**WSL Introduction**

什么是WSL？

Windows Subsystem for Linux，适用于 Linux 的 Windows 子系统可让开发人员按原样运行 GNU/Linux 环境 - 包括大多数命令行工具、实用工具和应用程序 - 且不会产生虚拟机开销。

什么是WSL2？

1. WSL 2 是 WSL 中体系结构的新版本，它更改 Linux 发行版与 Windows 交互的方式。
2. WSL 2 的主要目标是提高文件系统性能并增加系统调用的完全兼容性。
3. 每个 Linux 发行版都可以作为 WSL 1 或 WSL 2 发行版运行，并可随时进行切换。
4. WSL 2 是底层体系结构的主要功能，它使用虚拟化技术和 Linux 内核来实现其新功能。

在WSL中，你可以?

1. 在 Microsoft Store 中选择你喜欢的Linux （Ubuntu/ Debian/ SUSE 等等）。
2. 运行常用的软件/Bash shell脚本/docker 等。包括但不限于：

工具：vim、emacs、tmux

语言：Javascript/node.js、Ruby、Python、C/C++、 C# 和 F#、Rust、Go 等等

服务：sshd、MySQL、Apache、lighttpd

1. 在 Windows 上调用 GNU/Linux 应用程序。

**WSL Related Commands**

|  |  |
| --- | --- |
| 介绍 | 指令 |
| 查看当前的版本 | wsl -l -v; wsl –list; wslconfig /list |
| 在以后安装 Linux 的时候默认启用 WSL2 | wsl --set-default-version 2 |
| 启动WSL服务 | wsl |
| 关闭WSL服务 | wsl –terminate Ubuntu-20.04; wsl -t Ubuntu-20.04 |
| 更新WSL内核 | wsl.exe --update |

**Ubuntu Commands**

|  |  |
| --- | --- |
| 介绍 | 指令 |
| 更新软件源 | sudo apt-get update -y |
| 升级所有软件 | sudo apt-get upgrade -y |
| 安装systemctl | git clone <https://github.com/DamionGans/ubuntu-wsl2-systemd-script.git>  cd ubuntu-wsl2-systemd-script/  bash ubuntu-wsl2-systemd-script.sh |
| 安装netstat | sudo apt-get install net-tools |
| 监听所有端口 | sudo netstat -tlnp |
| 启动 SSH 服务 | sudo systemctl start ssh |
| 重启SSH服务 | sudo systemctl restart ssh |
| 检查SSH服务状态 | sudo systemctl status ssh |
| 访问SSH服务 | ssh name@ip -p 2222 # name是用户名，ip是windows的ip，2222设置的端口 |
| 测试cuda安装成功与否 | nvcc -V; nvcc --version |
| 测试cudnn安装成功与否  如果 cuDNN 已成功安装，则应显示 cuDNN 版本和其他相关信息。 | cat /usr/local/cuda/include/cudnn.h | grep CUDNN\_MAJOR -A 2 |
| 安装desktop environment | sudo apt install xfce4 |
| vcXsrv界面启动(GUI 界面) | sudo startxfce4 |
| 关闭vcXsrv服务 |  |
| chrome 启动 | google-chrome --no-sandbox |
| Check and delete /run/nologin | ls -l /run/nologin  rm /run/nologin |
| 启动cpolar服务 | sudo systemctl start cpolar |
| 停止cpolar服务 | sudo systemctl stop cpolar |
| 查看cpolar状态 | sudo systemctl status cpolar |
| 查看 cpolar 服务的活动 TCP 域名和端口 | sudo netstat -tnlp |
| Wsl2上打开win桌面软件 | /mnt/c/Windows/System32/cmd.exe /c start "C:\Users\Administrator\Desktop\向日葵远程控制.lnk" |
| 强制用户pts/0下线 | pkill -kill -t pts/0 |
| 显示计算资源（cpu, gpu等） | lscpu(cpu), free -h(内存), df -h(磁盘信息), nvidia-smi(gtx) |
| 配置wsl资源 | 打开Windows资源管理器,地址栏输入 %UserProfile%，修改.wslconfig文件内容 |
| apt install的所有包 | apt list --installed |
| 复制服务器A到服务器B | scp /home/work/source.txt work@192.168.0.10:/home/work/ #把本地的source.txt文件拷贝到192.168.0.10机器上的/home/work目录下 |

**CMD Commands**

|  |  |
| --- | --- |
| 介绍 | 指令 |
| 查看网络配置 | ifconfig |

**Screen Commands**

|  |  |
| --- | --- |
| 介绍 | 指令 |
| 新建名字为name的session | screen -S name |
| 远程detach 某个session | screen -d name |
| 恢复特定id窗口(name) | screen -r id(name) |
| 查看当前所有session | screen -ls |
| 切换至下一个窗口 | CTRL+a+n |
| 切换至上一个窗口 | CTRL+a+p |
| Detach当前窗口 | CTRL+a+d |
| 进入screen后，当按下tab键时，会闪屏，可通过ctrl + a & ctrl + g来停止当前screen的闪屏，如果要对所有的screen生效，在~/.screenrc中加入vbell off。 | |

**Docker 配置（wsl）**

|  |  |
| --- | --- |
| 介绍 | 指令 |
| 安装依赖包 | sudo apt update  sudo-apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common |
| 添加 Docker 的官方 GPG 密钥 | curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - |
| 对于amd64架构计算机添加清华大学TUNA软件源 | sudo add-apt-repository \  "deb [arch=amd64] https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu \  $(lsb\_release -cs) \  stable" |
| 安装最新版Docker Engine-Community和containerd | sudo apt-get install docker-ce docker-ce-cli containerd.io |
| 将当前用户加入新docker组中，重启docker，切换当前bash到新group中 | sudo groupadd docker  sudo gpasswd -a ${USER} docker  sudo service docker restart  newgrp - docker |
| 常用docker指令  镜像相关 | 根据 Dockerfile 构建一个名为myapp的镜像------------------docker build -t myapp .  从镜像仓库中下载指定的镜像----------docker pull <镜像名称>  强制删除的镜像--------------- docker rmi -f <镜像名称或镜像ID>  查看本地docker所有镜像------------------docker images  将本地的镜像推送到镜像仓库中------------------docker push |
| 常用docker指令  容器相关 | 从镜像myapp创建并运行一个新容器------------------docker run --rm myapp –name container\_id  使用镜像启动一个容器，参数为以命令行模式进入--------- docker run -it <ubuntu> /bin/bash  查看正在运行的容器------------------docker ps(-a, 显示全部)  停止一个运行中的容器------------------docker stop <容器名称或容器ID>  删除一个容器---------docker rm <容器名称或容器ID>  在运行的容器中执行指定的命令---------docker exec -it <容器名称或容器ID> <命令>  查看容器的日志---------docker logs <容器名称或容器ID>  重新启动已停止的容器---------docker start <容器名称或容器ID>( docker exec -it <容器ID或容器名称> /bin/bash) |
| 常用docker指令  仓库和网络相关 | 在本地登录到远程仓库---------------docker login --username=[用户名] -p=[密码] 远程ip:端口  注销当前 Docker 镜像仓库的登录状态-------docker logout  创建一个新的网络--------docker network create <网络名称>  从镜像仓库中下载指定的镜像--------docker pull  将本地的镜像推送到镜像仓库中--------先docker tag再 docker push |
| 其他方面指令 | 查看 Docker 的磁盘使用情况--------docker system df  docker tag ubuntu:20.04 username/ubuntu:20.04  登录Docker hub: docker login, docker search ubuntu, docker logout, docker pull ubuntu, docker image ls  重启docker--------sudo systemctl restart docker |
| 启动一个新的Docker 容器，并将容器内部的端口80映射到主机上的端口8080 | docker run -p 8080:80 myapp  能够在浏览器中访问 “http://localhost:8080”并看到 “Hello, Docker!”的输出。 |

**机架CPU型号**

Intel ark（Google search）; 华硕服务器与工作站（ESC4000-E10S，<https://servers.asus.com.cn/products/servers/gpu-servers/ESC4000-E10S>）；

详细配置：

CPU Intel Xeon Scalable Gold 5318Y, 2.1GHz, 24C, 36M;

主板 Intel C621A PCH

内 存 Samsung 64G DDR4 RECC 3200Mhz

HDD 西部数据SATA 8TB HUS728T8TALE614企业级

SSD 三星 PM9A1 2TB NVME

电 源 2200W (1+1) Pitanium 冗 余 电 源

机 型 ASUS E5C4000 -E10

**安装逻辑**

1. Install wsl2和Ubuntu-20.04
2. 修改apt源\*\*\*\*\*\*sudo cp /etc/apt/sources.list /etc/apt/sources.list.bat

deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse

1. 更新apt\*\*\*\*\*\*sudo apt-get update
2. 安装cmake等工具\*\*\*\*\*\*sudo apt-get install build-essential
3. Install miniconda3
4. wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\\_64.sh
5. bash Miniconda3-latest-Linux-x86\\_64.sh
6. 更换源

生成condarc文件\*\*\*\*\*\*conda config --set show\_channel\_urls yes

vim .condarc内容修改如下

channels:

- defaults

show\_channel\_urls: true

default\_channels:

- http://mirrors.aliyun.com/anaconda/pkgs/main

- http://mirrors.aliyun.com/anaconda/pkgs/r

- http://mirrors.aliyun.com/anaconda/pkgs/msys2

custom\_channels:

conda-forge: http://mirrors.aliyun.com/anaconda/cloud

msys2: http://mirrors.aliyun.com/anaconda/cloud

bioconda: http://mirrors.aliyun.com/anaconda/cloud

menpo: http://mirrors.aliyun.com/anaconda/cloud

pytorch: http://mirrors.aliyun.com/anaconda/cloud

simpleitk: <http://mirrors.aliyun.com/anaconda/cloud>

1. 更新缓存\*\*\*\*\*\*conda clean -i
2. Install cuda和cudnn
3. 首先安装带WSL cuda driver的win11 x86\_64的匹配gtx2060super驱动
4. 采用Runfile安装模式\*\*\*\*\*\*wget https://developer.download.nvidia.com/compute/cuda/12.1.1/local\_installers/cuda\_12.1.1\_530.30.02\_linux.run； sudo sh cuda\_12.1.1\_530.30.02\_linux.run

如果系统自带driver则取消这一项的安装

安装成功后配置

sudo gedit ~/.bashrc

# 在文件末尾添加：export PATH=/usr/local/cuda-11.0/bin${PATH:+:${PATH}}

source ~/.bashrc

1. Cudnn下载Local Installer for Linux x86\_64 (Tar)

解压tar -xvJf cudnn-linux-x86\_64-8.9.1.23\_cuda12-archive.tar.xz

复制

sudo cp cudnn-linux-x86\_64-8.9.1.23\_cuda12-archive/include/cudnn.h /usr/local/cuda/include

sudo cp cudnn-linux-x86\_64-8.9.1.23\_cuda12-archive/lib64/libcudnn\* /usr/local/cuda/lib64

修改权限

sudo chmod a+r /usr/local/cuda/include/cudnn.h

sudo chmod a+r /usr/local/cuda/lib64/libcudnn\*

1. Cudnn的测试包采用Debian Local Installation \*\*\*\*\*\*

sudo dpkg -i cudnn-local-repo-${distro}-8.x.x.x\_1.0-1\_amd64.deb # Enable the local repository

sudo cp /var/cudnn-local-repo-\*/cudnn-local-\*-keyring.gpg /usr/share/keyrings/ # CUDA GPG key

sudo apt-get update # Refresh the repository metadata

进入/var/cudnn-local-repo-ubuntu2004-8.9.3.28/文件夹进行本地安装如下三个文件

sudo apt-get install libcudnn8=8.x.x.x-1+cudaX.Y # runtime library

sudo apt-get install libcudnn8-dev=8.x.x.x-1+cudaX.Y # developer library

sudo apt-get install libcudnn8-samples=8.x.x.x-1+cudaX.Y # code samples

1. 测试

cp -r /usr/src/cudnn\_samples\_v8/ ~

cd ~/cudnn\_samples\_v8/mnistCUDNN/

make clean && make

./mnistCUDNN

如果遇到如下问题

mnistCUDNN sudo make

CUDA\_VERSION is 11010

Linking agains cublasLt = true

CUDA VERSION: 11010

TARGET ARCH: x86\_64

HOST\_ARCH: x86\_64

TARGET OS: linux

SMS: 35 50 53 60 61 62 70 72 75 80 86

test.c:1:10: fatal error: FreeImage.h: No such file or directory

1 | #include "FreeImage.h"

| ^~~~~~~~~~~~~

compilation terminated.

**则安装sudo apt-get install libfreeimage3 libfreeimage-dev**

显示Test passed!即为成功

1. Install pytorch

conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia

1. 配置 ssh
2. 重装open ssh-server\*\*\*\*\*\*sudo apt-get remove openssh-server
3. 修改sshd\_config\*\*\*\*\*\*sudo vim /etc/ssh/sshd\_config

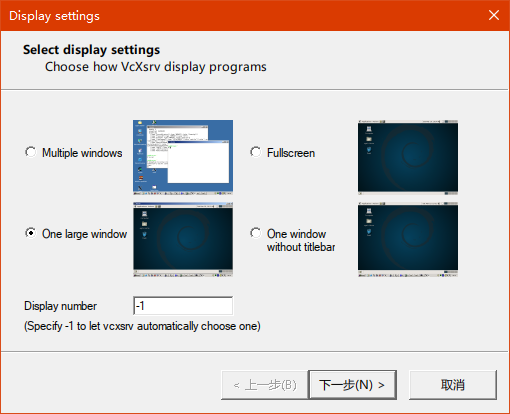
Port 2222 # 监听的端口，可以是其它的

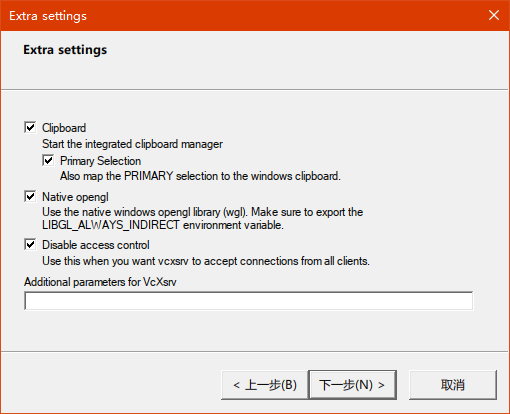
ListenAddress 0.0.0.0 # 0.0.0.0 表示所有的地址

PasswordAuthentication yes # 把原来的no改成yes，意思是可以用密码登录

PermitRootLogin yes # 把原来的prohibit-password改成yes

1. 重启ssh服务\*\*\*\*\*\*service ssh restart
2. 监听本机ip&port转发到wsl的ip&port\*\*\*\*\*\*netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=2222 connectaddress=IP connectport=2222 # IP为你的WSL的IP地址
3. 设置防火墙入站规则\*\*\*\*\*\*netsh advfirewall firewall add rule name=WSL2 dir=in action=allow protocol=TCP localport=2222
4. Vscode相关
5. Install remote SSH plugin&login via name@ip -p
6. Install python on remote SSH’s vscode
7. Install Jupyter
8. Install copilot etc…
9. 安装GUI界面
10. Win11安装VcXsrv, wsl2中安装桌面环境\*\*\*\*\*\*sudo apt install xfce4





1. 配置VcXsrv投影，vim .bashrc\*\*\*\*\*\*export DISPLAY=192.168.31.168:0
2. 重启配置source .bashrc

其中192.168.31.168是以太网适配器WSL的ip地址

！ 所有id和密码 见 id\_pass.txt

**训练逻辑**

硬件需求

|  |  |  |
| --- | --- | --- |
| 量化需求 | 最低GPU显存（推理） | 最低GPU显存（高效参数微调） |
| FP16（无量化） | 13GB | 14GB |
| INT8 | 8GB | 9GB |
| INT4 | 6GB | 7GB |

1. 基于P-Tuning V2微调ChatGLM2-6b-int（微调前缀编码器）

P-Tuning是指在预训练模型的输入层插入一些可训练的连续向量（Prompt），作为任务相关的信息，然后只对这些向量进行微调，而冻结预训练模型的其他参数。这种方法可以减少微调的参数量和数据量，提高微调的效率和泛化能力，但也可能会降低模型的交互性和生成质量。

1. 安装必要库

pip install rouge\_chinese nltk jieba datasets

rouge\_chinese：这是一个用于中文文本摘要评估的库。它提供了一些评估指标，可以用来评估生成的摘要与参考摘要之间的相似度。

nltk：这是自然语言处理（NLP）领域最常用的Python库之一。它提供了许多用于文本处理和分析的功能，例如分词、词性标注、句法分析等。nltk还包含了大量的语料库和数据集，可以用于训练和评估NLP模型。

jieba：这是一个用于中文分词的开源库。它可以将中文文本切分成词语，方便后续的文本处理和分析。

datasets：这是一个用于访问和处理各种数据集的库。它提供了一个统一的接口，可以方便地下载、加载和预处理各种常见的NLP数据集，例如文本分类、命名实体识别等。

1. 禁用W&B
2. python -c "import os; os.environ['WANDB\_DISABLED']=True"
3. 或者export WANDB\_DISABLED=true
4. 准备数据集jsonline格式
5. 修改train.sh和evaluate.sh中的train\_file、validation\_file和test\_file为你自己的 JSON格式数据集路径，并将prompt\_column和response\_column改为JSON文件中输入文本和输出文本对应的KEY。可能还需要增大max\_source\_length和 max\_target\_length来匹配你自己的数据集中的最大输入输出长度。并将模型路径 THUDM/chatglm-6b改为你本地的模型路径。
6. CHECKPOINT 实际就是 train.sh 中的 output\_dir
7. 参数含义：

ChatGLM-6B 训练脚本 train.sh 的各个参数含义如下：

–do\_train：表示进行训练。

–train\_file：表示训练数据集的路径，需要是 JSON 格式的文件。

–validation\_file：表示验证数据集的路径，需要是 JSON 格式的文件。

–prompt\_column：表示 JSON 文件中输入文本对应的 KEY。

–response\_column：表示 JSON 文件中输出文本对应的 KEY。

–overwrite\_cache：表示是否覆盖缓存文件。

–model\_name\_or\_path：表示预训练模型的名称或路径。

–output\_dir：表示输出目录，用于保存模型和日志等文件。

–overwrite\_output\_dir：表示是否覆盖输出目录。

–max\_source\_length：表示输入文本的最大长度，单位是 token。

–max\_target\_length：表示输出文本的最大长度，单位是 token。输入文本和输出文本的最大长度是影响模型处理能力的重要参数，一般建议根据任务和数据集的特点选择合适的值，但不要超过 ChatGLM-6B 的序列长度限制（2048）。

–per\_device\_train\_batch\_size：这个参数是指每个设备上的训练批次大小，即每个 GPU / TPU 内核/ CPU 上用于训练的数据量。这个参数会影响显存的占用和训练的速度，一般来说，批次大小越大，显存占用越高，训练速度越快，但是也可能导致梯度爆炸或者欠拟合。因此，需要根据具体的任务和数据集来调整这个参数，以达到最佳的效果。

–per\_device\_eval\_batch\_size：表示每个设备的评估批处理大小。

–gradient\_accumulation\_steps：这个参数是指梯度累积的次数，即在进行一次梯度更新之前，需要累积多少个小批量的梯度。这个参数可以用来解决显存不足的问题，通过减小每个小批量的大小，但是增加梯度累积的次数，来保持总的有效批量大小不变。gradient\_accumulation\_steps 和 per\_device\_train\_batch\_size 的关系是：它们都可以影响训练过程中的有效批次大小。有效批次大小是指每次梯度更新时使用的数据量，它等于 per\_device\_train\_batch\_size 乘以 gradient\_accumulation\_steps 乘以 设备数。有效批次大小会影响模型的训练速度和效果，一般来说，批次大小越大，训练速度越快，但是也可能导致梯度爆炸或者欠拟合。

怎么结合这两个参数使用呢？一种常见的做法是，当显存不足以支持较大的 per\_device\_train\_batch\_size 时，可以通过增加 gradient\_accumulation\_steps 来实现梯度累积，相当于用更多的小批次来模拟一个大批次。这样可以在不牺牲有效批次大小的情况下，节省显存和提高训练效率。但是也要注意不要设置过大的 gradient\_accumulation\_steps ，否则可能会导致梯度更新不及时或者内存溢出。

–predict\_with\_generate：表示是否使用生成模式进行预测。

–max\_steps：表示最大训练步数，即模型在训练集上的迭代次数，可以理解为训练多少轮。

–logging\_steps：表示记录日志的步数间隔。

–save\_steps：表示保存模型的步数间隔，即训练多少轮保存一次训练结果（checkpoint）。

–learning\_rate：表示训练的学习率，可以进行调节以取得最佳效果。用于控制模型训练的速度和效果。它的取值范围可以根据具体的任务和数据集进行调节，以取得最佳的效果。一般来说，learning\_rate 越大，模型训练的速度越快，但也可能导致过拟合或不收敛。learning\_rate 越小，模型训练的速度越慢，但也可能获得更好的泛化能力或收敛性。

值范围在 0 到 1 之间，但也可以超过 1 或小于 0。不过，如果 learning\_rate 太大或太小，都可能导致模型训练失败或效果不佳。因此，建议您从一个较小的值开始，比如 0.01 (1e-2)或 0.001 (1e-3)，然后逐渐增大或减小，观察模型的表现，找到一个合适的值。

–pre\_seq\_len：pre\_seq\_len的取值范围一般是1到512，它表示自然语言指令的长度，即输入序列中的前pre\_seq\_len个token，具体的值需要根据自然语言指令的长度和复杂度来确定。一般来说，指令越长越复杂，pre\_seq\_len就需要越大，以便模型能够充分理解指令的含义。但是，pre\_seq\_len也不能太大，否则会占用过多的输入序列长度，导致生成内容过于单一或重复。因此，pre\_seq\_len的取值需要在保证指令完整性和生成多样性之间进行权衡。一种可能的方法是，根据不同的指令类型设置不同的pre\_seq\_len值，例如，对于简单的指令，如“生成一个笑话”，可以设置pre\_seq\_len为4；对于复杂的指令，如“生成一个关于环保主题的故事”，可以设置pre\_seq\_len为16。当然，这些值也需要根据实际的效果进行调整和优化。

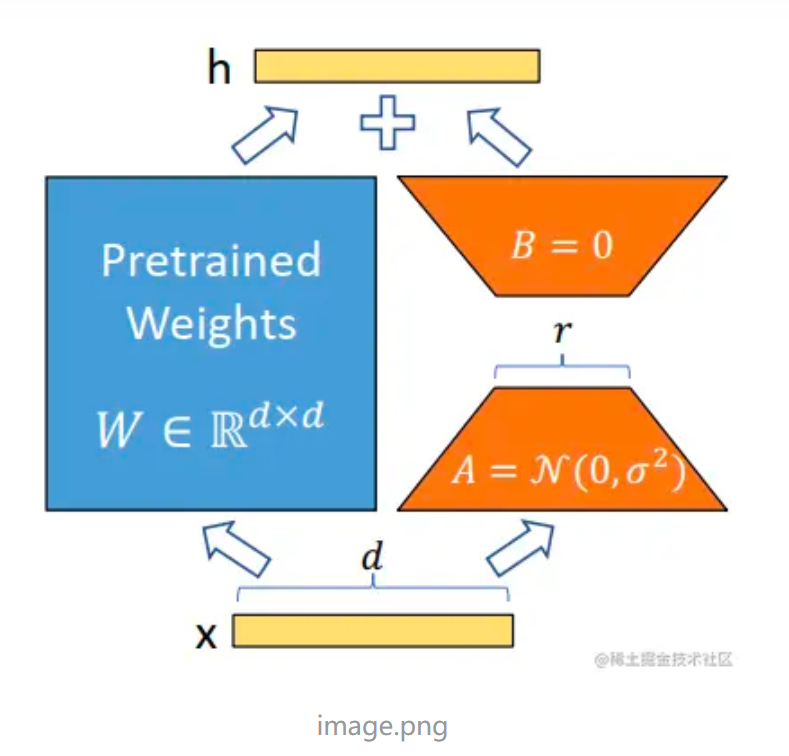
–quantization\_bit：表示模型参数的量化位数，用于降低显存需求。

pre\_seq\_len和max\_source\_length的含义是差不多的，或者取值是可以参照的？

不完全是的，pre\_seq\_len和max\_source\_length的含义是不同的，但是它们之间有一定的关系。pre\_seq\_len是指自然语言指令的长度，而max\_source\_length是指整个输入序列的最大长度。一般来说，pre\_seq\_len应该小于或等于max\_source\_length，因为输入序列除了包含指令之外，还可能包含其他内容，例如上下文信息或对话历史。如果pre\_seq\_len大于max\_source\_length，那么模型就无法接收到完整的指令，也就无法正确地生成输出内容。因此，pre\_seq\_len和max\_source\_length的取值需要协调一致，以保证模型能够有效地利用输入信息。

1. 基于LoRA微调ChatGLM2-6b-int4（微调低秩适应器）

LoRA：LoRA是指在预训练模型的每一层注入一些可训练的低秩矩阵（Low-Rank Adaptation），用于捕捉下游任务的低秩变化，然后只对这些矩阵进行微调，而冻结预训练模型的其他参数。这种方法可以减少微调的参数量和计算量，提高微调的效率和推理速度，同时保持模型的生成质量。



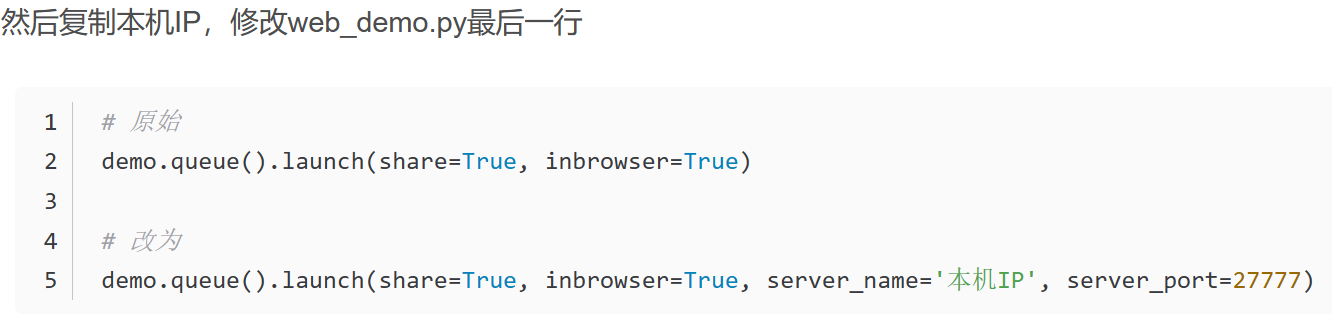
1. 基于Freeze Tuning微调ChatGLM2-6b-int4（微调后几层的全连接层）

即参数冻结，对原始模型部分参数进行冻结操作，仅训练部分参数，以达到在单卡或不进行TP或PP操作，就可以对大模型进行训练。

1. 全量微调ChatGLM2-6b-int4（微调模型所有参数）

Finetune：Finetune是指对预训练模型的所有参数进行微调，以适应下游任务。这种方法可以充分利用预训练模型的知识，但也需要较多的计算资源和数据量，可能会导致过拟合或灾难性遗忘。

python3 web\_demo.py启动web\_ui修改成公网ip+端口



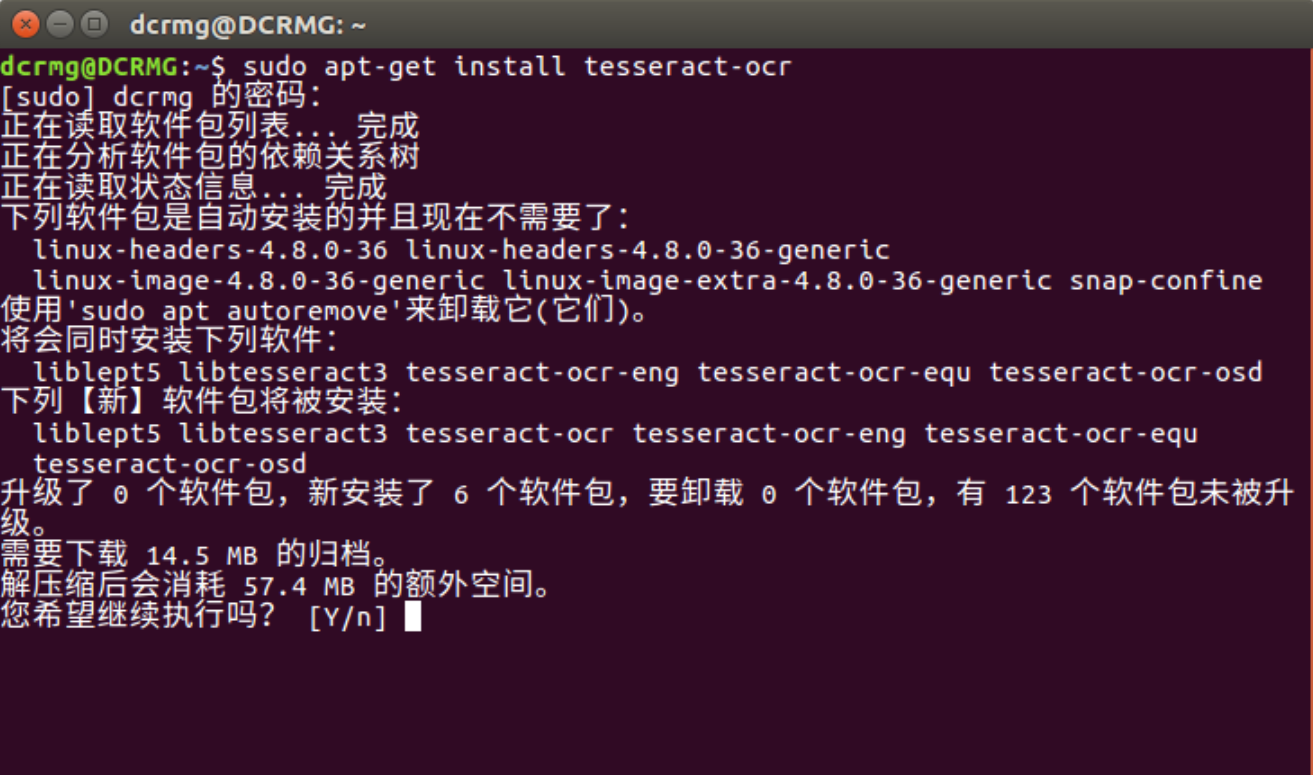
**OCR工具—pytesseract**

1. install Tesseract

Ubuntu

sudo apt install tesseract-ocr

sudo apt install libtesseract-dev



1. install PIL, (python imaging library)是python中的图像处理库

sudo apt-get install python-imaging

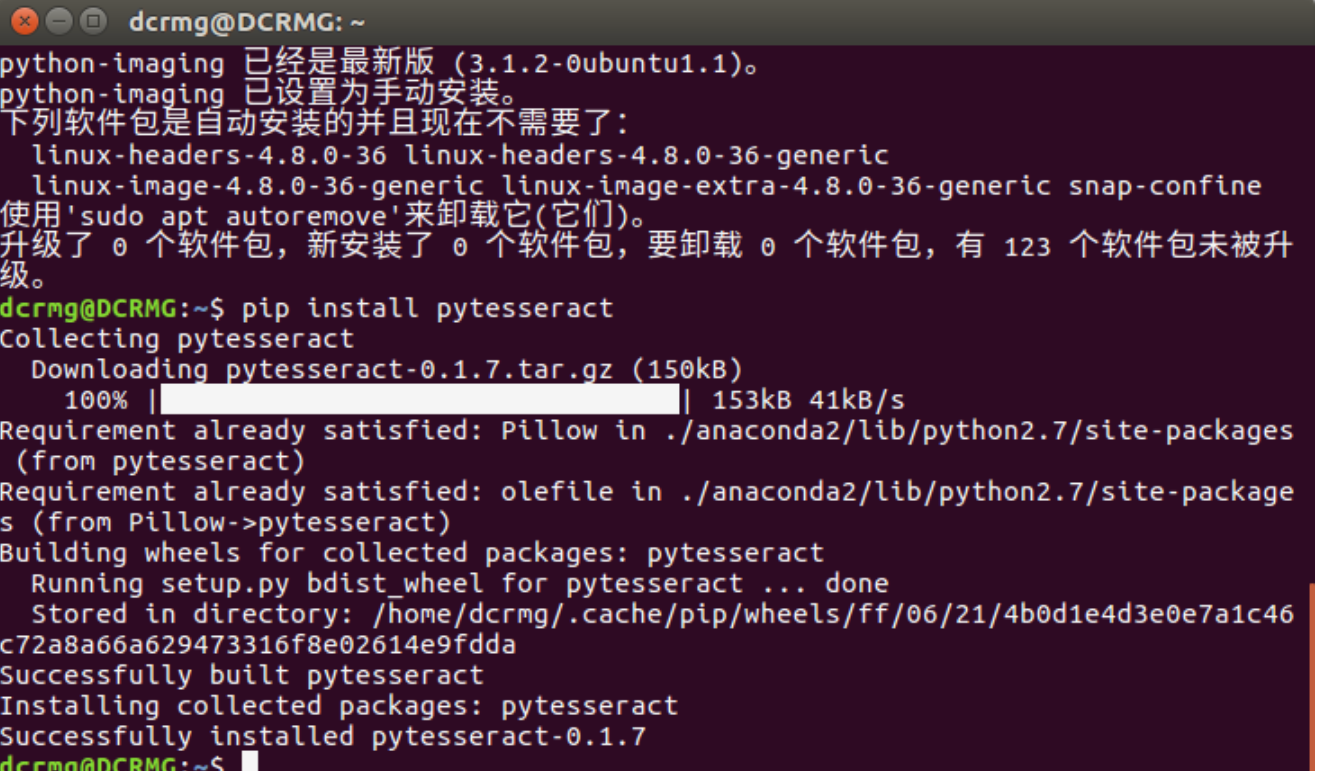


1. 下载文语言文件名为：chi\_sim.traineddata

下载的chi\_sim.traineddata文件，放在Tesseract-OCR\tessdata目录下，例如D:\Program Files\Tesseract-OCR\tessdata（/usr/share/tesseract-ocr/4.00/tessdata）

1. install pytesseract

pip install pytesseract



1. example

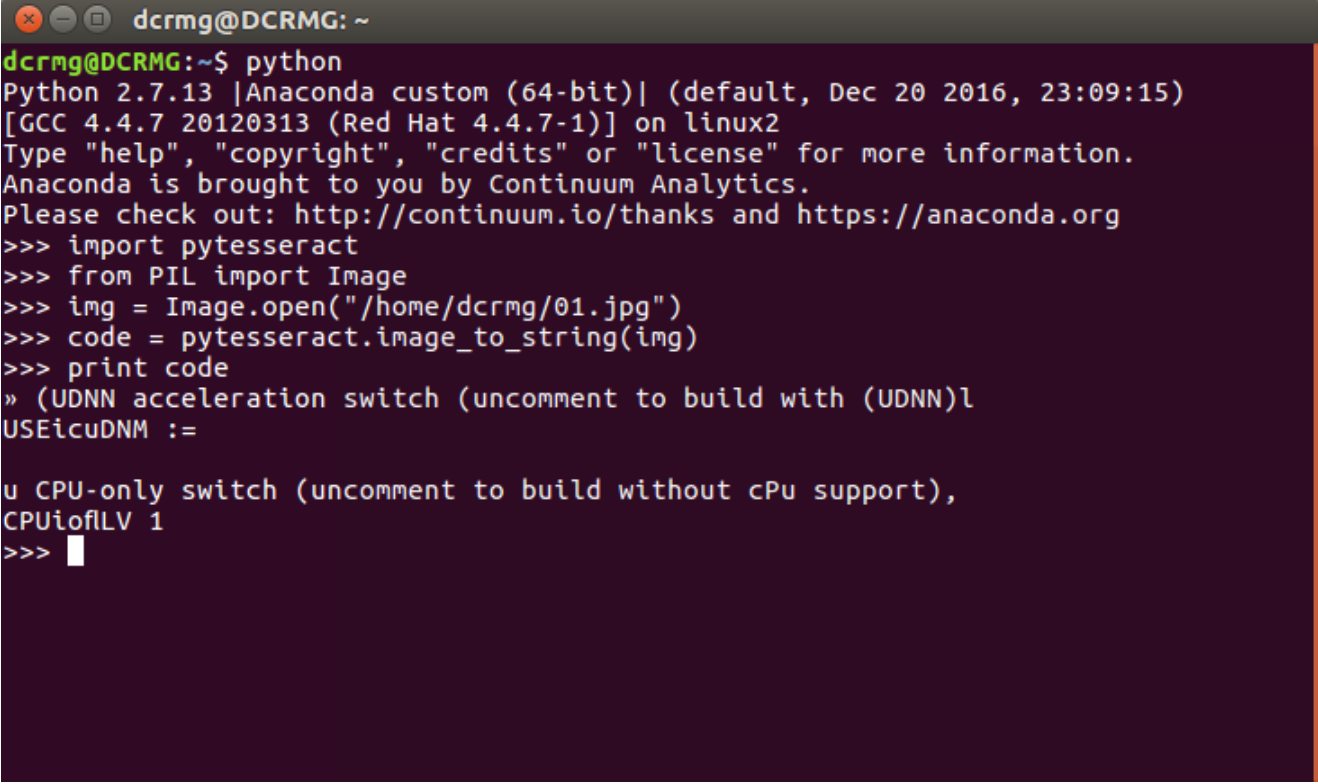
import pytesseract

from PIL import Image

img = Image.open("./01.jpg")

code = pytesseract.image\_to\_string(img)

print code



**Squid代理服务器配置**

**（A:台式机以太网192.68.108.5+WiFi，B局域网服务器192.68.108.3，wsl2—ip192.168.144.126）**

1. install squid

sudo apt update

sudo apt install squid

sudo systemctl status squid

1. win11 power shell端口转发

查看转发规则

netsh interface portproxy show all

netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8080 connectaddress=192.168.144.126 connectport=80（将本地端口8080转发到wsl2）

侦听 ipv4: 连接到 ipv4:

地址 端口 地址 端口

--------------- ---------- --------------- ----------

0.0.0.0 2222 IP 2222

0.0.0.0 23 192.168.144.126 23

0.0.0.0 8080 192.168.144.126 80

1. SSH端口转发

power shell中查询server和client是否已经安装

Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH\*'

安装ssh-server

# 安装 OpenSSH 客户端

Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0

# 安装 OpenSSH 服务端

Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0

检测ssh-server是否启动Get-Service sshd

检测ssh-client是否启动Get-Service ssh-agent

在power shell中创建SSH隧道直连wsl2

ssh -L \*:服务器端口:WSL2的IP地址:WSL2的端口 WSL2的用户名@WSL2的IP地址

Wsl2中安装

**ssh -L \*:8080:192.168.144.126:81 egnail@192.168.144.126 -p 23**

sudo apt update

sudo apt install openssh-server

启动ssh-server服务

sudo service ssh start

**核心逻辑：需要先在power shell中建立从本机到wsl2的隧道；然后在wsl2中配置squid建立从81端口到代理服务器的连接。**

1. 修改/etc/squid/squid.conf文件

备份sudo cp /etc/squid/squid.conf{,.orginal}

# Squid normally listens to port 3128

http\_port IP\_ADDR:PORT （http\_port 192.168.144.126:80）

<https://www.myfreax.com/how-to-install-and-configure-squid-proxy-on-ubuntu-20-04/>

测试squid服务 sudo squid -N

查看squid日志 vim /var/log/squid/cache.log

1. 修改~/.bashrc文件

添加

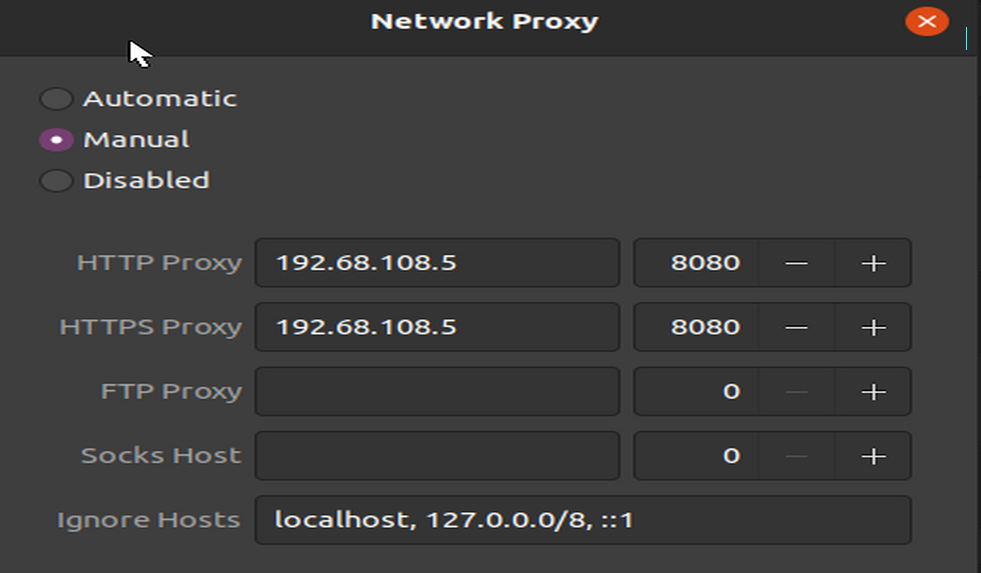
export http\_proxy=http://192.68.108.5:8080

export https\_proxy=http://192.68.108.5:8080

这将配置wget和curl命令通过你的SSH隧道访问外网。

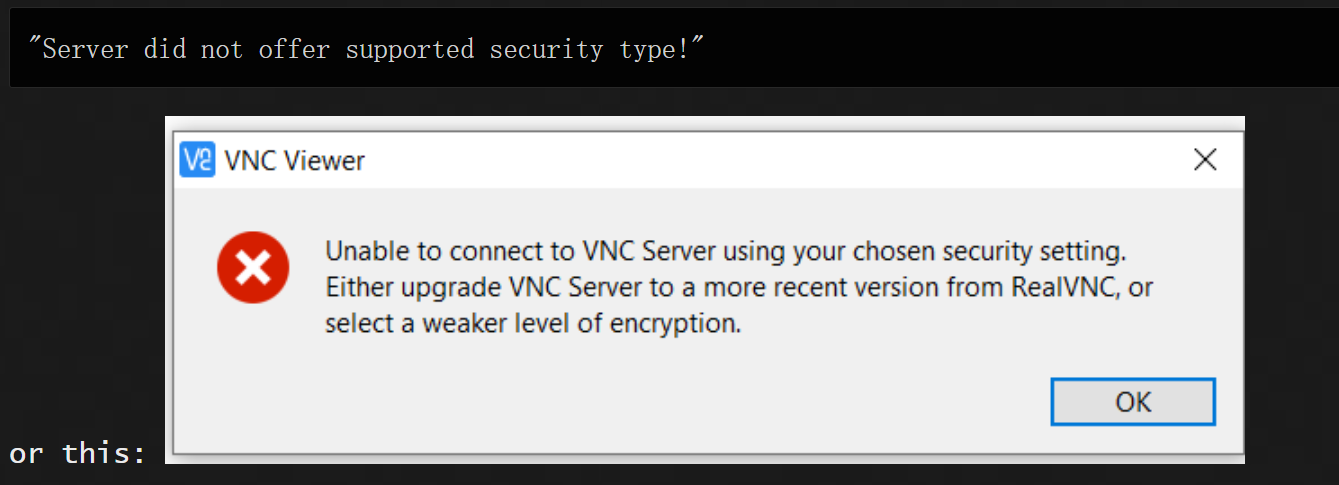
1. Desktop proxy配置

Network proxy



**VNC设置**

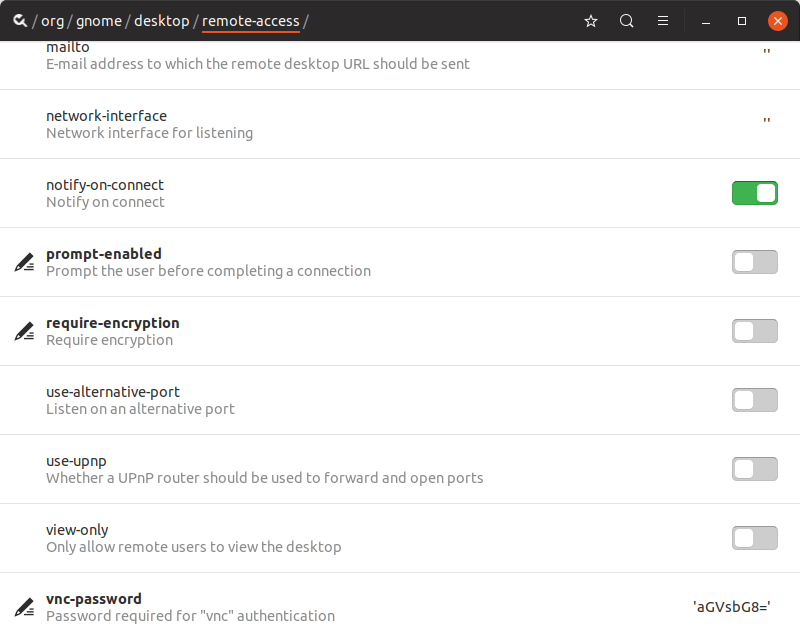
Mobaxterm vnc 连接发现问题



安装dconf editor

sudo apt-get install dconf-editor

Set require-encryption at off in the （/org/gnome/desktop/remote-access）



**监控硬件温度**

sudo apt-get install lm-sensors hddtemp

sudo sensors-detect

sensors

**SSH 通过跳板机直接访问内网机器**

ssh username@目标机器IP -p 22 -J username1@跳板机IP1:22,username2@跳板机IP2:22