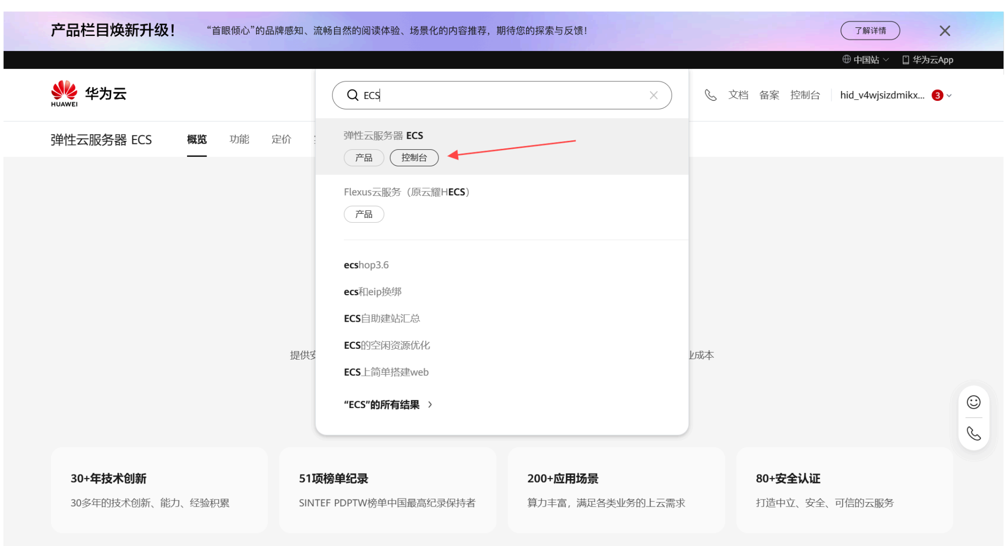
Assignment4: Deepspeech2

1. Development Environment Construction

1.1. Server Purchase

On the Huawei Cloud ECS homepage, search "ECS" and click "Management Console" to enter the ECS management page.



Create an elastic cloud server Select "North China-Beijing Four" in the console area, select "Elastic Cloud Server" in the left menu bar, and "Buy Elastic Cloud Server" in the upper right corner.

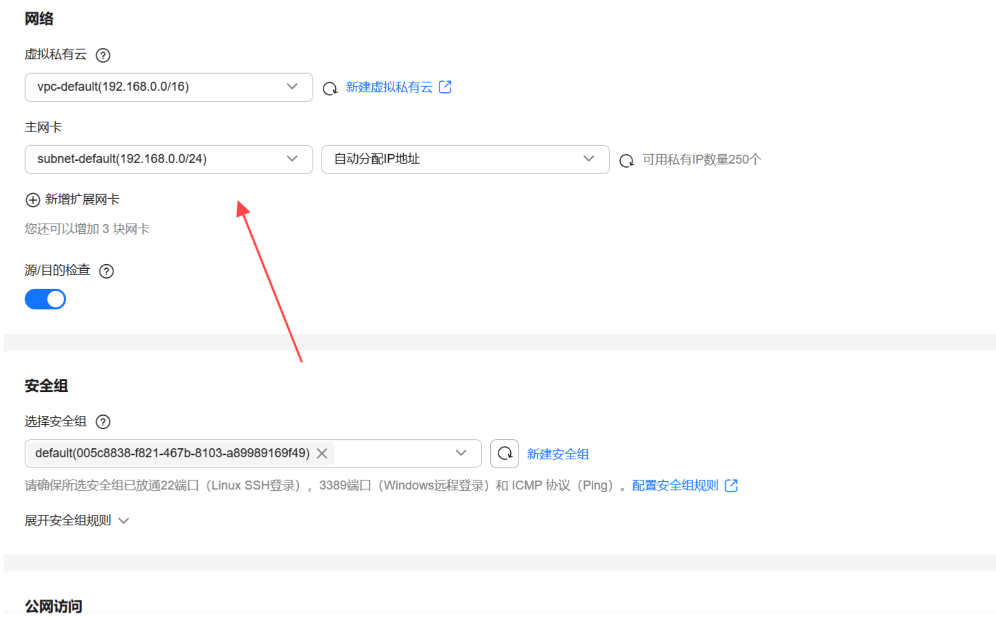
In the Basic Configuration, select the following configuration:

- Billing mode: billing on demand.
- Region: North China-Beijing 4th.
- Available area: Random allocation.
- CPU architecture: x86 calculation.
- Specification: General calculation enhanced type | c7.2xlarge.2 | 8vCPUs | 16 GiB
- Mirror: Public mirror, Ubuntu, Ubuntu 18.04 server 64bit
- System disk: Universal SSD, 100GB.

Click "Next: Network Configuration" in the next right corner of the window In the Network Configuration, select the following configuration:

- Network: You can go to the console to create a new virtual private cloud.
- Expand the network card: no.
- Security group: You can create a new security group.
- Flexible public network IP:
- Buy it now. Line: full dynamic BGP.
- Public network bandwidth: charge by traffic.
- Broadband size: custom, 200 Mbit/s.

- Release behavior: Check the Release with the instance.



Click "Next: the advanced configuration" in the lower right corner of the window In the Advanced Configuration section, select the following configuration:

Cloud server name: It can be customized.

Login certificate: password.

User name: root.

Password: custom (subsequent login use, remember). C

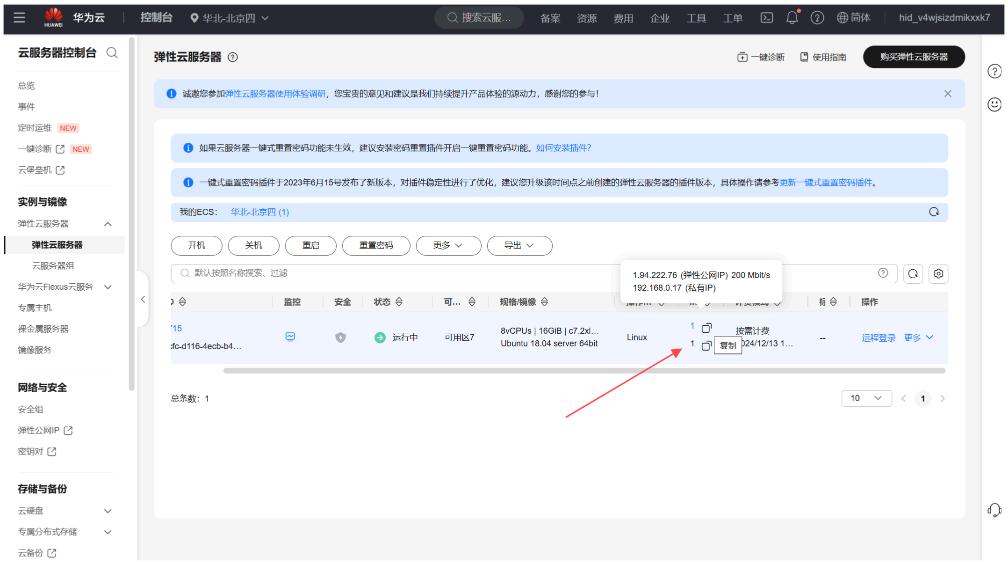
loud backup: it is not purchased temporarily.

Cloud Server Group: None. Advanced option: None.

Click "Next: Confirm configuration" in the lower right corner of the window, In Confirm Configuration, select the following configuration: Agreement: Check that I have read and agreed to the Mirror Disclaimer.

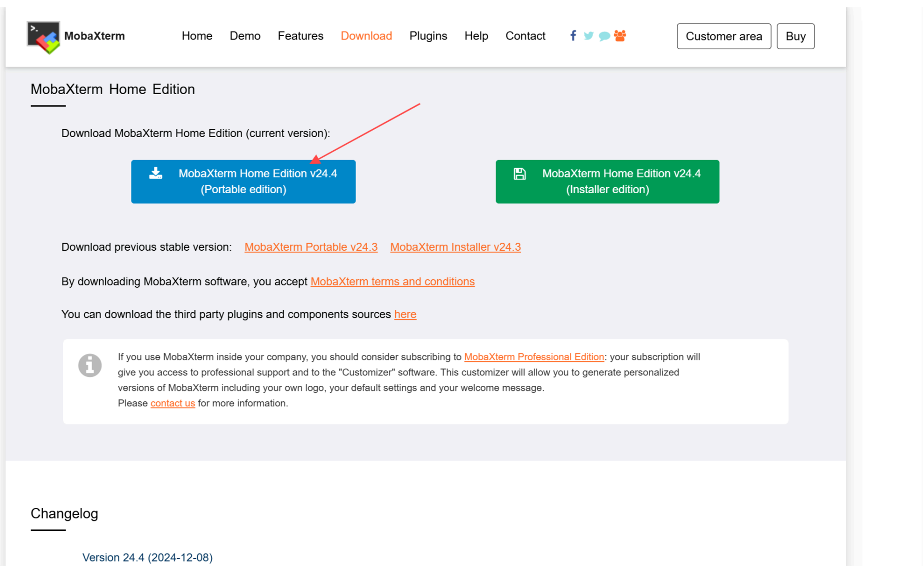
Click "Buy Now" in the lower right corner of the window. After the Task Submission succeeded, select Return to Server List to return to the management console of the elastic cloud server and see that the ECS created is running.

Note the flexible public network IP address displayed in the IP Address, which will be used later.

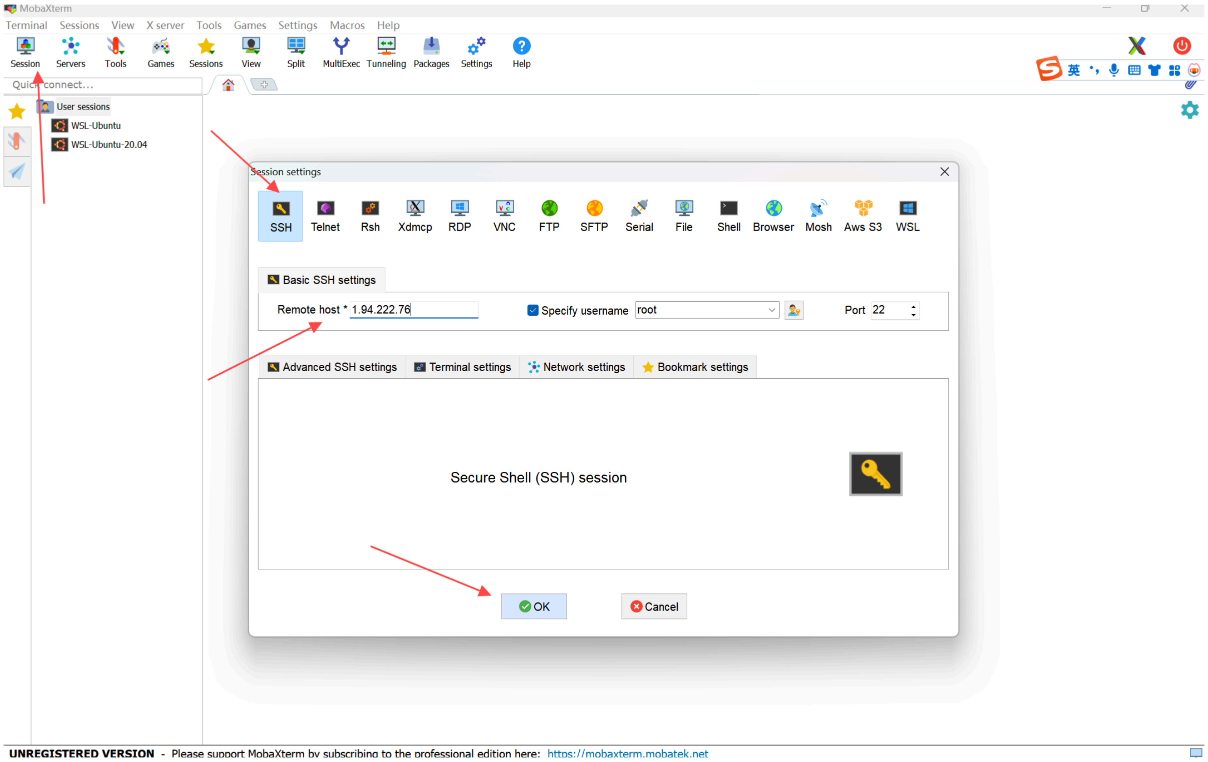


1.2. MobaXterm Connect ECS

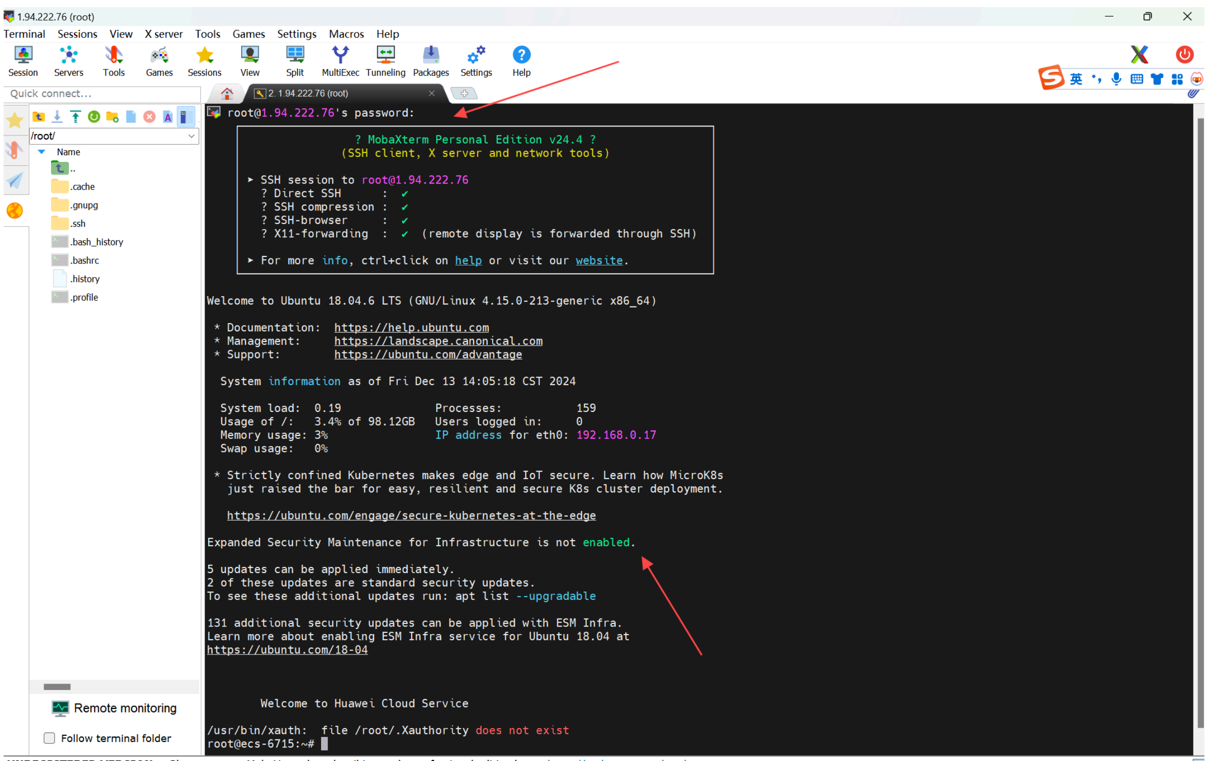
Download the MobaXterm Go to MobaXterm's official website home page: https://mobaxterm.mobatek.net/ Select the Home Edition and download the MobaXterm Home Edition v21.x (Portable edition). Unpack the MobaXterm_Portable_v21.x.zip file after the download is complete.



Use MobaXterm to connect to an elastic cloud server Enter the decompression MobaXterm_Portable_v21.x folder, Open the MobaXterm_Personal_21.x.exe file, Select the "Session" of the menu bar, Then enter the "Session settings" page, Remote link selection "SSH" protocol, Enter Figure 2-12 The elastic public network IP address displayed when the ECS elastic cloud server is created, Select the specified user name "Specify username", User name is "root", Select OK for submission after the configuration is complete.



MobaXterm Login to ECS requires a password. In step 4 of ECS elastic cloud server, the elastic cloud server root user password has been customized in the advanced configuration. You can enter here. MobaXterm The remote link to the elastic cloud server is successful, and the cloud environment of the elastic cloud server should be further configured later.



2. Code and Data Download

2.1. Get Code Files

Use git to download the source code of the training script from mindspore, switch to the home directory, create a working directory such as / work, and execute the following command:

```
git clone https://gitee.com/mindspore/models.git
```

The deepspeech2 project code for this experiment is located at models / research / audio / deepspeech2.

Training and inference-related parameters in the config.py file.

```
root@ecs-6715:~# cd "/home/"
root@ecs-6715:/home# mkdir work
root@ecs-6715:/home# cd work
root@ecs-6715:/home/work# ^C
root@ecs-6715:/home/work# git clone https://gitee.com/mindspore/models.git
Cloning into 'models'...
remote: Enumerating objects: 87074, done.
remote: Total 87074 (delta 0), reused 0 (delta 0), pack-reused 87074
Receiving objects: 100% (87074/87074), 502.99 MiB | 22.09 MiB/s, done.
Resolving deltas: 100% (62429/62429), done.
Checking out files: 100% (18471/18471), done.
root@ecs-6715:/home/work#
```

2.2. Dataset and its Preprocessing

2.2.1. Download the LibriSpeech Dataset

The link to download the data set is: http://www.openslr.org/12.

- **training set**

  Train-clean-100: [6.3G] (100 Hour No Noise Speech Training Set) (just download this file)

- **validation set**

  dev-clean.tar.gz [337M] (No Noise)

  dev-other.tar.gz [314M] (with noise)

- **test set**

  test-clean.tar.gz [346M] (Test set, No Noise)

  test-other.tar.gz [328M] (Test set, noisy)

LibriSpeech Data directory structure, is as follows:

```
|——LibriSpeech
|—— train
|    |—train-clean-100
|—— val
|    |—dev-clean.tar.gz
|    |—dev-other.tar.gz
|—— test_other
|    |—test-other.tar.gz
|——test_clean
|    |—test-clean.tar.gz
```

2.2.2. Install the Python3.9.0

Installing python dependency and software such as gcc.

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev libgmp-dev sox libjpeg8-dev
```

```
root@ecs-6715:/home/Python-3.9.0# sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip
pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev libgmp-dev sox libjpeg8-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
make is already the newest version (4.1-9.1ubuntu1).
make set to manually installed.
net-tools is already the newest version (1.60+git20161116.90da8a0-1ubuntu1).
g++ is already the newest version (4:7.4.0-1ubuntu2.3).
g++ set to manually installed.
gcc is already the newest version (4:7.4.0-1ubuntu2.3).
gcc set to manually installed.
openssl is already the newest version (1.1.1-1ubuntu2.1~18.04.23).
pciutils is already the newest version (1:3.5.2-1ubuntu1.1).
unzip is already the newest version (6.0-21ubuntu1.2).
```

Use wget to download the python3.9.0 source package, which can be downloaded to any directory of the installation environment with the command:

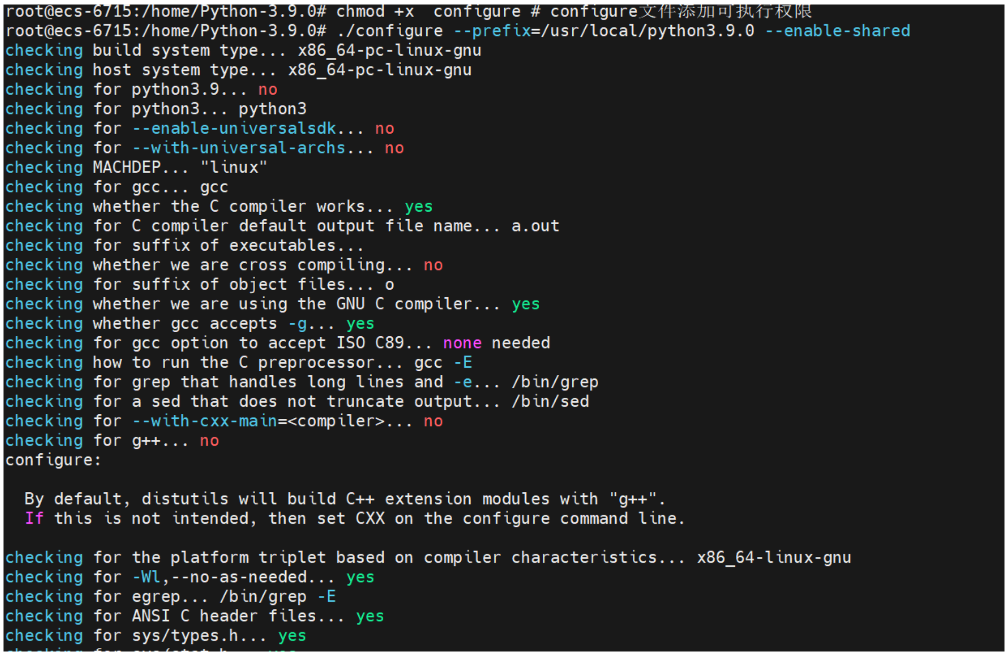
```
wget https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
tar -zxvf Python-3.9.0.tgz
```

```
root@ecs-6715:/home# wget https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
--2024-12-13 14:37:26--  https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
Resolving www.python.org (www.python.org)... 151.101.88.223, 2a04:4e42:15::223
Connecting to www.python.org (www.python.org)|151.101.88.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26724009 (25M) [application/octet-stream]
Saving to: ‘Python-3.9.0.tgz’

Python-3.9.0.tgz                  100%[==================================================================>]  25.49M   142KB/s    in 3m 37s

2024-12-13 14:41:03 (120 KB/s) - ‘Python-3.9.0.tgz’ saved [26724009/26724009]
```

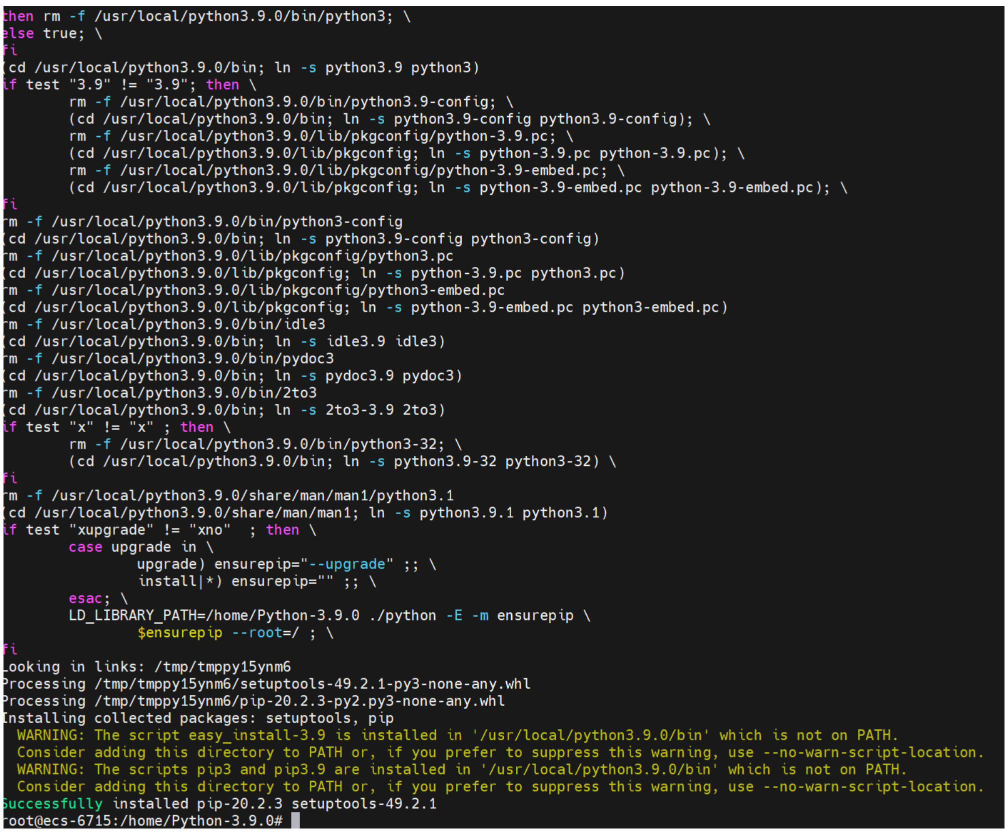
```
root@ecs-6715:/home# tar -zxvf Python-3.9.0.tgz
Python-3.9.0/
Python-3.9.0/CODE_OF_CONDUCT.md
Python-3.9.0/README.rst
Python-3.9.0/Doc/
Python-3.9.0/Doc/howto/
Python-3.9.0/Doc/howto/pyporting.rst
Python-3.9.0/Doc/howto/logging-cookbook.rst
Python-3.9.0/Doc/howto/logging_flow.png
Python-3.9.0/Doc/howto/sorting.rst
Python-3.9.0/Doc/howto/functional.rst
Python-3.9.0/Doc/howto/regex.rst
Python-3.9.0/Doc/howto/ipaddress.rst
Python-3.9.0/Doc/howto/argparse.rst
Python-3.9.0/Doc/howto/urllib2.rst
Python-3.9.0/Doc/howto/unicode.rst
Python-3.9.0/Doc/howto/index.rst
Python-3.9.0/Doc/howto/logging.rst
Python-3.9.0/Doc/howto/curses.rst
Python-3.9.0/Doc/howto/descriptor.rst
Python-3.9.0/Doc/howto/sockets.rst
Python-3.9.0/Doc/howto/instrumentation.rst
Python-3.9.0/Doc/howto/cporting.rst
Python-3.9.0/Doc/howto/clinic.rst
Python-3.9.0/Doc/runtime.txt
Python-3.9.0/Doc/README.rst
Python-3.9.0/Doc/install/
```

Go to the decompression folder and execute the configuration, compile, and installation commands:

```
cd Python-3.9.0
chmod +x  configure
./configure --prefix=/usr/local/python3.9.0 --enable-shared
make
sudo make install
```



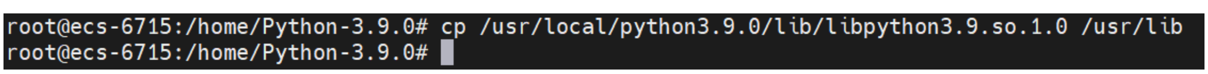
```
root@ecs-6715:/home/Python-3.9.0# chmod +x  configure # configure文件添加可执行权限
root@ecs-6715:/home/Python-3.9.0# ./configure --prefix=/usr/local/python3.9.0 --enable-shared
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for python3.9... no
checking for python3... python3
checking for --enable-universalsdk... no
checking for --with-universal-archs... no
checking MACHDEP... "linux"
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for a sed that does not truncate output... /bin/sed
checking for --with-cxx-main=<compiler>... no
checking for g++... no
configure:

  By default, distutils will build C++ extension modules with "g++".
  If this is not intended, then set CXX on the configure command line.

checking for the platform triplet based on compiler characteristics... x86_64-linux-gnu
checking for -Wl,--no-as-needed... yes
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
```



```
then rm -f /usr/local/python3.9.0/bin/python3; \
else true; \
fi
(cd /usr/local/python3.9.0/bin; ln -s python3.9 python3)
if test "3.9" != "3.9"; then \
        rm -f /usr/local/python3.9.0/bin/python3.9-config; \
        (cd /usr/local/python3.9.0/bin; ln -s python3.9-config python3.9-config); \
        rm -f /usr/local/python3.9.0/lib/pkgconfig/python-3.9.pc; \
        (cd /usr/local/python3.9.0/lib/pkgconfig; ln -s python-3.9.pc python-3.9.pc); \
        rm -f /usr/local/python3.9.0/lib/pkgconfig/python-3.9-embed.pc; \
        (cd /usr/local/python3.9.0/lib/pkgconfig; ln -s python-3.9-embed.pc python-3.9-embed.pc); \
fi
rm -f /usr/local/python3.9.0/bin/python3-config
(cd /usr/local/python3.9.0/bin; ln -s python3.9-config python3-config)
rm -f /usr/local/python3.9.0/lib/pkgconfig/python3.pc
(cd /usr/local/python3.9.0/lib/pkgconfig; ln -s python-3.9.pc python3.pc)
rm -f /usr/local/python3.9.0/lib/pkgconfig/python3-embed.pc
(cd /usr/local/python3.9.0/lib/pkgconfig; ln -s python-3.9-embed.pc python3-embed.pc)
rm -f /usr/local/python3.9.0/bin/idle3
(cd /usr/local/python3.9.0/bin; ln -s idle3.9 idle3)
rm -f /usr/local/python3.9.0/bin/pydoc3
(cd /usr/local/python3.9.0/bin; ln -s pydoc3.9 pydoc3)
rm -f /usr/local/python3.9.0/bin/2to3
(cd /usr/local/python3.9.0/bin; ln -s 2to3-3.9 2to3)
if test "x" != "x" ; then \
        rm -f /usr/local/python3.9.0/bin/python3-32; \
        (cd /usr/local/python3.9.0/bin; ln -s python3.9-32 python3-32) \
fi
rm -f /usr/local/python3.9.0/share/man/man1/python3.1
(cd /usr/local/python3.9.0/share/man/man1; ln -s python3.9.1 python3.1)
if test "xupgrade" != "xno"  ; then \
        case upgrade in \
                upgrade) ensurepip="--upgrade" ;; \
                install|*) ensurepip="" ;; \
        esac; \
        LD_LIBRARY_PATH=/home/Python-3.9.0 ./python -E -m ensurepip \
                $ensurepip --root=/ ; \
fi
Looking in links: /tmp/tmppy15ynm6
Processing /tmp/tmppy15ynm6/setuptools-49.2.1-py3-none-any.whl
Processing /tmp/tmppy15ynm6/pip-20.2.3-py2.py3-none-any.whl
Installing collected packages: setuptools, pip
  WARNING: The script easy_install-3.9 is installed in '/usr/local/python3.9.0/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The scripts pip3 and pip3.9 are installed in '/usr/local/python3.9.0/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pip-20.2.3 setuptools-49.2.1
root@ecs-6715:/home/Python-3.9.0#
```

Query whether there is libpython3.9.so.1.0 under / usr / lib 64 or / usr / lib, skip this step or back up the libpython3.9.so.1.0 file with the following command.

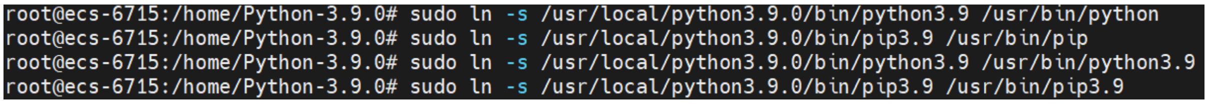
```
cp /usr/local/python3.9.0/lib/libpython3.9.so.1.0 /usr/lib
```



```
root@ecs-6715:/home/Python-3.9.0# cp /usr/local/python3.9.0/lib/libpython3.9.so.1.0 /usr/lib
root@ecs-6715:/home/Python-3.9.0#
```

Perute the following command setting soft link:

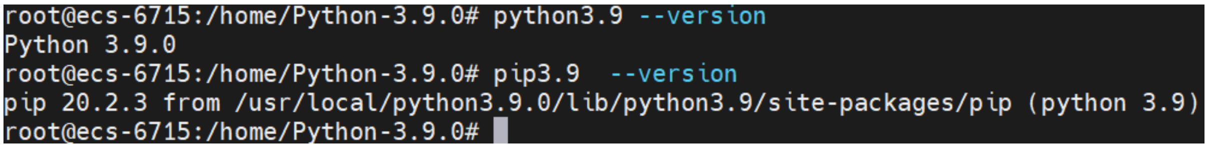
```
sudo ln -s /usr/local/python3.9.0/bin/python3.9 /usr/bin/python
sudo ln -s /usr/local/python3.9.0/bin/pip3.9 /usr/bin/pip
sudo ln -s /usr/local/python3.9.0/bin/python3.9 /usr/bin/python3.9
sudo ln -s /usr/local/python3.9.0/bin/pip3.9 /usr/bin/pip3.9
```



```
root@ecs-6715:/home/Python-3.9.0# sudo ln -s /usr/local/python3.9.0/bin/python3.9 /usr/bin/python
root@ecs-6715:/home/Python-3.9.0# sudo ln -s /usr/local/python3.9.0/bin/pip3.9 /usr/bin/pip
root@ecs-6715:/home/Python-3.9.0# sudo ln -s /usr/local/python3.9.0/bin/python3.9 /usr/bin/python3.9
root@ecs-6715:/home/Python-3.9.0# sudo ln -s /usr/local/python3.9.0/bin/pip3.9 /usr/bin/pip3.9
```

After the installation is complete, perform the following command to view the installation version, and if the relevant version information is returned, the installation is successful.

```
python3.9 --version
pip3.9  --version
```

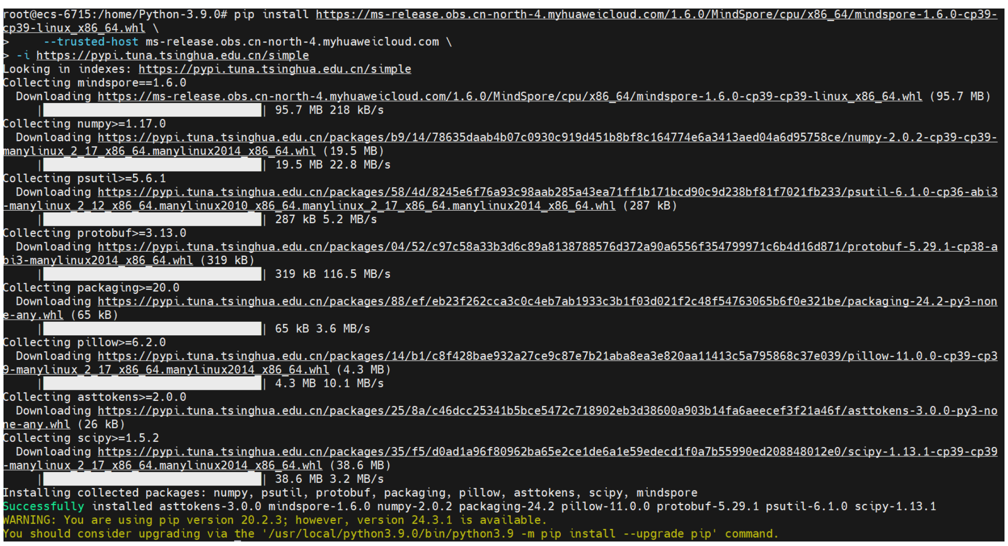


```
root@ecs-6715:/home/Python-3.9.0# python3.9 --version
Python 3.9.0
root@ecs-6715:/home/Python-3.9.0# pip3.9  --version
pip 20.2.3 from /usr/local/python3.9.0/lib/python3.9/site-packages/pip (python 3.9)
root@ecs-6715:/home/Python-3.9.0#
```

2.2.3. Install the MindSpore and the Required Dependency Package

Install mindspore, install according to the actual server architecture, please refer to https://www.mindspore.cn/install.

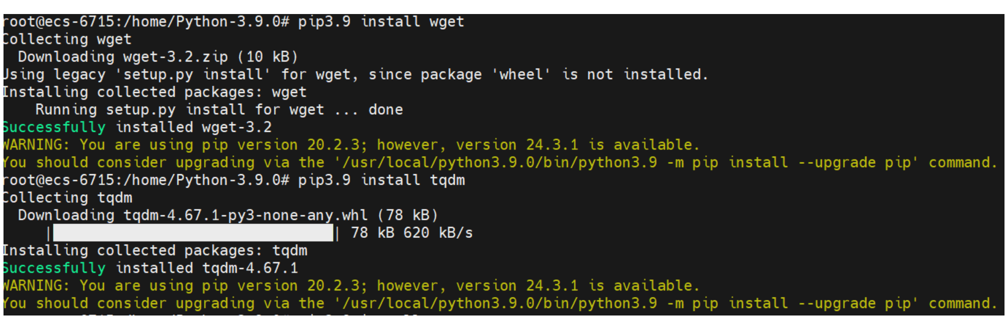
```
pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.6.0/MindSpore/cpu/x86_64/mindspore-1.6.0-cp39-cp39-linux_x86_64.whl \
    --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com \
-i https://pypi.tuna.tsinghua.edu.cn/simple
```



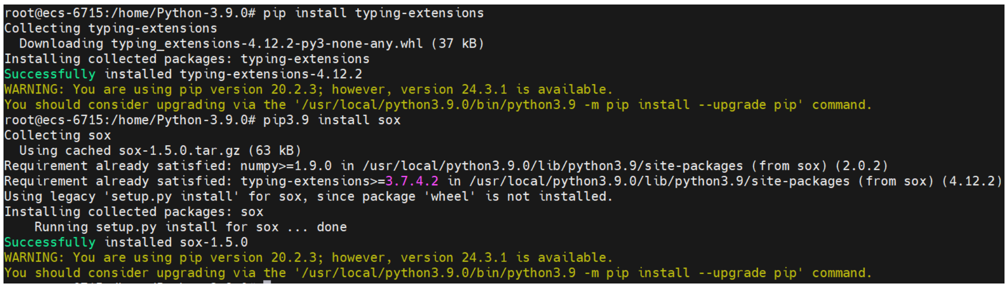
```
root@ecs-6715:/home/Python-3.9.0# pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.6.0/MindSpore/cpu/x86_64/mindspore-1.6.0-cp39-cp39-linux_x86_64.whl \
>     --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com \
> -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting mindspore==1.6.0
  Downloading https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.6.0/MindSpore/cpu/x86_64/mindspore-1.6.0-cp39-cp39-linux_x86_64.whl (95.7 MB)
     |████████████████████████████████| 95.7 MB 218 kB/s
Collecting numpy>=1.17.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/b9/14/78635daab4b07c0930c919d451b8bf8c164774e6a3413aed04a6d95758ce/numpy-2.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.5 MB)
     |████████████████████████████████| 19.5 MB 22.8 MB/s
Collecting psutil>=5.6.1
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/58/4d/8245e6f76a93c98aab285a43ea71ff1b171bcd90c9d238bf81f7021fb233/psutil-6.1.0-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (287 kB)
     |████████████████████████████████| 287 kB 5.2 MB/s
Collecting protobuf>=3.13.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/04/52/c97c58a33b3d6c89a8138788576d372a90a6556f354799971c6b4d16d871/protobuf-5.29.1-cp38-abi3-manylinux2014_x86_64.whl (319 kB)
     |████████████████████████████████| 319 kB 116.5 MB/s
Collecting packaging>=20.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/88/ef/eb23f262cca3c0c4eb7ab1933c3b1f03d021f2c48f54763065b6f0e321be/packaging-24.2-py3-none-any.whl (65 kB)
     |████████████████████████████████| 65 kB 3.6 MB/s
Collecting pillow>=6.2.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/14/b1/c8f428bae932a27ce9c87e7b21aba8ea3e820aa11413c5a795868c37e039/pillow-11.0.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.3 MB)
     |████████████████████████████████| 4.3 MB 10.1 MB/s
Collecting asttokens>=2.0.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/25/8a/c46dcc25341b5bce5472c718902eb3d38600a903b14fa6aeecef3f21a46f/asttokens-3.0.0-py3-none-any.whl (26 kB)
Collecting scipy>=1.5.2
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/35/f5/d0ad1a96f80962ba65e2ce1de6a1e59edecd1f0a7b55990ed208848012e0/scipy-1.13.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
     |████████████████████████████████| 38.6 MB 3.2 MB/s
Installing collected packages: numpy, psutil, protobuf, packaging, pillow, asttokens, scipy, mindspore
Successfully installed asttokens-3.0.0 mindspore-1.6.0 numpy-2.0.2 packaging-24.2 pillow-11.0.0 protobuf-5.29.1 psutil-6.1.0 scipy-1.13.1
WARNING: You are using pip version 20.2.3; however, version 24.3.1 is available.
You should consider upgrading via the '/usr/local/python3.9.0/bin/python3.9 -m pip install --upgrade pip' command.
```

Pip source installation, you can add a mirror source installation when the dependent package file is large, such as pip install-i https://pypi.tuna.tsinghua.edu.cn/simple sox.

```
pip3.9 install wget
pip3.9 install tqdm
pip3.9 install sox
```



```
root@ecs-6715:/home/Python-3.9.0# pip3.9 install wget
Collecting wget
  Downloading wget-3.2.zip (10 kB)
Using legacy 'setup.py install' for wget, since package 'wheel' is not installed.
Installing collected packages: wget
    Running setup.py install for wget ... done
Successfully installed wget-3.2
WARNING: You are using pip version 20.2.3; however, version 24.3.1 is available.
You should consider upgrading via the '/usr/local/python3.9.0/bin/python3.9 -m pip install --upgrade pip' command.
root@ecs-6715:/home/Python-3.9.0# pip3.9 install tqdm
Collecting tqdm
  Downloading tqdm-4.67.1-py3-none-any.whl (78 kB)
     |████████████████████████████████| 78 kB 620 kB/s
Installing collected packages: tqdm
Successfully installed tqdm-4.67.1
WARNING: You are using pip version 20.2.3; however, version 24.3.1 is available.
You should consider upgrading via the '/usr/local/python3.9.0/bin/python3.9 -m pip install --upgrade pip' command.
```

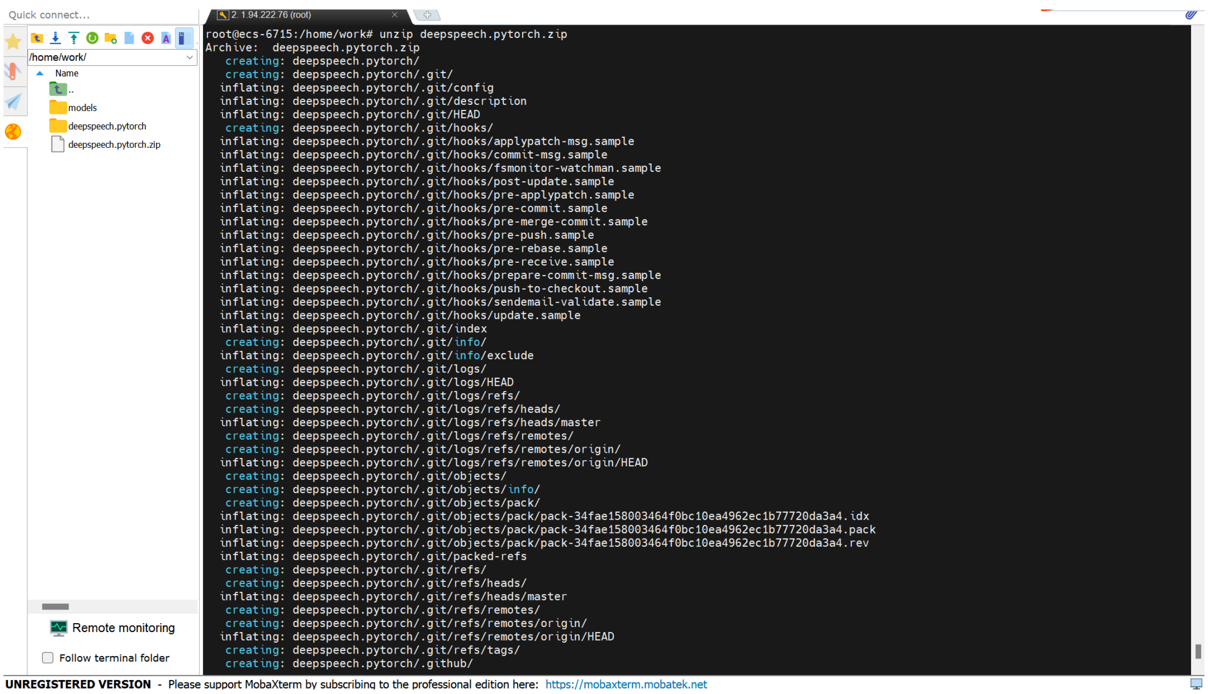


2.2.4. The Data Preprocessing SeanNaren Scripts was Downloaded

After MobaXterm / Finalshell (recommended) connects to the ECS server, switch to the home directory, create the working directory, and then use the scripts in SeanNaren to process the data. SeanNaren Script link: https://github.com/SeanNaren/deepspeech.pytorch.

```
cd ../home
mkdir work
cd work
git clone https://github.com/SeanNaren/deepspeech.pytorch.git
```

Here, you need to access the external network, it is recommended to download directly to the local and then upload the zip file before decompression



2.2.5. LibriSpeech Data Preprocessing

The training set of train-clean-100 was downloaded locally via the dataset link http://www.openslr.org/12, validation sets dev-clean.tar.gz and dev-other.tar.gz, and test sets test-clean.tar.gz and test-other.tar.gz. Upload the local data set to the MobaXterm server.

The structure of the data set is shown below:

```
root@ecs-6715:/home/work/deepspeech.pytorch# tree LibriSpeech_dataset
LibriSpeech_dataset
├── test_clean
│   └── test-clean.tar.gz
├── test_other
│   └── test-other.tar.gz
├── train
│   └── train-clean-100.tar.gz
└── val
    ├── dev-clean.tar.gz
    └── dev-other.tar.gz

4 directories, 5 files
```

Copy the librispeech.py in the data directory of the deepspeech.pytorch to the deepspeech.pytorch directory and execute the following command:

```
cd deepspeech.pytorch
cp ./data/librispeech.py ./
```

```
root@ecs-6715:/home/work# cd deepspeech.pytorch
root@ecs-6715:/home/work/deepspeech.pytorch# cp ./data/librispeech.py ./
```

Modify the librispeech.py code data set path, refer to step 3, set the directory structure in the current directory, and change the code path to the data set actual path, as shown in the figure below:

```
LIBRI_SPEECH_URLS = {
    "train": ["LibriSpeech_dataset/train/train-clean-100.tar.gz"],

    "val": ["LibriSpeech_dataset/val/dev-clean.tar.gz",
            "LibriSpeech_dataset/val/dev-other.tar.gz"],

    "test_clean": ["LibriSpeech_dataset/test-clean/test-clean.tar.gz"],
    "test_other": ["LibriSpeech_dataset/test-other/test-other.tar.gz"]
}
```

Execute the data set processing command, and execute the command as follows.

```
python librispeech.py
```

```
root@ecs-6715:/home/work/deepspeech.pytorch# python librispeech.py
Unpacking train-clean-100.tar.gz...
Converting flac files to wav and extracting transcripts...
838it [03:53,  3.59it/s]
Finished LibriSpeech_dataset/train/train-clean-100.tar.gz
Gathering durations...
100%|██████████| 28539/28539 [00:23<00:00, 1197.48it/s]
Sorting manifests...
Pruning manifests between 1 and 15 seconds
Total duration of split: 233513.3018s
100%|██████████| 20336/20336 [00:00<00:00, 54626.48it/s]
Unpacking dev-clean.tar.gz...
Converting flac files to wav and extracting transcripts...
139it [00:16,  8.35it/s]
Finished LibriSpeech_dataset/val/dev-clean.tar.gz
Unpacking dev-other.tar.gz...
Converting flac files to wav and extracting transcripts...
126it [00:16,  7.47it/s]
Finished LibriSpeech_dataset/val/dev-other.tar.gz
Gathering durations...
100%|██████████| 5567/5567 [00:03<00:00, 1398.12it/s]
Sorting manifests...
Total duration of split: 37832.3861s
100%|██████████| 5567/5567 [00:00<00:00, 58691.16it/s]
Unpacking test-clean.tar.gz...
Converting flac files to wav and extracting transcripts...
129it [00:15,  8.17it/s]
Finished LibriSpeech_dataset/test_clean/test-clean.tar.gz
Gathering durations...
100%|██████████| 2620/2620 [00:01<00:00, 1419.30it/s]
Sorting manifests...
Total duration of split: 19452.4806s
100%|██████████| 2620/2620 [00:00<00:00, 56671.31it/s]
Unpacking test-other.tar.gz...
Converting flac files to wav and extracting transcripts...
125it [00:16,  7.37it/s]
Finished LibriSpeech_dataset/test_other/test-other.tar.gz
Gathering durations...
100%|██████████| 2939/2939 [00:02<00:00, 1300.47it/s]
Sorting manifests...
Total duration of split: 19229.5701s
100%|██████████| 2939/2939 [00:00<00:00, 58223.68it/s]
root@ecs-6715:/home/work/deepspeech.pytorch#
```

After the data processing, the data directory structure is as follows:

```
├ LibriSpeech_dataset
  |   ├── train
  |   |    ├ wav
  |   |    └ txt
```

```
    |   ├── val
    |   |     ├ wav
    |   |     └ txt
    |   ├── test_clean
    |   |     ├ wav
    |   |     └ txt
    |   └── test_other
    |         ├ wav
    |         └ txt
    └ libri_test_clean_manifest.json, libri_test_other_manifest.json,
libri_train_manifest.json, libri_val_manifest.json
```

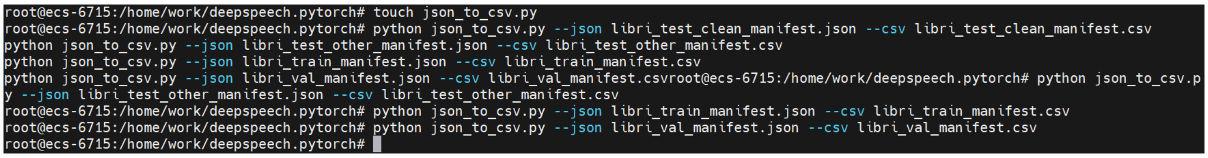
Go from the json file to the csv file, create the json_to_csv.py in the deepspeech.pytorch directory, and copy the code to the file with the following code:

```
touch json_to_csv.py
```

```
# json_to_csv.py:
import json
import csv
import argparse


parser = argparse.ArgumentParser(description='Image classification')
parser.add_argument("--json", type=str, default="", help="")
parser.add_argument("--csv", type=str, default="", help="")
config = parser.parse_args()

def trans(jsonfile, csvfile):
    jsonData = open(jsonfile)
    csvfile = open(csvfile, "a")
    for i in jsonData:
        dic = json.loads(i[0:])
        root_path = dic["root_path"]
        for j in dic["samples"]:
            wav_path = j["wav_path"]
            transcript_path =j["transcript_path"]
            res_wav = root_path + '/' + wav_path
            res_txt = root_path + '/' + transcript_path
            res = [res_wav, res_txt]
            writer = csv.writer(csvfile)
            writer.writerow(res)
    jsonData.close()
    csvfile.close()

if __name__ == "__main__":
    trans(config.json, config.csv)
```

Run the command as shown in the following figure:

```
python json_to_csv.py --json libri_test_clean_manifest.json --csv
libri_test_clean_manifest.csv
python json_to_csv.py --json libri_test_other_manifest.json --csv
libri_test_other_manifest.csv
python json_to_csv.py --json libri_train_manifest.json --csv libri_train_manifest.csv
python json_to_csv.py --json libri_val_manifest.json --csv libri_val_manifest.csv
```



3. Model Training Results and Evaluation

3.1. Model Training

Switch to the models / official / audio / DeepSpeech2 directory, Model training requires the creation of the deepspeech_pytorch directory under the DeepSpeech2 directory and the decoder.py file under the deepspeech_pytorch directory.

```
mkdir deepspeech_pytorch
cd deepspeech_pytorch
touch decoder.py
```

Copy the code to the decoder.py file with the following code:

```
#!/usr/bin/env python
# ----------------------------------------------------------------------------
# Copyright 2015-2016 Nervana Systems Inc.
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ----------------------------------------------------------------------------
# Modified to support pytorch Tensors

import Levenshtein as Lev
import torch
from six.moves import xrange


class Decoder(object):
    """
    Basic decoder class from which all other decoders inherit. Implements several
    helper functions. Subclasses should implement the decode() method.

    Arguments:
```

```
        labels (list): mapping from integers to characters.
        blank_index (int, optional): index for the blank '_' character. Defaults to 0.
    """

    def __init__(self, labels, blank_index=0):
        self.labels = labels
        self.int_to_char = dict([(i, c) for (i, c) in enumerate(labels)])
        self.blank_index = blank_index
        space_index = len(labels)  # To prevent errors in decode, we add an out of bounds
index for the space
        if ' ' in labels:
            space_index = labels.index(' ')
        self.space_index = space_index

    def wer(self, s1, s2):
        """
        Computes the Word Error Rate, defined as the edit distance between the
        two provided sentences after tokenizing to words.
        Arguments:
            s1 (string): space-separated sentence
            s2 (string): space-separated sentence
        """

        # build mapping of words to integers
        b = set(s1.split() + s2.split())
        word2char = dict(zip(b, range(len(b))))

        # map the words to a char array (Levenshtein packages only accepts
        # strings)
        w1 = [chr(word2char[w]) for w in s1.split()]
        w2 = [chr(word2char[w]) for w in s2.split()]

        return Lev.distance(''.join(w1), ''.join(w2))
    def cer(self, s1, s2):
        """
        Computes the Character Error Rate, defined as the edit distance.

        Arguments:
            s1 (string): space-separated sentence
            s2 (string): space-separated sentence
        """
        s1, s2, = s1.replace(' ', ''), s2.replace(' ', '')
        return Lev.distance(s1, s2)

    def decode(self, probs, sizes=None):
        """
        Given a matrix of character probabilities, returns the decoder's
        best guess of the transcription

        Arguments:
            probs: Tensor of character probabilities, where probs[c,t]
                            is the probability of character c at time t
            sizes(optional): Size of each sequence in the mini-batch
        Returns:
            string: sequence of the model's best guess for the transcription
```

```
        """
        raise NotImplementedError


class BeamCTCDecoder(Decoder):
    def __init__(self,
                 labels,
                 lm_path=None,
                 alpha=0,
                 beta=0,
                 cutoff_top_n=40,
                 cutoff_prob=1.0,
                 beam_width=100,
                 num_processes=4,
                 blank_index=0):
        super(BeamCTCDecoder, self).__init__(labels)
        try:
            from ctcdecode import CTCBeamDecoder
        except ImportError:
            raise ImportError("BeamCTCDecoder requires paddledecoder package.")
        labels = list(labels)  # Ensure labels are a list before passing to decoder
        self._decoder = CTCBeamDecoder(labels, lm_path, alpha, beta, cutoff_top_n,
cutoff_prob, beam_width,
                                       num_processes, blank_index)

    def convert_to_strings(self, out, seq_len):
        results = []
        for b, batch in enumerate(out):
            utterances = []
            for p, utt in enumerate(batch):
                size = seq_len[b][p]
                if size > 0:
                    transcript = ''.join(map(lambda x: self.int_to_char[x.item()],
utt[0:size]))
                else:
                    transcript = ''
                utterances.append(transcript)
            results.append(utterances)
        return results

    def convert_tensor(self, offsets, sizes):
        results = []
        for b, batch in enumerate(offsets):
            utterances = []
            for p, utt in enumerate(batch):
                size = sizes[b][p]
                if sizes[b][p] > 0:
                    utterances.append(utt[0:size])
                else:
                    utterances.append(torch.tensor([], dtype=torch.int))
            results.append(utterances)
        return results

    def decode(self, probs, sizes=None):
        """
```

```
        Decodes probability output using ctcdecode package.
        Arguments:
            probs: Tensor of character probabilities, where probs[c,t]
                            is the probability of character c at time t
            sizes: Size of each sequence in the mini-batch
        Returns:
            string: sequences of the model's best guess for the transcription
        """
        probs = probs.cpu()
        out, scores, offsets, seq_lens = self._decoder.decode(probs, sizes)

        strings = self.convert_to_strings(out, seq_lens)
        offsets = self.convert_tensor(offsets, seq_lens)
        return strings, offsets


class GreedyDecoder(Decoder):
    def __init__(self, labels, blank_index=0):
        super(GreedyDecoder, self).__init__(labels, blank_index)
        def convert_to_strings(self,
                           sequences,
                           sizes=None,
                           remove_repetitions=False,
                           return_offsets=False):
        """Given a list of numeric sequences, returns the corresponding strings"""
        strings = []
        offsets = [] if return_offsets else None
        for x in xrange(len(sequences)):
            seq_len = sizes[x] if sizes is not None else len(sequences[x])
            string, string_offsets = self.process_string(sequences[x], seq_len,
remove_repetitions)
            strings.append([string])  # We only return one path
            if return_offsets:
                offsets.append([string_offsets])
        if return_offsets:
            return strings, offsets
        else:
            return strings

    def process_string(self,
                       sequence,
                       size,
                       remove_repetitions=False):
        string = ''
        offsets = []
        for i in range(size):
            char = self.int_to_char[sequence[i].item()]
            if char != self.int_to_char[self.blank_index]:
                # if this char is a repetition and remove_repetitions=true, then skip
                if remove_repetitions and i != 0 and char == self.int_to_char[sequence[i -
1].item()]:
                    pass
                elif char == self.labels[self.space_index]:
                    string += ' '
                    offsets.append(i)
```

```
                else:
                    string = string + char
                    offsets.append(i)
        return string, torch.tensor(offsets, dtype=torch.int)

    def decode(self, probs, sizes=None):
        """
        Returns the argmax decoding given the probability matrix. Removes
        repeated elements in the sequence, as well as blanks.

        Arguments:
            probs: Tensor of character probabilities from the network. Expected shape of
batch x seq_length x output_dim
            sizes(optional): Size of each sequence in the mini-batch
            Returns:
            strings: sequences of the model's best guess for the transcription on inputs
            offsets: time step per character predicted
        """
        _, max_probs = torch.max(probs, 2)
        strings, offsets = self.convert_to_strings(max_probs.view(max_probs.size(0),
max_probs.size(1)),
                                                   sizes,
                                                   remove_repetitions=True,
                                                   return_offsets=True)
        return strings, offsets
```

Model configuration Modify the config.py under the src. After the modification, "ctrl + s" is saved and exits.

Modify batch_size to 1 (one-process data size that is related to server device performance).

Modify epochs to 1 (about 48h, can be adjusted according to the actual demand).

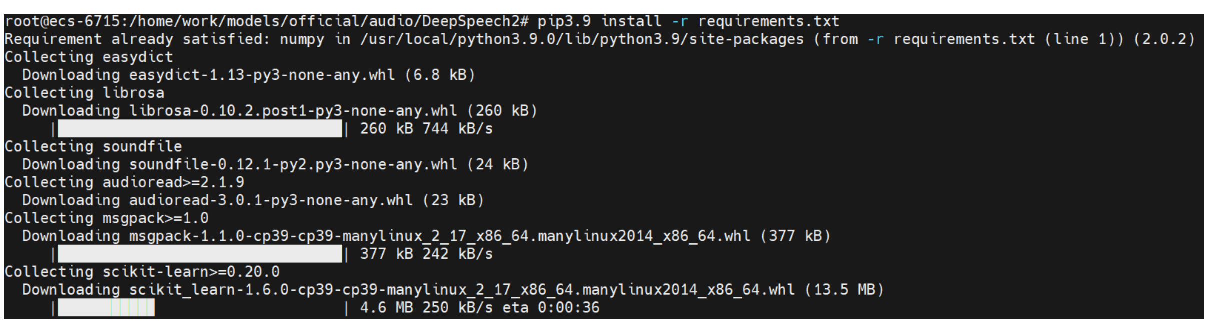
Modifies the train_manifest to the libri_train_manifest.csv actual path.

Modifies test_manifest to libri_test_clean_manifest.csv to the actual path.

Modify the window type of eval_config to change hanning to hann.

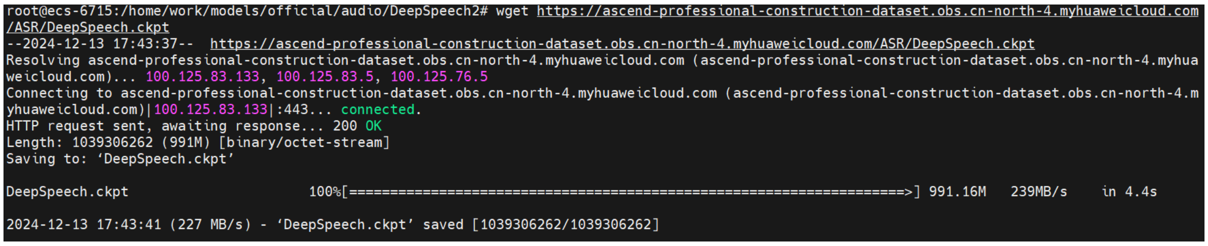
Install the model python dependency:

```
cd /home/work/models/official/audio/DeepSpeech2
pip3.9 install -r requirements.txt
pip3.9 install Levenshtein
pip3.9 install -i https://pypi.tuna.tsinghua.edu.cn/simple torch==1.7.1
pip3.9 install numpy==1.20.0
pip install numba==0.53.1
```



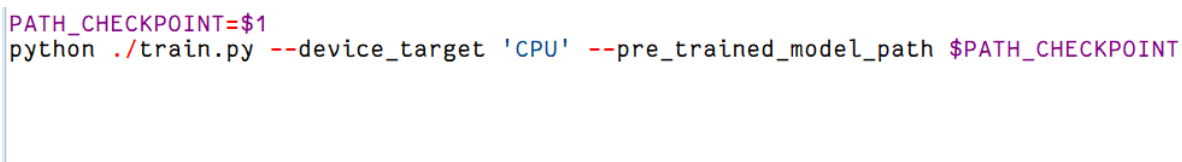
Download the pre-training model, the download link is https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/DeepSpeech.ckpt, and the download command is as follows:

```
wget https://ascend-professional-construction-dataset.obs.cn-north-
4.myhuaweicloud.com/ASR/DeepSpeech.ckpt
```



Modify the run_standalone_train_cpu.sh in the scripts directory to load the pre-training model The modification is as follows:

```
PATH_CHECKPOINT=$1
python ./train.py --device_target 'CPU' --pre_trained_model_path $PATH_CHECKPOINT
```



Train the model in the DeepSpeech2 directory and enter the following command.

```
bash scripts/run_standalone_train_cpu.sh PATH_CHECKPOINT
# PATH_CHECKPOINT: Pre-training file path
```

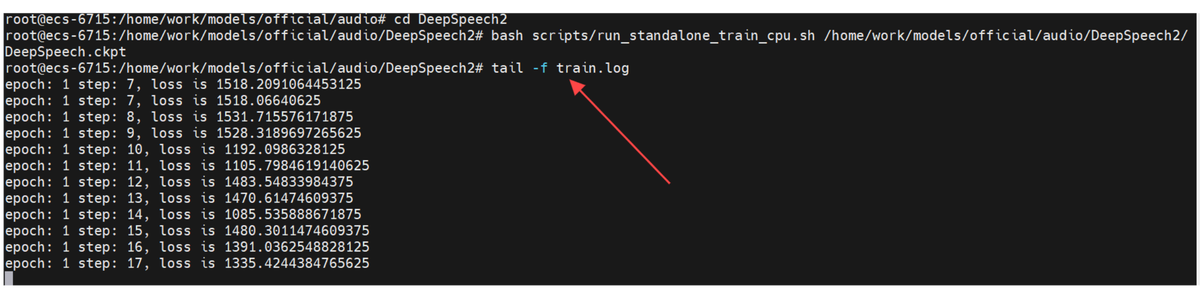
So that, the model is now being trained.

3.2. Model Training Results and Evaluation

3.2.1. View the Training Log

If we want to view the training log, the current directory under the train.log. Enter the command as follows:

```
tail -f train.log
```



So you can observe the training log all the time.

3.2.2. Training Results and Evaluation

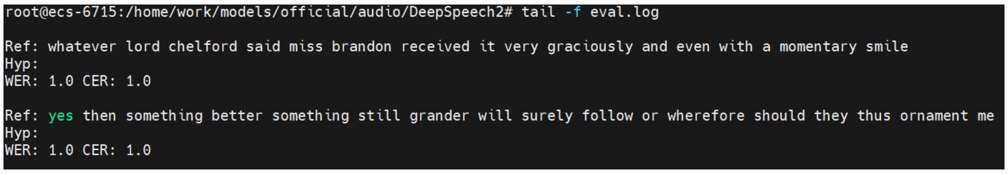
Model evaluation, enter the following command for evaluation.

```
bash scripts/run_eval_cpu.sh [PATH_CHECKPOINT]
# [PATH_CHECKPOINT] The Model checkpoint file
```

View the evaluation log, the eval.log in the current directory. Enter the command as follows:

```
tail -f eval.log
```

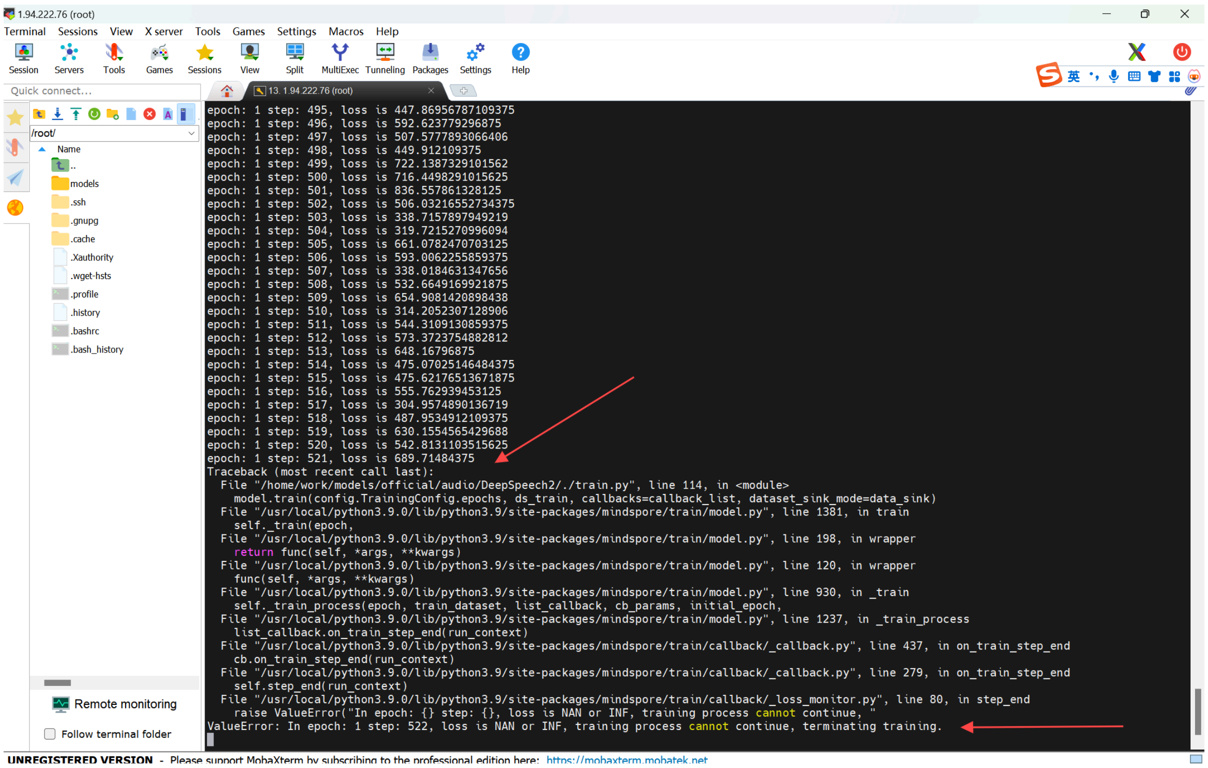
On my computer, the evaluation result is shown in the figure below:

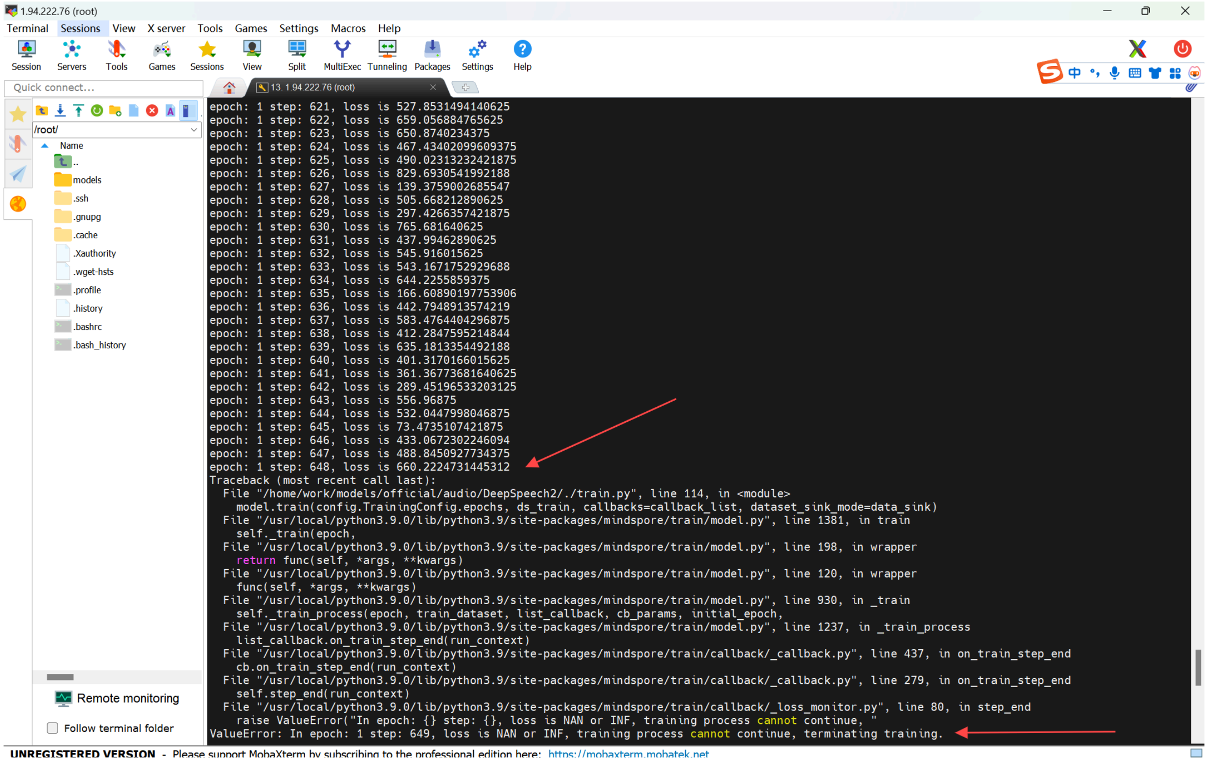


ASR refers to the automatic speech recognition technology (Automatic Speech Recognition), which is a technology to convert human speech into text. WER is the word error rate, Word Error Rate (WER) is an important indicator used to evaluate ASR performance, used to evaluate the error rate between predicted text and standard text, so the biggest characteristic of the word error rate is that the smaller, the better.

The two indicators on my computer are probably the same as those above in the experimental manual, and the model evaluation effect is good.

Here I have a problem, when the loss becomes nan or inf after the model training for a long time, it will automatically stop. Just like the following two pictures, I trained twice, one until the step length was more than 500, and the other was more than 600, about half an hour.





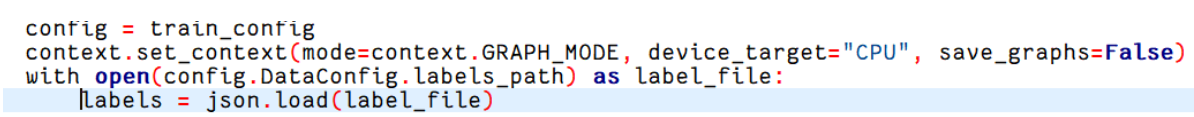
After many times of training, it is inevitable that this situation, I combine the data here to think there are the following several possibilities:

1. The learning rate is too high, which may lead to the gradient explosion, and the parameter update amplitude is too large, which makes the model weight becomes unstable. In the early stage of training, the update range of model weights is large. If the learning rate is too high, it is easy to lead to too large gradient and cause numerical overflow.
2. Gradient explosion, in deep networks the gradient may grow exponentially with backpropagation, leading in numerical spillover. When dealing with long sequences, the gradients can easily accumulate and eventually explode.
3. If the value of the loss function is unstable, the loss function may return a large value or directly return an invalid value. If the output of the model is logits and the value is too large, the numerical overflow may be caused when calculating the Softmax. Cross-entropy loss is prone to extreme values with probability approaching 0 or 1.
4. Sequence length or batch size problems, increasing memory requirements and computational complexity when handling long speech sequences or large volumes of data, may trigger numerical overflows. The longer the sequence length, the more information needed to process, easily leading to gradient problems. When the batch size is too large, the gradient of each update may be too intense.

3.3. Model Export

Model export that requires the following code to modify the export.py file.

```
config = train_config
context.set_context(mode=context.GRAPH_MODE, device_target="CPU", save_graphs=False)
with open(config.DataConfig.labels_path) as label_file:
labels = json.load(label_file)
```



Enter the following command for converting and exporting the model file.

```
root@ecs-6715:/home/work/models/official/audio/DeepSpeech2# python export.py --pre_trained_model_path /home/work/models/official/audio/DeepSpeech2
/DeepSpeech.ckpt
Successfully loading the pre-trained model
```

Then enter the command ll in the directory, and you can see the model file:

```
root@ecs-6715:/home/work/models/official/audio/DeepSpeech2# ll
total 1873172
drwxr-xr-x 6 root root       4096 Dec 14 13:37 ./
drwxr-xr-x 7 root root       4096 Dec 13 14:27 ../
drwx------ 2 root root       4096 Dec 14 13:20 checkpoint/
-r-------- 1 root root  346524583 Dec 14 13:45 deepspeech2.mindir
-rw-r--r-- 1 root root 1039306262 Jan 10  2023 DeepSpeech.ckpt
drwxr-xr-x 3 root root       4096 Dec 13 18:16 deepspeech_pytorch/
-rw-r--r-- 1 root root       8216 Dec 14 02:36 eval.log
-rw-r--r-- 1 root root       5690 Dec 13 14:27 eval.py
-rw-r--r-- 1 root root       3003 Dec 14 13:37 export.py
-rw-r--r-- 1 root root        205 Dec 13 14:27 labels.json
-rw-r--r-- 1 root root  532178606 Dec 14 01:12 librispeech_val_output.bin
-rw-r--r-- 1 root root       4765 Dec 13 14:27 quick_start.py
-rw-r--r-- 1 root root      13387 Dec 13 14:27 README-CN.md
-rw-r--r-- 1 root root      14740 Dec 13 14:27 README.md
-rw-r--r-- 1 root root         33 Dec 13 14:27 requirements.txt
drwxr-xr-x 2 root root       4096 Dec 13 14:27 scripts/
drwxr-xr-x 3 root root       4096 Dec 13 18:13 src/
-rw-r--r-- 1 root root       3258 Dec 14 13:20 train.log
-rw-r--r-- 1 root root       5442 Dec 13 14:27 train.py
```

4. Answer the Question

4.1. Question Description

1 round of model training, need 50 hours, how to improve the speed?

4.2. Problem Thinking

- **Replace the stronger hardware**

Using either a GPU or a TPU.

GPU: When training a deep learning model, the GPU is the most commonly used acceleration hardware. Compared with CPU, GPU can greatly improve the training speed when processing matrix operations. A NVIDIA GPU, such as A100, V100, or RTX 3090, is recommended.

TPU: If you use the Google Cloud, or any other TPU-enabled platform, the TPU performance may be stronger and is suitable for large-scale deep learning tasks.

Multi-GPU parallel training: If a single GPU is not enough, multiple GPU can be used for parallel training (data parallel or model parallel). Frameworks such as TensorFlow, PyTorch, and Mindspore all support this training approach.

Using a high-performance server:

If train on a cloud server, consider using stronger instances (such as NVIDIA Tesla V100, A100, or TPU instance). Cloud computing platforms: AWS, Google Cloud, and Azure all provide powerful computing resources that can be expanded as needed.

- **Training with a mixed-precision approach**

FP16 (Hybrid Precision Training): Hybrid precision training (FP16) can accelerate training while saving memory. It uses most of the calculations using 16-bit floating-points instead of 32-bit, thereby increasing computation speed and reducing memory footprint.

- **Data preprocessing and loading optimization**

Data preprocessing acceleration: If the data preprocessing before the training (such as audio feature extraction, image enhancement, etc.) is too time-consuming, it will significantly affect the training speed.

- **Adjust the bulk size**

Increase the batch size (Batch Size): Increasing the batch size can reduce the time required to calculate each step. However, increasing the batch size increases memory consumption and needs to be adjusted for hardware resources.

Increasing batch size: Sometimes a large batch size can cause video memory outages. In this case, the "progressive batch size" strategy can be adopted to gradually increase the batch size until the video memory reaches the maximum capacity.

- **Distributed training**

Data parallel: use multiple GPU or computing nodes, using data parallel training. Each compute node or GPU processes a different batch of the training data to accelerate the training by synchronously updating the model parameters.

Hybrid parallelism: combine model parallelism and data parallelism, distribute the model to multiple devices, and accelerate the training through data parallelism. This applies to very large models.

- **Model architecture optimization**

Pruning (Pruning): Pruning is a technique that reduces the number of model parameters to accelerate reasoning and training by removing redundant neurons or connections.

Knowledge distillation (Knowledge Distillation): Train a small model to simulate the output of a large pre-training model to accelerate the inference and training process.

Network compression: smaller networks (such as MobileNet, EfficientNet) are used to replace complex network models. For some tasks, using a lighter architecture significantly reduces the training time.