实验　深度学习的编解码 Encoder-Decoder 实现 2

学号：21030031009　姓名：惠欣宇　电话：18149045867

本实验要求独立完成。实验报告正文要求采用小四号字，中文使用宋体，英文使用 Times New Roman。对于实验中出现的代码文字，可使用 Consolas 字体。实验报告要求提交 pdf 电子版。电子版文件命名格式要求为：学号-姓名.pdf。

1. 实验目的

1) 深入理解编解码概念、知识。
2) 掌握编解码的基本用法。
3) 利用 AutoEncoder 实现编解码框架还原生成 MNIST 数字图。

2. 实验预备知识及实验要求

2.1. 实验预备知识

1) 编解码、AutoEncoder 相关知识
2) MNIST 数据集类型、尺寸、数量
3) MNIST 下载链接：http://yann.lecun.com/exdb/mnist/

2.2. 实验要求：

1) 上交可以直接运行的代码（包括数据集）。
2) 展示整个实验过程，可适当截图，并标注说明。

用 python 实现 AutoEncoder 网络对 MNIST 的压缩再还原：

①encoder 压缩图片，得到 code；

②decoder 从 code 中还原图片。

首先记录程序开始时间，定义超参数，学习率 LR 为 0.005，N_TEST_IMG 表示显示五张图片的效果，

```
starttime = time.time()

torch.manual_seed(1)#随机种子

# 超参数
EPOCH = 10
BATCH_SIZE = 64
LR = 0.005
#DOWNLOAD_MNIST = True   # 下过数据的话，就可以设置成 False
N_TEST_IMG = 5          # 到时候显示 5张图片看效果
```

本次实验的数据集与上次实验相同，但只使用训练集：

```
# Mnist digits dataset
train_data = torchvision.datasets.MNIST(
    root='./data/',
    train=True,                                     # this is training data
    transform=torchvision.transforms.ToTensor(),    # Converts a PIL.Image or numpy.
                                                    # torch.FloatTensor of shape (C
    download=True,                          # download it if you don't have it
)

loader = Data.DataLoader(dataset=train_data, batch_size=BATCH_SIZE, shuffle=True)
```

下面构建 AutoEncoder 的网络结构：
AutoEncoder 分为 encoder 与 decoder 两部分。

在 encoder 中将 28x28 的神经元压缩为 128 个神经元，通过激活函数 Tanh 之后再从 128 个神经元压缩为 64 个神经元，如此重复压缩至 3 个神经元。

decoder 对图片进行解压操作，将 encoder 压缩出来的特征向量解压，解压过程仍需要使用激活函数 Tanh，将其解压为原来的 28x28，最后再使用 Sigmoid 函数将输出值固定在区间(0, 1)中。

AutoEncoder 的具体流程便是顺序使用 encoder 压缩与 decoder 解压，根据特征向量还原出最终图片。

```
class AutoEncoder(nn.Module):
    def __init__(self):
        super(AutoEncoder,self).__init__()
        # 压缩
        self.encoder = nn.Sequential(
            nn.Linear(28*28,128),
            nn.Tanh(),
            nn.Linear(128, 64),
            nn.Tanh(),
            nn.Linear(64, 32),
            nn.Tanh(),
            nn.Linear(32, 16),
            nn.Tanh(),
            nn.Linear(16, 3)
        )
        # 解压
        self.decoder = nn.Sequential(
            nn.Linear(3,16),
            nn.Tanh(),
            nn.Linear(16, 32),
            nn.Tanh(),
            nn.Linear(32, 64),
            nn.Tanh(),
            nn.Linear(64, 128),
            nn.Tanh(),
            nn.Linear(128, 28*28),
            nn.Sigmoid()  # 激励函数让输出值在 (0, 1)
        )
    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

打印 autoencoder 的网络结构，选取 Adam 优化器以及 MSE 损失函数。

```
#training
autoencoder = AutoEncoder()
print(autoencoder)

optimizer = torch.optim.Adam(autoencoder.parameters(),lr=LR)
loss_func = nn.MSELoss()
```

调用 matplotlib 模块进行画图展示：

在 subplots 函数中的参数表示绘制两行，每行绘制五张图片，figsize 表示画布大小，然后调用 imshow 开始画图。

```
# initialize figure
f, a = plt.subplots(2, N_TEST_IMG, figsize=(5, 2))  #return fig, ax
plt.ion()   # continuously plot
```

```
for i in range(N_TEST_IMG):
    a[0][i].imshow(np.reshape(view_data.data.numpy()[i], (28, 28)), cmap='gray'); a[0][i].set_xticks(()); a[0][i].set_yticks(())
```

因为图像数据需要按正确的类型输入给模型，则使用 view 函数对图像的 shape 类型进行改变操作，并进行归一化。

```
# original data (first row) for viewing
view_data = train_data.data[:N_TEST_IMG].view(-1, 28*28).type(torch.FloatTensor)/255
```

画图完成后进行训练过程：

使用 loader 分批读取训练集数据，每次返回三个值（step 表示读取进度，x 表示 image 数据，y 表示 label 数据）。将图像进行 shape 类型转换之后将 b_x 输入至模型中，得到压缩并解压后还原出的图。

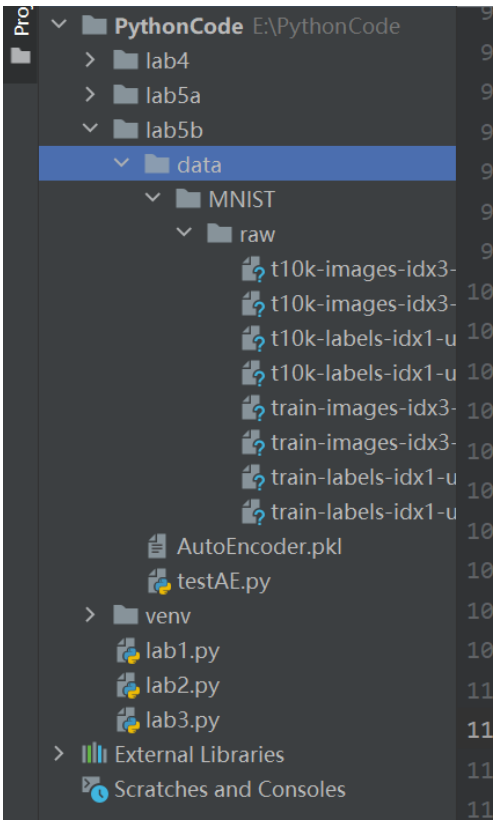
将还原图与目标值进行损失函数计算，将梯度置零后调用 backward 回传梯度，再进行梯度更新。

在每次读取完 50 个数据后验证学习效果，输出 Epoch 与 train_loss 观察效果。将训练集中的前五张图片传入模型得到还原之后的图，绘制的画布中第一行为真实数据，第二行为经过模型还原后的结果数据。同样调用 imshow 对第二行数据进行绘制。pause 表示绘制图片的间隔时长。

最后调用 ioff 关闭画图。

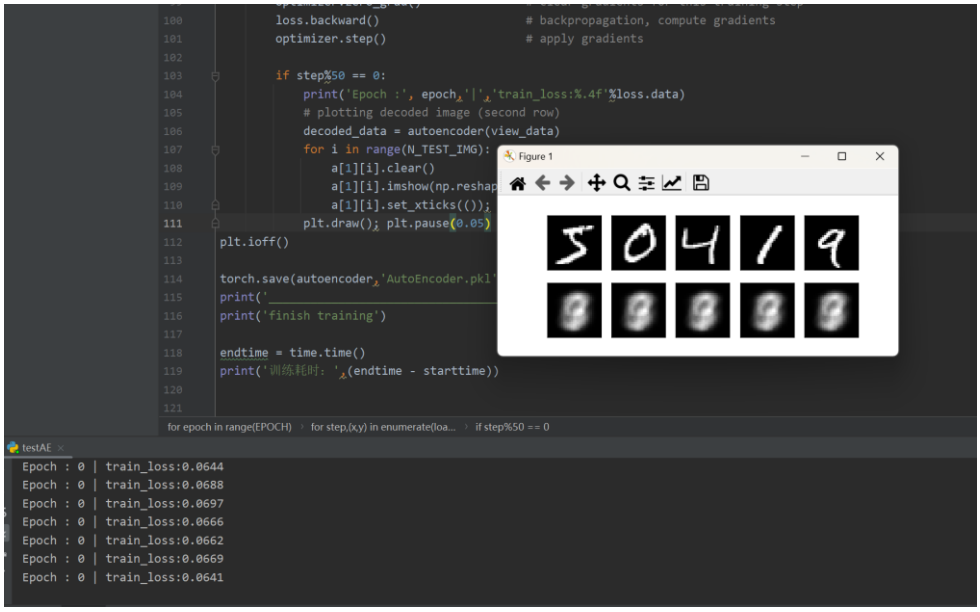
```
for epoch in range(EPOCH):
    for step,(x,y) in enumerate(loader):
        b_x = x.view(-1,28*28)  # batch x, shape (batch, 28*28)
        b_y = x.view(-1,28*28)  # batch x, shape (batch, 28*28)
        decoded = autoencoder(b_x)

        loss = loss_func(decoded, b_y)      # mean square error
        optimizer.zero_grad()               # clear gradients for this training step
        loss.backward()                     # backpropagation, compute gradients
        optimizer.step()                    # apply gradients

        if step%50 == 0:
            print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
            # plotting decoded image (second row)
            decoded_data = autoencoder(view_data)
            for i in range(N_TEST_IMG):
                a[1][i].clear()
                a[1][i].imshow(np.reshape(decoded_data.data.numpy()[i], (28, 28)), cmap='gray')
                a[1][i].set_xticks(()); a[1][i].set_yticks(())
            plt.draw(); plt.pause(0.05)
plt.ioff()

torch.save(autoencoder,'AutoEncoder.pkl')
print('_____')
print('finish training')

endtime = time.time()
print('训练耗时：',(endtime - starttime))
```

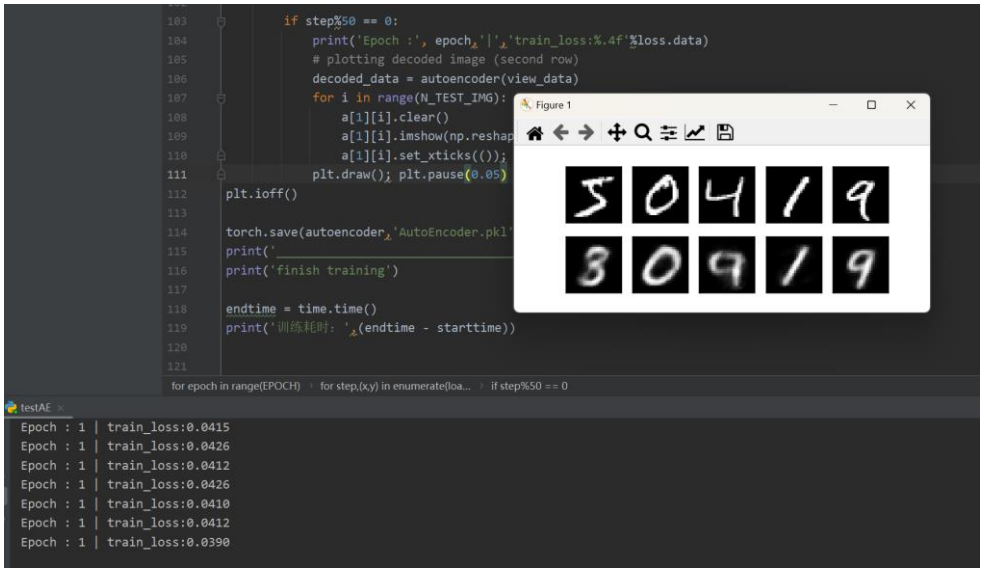
使用 `torch.save` 保存模型（AutoEncoder.pkl）：

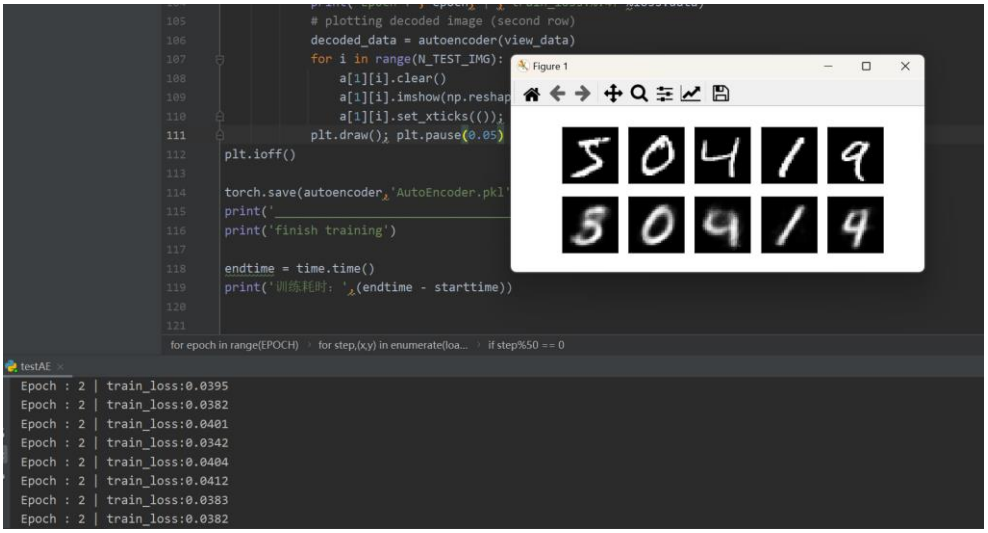


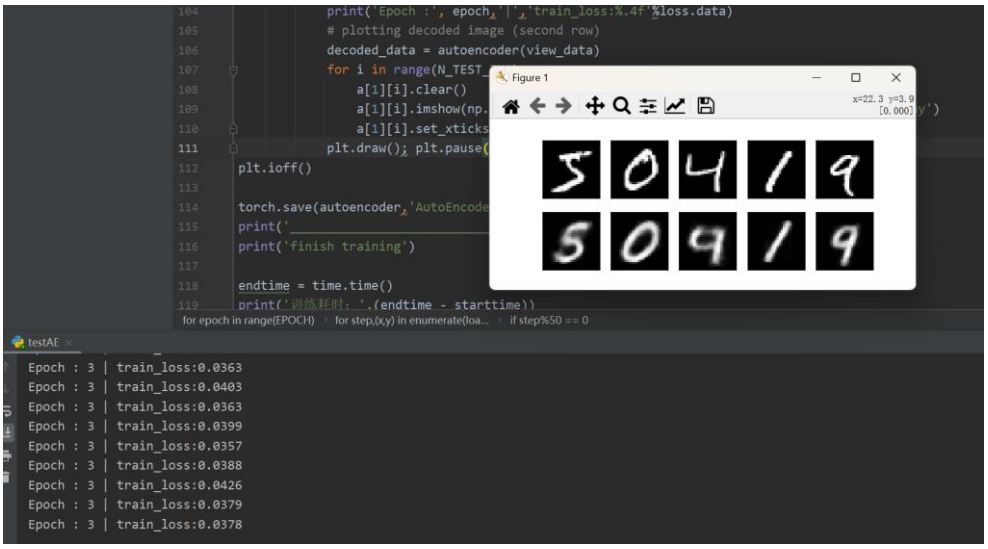
3） 实验结果。

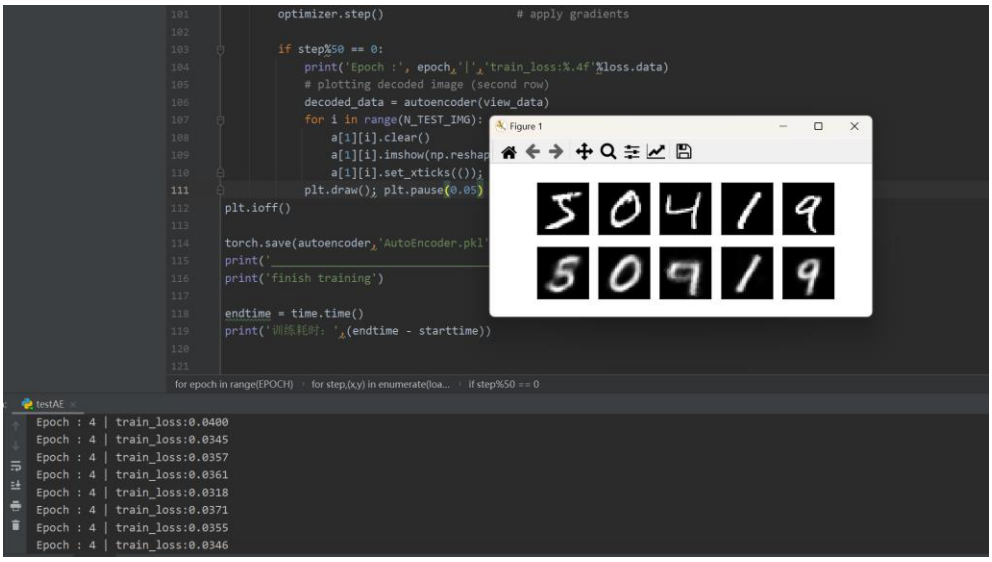
训练过程的输出结果：

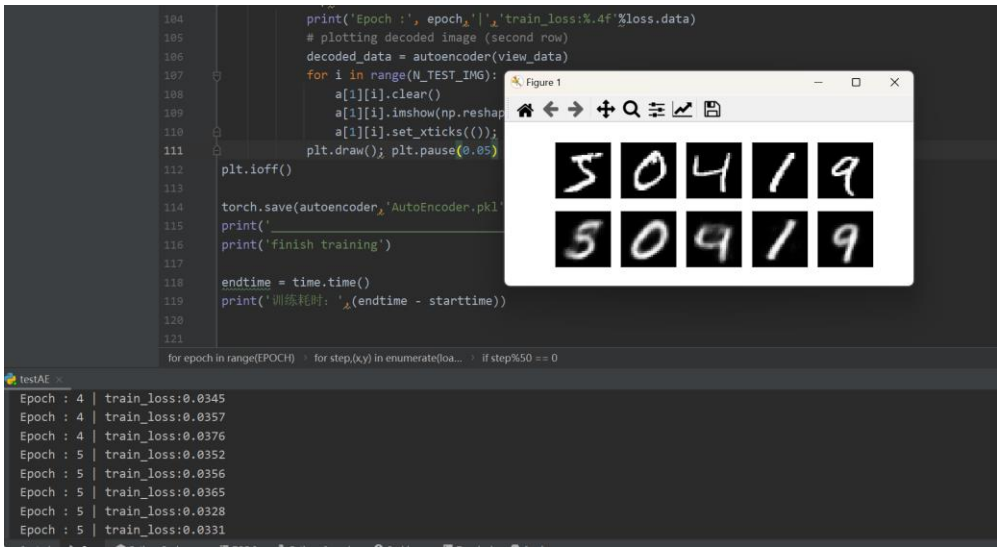




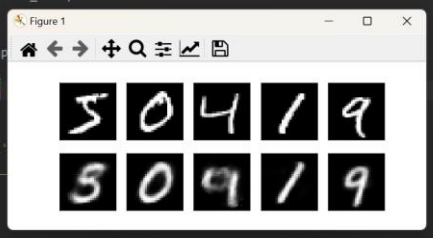



print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
# plotting decoded image (second row)
decoded_data = autoencoder(view_data)
for i in range(N_TEST_IMG):
a[1][i].clear()
a[1][i].imshow(np.
a[1][i].set_xticks
plt.draw(); plt.pause(
plt.ioff()
torch.save(autoencoder,'AutoEncode
print('
print('finish training')
endtime = time.time()
print('训练耗时: ',(endtime - starttime))
Figure 1
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0
testAE
Epoch : 3 | train_loss:0.0363
Epoch : 3 | train_loss:0.0403
Epoch : 3 | train_loss:0.0363
Epoch : 3 | train_loss:0.0399
Epoch : 3 | train_loss:0.0357
Epoch : 3 | train_loss:0.0388
Epoch : 3 | train_loss:0.0426
Epoch : 3 | train_loss:0.0379
Epoch : 3 | train_loss:0.0378


optimizer.step() # apply gradients
if step%50 == 0:
print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
# plotting decoded image (second row)
decoded_data = autoencoder(view_data)
for i in range(N_TEST_IMG):
a[1][i].clear()
a[1][i].imshow(np.reshap
a[1][i].set_xticks(());
plt.draw(); plt.pause(0.05)
plt.ioff()
torch.save(autoencoder,'AutoEncoder.pkl'
print('
print('finish training')
endtime = time.time()
print('训练耗时: ',(endtime - starttime))
Figure 1
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0
testAE
Epoch : 4 | train_loss:0.0400
Epoch : 4 | train_loss:0.0345
Epoch : 4 | train_loss:0.0357
Epoch : 4 | train_loss:0.0361
Epoch : 4 | train_loss:0.0318
Epoch : 4 | train_loss:0.0371
Epoch : 4 | train_loss:0.0355
Epoch : 4 | train_loss:0.0346


print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
# plotting decoded image (second row)
decoded_data = autoencoder(view_data)
for i in range(N_TEST_IMG):
a[1][i].clear()
a[1][i].imshow(np.reshap
a[1][i].set_xticks(());
plt.draw(); plt.pause(0.05)
plt.ioff()
torch.save(autoencoder,'AutoEncoder.pkl'
print('
print('finish training')
endtime = time.time()
print('训练耗时: ',(endtime - starttime))
Figure 1
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0
testAE
Epoch : 4 | train_loss:0.0345
Epoch : 4 | train_loss:0.0357
Epoch : 4 | train_loss:0.0376
Epoch : 5 | train_loss:0.0352
Epoch : 5 | train_loss:0.0356
Epoch : 5 | train_loss:0.0365
Epoch : 5 | train_loss:0.0328
Epoch : 5 | train_loss:0.0331

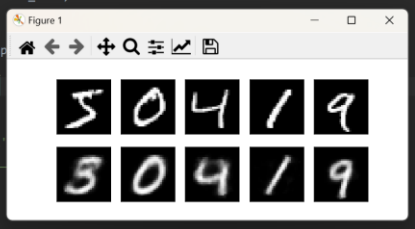
```
        if step%50 == 0:
            print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
            # plotting decoded image (second row)
            decoded_data = autoencoder(view_data)
            for i in range(N_TEST_IMG):
                a[1][i].clear()
                a[1][i].imshow(np.reshap
                a[1][i].set_xticks(());
            plt.draw(); plt.pause(0.05)
    plt.ioff()

    torch.save(autoencoder,'AutoEncoder.pkl'
    print('
    print('finish training')

    endtime = time.time()
    print('训练耗时：',(endtime - starttime))
```



for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0

testAE

```
Epoch : 6 | train_loss:0.0319
Epoch : 6 | train_loss:0.0326
Epoch : 6 | train_loss:0.0309
Epoch : 6 | train_loss:0.0377
Epoch : 6 | train_loss:0.0351
Epoch : 6 | train_loss:0.0338
Epoch : 6 | train_loss:0.0351
```



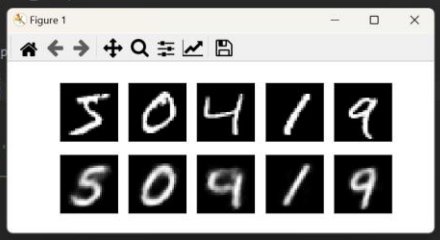
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0

Run: testAE

```
Epoch : 7 | train_loss:0.0356
Epoch : 7 | train_loss:0.0329
Epoch : 7 | train_loss:0.0337
Epoch : 7 | train_loss:0.0353
Epoch : 7 | train_loss:0.0353
Epoch : 7 | train_loss:0.0338
Epoch : 7 | train_loss:0.0366
```

Version Control | Run | Python Packages | TODO | Python Console | Problems | Terminal | Services

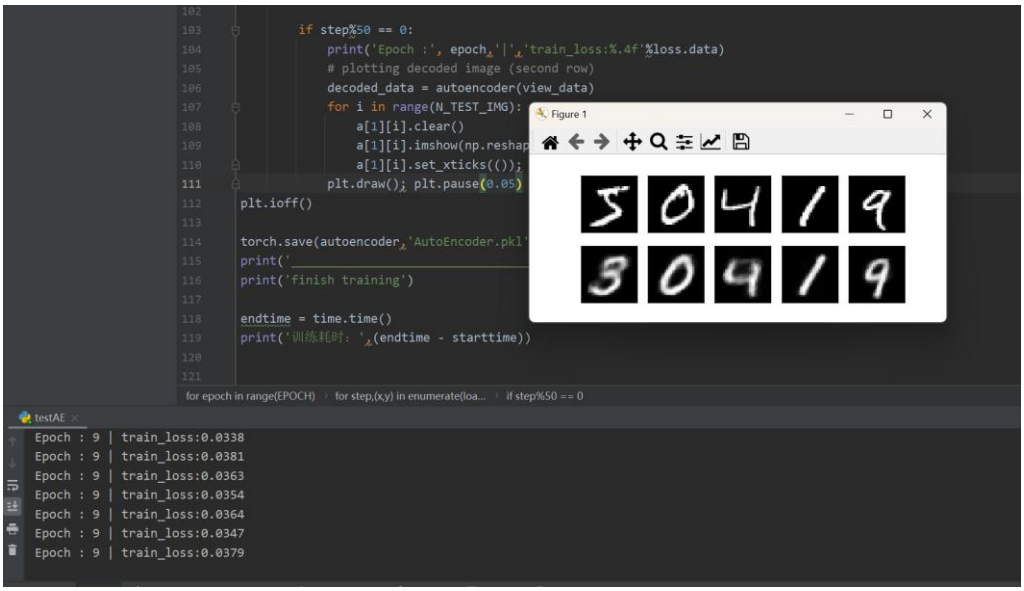
```
        if step%50 == 0:
            print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
            # plotting decoded image (second row)
            decoded_data = autoencoder(view_data)
            for i in range(N_TEST_IMG):
                a[1][i].clear()
                a[1][i].imshow(np.reshap
                a[1][i].set_xticks(());
            plt.draw(); plt.pause(0.05)
    plt.ioff()

    torch.save(autoencoder,'AutoEncoder.pkl'
    print('
    print('finish training')

    endtime = time.time()
    print('训练耗时：',(endtime - starttime))
```



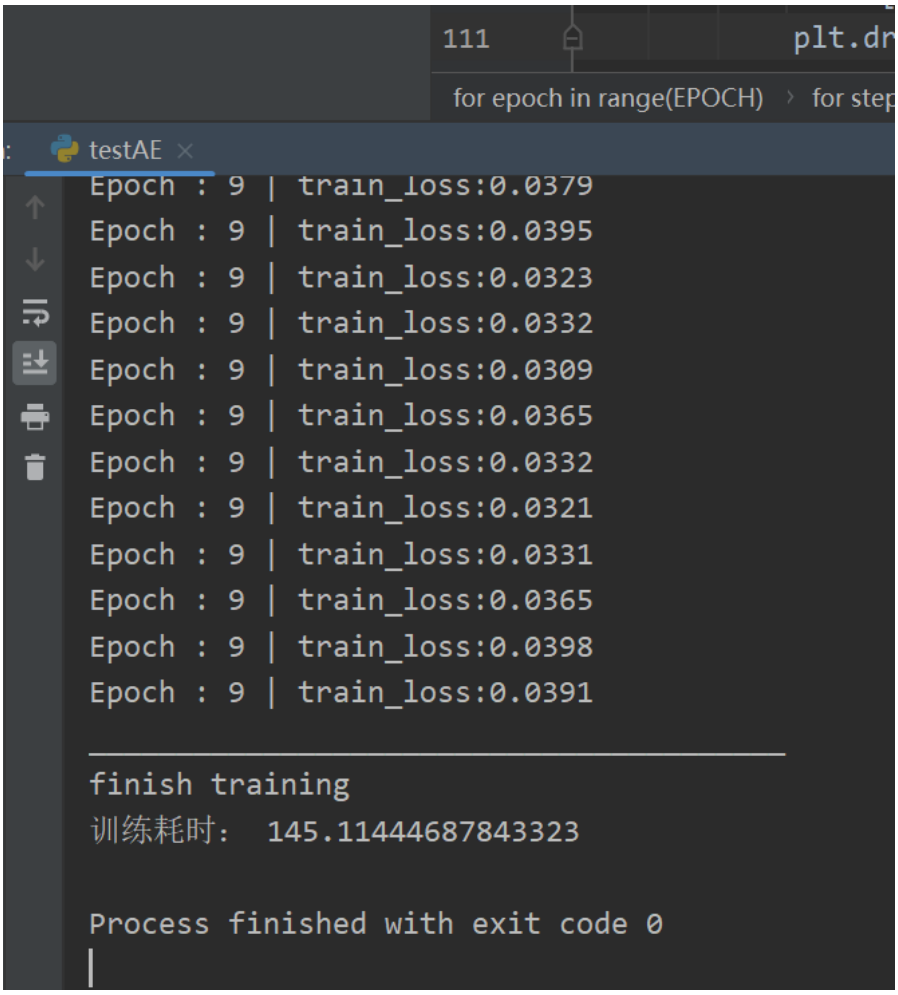
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0

testAE

```
Epoch : 8 | train_loss:0.0324
Epoch : 8 | train_loss:0.0326
Epoch : 8 | train_loss:0.0330
Epoch : 8 | train_loss:0.0366
Epoch : 8 | train_loss:0.0319
Epoch : 8 | train_loss:0.0391
Epoch : 8 | train_loss:0.0359
```


if step%50 == 0:
print('Epoch :', epoch,'|','train_loss:%.4f'%loss.data)
# plotting decoded image (second row)
decoded_data = autoencoder(view_data)
for i in range(N_TEST_IMG):
a[1][i].clear()
a[1][i].imshow(np.reshap
a[1][i].set_xticks(());
plt.draw(); plt.pause(0.05)
plt.ioff()
torch.save(autoencoder,'AutoEncoder.pkl'
print('
print('finish training')
endtime = time.time()
print('训练耗时：',(endtime - starttime))
Figure 1
for epoch in range(EPOCH) › for step,(x,y) in enumerate(loa... › if step%50 == 0
testAE
Epoch : 9 | train_loss:0.0338
Epoch : 9 | train_loss:0.0381
Epoch : 9 | train_loss:0.0363
Epoch : 9 | train_loss:0.0354
Epoch : 9 | train_loss:0.0364
Epoch : 9 | train_loss:0.0347
Epoch : 9 | train_loss:0.0379

训练完成：


111
plt.dr
for epoch in range(EPOCH) › for step
testAE
Epoch : 9 | train_loss:0.0379
Epoch : 9 | train_loss:0.0395
Epoch : 9 | train_loss:0.0323
Epoch : 9 | train_loss:0.0332
Epoch : 9 | train_loss:0.0309
Epoch : 9 | train_loss:0.0365
Epoch : 9 | train_loss:0.0332
Epoch : 9 | train_loss:0.0321
Epoch : 9 | train_loss:0.0331
Epoch : 9 | train_loss:0.0365
Epoch : 9 | train_loss:0.0398
Epoch : 9 | train_loss:0.0391
finish training
训练耗时： 145.11444687843323
Process finished with exit code 0

4)总结分析。总结实现模型搭建和训练过程中所遇到的困难和问题(给出你的解决办法)。总结关键步需要的注意事项。

AutoEncoder 是一个自动编码器是一个非监督的学习模式，只需要输入数据，不需要 `label` 或者输入输出对的数据。

虽然 AutoEncoder 是一个非监督学习算法，如果它的解码器是线性重建数据，可以用 `MSE` 来表示它的损失函数：

$$L(x,y) = \sum (x - h_{W,b}(x))^2 y = h_{W,b}(x)$$

如果解码器用 Sigmoid 的激活函数，主要使用交叉熵损失函数：

$$L(x,y) = -\sum_{i=1}^{d_x} x_i log(y_i) + (1 - x_i) log(1 - y_i)$$