**一、深度学习环境配置**(Anaconda +Pytorch + Jupyter Notebook + PyCharm)

网址：[面向零基础的 PyTorch 环境配置教程 (Anaconda +Pytorch + Jupyter Notebook + PyCharm)\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1jK2dYgEw3/?spm_id_from=333.1007.top_right_bar_window_history.content.click&vd_source=635f0af1a1c7fdacc2edef0784b9b9c5)

作者：进击的兵长啊啊啊

**二、PyTorch手写数字识别**[[

安装库：pip install numpy torch torchvision matplotlib

运行代码如下：

import torch

from torch.utils.data import DataLoader

from torchvision import transforms

from torchvision.datasets import MNIST

import matplotlib.pyplot as plt

class Net(torch.nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.fc1 = torch.nn.Linear(28\*28, 64)

self.fc2 = torch.nn.Linear(64, 64)

self.fc3 = torch.nn.Linear(64, 64)

self.fc4 = torch.nn.Linear(64, 10)

# 定义一个Net类，包含四个全链接层输入28×28尺寸的图像，中间64个节点，输出为10个数字类别。

def forward(self, x):

x = torch.nn.functional.relu(self.fc1(x))

x = torch.nn.functional.relu(self.fc2(x))

x = torch.nn.functional.relu(self.fc3(x))

x = torch.nn.functional.log\_softmax(self.fc4(x), dim=1)

return x

# forward定义前向传播过程，x为图像输入，每层传播中先做全链接线性计算，

# 外层为激活函数，输出层通过softmax归一化，log提高计算稳定性。

def get\_data\_loader(is\_train):

to\_tensor = transforms.Compose([transforms.ToTensor()])

data\_set = MNIST("", is\_train, transform=to\_tensor, download=True)

return DataLoader(data\_set, batch\_size=15, shuffle=True)

# 导入数据，首先定义数据转化类型，

# 下载mnist数据集，""下载目录，

# 空为当前目录，is\_train训练集还是测试集，

# batch\_size=15一个批次15张图片，

# shuffle=True随机打乱。

def evaluate(test\_data, net):

n\_correct = 0

n\_total = 0

with torch.no\_grad():

for (x, y) in test\_data:

outputs = net.forward(x.view(-1, 28\*28))

for i, output in enumerate(outputs):

if torch.argmax(output) == y[i]:

n\_correct += 1

n\_total += 1

return n\_correct / n\_total

# 评估神经网络的正确率，

# 从测试集按批次取出数据，计算神经网络的预测值，在对批次中的每个结果比较，累计正确预测数量。

def main():

train\_data = get\_data\_loader(is\_train=True)

test\_data = get\_data\_loader(is\_train=False)

net = Net()

# 导入训练集和测试集，初始化神经网络。

print("initial accuracy:", evaluate(test\_data, net))

# 打印初始神经网络的正确率。

optimizer = torch.optim.Adam(net.parameters(), lr=0.001)

for epoch in range(2):

for (x, y) in train\_data:

net.zero\_grad()

# 初始化

output = net.forward(x.view(-1, 28\*28))

# 正向传播

loss = torch.nn.functional.nll\_loss(output, y)

# 计算差值

loss.backward()

# 反向误差传播

optimizer.step()

# 优化网络参数

print("epoch", epoch, "accuracy:", evaluate(test\_data, net))

# 训练神经网络，pytorch的固定写法。

# epoch训练轮次

for (n, (x, \_)) in enumerate(test\_data):

if n > 3:

break

predict = torch.argmax(net.forward(x[0].view(-1, 28\*28)))

plt.figure(n)

plt.imshow(x[0].view(28, 28))

plt.title("prediction: " + str(int(predict)))

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

main()

网址：[10分钟入门神经网络 PyTorch 手写数字识别\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1GC4y15736/?spm_id_from=333.337.search-card.all.click&vd_source=635f0af1a1c7fdacc2edef0784b9b9c5)

作者：[孔工码字](https://space.bilibili.com/302545359)

注：

问题展示：

OMP: Error #15: Initializing libiomp5md.dll, but found libiomp5md.dll already initialized.

OMP: Hint This means that multiple copies of the OpenMP runtime have been linked into the program. That is dangerous, since it can degrade performance or cause incorrect results. The best thing to do is to ensure that only a single OpenMP runtime is linked into the process, e.g. by avoiding static linking of the OpenMP runtime in any library. As an unsafe, unsupported, undocumented workaround you can set the environment variable KMP\_DUPLICATE\_LIB\_OK=TRUE to allow the program to continue to execute, but that may cause crashes or silently produce incorrect results. For more information, please see http://www.intel.com/software/products/support/.

解决方法：

1、在PyCharm里调试程序时可以直接通过在程序前添加这两个语句解决

import os

os.environ[“KMP\_DUPLICATE\_LIB\_OK”]=“TRUE”

2、究其原因是，anaconda的环境下存在两个libiomp5md.dll文件。所以直接去虚拟环境的路径下搜索这个文件，可以看到在环境里有两个dll文件：

D:\anaconda3\envs\L7\Library\bin