

山东大学计算机科学与技术学院

大数据分析与实践实验报告

学号：202300130214	姓名：于博雅	班级：23级数据班
实验题目：BERT环境配置		
实验学时：2	实验日期：2025. 11. 7	
实验目标： 创建大规模数据的处理平台实例，利用远程环境进行代码的调试		

实验步骤与结果：

选择云服务器平台恒源云，选择各项参数，创建实例：

Gpushare Cloud 恒源云

云市场模型市场数据集镜像合作伙伴解决方案帮助与支持中控制台

购买实例

计费模式

按量付费

包天

包周

包月

包年

对比说明

系统

Linux

地区

全部

华中

华东

华北

东北

西北

华南

西南

活动标签

☐ 限时特价

☐ 活动价

☐ 代金券

☐ 优惠券

☐ 夜间折扣

清空

机器特性

☐ 高可用

☐ 大硬盘

☐ 大内存

☐ 多核心

☐ 1元GPU

☐ 大显存

☐ cuda10

☐ Nvme

☐ 2元GPU

☐ 8卡GPU

☐ 10卡GPU

☐ 高速公网

☐ OSS加速

☐ 独立带宽

☐ SSD

☐ 高主频

☐ 动态公网

☐ 固定公网

☐ 超大内存

☐ 超大硬盘

☐ 独享带宽

☐ 优选

☐ NVLINK

☐ 内网存储

清空

Gpushare Cloud 恒源云

云市场模型市场数据集镜像合作伙伴解决方案帮助与支持中控制台

显卡类型

A30-24G

A10-24G

PRO 6000-96G

5080-16G

H100-80GB

A800-80GB

如何选卡?

A100-80GB

A100-40GB

5880Ada-48G

A6000-48G

L40S-48G

8000-48G

5000-16G

V100-16G

P100-16G

5090-32G

4090-48G

4090-24G

4090D-24G

M40-24G

4080-16G

4070 Ti-12G

3090 Ti-24G

4070-12G

4060 Ti-16G

3090-24G

3080 Ti-12G

3080-10G

3080-20G

3070-8G

4060-8G

3060 Ti-8G

3060-12G

2080 Ti-22G

2070 SUPER-8G

1080 Ti-12G

5060 Ti-16G

V100-32G

P40-24G

T4-16G

P4-8G

TITAN X-12G

GPU数量

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

多机多卡需求请联系客服开通

GPU 类型	地区	数量	显存	显卡驱动版本	最高 CUDA 版本	CPU 型号	CPU 配置	内存	实例硬盘	网络	价格	到期时间	可用券
3090-24G	西南	1	24 GB	535.183.06	12.2	AMD EPYC 7B12 64-Core Processor	8核	64G	系统盘：20G 数据盘：50GB NVME (关机保留24小时可扩容至5400 GB)	U: 700 Mbps/s D: 800 Mbps/s	¥0.89/小时	2026-10-13	-

数据盘：免费 50GB 需要扩容

实例镜像

官方镜像

备份镜像

镜像市场

PyTorch / 2.0.0 / 11.8.0 / 3.8

没有需要的镜像?

费用金额

¥0.89/小时

计费说明

余额不足，当前可用余额 ¥0.00，前往充值

创建实例

正在创建中：



最后创建完成如下图所示：

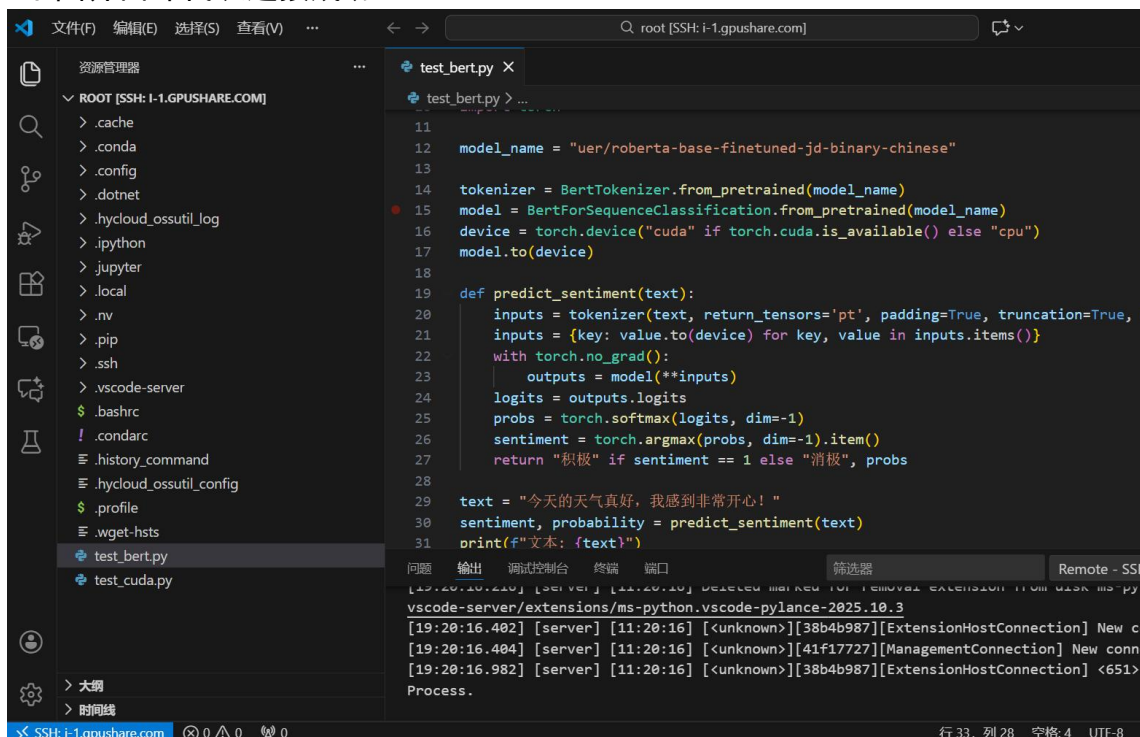


使用VSCode进行连接：

输入密码：（复制这一栏）



出现如下图界面即代表连接成功：



接下来是环境的搭配：

因为在创建服务器时就已经装好了 PyTorch，所以运行以下测试代码： test1.py

```
test1.py > ...
1 import torch
2 # 检查CUDA是否可用
3 cuda_available = torch.cuda.is_available()
4 print(f"CUDA 是否可用: {cuda_available}")
5 if cuda_available:
6     # 检查当前可用的GPU数量
7     gpu_count = torch.cuda.device_count()
8     print(f"可用的 GPU 数量: {gpu_count}")
9     # 获取当前GPU名称
10    for i in range(gpu_count):
11        print(f"GPU {i} 名称: {torch.cuda.get_device_name(i)}")
12    # 检查当前设备
13    current_device = torch.cuda.current_device()
14    print(f"当前使用的设备索引: {current_device}")
15 else:
16    print("未检测到可用的 CUDA 设备")
```

得到如下结果：

```
(base) root@I2562e4f31701501068:~# /usr/local/miniconda3/bin/python /root/test1.py
CUDA 是否可用: True
可用的 GPU 数量: 1
GPU 0 名称: NVIDIA GeForce RTX 3090
当前使用的设备索引: 0
(base) root@I2562e4f31701501068:~#
```

得到如上图输出，说明 torch包 的 CUDA 和 GPU 都是可用的。

但想要运行 BERT，还需要安装transformers

在终端输入 pip install transformers安装完成后运行以下测试代码： test2.py

```
test2.py > predict_sentiment
6 import os
7 os.environ['HF_ENDPOINT'] = 'https://hf-mirror.com'
8
9 from transformers import BertTokenizer, BertForSequenceClassification
10 import torch
11
12 model_name = "uer/roberta-base-finetuned-jd-binary-chinese"
13
14 tokenizer = BertTokenizer.from_pretrained(model_name)
15 model = BertForSequenceClassification.from_pretrained(model_name)
16 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
17 model.to(device)
18
19 def predict_sentiment(text):
20     inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True, max_length=512)
21     inputs = {key: value.to(device) for key, value in inputs.items()}
22     with torch.no_grad():
23         outputs = model(**inputs)
24     logits = outputs.logits
25     probs = torch.softmax(logits, dim=-1)
26     sentiment = torch.argmax(probs, dim=-1).item()
27     return "积极" if sentiment == 1 else "消极", probs
28
29 text = "今天的天气真好，我感到非常开心！"
30 sentiment, probability = predict_sentiment(text)
31 print(f"文本: {text}")
32 print(f"预测的情感: {sentiment}")
33 print(f"概率: {probability}")
```

得到如下的结果：

```
(base) root@I2562e4f31701501068:~# /usr/local/miniconda3/bin/python /root/test2.py
文本: 今天的天气真好，我感到非常开心！
预测的情感: 积极
概率: tensor([[0.0114, 0.9886]], device='cuda:0')
```

通过以上操作，BERT 的基础环境已经配好

遇到的问题以及解决方法：

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

之前报错的关键原因之一是服务器无法访问 HuggingFace 国外源（huggingface.co），导致模型下载超时 / 失败，现在将 HuggingFace 的模型下载端点强制切换为国内镜像源（hf-mirror.com），替代国外的 huggingface.co 帮助服务器快速拉取模型文件。

结论分析与体会：

本次实验成功在恒源云搭建远程实例，经 VSCode 连接调试，配好 PyTorch+transformers 的 BERT 环境，验证了 GPU 可用。遇到的 HuggingFace 源问题，换国内镜像解决。云镜像省环境配置，远程工具适配高效，国内源是海外工具国内用的关键。