



输入指令，进入虚拟环境。

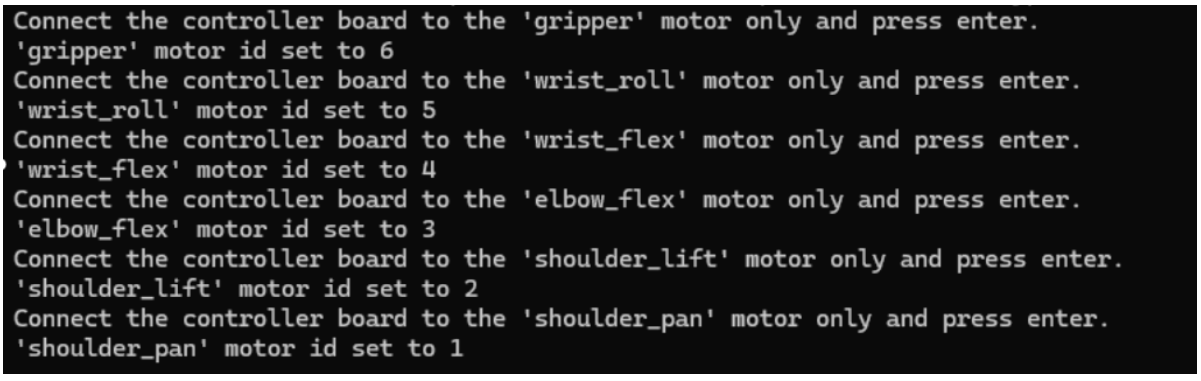
```
conda activate lerobot
```



使用`cd`工具跳到工程文件夹下，输入指令，进行`ID`设置。

```
cd Desktop\lerobot
lerobot-setup-motors --robot.type=so101_follower --robot.port=COM24
```

每出现一个舵机名提示，要确保单独插入了对应`ID`舵机，然后回车即可自动设置。当出现下一个舵机名称提示，代表上一个舵机设置完成，后续插拔重复回车操作即可。



3.5.2主机械臂（领导者）

输入指令，进行`ID`设置。

```
lerobot-setup-motors --teleop.type=so101_leader --teleop.port=COM22
```

步骤同从机械臂。

4.机械臂控制

4.1 校准

接下来，需要校准机器人，以确保主机械臂和从机械臂在处于相同物理位置时具有相同的位置值。

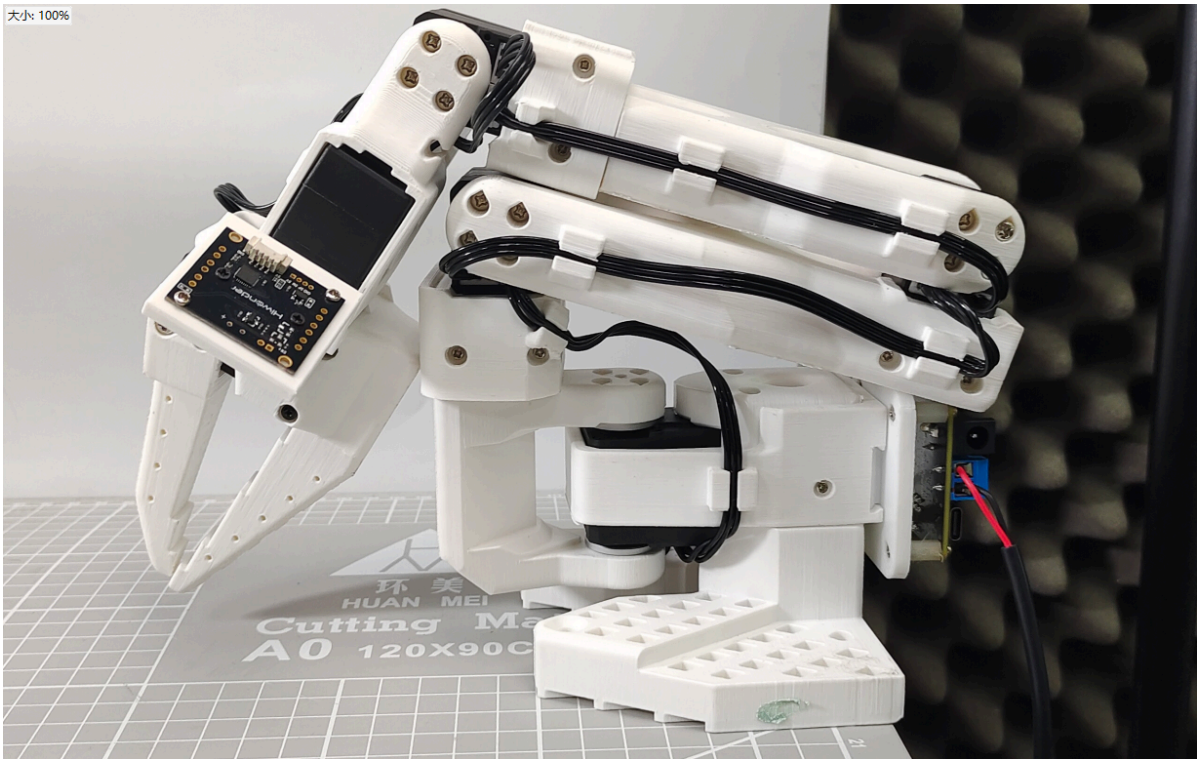
4.1.1从机械臂（跟随者）

输入指令，进行校准。

```
lerobot-calibrate --robot.type=so101_follower --robot.port=COM24 --
robot.id=my_awesome_follower_arm
```

```
(lerobot) C:\Users\Admin\Desktop\lerobot>lerobot-calibrate --robot.type=so101_follower --robot.port=CO
M24 --robot.id=my_awesome_follower_arm
```

需要将机器人所有关节都移动到机械臂初始位置，如下图所示。



如果没有校准过，按下**Enter**即可进入校准，校准过需要重新校准的，需要输入**c**，再按**Enter**，才能进行重新校准。

```
Press ENTER to use provided calibration file associated with the id my_awesome_follower_arm, or type '
c' and press ENTER to run calibration: c
```

校准通过手动掰动机械臂每个关节进行校准，具体的看查看视频进行了解。

4.1.2主机械臂（领导者）

输入指令，进行校准。

```
lerobot-calibrate --teleop.type=so101_leader --teleop.port=COM22 --
teleop.id=my_awesome_leader_arm
```

步骤同从机械臂。

4.2 摇操作不带视觉

输入指令，进行机械臂控制。

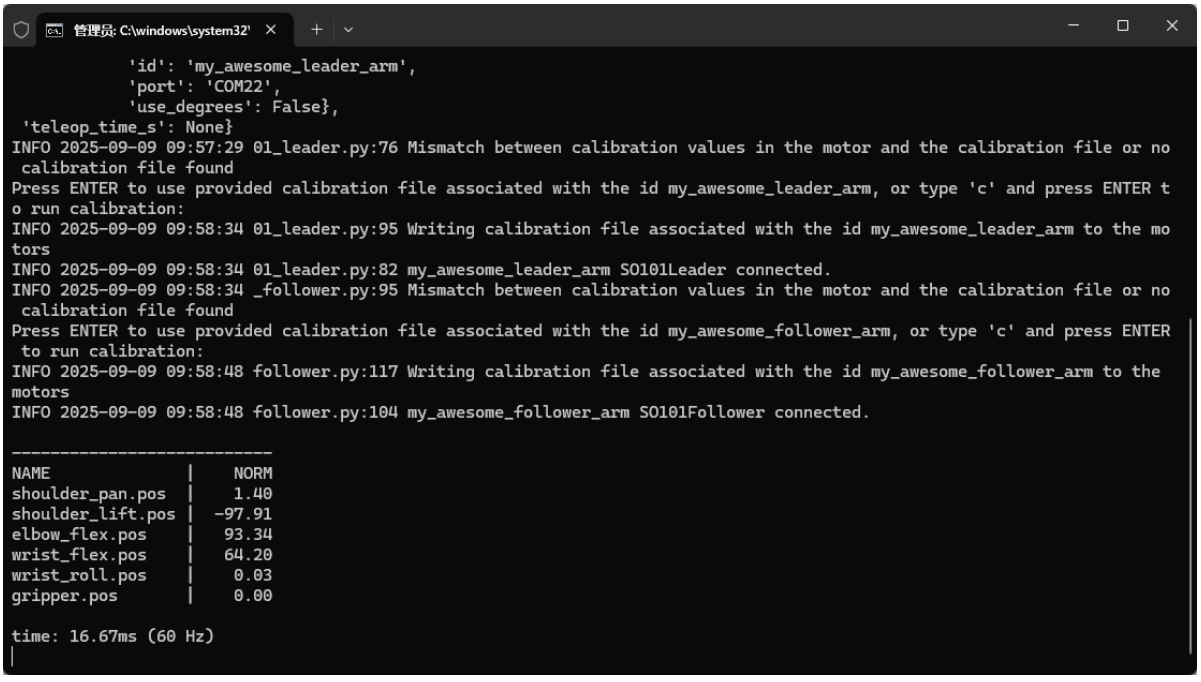
```
python -m lerobot.teleoperate --robot.type=so101_follower --robot.port=COM24 --robot.id=my_awesome_follower_arm --teleop.type=so101_leader --teleop.port=COM22 --teleop.id=my_awesome_leader_arm
```

```
(lerobot) C:\Users\Admin\Desktop\lerobot>python -m lerobot.teleoperate --robot.type=so101_follower --robot.port=COM24 --robot.id=my_awesome_follower_arm --teleop.type=so101_leader --teleop.port=COM22 --teleop.id=my_awesome_leader_arm
```

当出现下图提示时，按下**Enter**继续。

```
Press ENTER to use provided calibration file associated with the id my_awesome_leader_arm, or type 'c' and press ENTER to run calibration:
```

出现下图界面，即可通过主机械臂遥控从机械臂。



按下**Ctrl+c**即可终止终止程序。

4.3 摇操作带视觉

将两个摄像头的USB插到电脑上。

> **Note**
>
> **①如果使用拓展坞，两个摄像头的USB不能都接在拓展坞上。**
>
> **②固定环境摄像头的画面必须能够看到整个从机械臂的动作。**

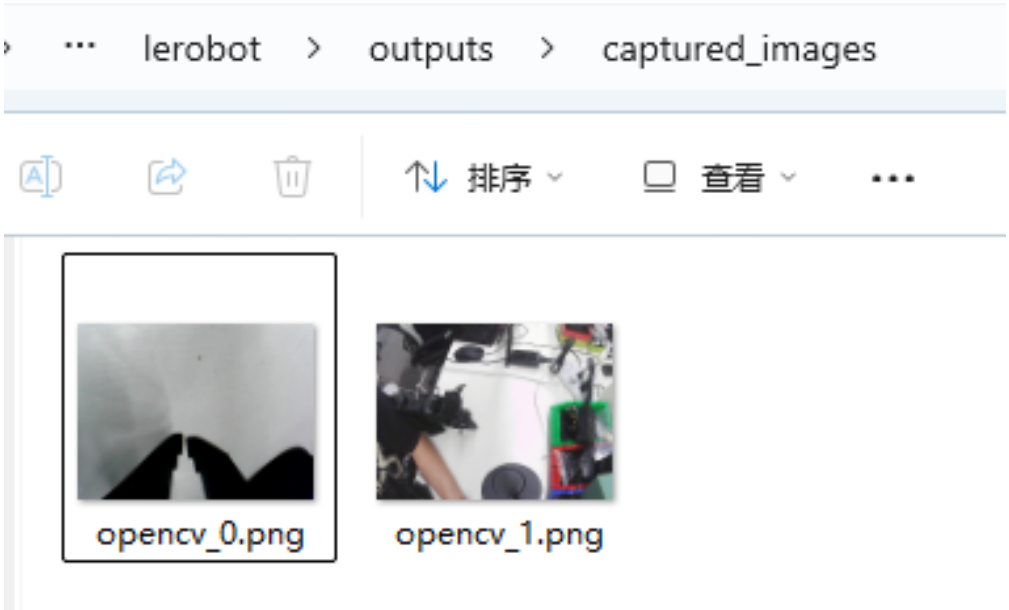
输入指令，查找摄像头`ID`，检查是否有两个摄像头的画面。

```
lerobot-find-cameras opencv
```

```
--- Detected Cameras ---
Camera #0:
  Name: OpenCV Camera @ 0
  Type: OpenCV
  Id: 0
  Backend api: MSMF
  Default stream profile:
    Format: -1.0
    Width: 640
    Height: 480
    Fps: 30.0
--------------------
Camera #1:
  Name: OpenCV Camera @ 1
  Type: OpenCV
  Id: 1
  Backend api: MSMF
  Default stream profile:
    Format: -1.0
    Width: 640
    Height: 480
    Fps: 30.0
--------------------
```

运行结束，会将捕获的图片保存到`outputs\captured_images`。

```
Image capture finished. Images saved to outputs\captured_images
```



可以通过文件名来区分摄像头的`ID`，`opencv_0`对应着`index_or_path`：0，`opencv_1`对应着`index_or_path`：1。

运行指令，进行遥控机械臂可视化。

```
python -m lerobot.teleoperate --robot.type=so101_follower --robot.port=COM24 --robot.id=my_awesome_follower_arm --robot.cameras="{ fixed: {type: opencv, index_or_path: 1, width: 640, height: 480, fps: 30}, handeye: {type: opencv, index_or_path: 0, width: 640, height: 480, fps: 30}}" --teleop.type=so101_leader --teleop.port=COM22 --teleop.id=my_awesome_leader_arm --display_data=true
```

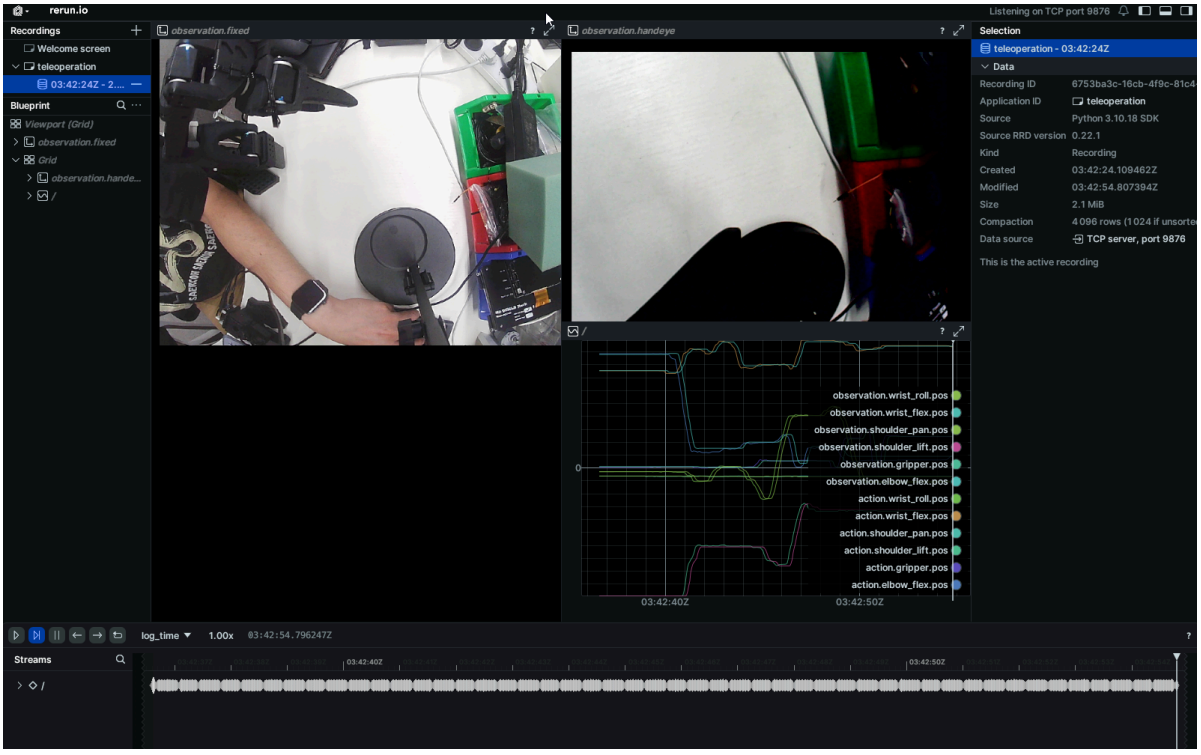
```
(lerobot) C:\Users\Admin\Desktop\lerobot>python -m lerobot.teleoperate --robot.type=so101_follower --robot.port=COM24 --robot.id=my_awesome_follower_arm --robot.cameras="{ fixed: {type: opencv, index_or_path: 1, width: 640, height: 480, fps: 30}, handeye: {type: opencv, index_or_path: 0, width: 640, height: 480, fps: 30}}" --teleop.type=so101_leader --teleop.port=COM22 --teleop.id=my_awesome_leader_arm --display_data=true
```

> **Note**
>
> `fixed`为固定环境摄像头，`handeye`为夹爪相机。

出现下面提示词，按**Enter**继续。

```
Press ENTER to use provided calibration file associated with the id my_awesome_follower_arm, or type 'c' and press ENTER to run calibration:
```

通过主机械臂控制从机械臂的同时会回传摄像头捕捉到的画面。若想关闭程序，在终端中**Ctrl+c**即可关闭。



5.数据采集

数据采集步骤：

1. 初始化机器人和传感器。
2. 启动遥操作界面。
3. 操作者控制机器人完成任务。
4. 系统同步记录所有传感器数据和控制指令。
5. 数据被保存为标准 LeRobot 数据集格式。
6. 可选的实时质量监控和可视化。

> **Note**
>
> **数据采集工作，需要熟练控制机械臂进行夹取方可进行操作，以免操作不规范导致数据不精准。**

运行指令，进行数据采集，`${HF_USER}`可以改为自己用户名（注意用户名必须为英文）。这里的指令我们以采集20次为例，如果需要采集更多次，需要将`dataset.num_episodes=20`这个20改为更多次数。

```
python -m lerobot.record --robot.type=so101_follower --robot.port=COM24 --
robot.id=my_awesome_follower_arm --robot.cameras="{ handeye: {type: opencv,
index_or_path: 0, width: 640, height: 480, fps: 30}, front: {type: opencv,
index_or_path: 1, width: 640, height: 480, fps: 30}}" --teleop.type=so101_leader
--teleop.port=COM22 --teleop.id=my_awesome_leader_arm --display_data=true --
dataset.repo_id=${HF_USER}/demo --dataset.num_episodes=20 --
dataset.single_task="Grab the screwdriver"
```

```
(lerobot) C:\Users\Admin\Desktop\lerobot>python -m lerobot.record --robot.type=so101_follower --robot.port=COM24 --robot
.id=my_awesome_follower_arm --robot.cameras="{ handeye: {type: opencv, index_or_path: 0, width: 640, height: 480, fps: 3
0}, front: {type: opencv, index_or_path: 1, width: 640, height: 480, fps: 30}}" --teleop.type=so101_leader --teleop.port
=COM22 --teleop.id=my_awesome_leader_arm --display_data=true --dataset.repo_id=Admin/demo --dataset.num_episodes=20 --da
taset.single_task="Grab the screwdriver"
```

出现下图提示，按下**Enter**继续。

```
Press ENTER to use provided calibration file associated with the id my_awesome_follower_arm, or type 'c' and press ENTER
 to run calibration:
```

出现下图提示，则说明已经进入数据录制，这时可以操控主机械臂来控制从机械臂进行目标物体夹取，放置目标位置。

```
INFO 2025-09-09 15:14:03 01_leader.py:82 my_awesome_leader_arm SO101Leader connected.
INFO 2025-09-09 15:14:03 ls\utils.py:227 Recording episode 0
```

> **Note**
>
> **录制的时候，除了机械臂，环境最好相对静止，这样的效果最好，例如：摄像头捕捉到的画面最好只要机械臂在运动，手臂、人等最好不要出现在画面里。**

动作执行完毕后，按下→继续，出现下图提示则可以将环境进行复位。

```
INFO 2025-09-09 15:22:14 ls\utils.py:227 Reset the environment
```

环境复位完毕后，按下→进行保存数据，当出现下图所示的进度条加载到100%则说明数据保存完毕。如果不想要当前轮的数据按←即可进行重新录制，本轮数据将不会保存。

```
Map: 100%|                                                                    | 1199/1199 [00:00<00:00, 2810.32 examples/s]
Creating parquet from Arrow format: 100%|                                     | 2/2 [00:00<00:00, 1963.17ba/s]
```

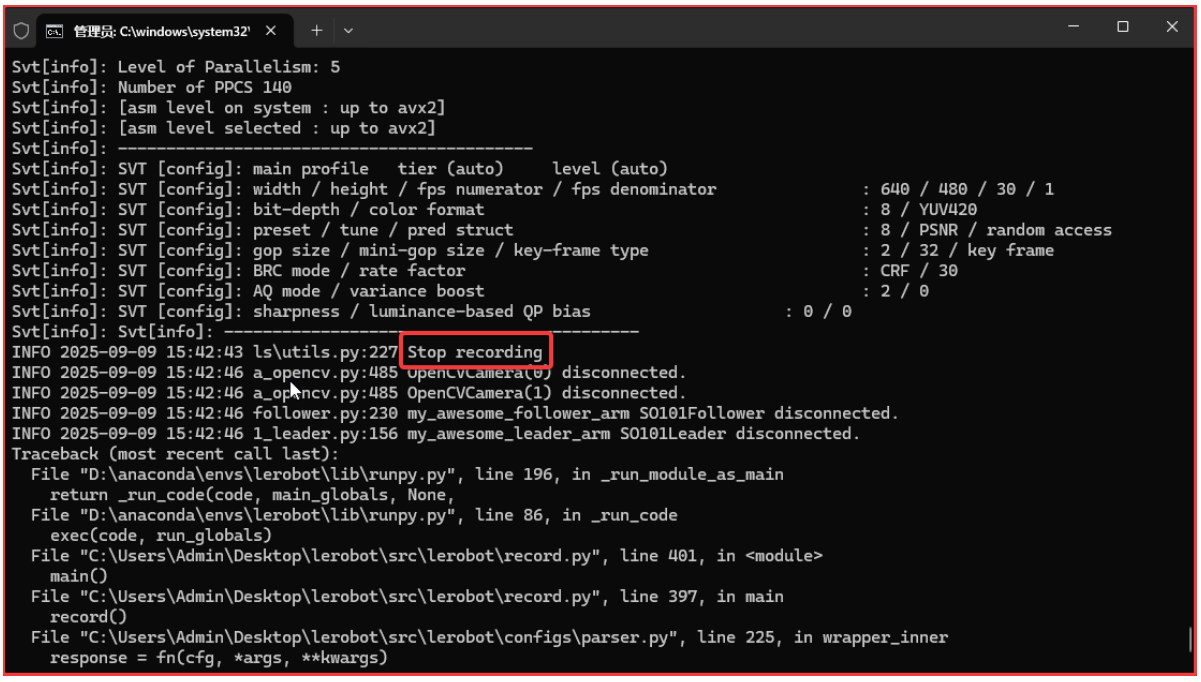
上一轮录制结束，等待下一次录制的标志，如下图所示，则说明已经进入下一轮数据录制。

```
INFO 2025-09-09 15:34:00 ls\utils.py:227 Recording episode 1
```

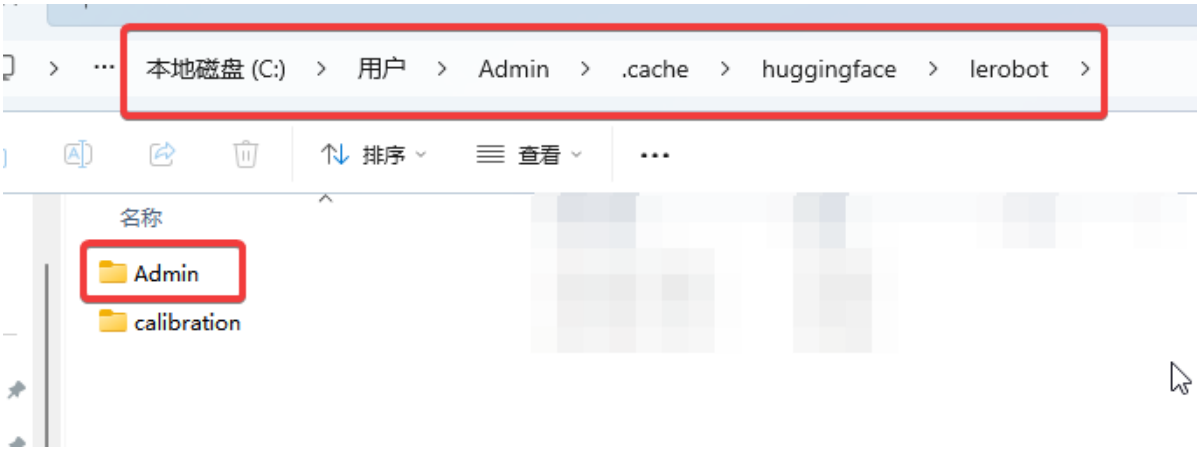
这里采集20轮数据为例，为了数据多样性，每轮目标物体放置的位置最好不要相同。

```
INFO 2025-09-09 15:42:42 ls\utils.py:227 Recording episode 19
Right arrow key pressed. Exiting loop...
Map: 100%|                                                                    | 9/9 [00:00<00:00, 1286.07 examples/s]
Creating parquet from Arrow format: 100%|                                     | 1/1 [00:00<?, ?ba/s]
```

当到**Recording episode 19**，则说明已进入第二十轮录制，录制完后会后打印**Stop recording**，说明已录制完毕，按下**Esc**终止程序就行。



数据默认保存到`Admin\.cache\huggingface\lerobot`中。



6.数据集训练

6.1 云服务器训练

6.1.1云主机租用

本次课程所选用的云主机平台为智星云。

[智星云 AI Galaxy | GPU云服务器 GPU服务器租用 远程GPU租用 深度学习服务器 | 免费GPU 便宜GPU]

如下图所示，右上角进行登录，没注册过的需要注册一下。

在算力市场中根据需求选择相对应的**GPU云主机**，本课程所采用的主机为**RTX4090**，训练时长大致约两个小时，配置如下。

> **Note**
>
> **RTX4090 1GPU 8核16G 系统200G**
>
> **镜像22.04带cuda12.4，宽带应该选80左右**

选择好系统镜像，点击**立即租用。**

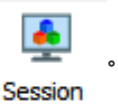
选择你的计费方式，根据自己需求进行选择，此次数据集训练大概需要3个小时，建议租用结束后保留全部磁盘，防止数据丢失，勾选好后，点击**创建实例。**

点击**控制台**即可看到你所创建的实例。

6.1.2连接方式查看

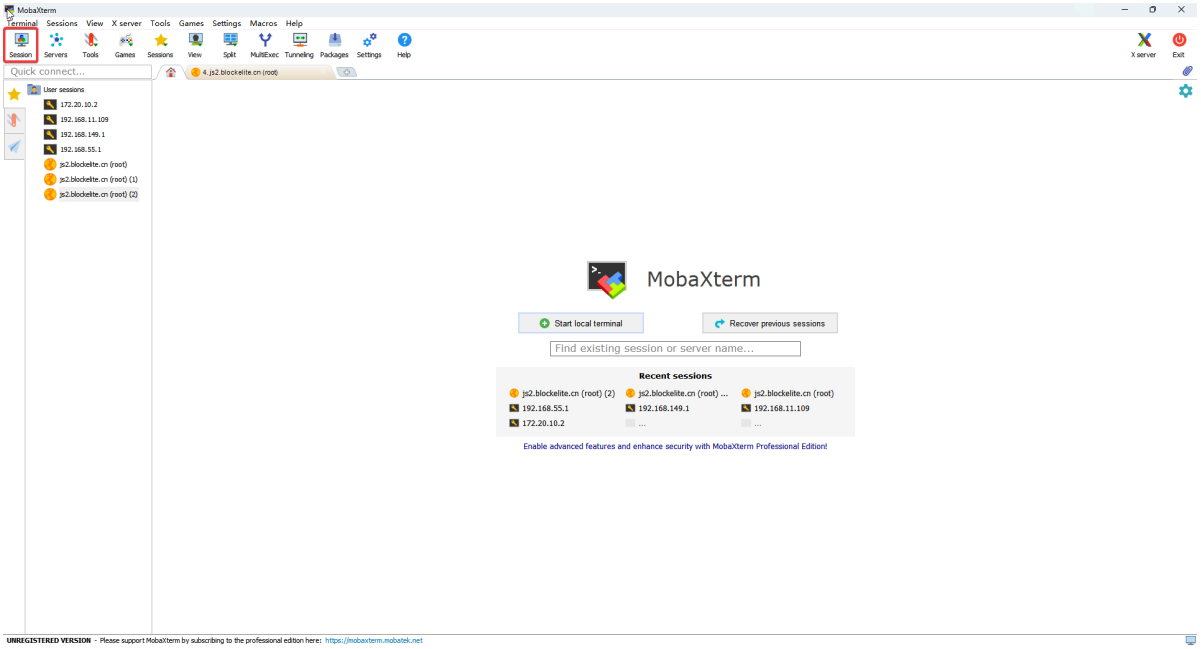
点击查看**连接方式**，选择ssh连接，即可看到**地址、账户、端口和密码。**

6.1.3数据集上传

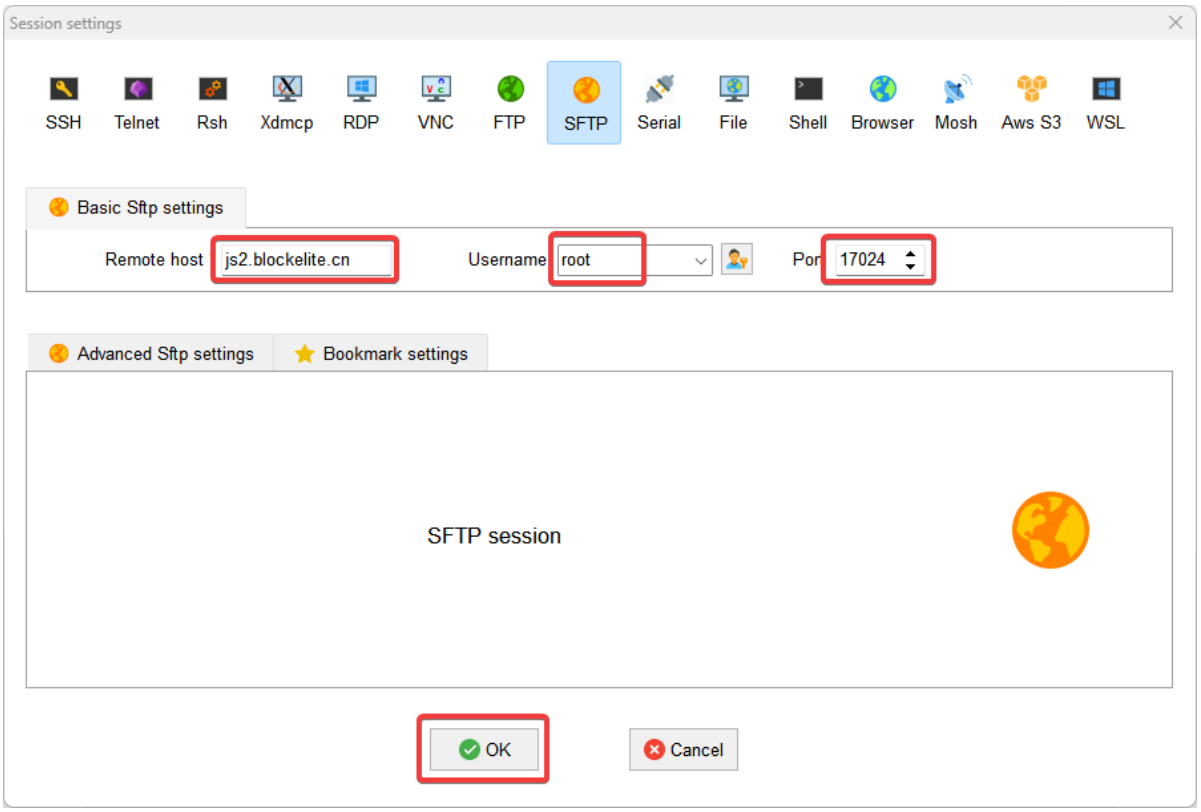
使用`MobaXterm`进行文件传输，打开`MobaXterm`后，点击**Session** 

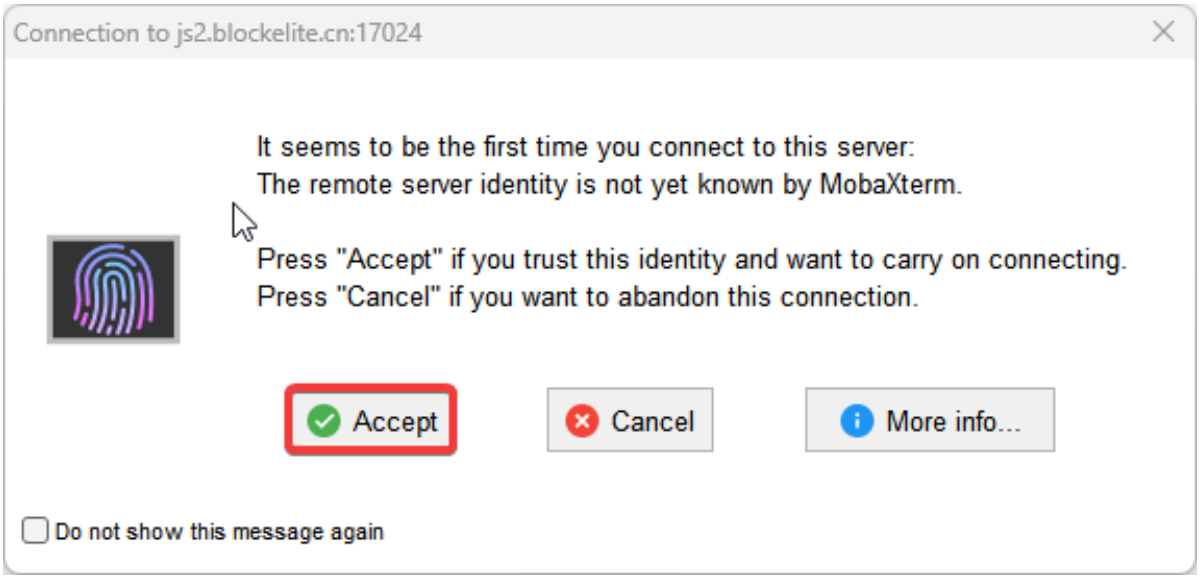
。

`MobaXterm`远程连接软件下载路径：**软件工具&源码程序\软件工具\MobaXterm_Installer_v22.1.zip**

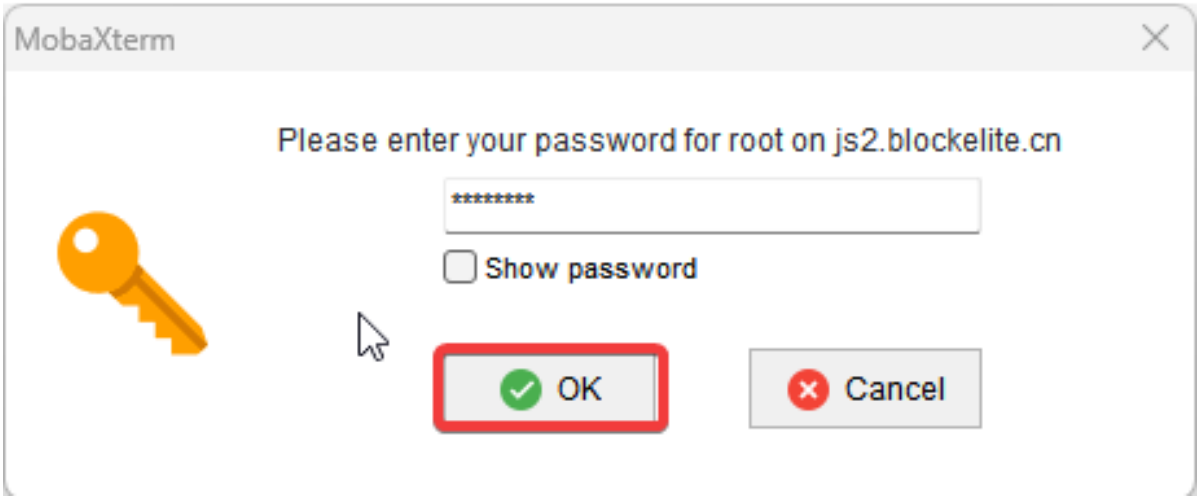


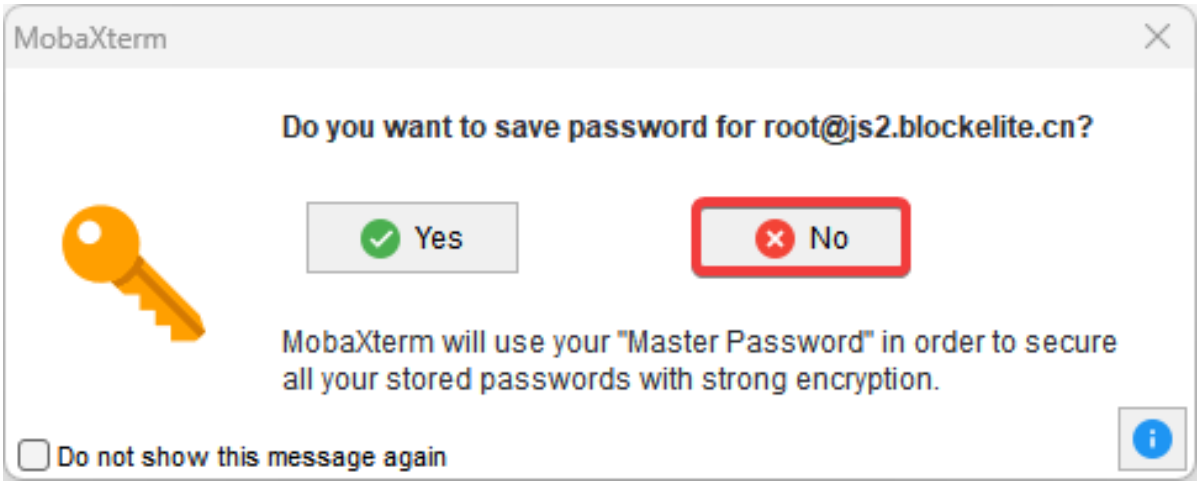
使用**SFTP**进行文件传输，将租用云主机的**地址、用户名和端口**进行填写，如下图所示，然后点击**OK。**



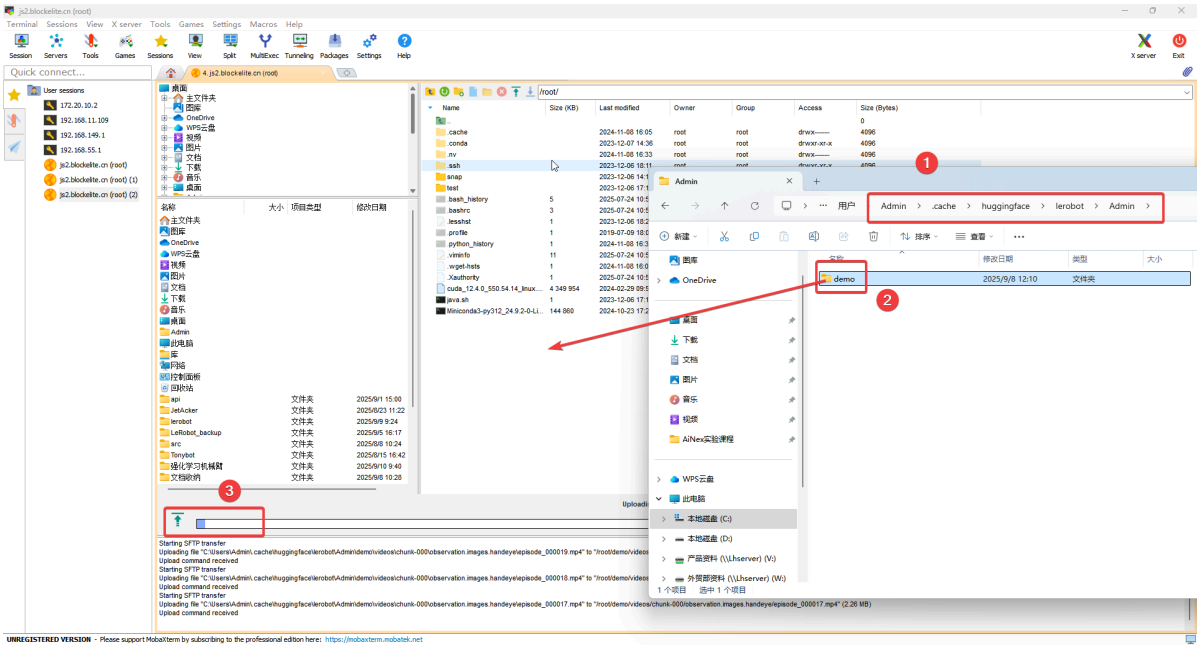


输入云主机的密码，进行登录。

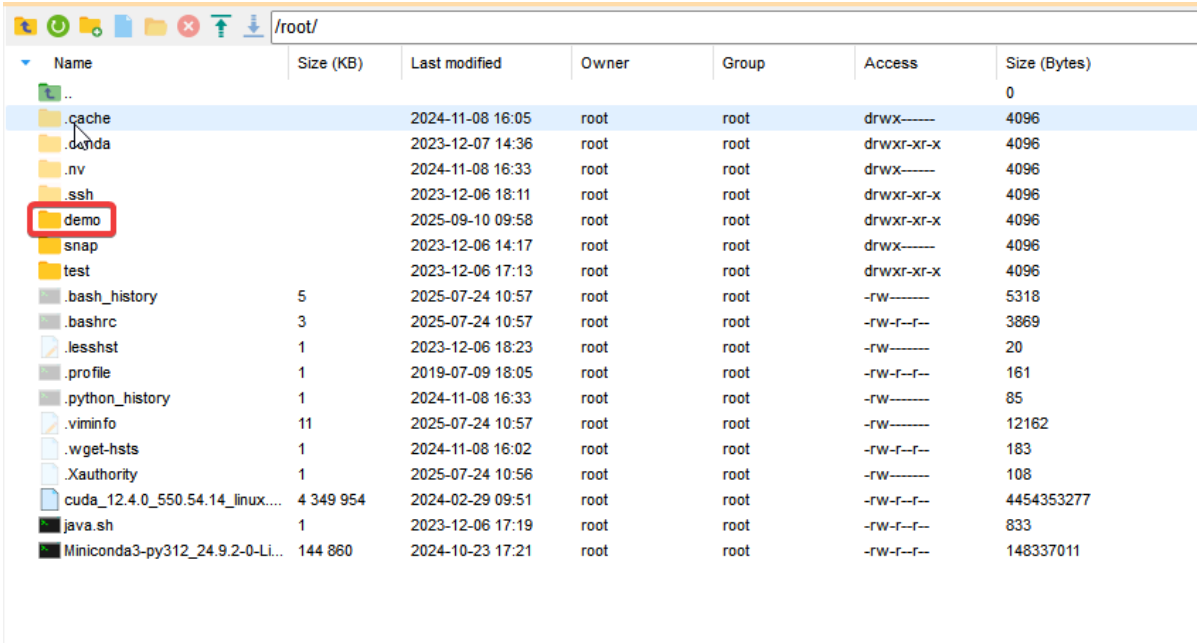




打开文件夹，找到采集的数据，这里存放的路径为`C:\Users\Admin\.cache\huggingface\lerobot\Admin`，将**demo**文件夹整个拖到云主机上。



传输完成后，可查看`root`目录下是否有一个**demo**文件夹。

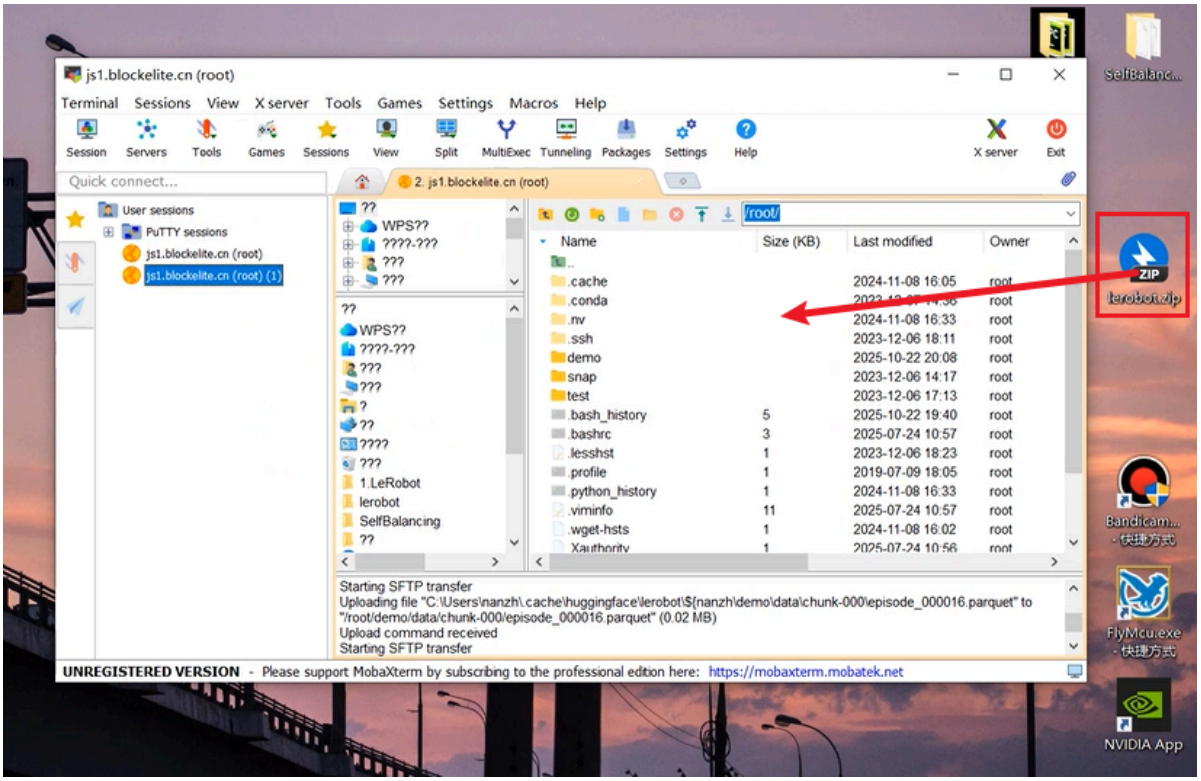


/root/

Name	Size (KB)	Last modified	Owner	Group	Access	Size (Bytes)
..						0
.cache		2024-11-08 16:05	root	root	drwx------	4096
.conda		2023-12-07 14:36	root	root	drwxr-xr-x	4096
.nv		2024-11-08 16:33	root	root	drwx------	4096
.ssh		2023-12-06 18:11	root	root	drwxr-xr-x	4096
demo		2025-09-10 09:58	root	root	drwxr-xr-x	4096
snap		2023-12-06 14:17	root	root	drwx------	4096
test		2023-12-06 17:13	root	root	drwxr-xr-x	4096
.bash_history	5	2025-07-24 10:57	root	root	-rw-------	5318
.bashrc	3	2025-07-24 10:57	root	root	-rw-r--r--	3869
.lesshst	1	2023-12-06 18:23	root	root	-rw-------	20
.profile	1	2019-07-09 18:05	root	root	-rw-r--r--	161
.python_history	1	2024-11-08 16:33	root	root	-rw-------	85
.viminfo	11	2025-07-24 10:57	root	root	-rw-------	12162
.wget-hsts	1	2024-11-08 16:02	root	root	-rw-r--r--	183
.Xauthority	1	2025-07-24 10:56	root	root	-rw-------	108
cuda_12.4.0_550.54.14_linux....	4 349 954	2024-02-29 09:51	root	root	-rw-r--r--	4454353277
java.sh	1	2023-12-06 17:19	root	root	-rw-r--r--	833
Miniconda3-py312_24.9.2-0-Li...	144 860	2024-10-23 17:21	root	root	-rw-r--r--	148337011

6.1.4lerobot工程包上传

将**lerobot.zip**文件拖到云主机上。

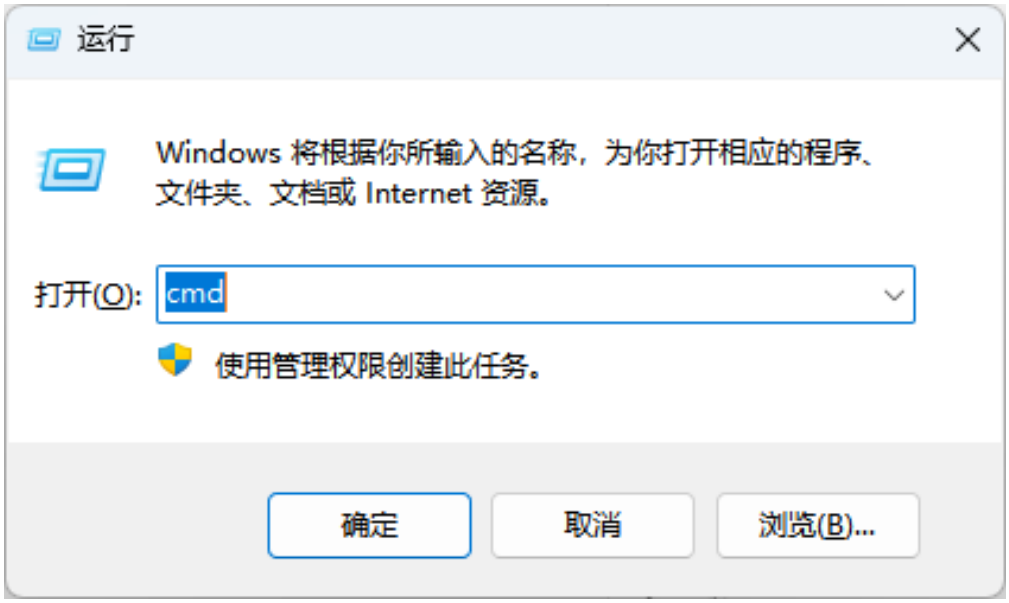


传输完成后，可查看`root`目录下是否有一个**lerobot.zip**文件。

6.1.5数据集训练

点击复制**SSH登录命令**。

win+R打开控制面板，输入**cmd**，打开终端。



将指令复制到终端中，点击**Enter**，输入**yes**后点击**Enter**。

```
C:\Users\Admin>ssh root@js2.blockelite.cn -p 17024
The authenticity of host '[js2.blockelite.cn]:17024 ([180.127.11.167]:17024)' can't be established.
ED25519 key fingerprint is SHA256:WEeKFd90sfhAztcvbci9bH98kReJlkMNgvlkskxTnJA.
This host key is known by the following other names/addresses:
    C:\Users\Admin/.ssh/known_hosts:13: [js2.blockelite.cn]:16928
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

输入云主机密码。

下图所示则为登录成功。

使用`apt`工具下载**screen**。

```
apt install -y screen
```

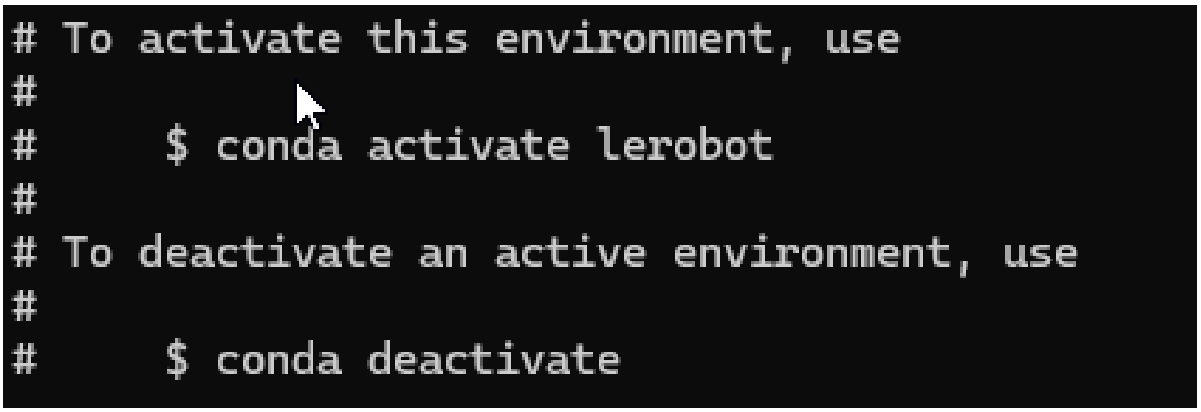
使用`unzip`工具解压**lerobot.zip**文件。

```
unzip lerobot.zip
```

使用**Miniconda**创建虚拟环境并安装**ffmpeg**。

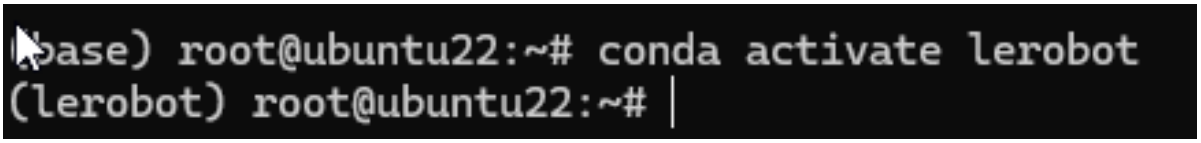
```
conda create -n lerobot python=3.10.18 ffmpeg=7.1.1 -c conda-forge
```

出现下图所示的提示，则说明创建成功。



激活虚拟环境。

```
conda activate lerobot
```

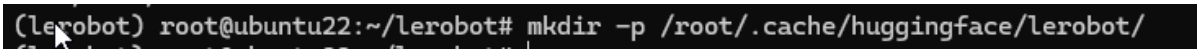


使用`cd`工具，跳转到**lerobot**目录下，安装**lerobot**的依赖包并且指定添加飞特舵机相关的驱动。

```
cd lerobot
pip install -e ".[feetech]" -i https://pypi.tuna.tsinghua.edu.cn/simple
```

输入指令，创建数据集存储路径。

```
mkdir -p /root/.cache/huggingface/lerobot/
```



将`/root/demo`移到`/root/.cache/huggingface/lerobot`下。

```
mv /root/demo/ /root/.cache/huggingface/lerobot/
```

创建一个名为**lerobot**的**session**，如需退出**session**，**Ctrl+a+d**即可退出**session**（**screen**是一个终端复用器，它允许你在一个终端窗口中创建多个虚拟终端会话）。

```
screen -S lerobot
```

输入指令，即可进入**session**。

```
screen -r lerobot
```

激活虚拟环境。

```
conda activate lerobot
```

```
(base) root@ubuntu22:~# conda activate lerobot
(lerobot) root@ubuntu22:~#
```

使用`cd`指令进行跳转目录，输入指令，跳转到`~/lerobot/src`下。

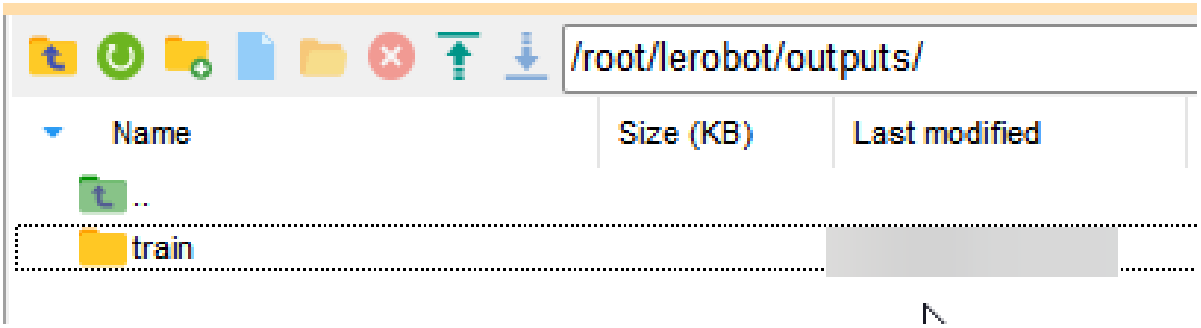
```
(lerobot) root@ubuntu22:~/lerobot# cd ~/lerobot/src/
```

输入指令，进行训练。

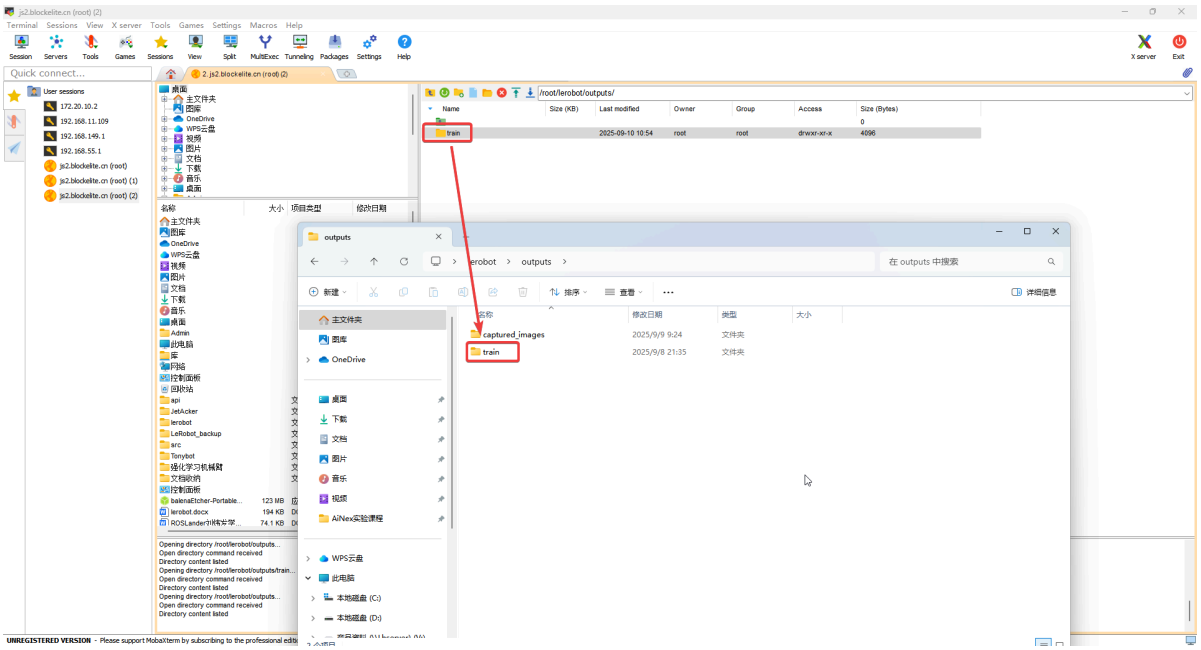
```
python lerobot/scripts/train.py --dataset.repo_id=demo --policy.type=act --
output_dir=outputs/train/act_so101_test --job_name=act_so101_test --
policy.device=cuda --wandb.enable=false  --policy.push_to_hub=false
```

```
(lerobot) root@ubuntu22:~/lerobot/src# python lerobot/scripts/train.py --dataset.repo_id=demo --policy.type=act --output
_dir=outputs/train/act_so101_test --job_name=act_so101_test --policy.device=cuda --wandb.enable=false  --policy.push_to_
hub=false
```

训练完，将在`lerobot/outputs`生成一个train的文件夹，具体路径为`/root/lerobot/outputs/train`。



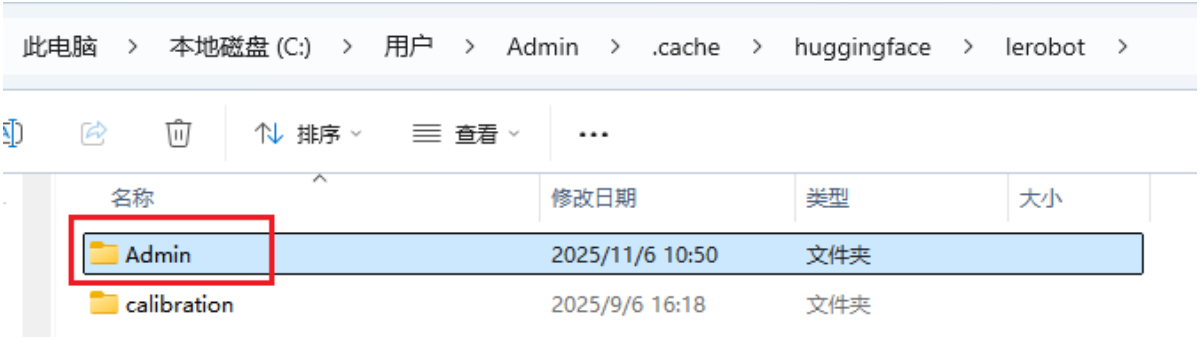
继续使用**SFTP**，将训练好的文件传输到本机上的工程文件下，路径为：`lerobot\outputs`。



至此就完成训练步骤了，接下来就可以推理测试了。

6.2 本机训练

数据集默认保存到`Admin\.cache\huggingface\lerobot`中。



win+X点击`终端管理员`，以管理员权限打开命令行终端。

输入指令，激活虚拟环境并跳到**lerobot**工作目录下。

```
conda activate lerobot
cd .\Desktop\lerobot\
```

```
(base) PS C:\Users\Admin> conda activate lerobot
(lerobot) PS C:\Users\Admin> cd .\Desktop\lerobot\
```

> **Note**
>
> - **没有独显的电脑不推荐在本机上训练。**
> - **下面的步骤需要电脑独显方可操作。**

输入指令，安装**cuda11.8**。

```
pip install torch==2.7.1 torchvision==0.22.1 torchaudio==2.7.1 --index-url
https://download.pytorch.org/whl/cu118
```

```
(lerobot) PS C:\Users\Admin\Desktop\lerobot> pip install torch==2.7.1 torchvision==0.22.1 torchaudio==2.7.1 --index-url https://download.pytorch.org/whl/cu118
```

模型训练的参数配置文件在`src\lerobot\configs\train.py`里，如下图所示，根据需求自行调整模型训练的配置。

```python
    dataset: DatasetConfig
    env: envs.EnvConfig | None = None
    policy: PreTrainedConfig | None = None
    # Set `dir` to where you would like to save all of the run outputs. If you run another training session
    # with the same value for `dir` its contents will be overwritten unless you set `resume` to true.
    output_dir: Path | None = None
    job_name: str | None = None
    # Set `resume` to true to resume a previous run. In order for this to work, you will need to make sure
    # `dir` is the directory of an existing run with at least one checkpoint in it.
    # Note that when resuming a run, the default behavior is to use the configuration from the checkpoint,
    # regardless of what's provided with the training command at the time of resumption.
    resume: bool = False
    # `seed` is used for training (eg: model initialization, dataset shuffling)
    # AND for the evaluation environments.
    seed: int | None = 1000
    # Number of workers for the dataloader.
    num_workers: int = 4
    batch_size: int = 8
    steps: int = 100_000
    eval_freq: int = 20_000
    log_freq: int = 200
    save_checkpoint: bool = True
    # Checkpoint is saved every `save_freq` training iterations and after the last training step.
    save_freq: int = 20_000
    use_policy_training_preset: bool = True
    optimizer: OptimizerConfig | None = None
    scheduler: LRSchedulerConfig | None = None
    eval: EvalConfig = field(default_factory=EvalConfig)
    wandb: WandBConfig = field(default_factory=WandBConfig)

```

参数说明：

- `dataset`：数据集配置对象，定义训练使用的数据集
- `env`：环境配置对象，用于定义机器人仿真或真实环境的参数（可选）
- `policy`：预训练模型配置，定义策略网络的架构和参数（可选）
- `output_dir`：训练输出目录路径，存储模型检查点、日志等文件。若使用相同路径再次训练会覆盖内容（除非启用 resume）
- `job_name`：任务名称，用于标识当前训练任务（可选）
- `resume`：是否从检查点恢复训练。设为 True 时需确保 output_dir 中存在至少一个检查点文件
- `seed`：随机种子（默认 1000），用于控制模型初始化、数据集打乱和评估环境的随机性，确保实验可复现
- `num_workers`：数据加载器的工作进程数（默认 4），影响数据加载速度

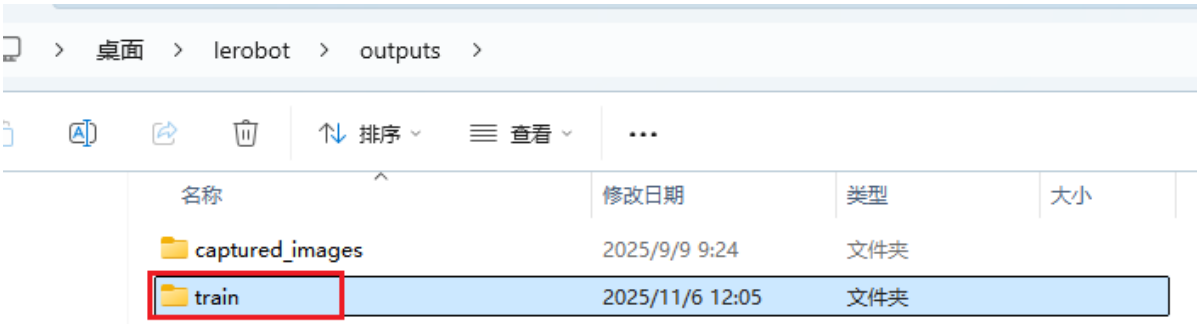
- `batch_size`：批次大小（默认 8），每次迭代处理的样本数量
- `steps`：总训练步数（默认 100,000）
- `eval_freq`：评估频率（默认每 20,000 步），定期在验证集上评估模型性能
- `log_freq`：日志记录频率（默认每 200 步），控制训练指标的记录间隔
- `save_checkpoint`：是否保存训练检查点（默认 True）
- `save_freq`：检查点保存频率（默认每 20,000 步），最后一步训练后也会保存
- `use_policy_training_preset`：是否使用策略训练预设配置（默认 True）
- `optimizer`：优化器配置对象，定义学习率、权重衰减等参数（可选）
- `scheduler`：学习率调度器配置，定义学习率衰减策略（可选）
- `eval`：评估配置对象，包含评估相关的详细设置
- `wandb`：Weights & Biases 配置对象，用于实验跟踪和可视化

输入指令，进行训练。

```
python src/lerobot/scripts/train.py --dataset.repo_id=${HF_USER}/demo --
policy.type=act --output_dir=outputs/train/act_so101_test --
job_name=act_so101_test --policy.device=cuda --wandb.enable=false  --
policy.push_to_hub=false
```

```
(lerobot) PS C:\Users\Admin\Desktop\lerobot> python src/lerobot/scripts/train.py --dataset.repo_id=Amdin/demo --policy.t
ype=act --output_dir=outputs/train/act_so101_test --job_name=act_so101_test --policy.device=cuda --wandb.enable=false  -
-policy.push_to_hub=false
```

训练完，将在`lerobot/outputs`生成一个train的文件夹，具体路径为`C:/Users/Admin/Desktop/lerobot/outputs/train`。



7.模型测试

SO-ARM101开源6轴机械臂推理是通过数据驱动的方式，让机器人模仿人类行为。它将复杂的机器人控制问题视为一个监督学习任务，其推理过程依赖于训练好的模型，该模型能够将从环境（主要是视觉观察）和自然语言指令中提取的信息进行多模态融合，并直接映射为控制机器人的具体动作。本质上，这是一种基于“行为克隆”的模仿学习，旨在让机器人通过学习人类演示数据，获得从感知到行动的端到端推理能力，从而完成特定任务。

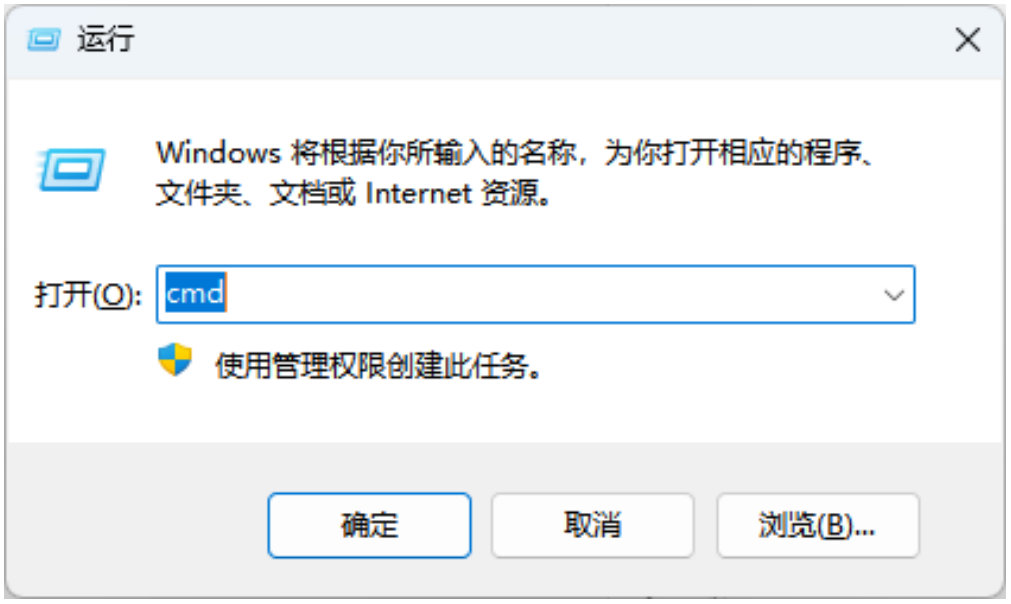
推理部署步骤：

1. 加载训练好的策略模型。
2. 初始化物理机器人及其传感器。
3. 建立传感器数据到策略输入的映射。
4. 建立策略输出到机器人控制指令的映射。
5. 在控制循环中执行策略推理和机器人控制。

6. 可选的实时可视化和监控。

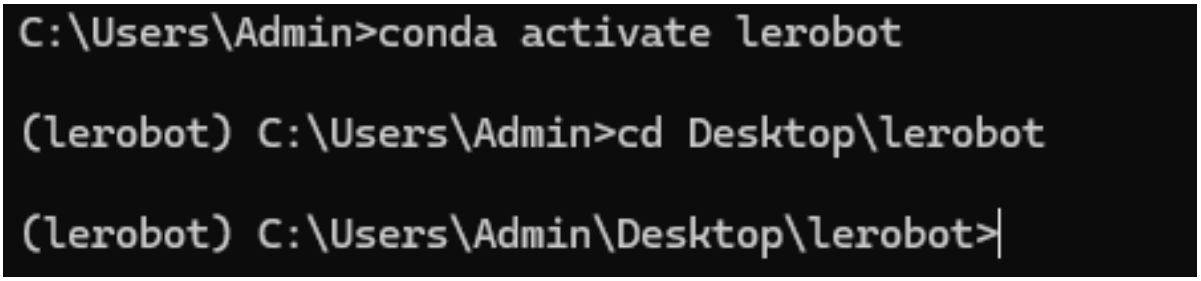
7.1实时推理测试

win+R打开控制面板，输入**cmd**，打开终端。



输入指令，进入虚拟环境并进入到工程目录下。

```
conda activate lerobot
cd Desktop\lerobot
```

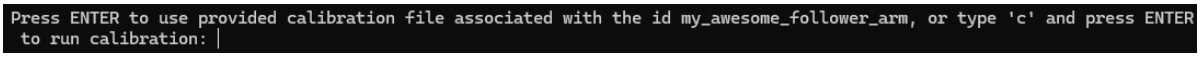


输入指令，进行推理。

> **Note**
>
> `outputs/train/act_so101_test/checkpoints/100000/pretrained_model`**这里的100000为你设置的** `steps` **参数（总训练步数）。**

```
python -m lerobot.record --robot.type=so101_follower --robot.port=COM24 --
robot.id=my_awesome_follower_arm --robot.cameras="{ handeye: {type: opencv,
index_or_path: 0, width: 640, height: 480, fps: 30}, front: {type: opencv,
index_or_path: 1, width: 640, height: 480, fps: 30}}" --display_data=true --
dataset.repo_id=${HF_USER}/eval_so101 --dataset.single_task="Grab the
screwdriver" --
policy.path=outputs/train/act_so101_test/checkpoints/100000/pretrained_model
```

当出现下图提示，按下**Enter**继续。



当出现下图标志，则说明机械臂开始推理。

出现下图标志，则说明机械臂推理结束。

按下→进入下一轮推理。

7.2推理时常见问题

问题1：分布偏移与泛化能力差

问题描述：模型在训练时见过的状态（如物体位置、光照、背景）下表现良好，但一旦遇到与训练数据稍有不同的情况（即“分布外”数据），性能就会急剧下降。例如，训练时是白色桌子，推理时换成木纹桌子，机器人可能就无法识别物体了。

根本原因：模型学习到的是训练数据中状态的静态分布，而非对物理世界的深层理解。它是在“记忆”而非真正“理解”任务。

问题2：复合误差

问题描述：这是行为克隆最经典的问题。在推理的每一步，模型都会产生微小的动作误差。由于是开环执行（没有根据结果实时修正），下一步的输入状态已经是上一个错误动作导致的结果，这与训练数据中的理想状态越来越远。误差会一步步累积，最终导致机器人完全偏离正确轨迹，任务失败。

根本原因：推理过程中存在误差，当误差一直在累积的话，就会逐渐偏离原本的轨道。

问题3：对演示风格的过度拟合

问题描述：模型可能不仅学习了完成任务所需的必要动作，还学习了演示者的个人习惯或特定风格（如一种奇怪的、低效的抓取姿势）。当环境变化时，这种僵化的风格可能不再有效。

根本原因：模型的目标是最大限度地模仿演示数据，而演示数据中可能包含非最优或与任务无关的行为模式。