D:\data_diver\work\homework.ini

```
#conda activate lecture6 ； 如果在miniconda等的命令行里
#python
#经过测试，该代码手动输入在终端（TERMINAL）里最终可以得到拉曼光谱的图片，
#但自动跑时就会出错
#原因在于空行太多？

from time import time #导入使用到的基本模块
t00 = time()
import numpy as np # type: ignore

X_fn = 'D:/data_diver/bacteria-ID/data/X_finetune.npy'
y_fn = 'D:/data_diver/bacteria-ID/data/y_finetune.npy'
X = np.load(X_fn)
y = np.load(y_fn)  #导入数据，打印数据的形状
print(X.shape, y.shape)


#已知resnet的文件在D:\data_diver\bacteria-ID中，
# 使用sys.path.append()方法将目标文件夹路径添加到Python的系统路径列表中
import sys
sys.path.append('D:\\data_diver\\bacteria-ID')
# 添加包含resnet模块的目录到系统路径
from resnet import ResNet # type: ignore

import os
import torch  #导入神经网络相关的模块

#已知预训练模型pretrained_model.ckpt在D:\data_diver\bacteria-ID 里，
# 需要确保在调用 torch.load() 函数时提供正确的文件路径


layers = 6
hidden_size = 100
block_size = 2
```

```
hidden_sizes = [hidden_size] * layers
num_blocks = [block_size] * layers
input_dim = 1000
in_channels = 64
n_classes = 30
os.environ['CUDA_VISIBLE_DEVICES'] = '{}'.format(0)
cuda = torch.cuda.is_available()  #设置CNN 参数（模型参数）

cnn = ResNet(hidden_sizes, num_blocks, input_dim=input_dim,
                in_channels=in_channels, n_classes=n_classes)
#第47行-50行命令需要一起使用；这一段if语句的命令很奇怪，
# 本身可运行，但有时会报错，需要多试（多点几下空格）
if cuda: cnn.cuda()
model_path = 'D:/data_diver/bacteria-ID/pretrained_model.ckpt' # 定义模型的完整路径
cnn.load_state_dict(torch.load(
    model_path, map_location=lambda storage, loc: storage)) # 为演示加载训练好的权重
#显示出<All keys matched successfully>字样则为成功激活

from datasets import spectral_dataloader # type: ignore
from training import run_epoch # type: ignore
from torch import optim  #导入额外的模块

p_val = 0.1
n_val = int(3000 * p_val)
idx_tr = list(range(3000))
np.random.shuffle(idx_tr)
idx_val = idx_tr[:n_val]
idx_tr = idx_tr[n_val:]
#生成两个索引 idx_val 和 idx_tr，将数据随机产分成训练集和测试集

# 微调 CNN
epochs = 1 # 将这个数字改为约30来进行完整训练
batch_size = 10
t0 = time()
# 设置 Adam 优化器
optimizer = optim.Adam(cnn.parameters(), lr=1e-3, betas=(0.5, 0.999))
# 设置数据加载器
```

```
dl_tr = spectral_dataloader(X, y, idxs=idx_tr,
                            batch_size=batch_size, shuffle=True)
dl_val = spectral_dataloader(X, y, idxs=idx_val,
                             batch_size=batch_size, shuffle=False)
# 微调 CNN 的第一阶段
best_val = 0
no_improvement = 0
max_no_improvement = 5
print('开始微调！ ')
for epoch in range(epochs):
    print(' Epoch {}: {:0.2f}s'.format(epoch+1, time()-t0))
    acc_tr, loss_tr = run_epoch(epoch, cnn, dl_tr, cuda,
        training=True, optimizer=optimizer)  # 训练
    print('  训练准确率: {:0.2f}'.format(acc_tr))
    acc_val, loss_val = run_epoch(epoch, cnn, dl_val, cuda,
        training=False, optimizer=optimizer)  # 验证
    print('  验证准确率: {:0.2f}'.format(acc_val))
    if acc_val > best_val or epoch == 0:    # 早停检查性能
        best_val = acc_val
        no_improvement = 0
    else:
        no_improvement += 1
    if no_improvement >= max_no_improvement:
        print('在 {} 轮后结束！ '.format(epoch+1))
        break
#使用时把空格和“#”字部分都删除，从第70行for开始至第85行break结束，
# 以上为一个循环语句

print('\n 这个演示完成耗时: {:0.2f}s'.format(time()-t00))

from resnet import ResNet # type: ignore
import os
import torch

# CNN parameters
layers = 6
hidden_size = 100
```

```
block_size = 2
hidden_sizes = [hidden_size] * layers
num_blocks = [block_size] * layers
input_dim = 1000
in_channels = 64
n_classes = 30 # instead of 30, we use the 8 empiric groupings


# 使用相同的参数重建模型，并载入权重
cnn = ResNet(hidden_sizes, num_blocks, input_dim=input_dim,
                in_channels=in_channels, n_classes=n_classes)

# 选择设备
# select the device for computation
if torch.cuda.is_available():
    device = torch.device("cuda")
elif torch.backends.mps.is_available():
    device = torch.device("mps")
else:
    device = torch.device("cpu")

# 载入模型权重
cnn.load_state_dict(torch.load('D:/data_diver/bacteria-ID/finetuned_model.ckpt',
        map_location=lambda storage, loc: storage))
#如果相对路径解析有问题，可以使用绝对路径
#出现<All keys matched successfully>字样则为载入成功

# 将模型移动到指定设备
cnn.to(device)

#检查模块是顶级模块还是子模块
for name, module in cnn.named_modules():
    # 如果名字是空，那么我们是最顶级；如果没有点，那么是顶级；有点的是子模块。
    if name == '':
        print(module)


```

```
#使用模型进行预测
import numpy as np
# 载入数据
X = np.load('D:/data_diver/bacteria-ID/data/X_test.npy')
y = np.load('D:/data_diver/bacteria-ID/data/y_test.npy')

# 打印数据形状
print(X.shape, y.shape)

#直接将整个数据集 X 转换为张量，并将其传递给模型进行预测
cnn.eval()

X_tensor = torch.tensor(X, dtype=torch.float32)
X_tensor = X_tensor.unsqueeze(1)
X_tensor = X_tensor.to(device)

with torch.no_grad():
    preds = cnn(X_tensor)


# 计算并打印准确性
y_hat = preds.argmax(dim=1).cpu().numpy()
acc = (y_hat == y).mean()
print('Accuracy: {:0.1f}%'.format(100*acc))


#读取菌株的名称
import config  # type: ignore
#来源于config.py文件（之前定义模型路径时可能顺带把这个也定义了？）

# 读取菌株名称顺序
order = config.ORDER

# 读取菌株名称
strains = config.STRAINS

# 打印菌株名称顺序
```

```
print(order)

# 打印菌株名称
print(strains)


from sklearn.metrics import confusion_matrix
#使用sklearn.metrics需在lecture6环境中（退出python环境）下载scikit-learn库

import seaborn as sns
import matplotlib.pyplot as plt

# 计算混淆矩阵
conf_matrix = confusion_matrix(y, y_hat, labels=order)

# 获取标签名称
label_names = [strains[i] for i in order]


# 绘制带有菌株名称的混淆矩阵
plt.figure(figsize=(10, 8))

# 创建热图
ax = sns.heatmap(conf_matrix,
            annot=True,
            fmt='d',
            cmap='YlGnBu',
            xticklabels=label_names,
            yticklabels=label_names)

# 将x轴标签移到顶部
ax.xaxis.set_ticks_position('top')
ax.xaxis.set_label_position('top')

plt.xticks(rotation=45, ha='left')
plt.yticks(rotation=0)
plt.xlabel('Predicted')
```

```
plt.ylabel('True')

# 调整布局以防止标签被切掉
plt.tight_layout()
plt.show()
```