

四足的强化学习入门

0.

本文档是一个“急功近利”的文档，可以快速入门强化学习的从训练到适配的过程，重点强调每个模块的输入和输出，细节需要慢慢学习。

可以在ubuntu22.04和ubuntu20.04上实现。

1.硬件要求以及驱动安装

1.1硬件要求

显卡准备，3060以上，12GB显存以上

1.2驱动安装

可以参考以下文档

<https://zhuanlan.zhihu.com/p/560826876>

豆先生：Anaconda介绍、安装及使用教程 *(关于anaconda的系统资料)*

Ubuntu下apt工具包安装NVIDIA driver *(驱动安装)*

CUDA Toolkit Archive *(Cuda离线安装、推荐这样安装)*

nvidia驱动安装：[https://link.zhihu.com/?](https://link.zhihu.com/?target=https%3A//blog.csdn.net/weixin_46584887/article/details/122726265)

[target=https%3A//blog.csdn.net/weixin_46584887/article/details/122726265](https://link.zhihu.com/?target=https%3A//blog.csdn.net/weixin_46584887/article/details/122726265)

安装完成后用nvidia-smi命令(查看nvidia驱动)，“CUDA Version”提示安装cuda版本的上限，它是12.2，

所以我选择安装12.0

cuda安装：<https://link.zhihu.com/?target=https%3A//developer.nvidia.com/cuda-toolkit-archive>

Cuda installation guide:<https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#ubuntu>

4. Choose an installation method: [local repo](#) or [network repo](#).

3.9.2. Local Repo Installation for Ubuntu

1. Install local repository on file system:

```
sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb
```

2. Enroll ephemeral public GPG key:

```
sudo cp /var/cuda-repo-<distro>-X-Y-local/cuda-*-keyring.gpg /usr/share/keyrings/
```

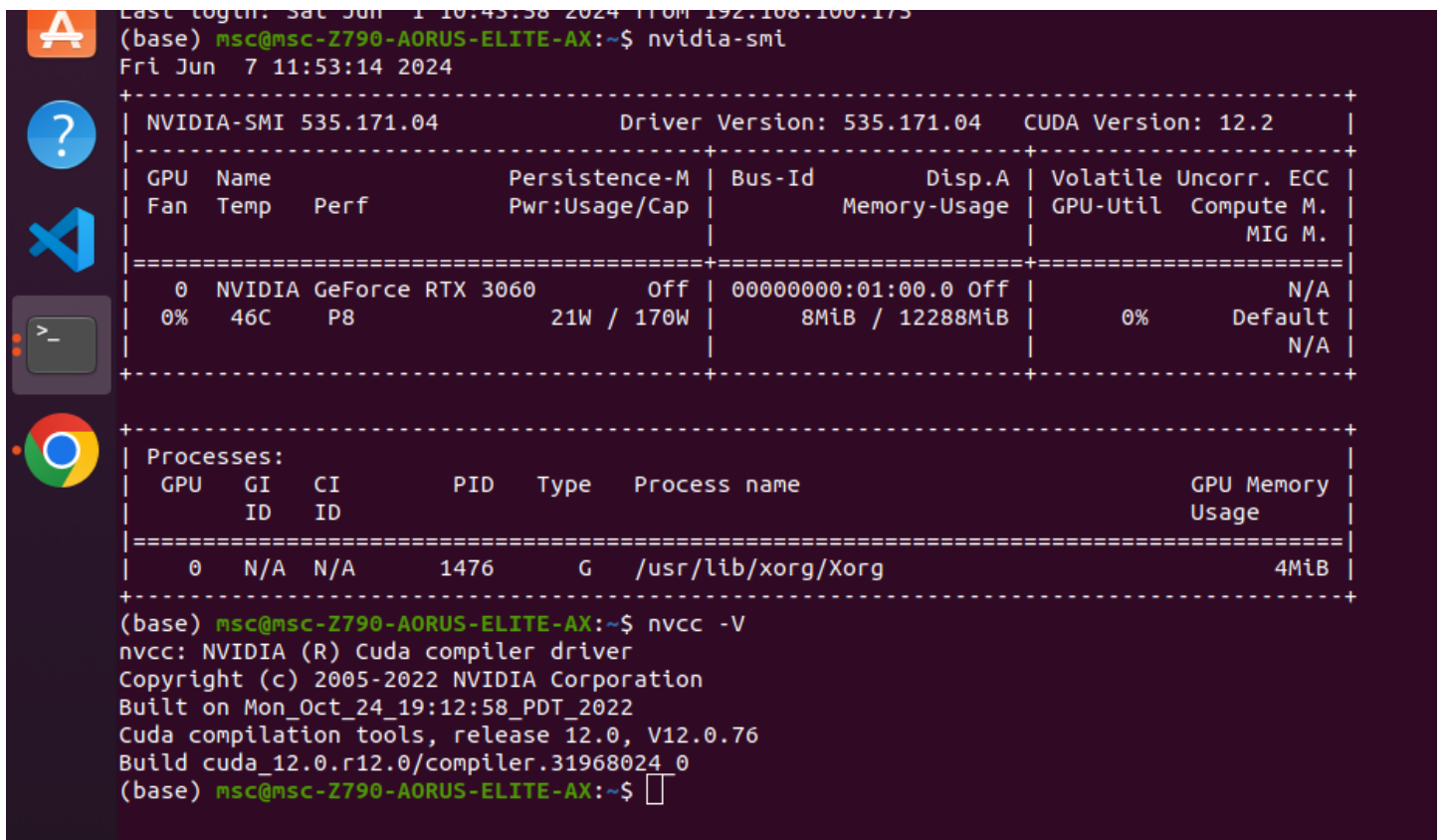
3. Add pin file to prioritize CUDA repository:

```
wget https://developer.download.nvidia.com/compute/cuda/repos/<distro>/x86_64/cuda-<distro>.pin  
sudo mv cuda-<distro>.pin /etc/apt/preferences.d/cuda-repository-pin-600
```

3.9.3. Network Repo Installation for Ubuntu

The new GPG public key for the CUDA repository is [3bf863cc](#). This must be enrolled on the system, either manually; the `apt-key` command is deprecated and not recommended.

安装完成后用`nvcc -V`(查看cuda版本)命令查看



```
Last login: Sat Jun  1 10:43:38 2024 from 192.168.100.173  
(base) msc@msc-Z790-AORUS-ELITE-AX:~$ nvidia-smi  
Fri Jun  7 11:53:14 2024  
+-----+  
| NVIDIA-SMI 535.171.04                Driver Version: 535.171.04      CUDA Version: 12.2      |  
+-----+-----+-----+  
| GPU   Name                Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |  
| Fan   Temp   Perf          Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |  
|====+=====+====+  
|  0  NVIDIA GeForce RTX 3060            Off | 00000000:01:00.0 Off |          N/A         |  
| 0%    46C    P8              21W / 170W |  8MiB / 12288MiB |      0%      Default |  
|====+=====+====+  
+-----+-----+-----+  
+-----+  
| Processes: |  
| GPU   GI    CI          PID    Type   Process name                  GPU Memory |  
| ID     ID     ID              |                 | Usage     |  
+-----+  
|  0     N/A  N/A           1476     G   /usr/lib/xorg/Xorg             4MiB      |  
+-----+  
(base) msc@msc-Z790-AORUS-ELITE-AX:~$ nvcc -V  
nvcc: NVIDIA (R) Cuda compiler driver  
Copyright (c) 2005-2022 NVIDIA Corporation  
Built on Mon_Oct_24_19:12:58_PDT_2022  
Cuda compilation tools, release 12.0, V12.0.76  
Build cuda_12.0.r12.0/compiler.31968024_0  
(base) msc@msc-Z790-AORUS-ELITE-AX:~$
```

1.3anaconda的安装

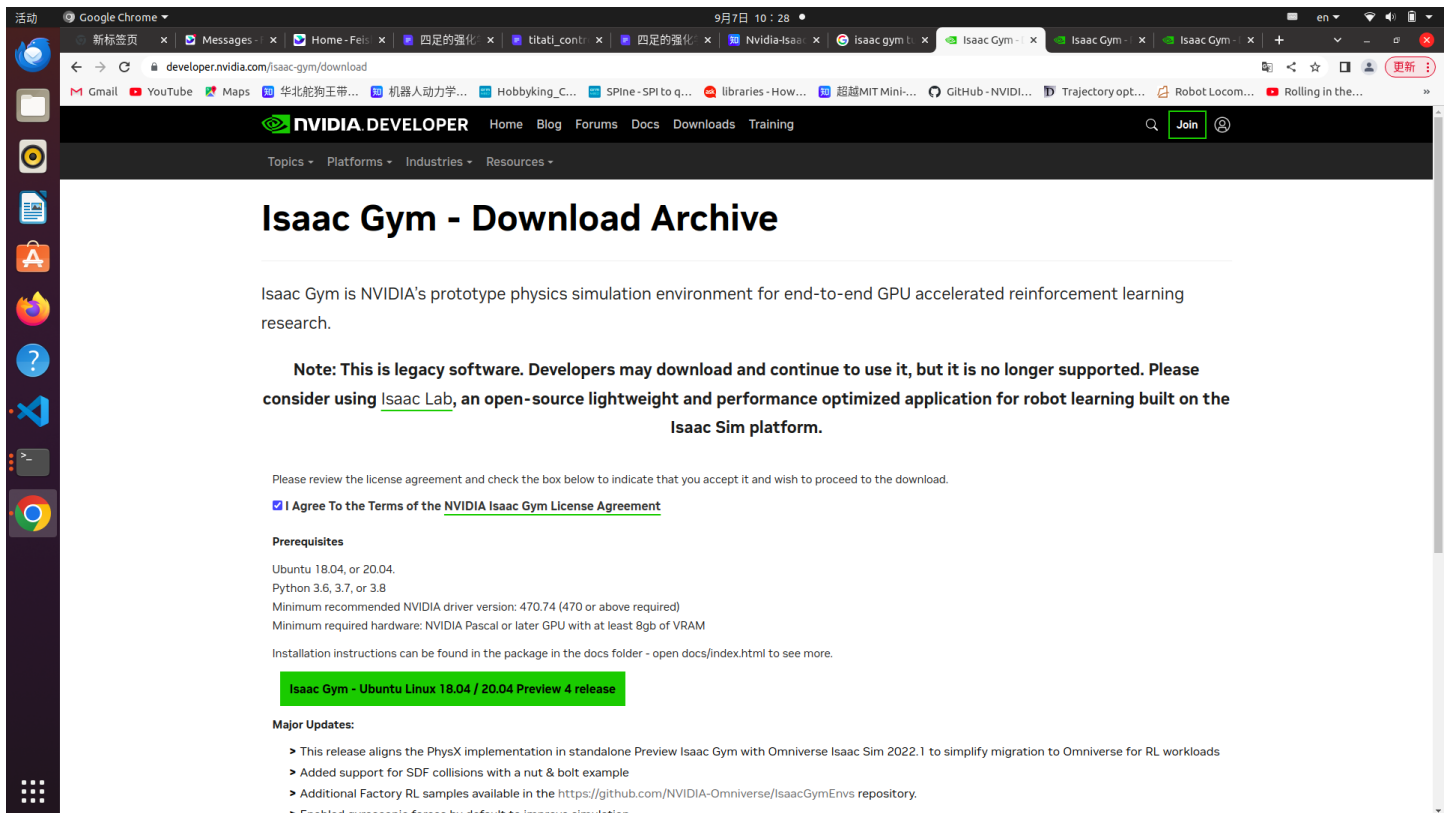
<https://zhuanlan.zhihu.com/p/32925500>,



conda相关的命令稍后会介绍，它能够建立一个虚拟环境，配置相应的python版本以及cuda-toolkit版本，在不影响本机版本的条件下保证算法和环境的适配。

2.安装issacgym

<https://developer.nvidia.com/isaac-gym/download>



安装后解压即可

3 conda配置虚拟环境

```
1 conda create -n your_env_name python=3.8
```

该环境配置，你能在你的路径/anaconda3/envs/your_env_name找到
之后，激活环境能在终端开头看到 “ (your_env_name) ”

```
1 conda activate your_env_name
2 export
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:your_path/anaconda3/envs/your_env_name/lib
```

3.1 一个简单的测试

```
1 pip3 install torch==1.10.0+cu113 torchvision==0.11.1+cu113
  torchaudio==0.10.0+cu113 -f
  https://download.pytorch.org/whl/cu113/torch_stable.html
```

```
1 cd 你的路径/isaacgym/python && pip install -e .
```

```
1 cd examples && python 1080_balls_of_solitude.py
```

看到一堆球落到地上表示成功

注意：如果不跑强化学习算法和强化学习环境，记得关闭虚拟环境，不然影响ros或者gazebo的正常运行

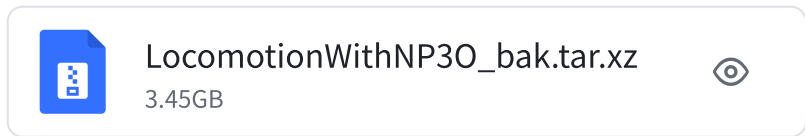
```
conda deactivate
```

相比 walk-these-way和 HIMLoco两个经典的强化学习git，该仓库具有训练时间较短并且训练效果比较好的特点，我的3060训一次7h，这两个要24h以上

源码作者https://space.bilibili.com/788372?spm_id_from=333.788.0.0

源码github<https://github.com/zeonsunlightyu/LocomotionWithNP30/tree/master>

我的导入titati近似模型的修改版本,the latest dev branch works **worse** than the older.

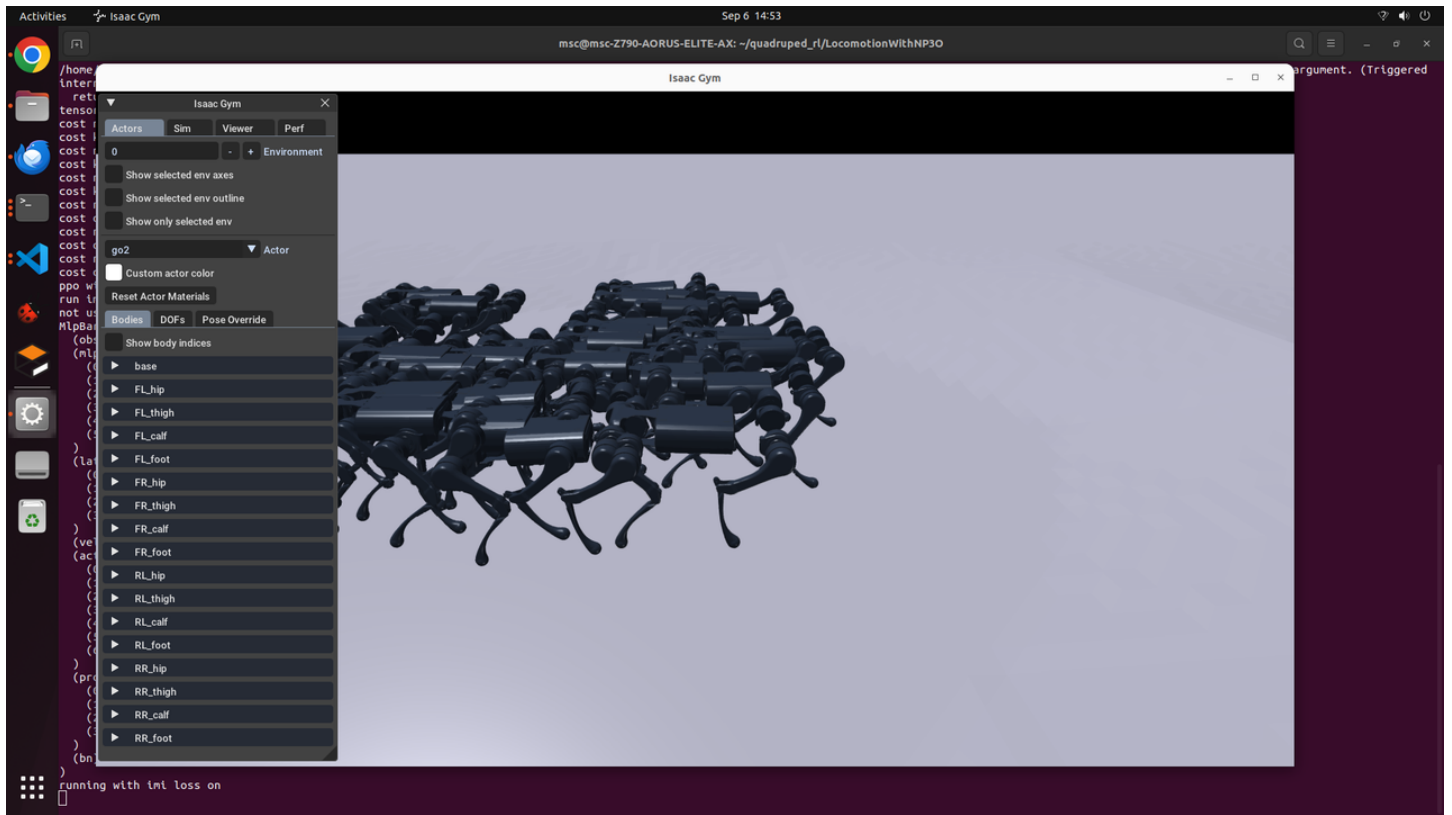


```
cd LocomotionWithNP30_bak
```

```
conda activate your_env_name
```

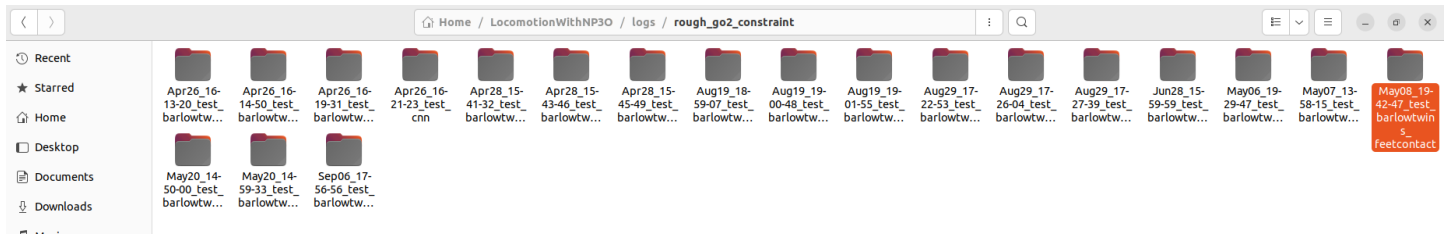
```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:your path/anaconda3/envs/msc_rl_env/lib
```

```
python train.py --task=go2N3poHim(显卡不好会非常卡，看到如下图片，表示程序正常执行，ctrl+c退出)
```



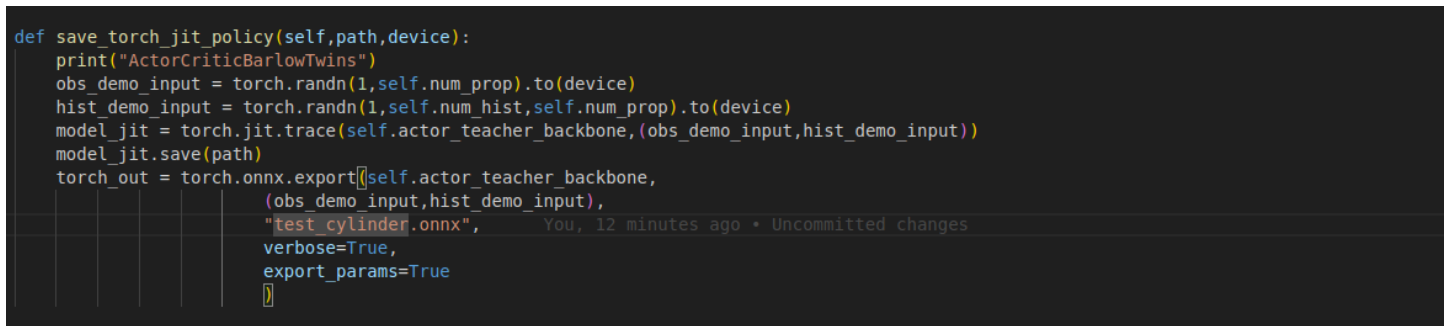
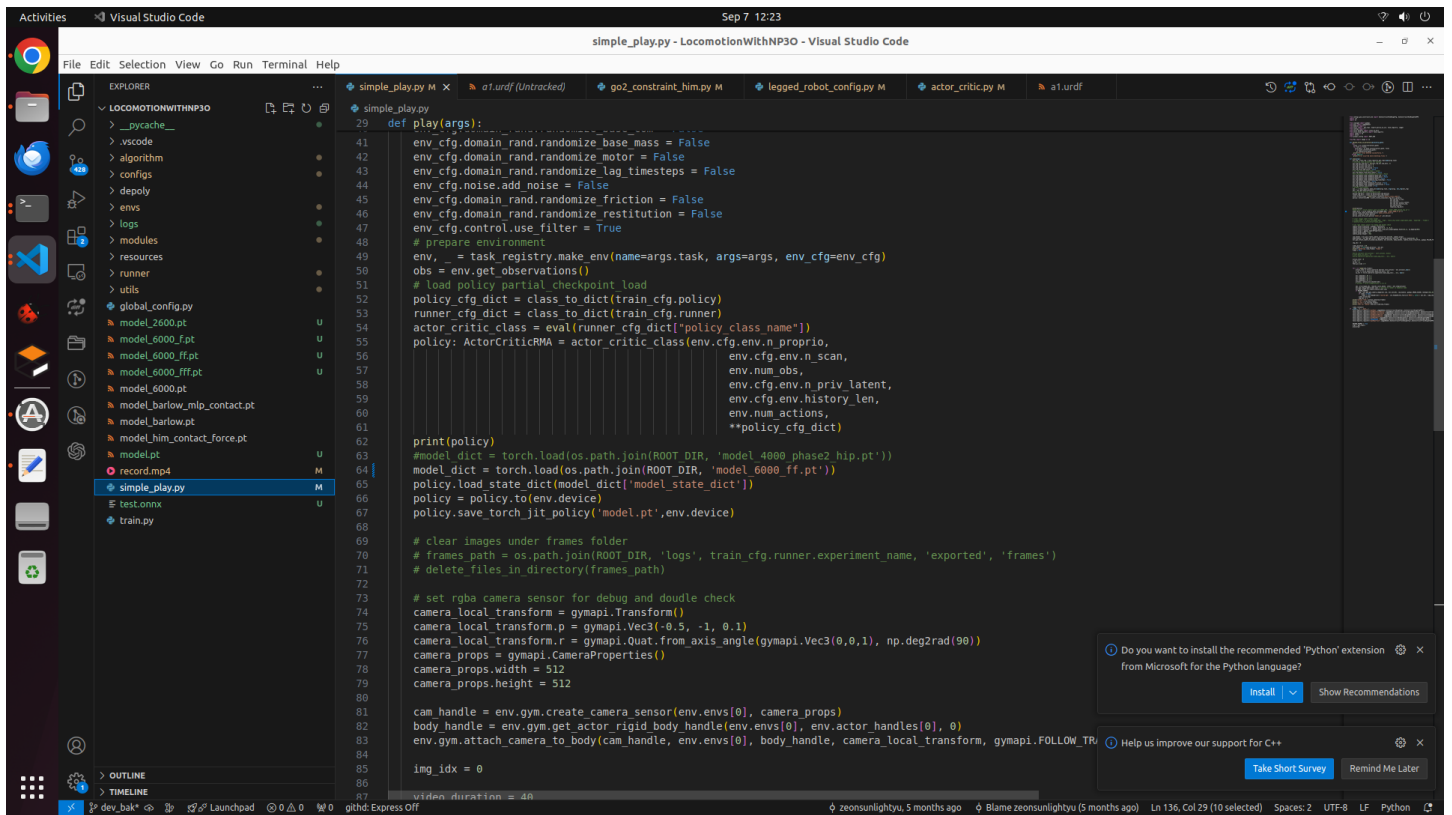
```
python train.py --task=go2N3poHim --headless (这个不显示强化学习的特色交互窗口，比较快，我的3060训6000次需要7h)
```

训练的结果在如下文件夹

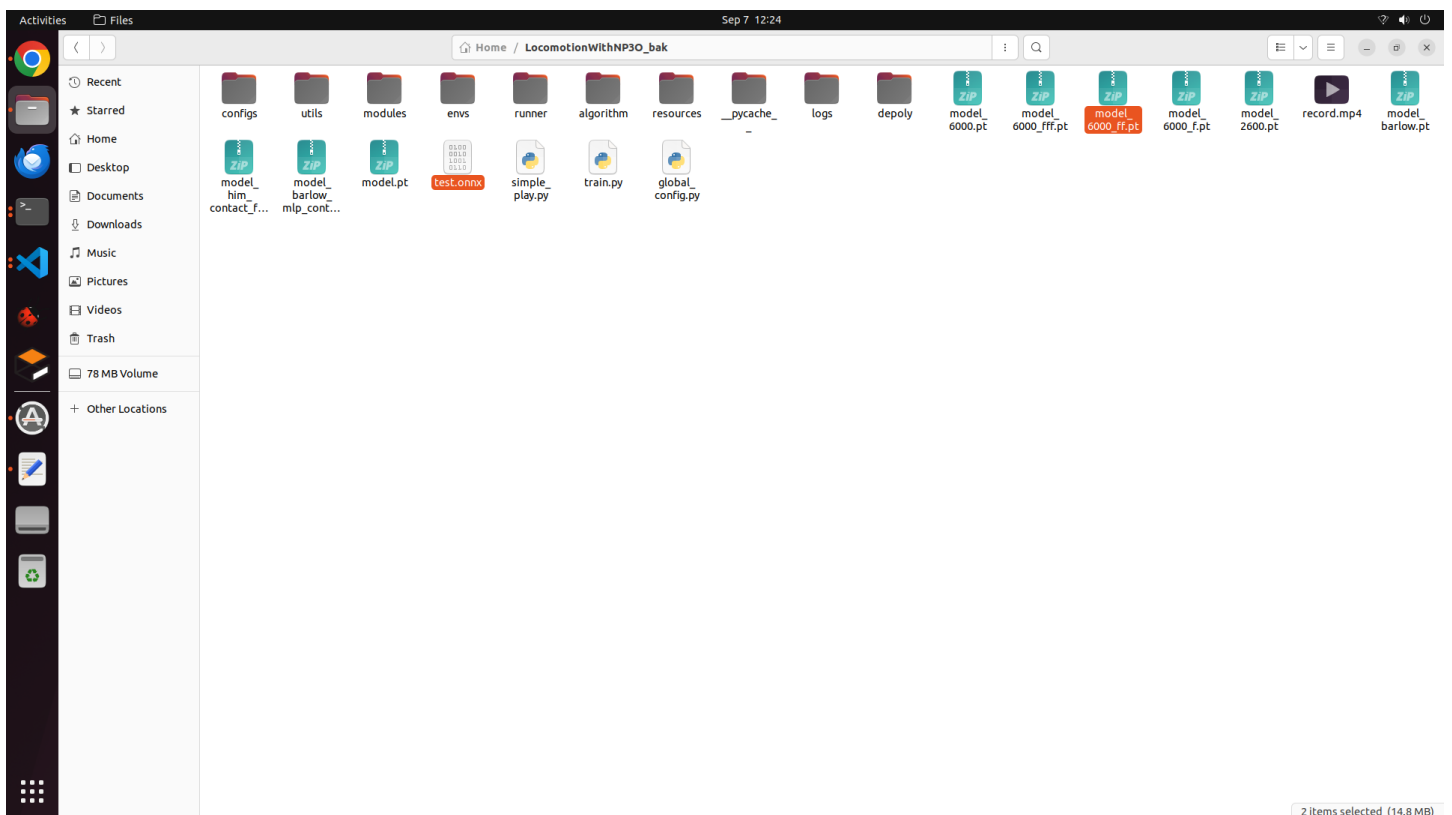


我将上述文件的最终训练结果model_6000.pt复制为model_6000_ff.pt，放到 LocomotionWithNP3O_bak目录下

python simple_play.py --task=go2N3poHim，（9000次训练完成后，用该命令来生成策略网路）



生成了我们需要用到的test.onnx文件，这是描述神经网络的通用格式文件



5代表policy的网络的输入和输出

policy的输入有两个，一个是1*45的观测值输入，一个是10*45的观测值输入

1*45维的观测值输入:

1*3 : local angular velocity

1*3 : Rotation_Matrix_from_world_to_base * (0, 0, -1): 表示局部重力矢量

1*3: x_vel_cmd, y_vel_cmd, yaw_turn_cmd

1*12: joint pos

1*12: joint vel

1*12: last joint action

$45 = 3 + 3 + 3 + 12 + 12 + 12$

10*45的观测值输入:

{前10个时刻的1*45观测值输入, 前9个时刻的1*45观测值输入,, 前1个时刻的1*45观测值输入}

policy的输出有一个, 是一个1*12维的动作值输出joint_action

joint_action与joint_desired_pos的关系

$\text{joint_desired_pos} = \text{joint_action} * \text{action_scale} + \text{default_joint_pos}$