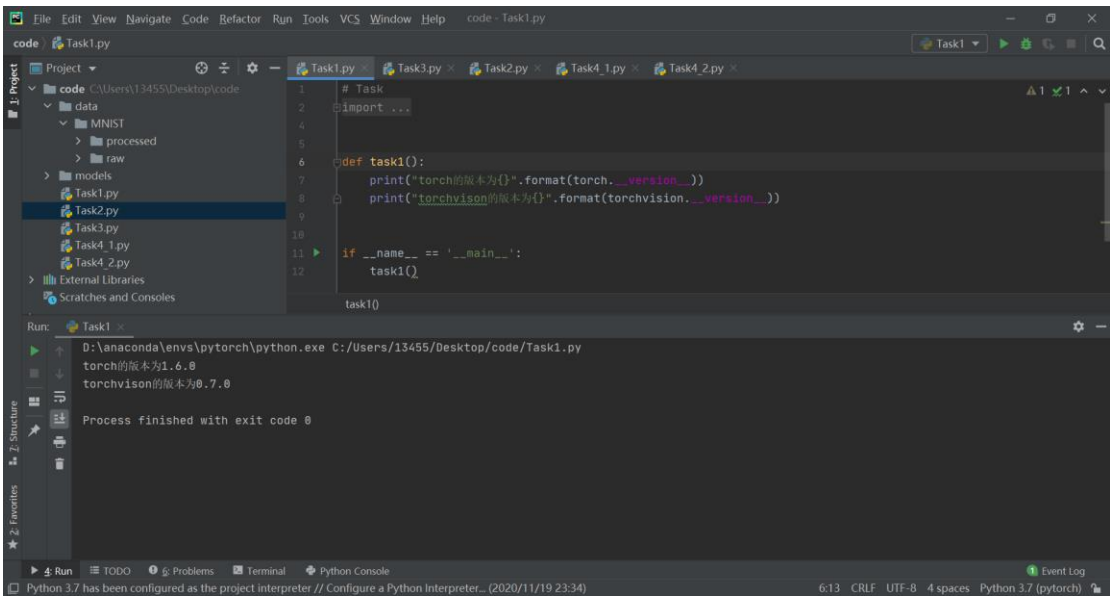


一：安装配置 Pytorch 环境，检测 Pytorch 安装情况。



```
# Task
import ...

def task1():
    print("torch的版本为{}".format(torch.__version__))
    print("torchvision的版本为{}".format(torchvision.__version__))

if __name__ == '__main__':
    task1()

task1()
```

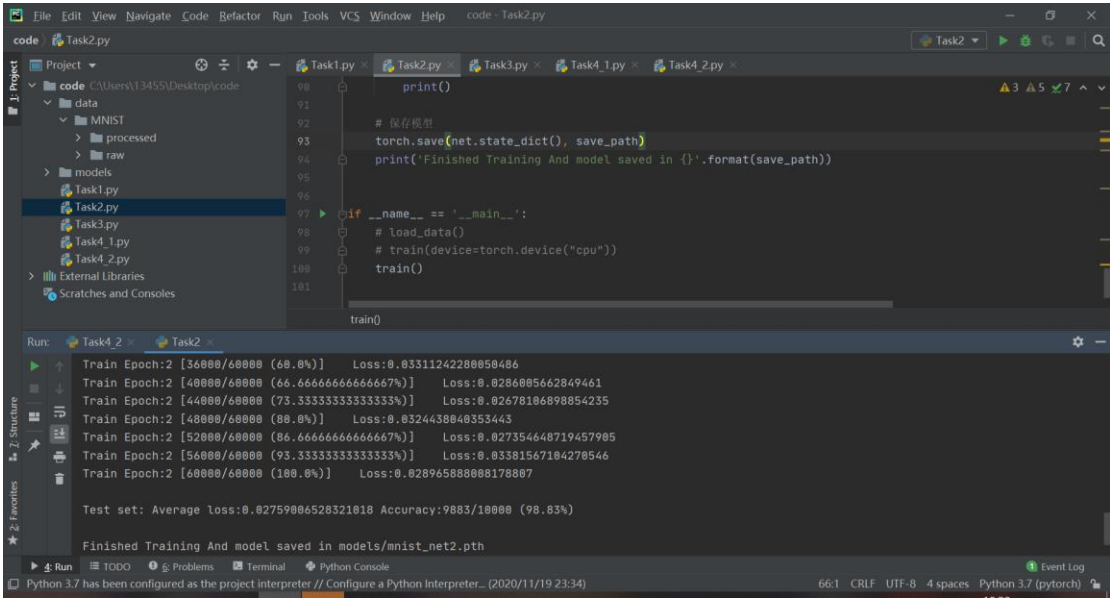
Run: Task1

D:\anaconda\envs\pytorch\python.exe C:/Users/13455/Desktop/code/Task1.py

torch的版本为1.6.0
torchvision的版本为0.7.0

Process finished with exit code 0

二、使用两个卷积层的神经网络对 MNIST 数据集分类。对生成网络结构进行截图（如例 1 所示），并对训练过程的精度增长情况进行截图：



```
print()

# 保存模型
torch.save(net.state_dict(), save_path)
print('Finished Training And model saved in {}'.format(save_path))

if __name__ == '__main__':
    # load_data()
    # train(device=torch.device("cpu"))
    train()

train()
```

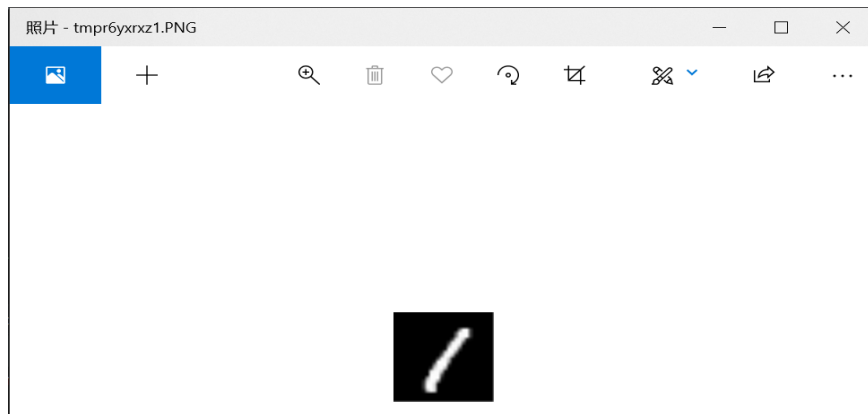
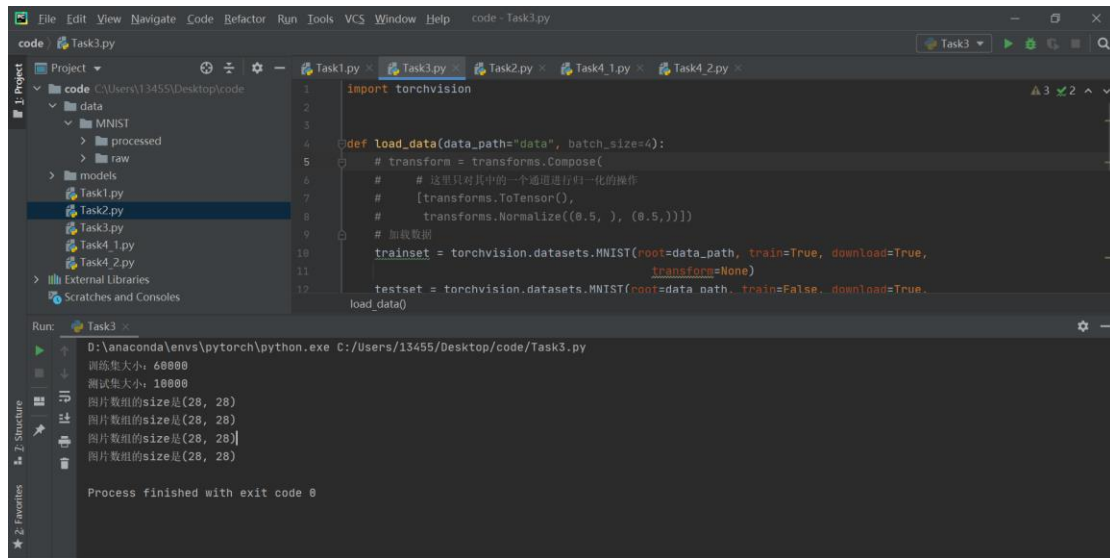
Run: Task2_2

Train Epoch:2 [36000/60000 (60.0%)] Loss:0.03311242280050486
Train Epoch:2 [40000/60000 (66.66666666666667%)] Loss:0.0286085662849461
Train Epoch:2 [44000/60000 (73.33333333333333%)] Loss:0.02678106898054235
Train Epoch:2 [48000/60000 (80.0%)] Loss:0.0324438048353443
Train Epoch:2 [52000/60000 (86.66666666666667%)] Loss:0.027354648719457905
Train Epoch:2 [56000/60000 (93.33333333333333%)] Loss:0.03381567104270546
Train Epoch:2 [60000/60000 (100.0%)] Loss:0.02896588808178807

Test set: Average loss:0.0275906528321018 Accuracy:9883/10000 (98.83%)

Finished Training And model saved in models/mnist_net2.pth

三：学习如何使用 MNIST 数据集。输出 MNIST 数据集训练集、验证集、测试大小；输出数据集中一张图片的数组 size，并将 MNIST 数据集中的一张手写体图片进行可视化展示。



四：修改网络结构（调整网络深度，使用不同的激活函数，调整神经元数量）或更改训练参数（学习率，batch_size），分析不同网络参数对于检测结果影响（至少分析两个变量）。

The screenshot shows an IDE window with the file `Task4_1.py` open. The code defines a `load_data` function and a neural network model. The output window shows the execution results for `Task4_1`.

```
code - Task4_1.py
Project
  code C:\Users\13455\Desktop\code
  data
    MNIST
      processed
      raw
  models
  Task1.py
  Task2.py
  Task3.py
  Task4_1.py
  Task4_2.py
  External Libraries
  Scratches and Consoles
Run: Task4_1
D:\anaconda\envs\pytorch\python.exe C:/Users/13455/Desktop/code/Task4_1.py
程序执行设备: cpu
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=9600, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=10, bias=True)
)
Train Epoch:0 [16000/60000 (26.666666666666668%)] Loss:0.6477198239304125
Python 3.7 has been configured as the project interpreter // Configure a Python Interpreter... (2020/11/19 23:34)
11:1 CRLF UTF-8 4 spaces Python 3.7 (pytorch)
```

The screenshot shows an IDE window with the file `Task4_2.py` open. The code shows the imports for the libraries used in the model.

```
code - Task4_2.py
Project
  code C:\Users\13455\Desktop\code
  data
    MNIST
      processed
      raw
  models
  Task1.py
  Task2.py
  Task3.py
  Task4_1.py
  Task4_2.py
  External Libraries
  Scratches and Consoles
Run: Task4_2
D:\anaconda\envs\pytorch\python.exe C:/Users/13455/Desktop/code/Task4_2.py
程序执行设备: cpu
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=9600, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=10, bias=True)
)
Python 3.7 has been configured as the project interpreter // Configure a Python Interpreter... (2020/11/19 23:34)
6:28 CRLF UTF-8 4 spaces Python 3.7 (pytorch)
```

分析：

- (1) **结点个数**：最后一个隐藏层结点个数越接近输出层的结点个数，效果相对越好。
- (2) **激活函数**：relu 比 sigmoid 效果要好。
- (3) **隐含层个数**：如果保持训练轮数不变，那么隐含层个数越多，效果相对越差；但如果同时增大训练轮数，那么效果就会相对较好。
- (4) **训练轮数**：增大训练轮数在一定程度上能够提高网络性能；尤其是在增大隐含层个数的情况下，增加训练轮数非常有必要。