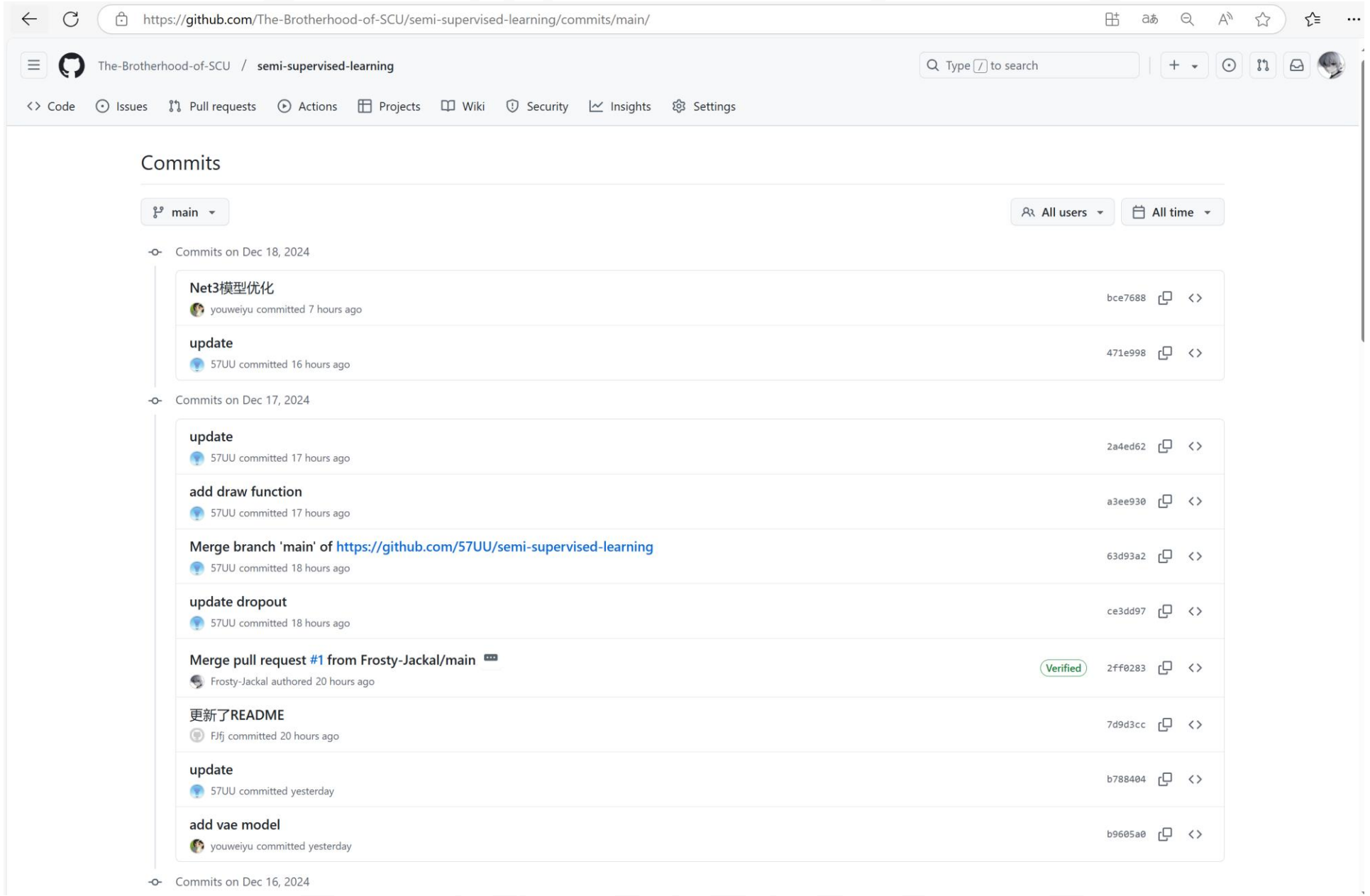
1

项目简介

1


https://github.com/The-Brotherhood-of-SCU/semi-supervised-learning/commits/main/
The-Brotherhood-of-SCU / semi-supervised-learning
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Commits
main
All users
All time
Commits on Dec 18, 2024
Net3模型优化
youweiyu committed 7 hours ago
bce7688
update
57UU committed 16 hours ago
471e998
Commits on Dec 17, 2024
update
57UU committed 17 hours ago
2a4ed62
add draw function
57UU committed 17 hours ago
a3ee930
Merge branch 'main' of https://github.com/57UU/semi-supervised-learning
57UU committed 18 hours ago
63d93a2
update dropout
57UU committed 18 hours ago
ce3dd97
Merge pull request #1 from Frosty-Jackal/main
Frosty-Jackal authored 20 hours ago
Verified
2ff0283
更新了README
FJfj committed 20 hours ago
7d9d3cc
update
57UU committed yesterday
b788404
add vae model
youweiyu committed yesterday
b9605a0
Commits on Dec 16, 2024

README: Neural Network Model for Classification

Overview

This repository contains a simple neural network model implemented using PyTorch for classification tasks for final project in the curriculum called "Introduction to AI". The model is designed to process input data with a shape of `batch_size x 784` (typically flattened 28x28 images, such as MNIST digits). The architecture includes convolutional layers, pooling layers, and fully connected layers to learn and classify the input data into one of ten classes. Thanks to 57U,yyw,and FrostyJ!

Model Architecture

under constructing

Usage

Prerequisites

- Python 3.x
- PyTorch

Installation

To use this model, you need to have PyTorch installed. You can install it using pip:

```
pip install torch torchvision
```

Running the Model

1. Clone this repository:

```
git clone https://github.com/The-Brotherhood-of-SCU/semi-supervised-learning.git
cd semi-supervised-learning
```

2. Create a Python script or use a Jupyter notebook to import and use the model.

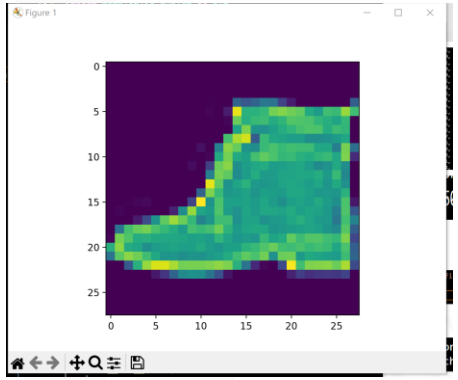
Notes

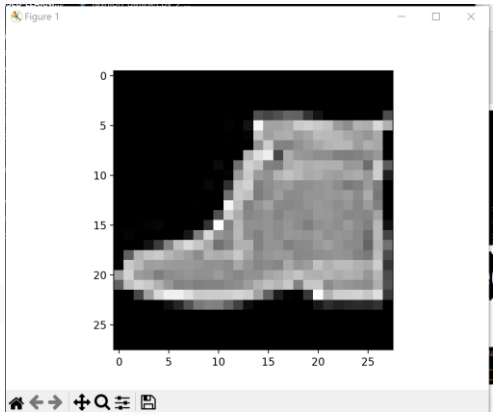
- The model expects input data to be of shape `batch_size x 784`, which is typically a flattened 28x28 image.
- The `forward` method of the `Net` class reshapes the input tensor to `batch_size x 1 x 28 x 28` before processing it through the convolutional layers.
- The output of the model is the raw scores for each class, which can be passed through a softmax function for probabilities (this is commented out in the `forward` method).

1

```
tensor([[  0,   1,   0,   0,   0,   4,  31,  64,  47,  11,   0,   0,   0,   0,
          10,  22,   0,   1,  14,  58,  62,  23,   0,   0,   0,   0,   0,   0],
        [  0,   1,   1,   0,  38, 104,  89,  83, 117, 141, 110,  74, 184, 116,
         122, 171,  83, 134, 143, 107,  79,  87,  74,   0,   0,   1,   0,   0],
        [  0,   0,   0,   8, 128,  82,  71,  71,  55,  80, 116, 116, 134, 147,
         131, 126, 137, 101,  58,  44,  62,  50,  90,  65,   0,   0,   0,   0],
        [  0,   2,   0,  64, 122,  74,  82,  73,  79,  68,  64,  65,  62,  70,
          65,  53,  37,  43,  58,  65,  61,  62,  67,  96,   2,   0,   0,   0],
        [  0,   0,   0,  99, 102,  79,  90,  71,  68,  74,  71,  76,  58,  61,
          61,  55,  67,  71,  61,  62,  59,  71,  70, 105,  32,   0,   2,   0],
        [  0,   0,   0, 132, 122,  95,  95,  79,  67,  71,  73,  74,  65,  70,
          71,  62,  71,  64,  59,  64,  59,  74,  74, 108,  59,   0,   0,   0],
        [  0,   0,  13, 147,  99, 155,  95, 102,  73,  76,  73,  71,  70,  67,
          68,  65,  71,  64,  62,  64,  68,  76,  95,  92,  83,   0,   0,   0],
        [  0,   0,  34, 165,  76, 174, 111, 113,  90,  77,  70,  65,  68,  71,
          76,  74,  82,  65,  64,  65,  71, 101, 111,  76, 108,  19,   0,   0],
        [  0,   0,  61, 158,  92, 170, 129, 111, 119,  79,  74,  76,  76,  76,
          74,  79,  89,  71,  82,  64,  74, 123,  92,  73, 116,  37,   0,   0],
        [  0,   0,  98, 146, 107, 207, 195, 104, 131, 101,  86,  80,  83,  79,
          74,  77,  85,  83,  96,  71, 110, 158, 101, 104, 125,  64,   0,   0],
```

```
print(x.shape)
x=torch.tensor(x).permute(1,0)
t=x[0].view(28,28)
plt.imshow(t)
plt.show()
```






```
torch.Size([784, 5000])
```

```
x=torch.tensor(np.load("fashion-dataset\\final_x.npy") ).to(device)
print(torch.max(x))
print(torch.min(x))
```

```
tensor(255, dtype=torch.uint8)
tensor(0, dtype=torch.uint8)
```

```
device="cuda:0" if torch.cuda.is_available() else "cpu"
floder_name="fashion-dataset"

print("current_device: ",device)
def load_npy(file_name:str,isDiv255=True)->torch.Tensor:
    tensor=torch.tensor(np.load(f"{floder_name}/{file_name}")).to(device).permute(1,0)
    tensor=tensor.float()
    if isDiv255:#feature
        tensor=tensor/255.0
    else:#label
        tensor=tensor.argmax(1)
    return tensor
```

1

```python
class TrainDataSet(Dataset):
    def __init__(self):
        self.x=load_npy("train_x.npy")
        self.y=load_npy("train_y.npy",isDiv255=False)
        print("x-shape",self.x.shape,self.x.dtype)
        print("y-shape",self.y.shape)
    def __getitem__(self, index):
        return self.x[index],self.y[index]
    def __len__(self):
        return len(self.x)
```

```
x-shape torch.Size([5000, 784]) torch.float32
y-shape torch.Size([5000])
```

```python
train_dataset_=TrainDataSet()
```

```python
class EnhancedDataset(Dataset):
    def __init__(self, original_dataset,offset=2):
        self.x = []
        self.y = []

        for data, label in original_dataset:
            # 将数据重塑为28x28的图像
            image = data.view(28, 28)

            # 左右上下移动3个像素
            left = torch.roll(image, shifts=-offset, dims=1)
            right = torch.roll(image, shifts=offset, dims=1)
            up = torch.roll(image, shifts=-offset, dims=0)
            down = torch.roll(image, shifts=offset, dims=0)
            flipped = torch.flip(image, dims=(1,))
            # 将修改后的图像添加到数据列表中，并保持标签不变
            self.x.append(left.view(-1))
            self.x.append(right.view(-1))
            self.x.append(up.view(-1))
            self.x.append(down.view(-1))
            self.x.append(data)
            self.x.append(flipped.view(-1))
            self.y.extend([label] * 6)

    def __len__(self):
        return len(self.x)

    def __getitem__(self, idx):
        return self.x[idx], self.y[idx]
```

```python
display_photo(enhanced_dataset_1,0)
display_photo(enhanced_dataset_1,1)
display_photo(enhanced_dataset_1,2)
display_photo(enhanced_dataset_1,3)
display_photo(enhanced_dataset_1,4)
```

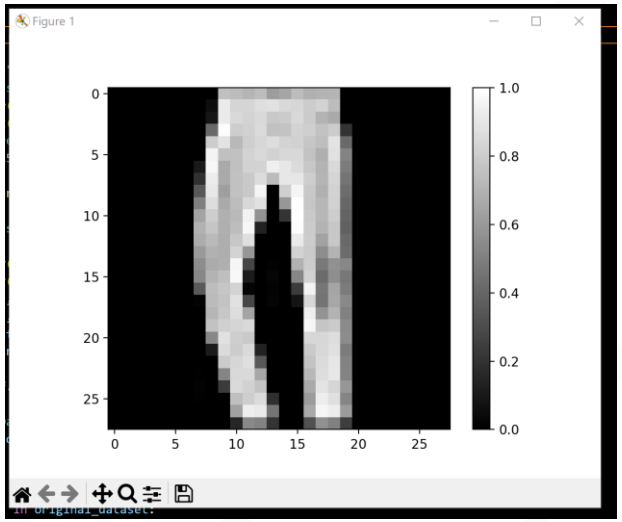


Figure 1

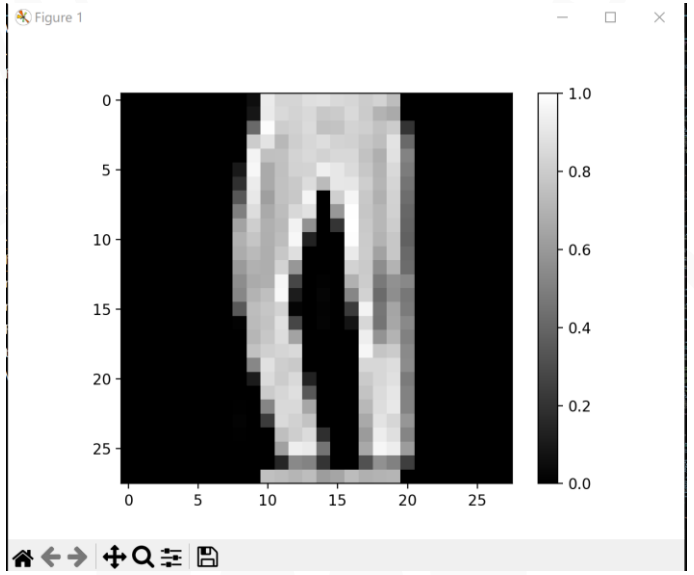
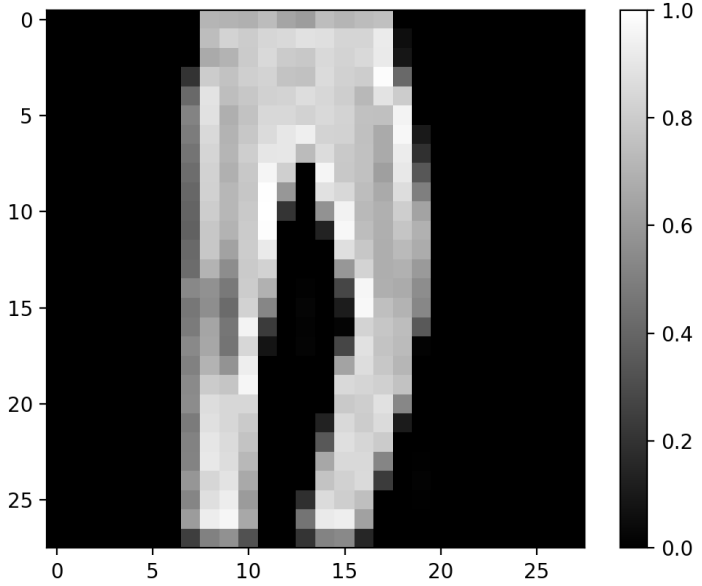


Figure 1



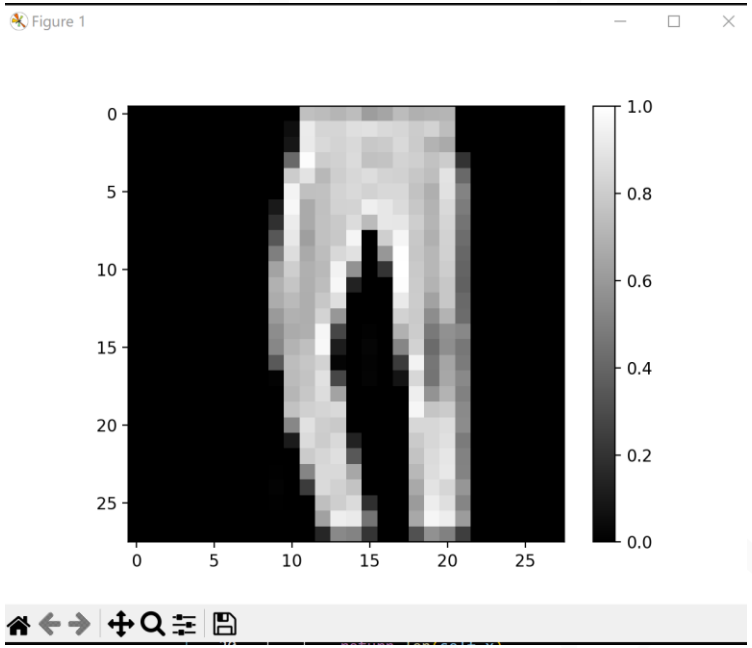


Figure 1

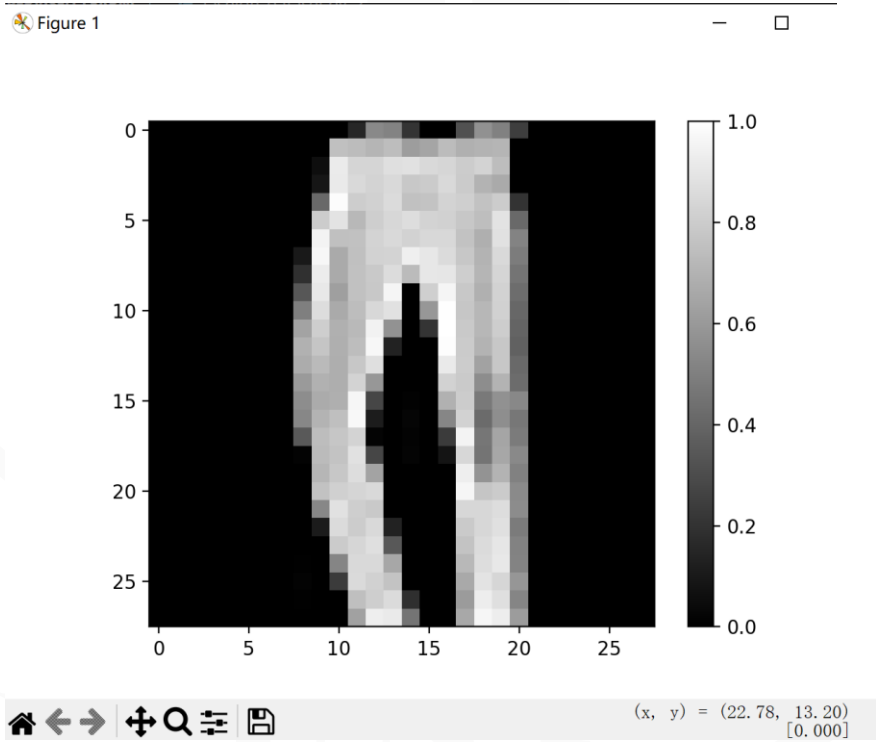


Figure 1
(x, y) = (22.78, 13.20)
[0.000]

```python
class RotatedDataset(Dataset):
    def __init__(self, original_dataset):
        self.x = []
        self.y = []

        for data, label in original_dataset:
            # 将数据重塑为28x28的图像
            image = data.view(28, 28)
            # 保存原始图像
            self.x.append(image)
            self.y.append(label)
            # 旋转90度
            rotated90 = torch.rot90(image, k=1, dims=(0, 1))
            self.x.append(rotated90)
            self.y.append(label)
            # 旋转180度
            rotated180 = torch.rot90(image, k=2, dims=(0, 1))
            self.x.append(rotated180)
            self.y.append(label)
            # 旋转270度
            rotated270 = torch.rot90(image, k=3, dims=(0, 1))
            self.x.append(rotated270)
            self.y.append(label)

        # 将图像重塑回784维向量
        self.x = [img.reshape(-1) for img in self.x]

    def __len__(self):
        return len(self.x)

    def __getitem__(self, idx):
        return self.x[idx], self.y[idx]
```

```
train_dataset_=TrainDataSet()

unlabeled_dataset=UnlabeledDataSet()
test_dataset=TestDataSet()
final_dataset=FinalTestDataSet()
# 直接用这个替代原始的
train_dataset=FlippedDataset(train_dataset_)
combined_unlabeled_dataset = CombinedUnlabeledDataset(unlabeled_dataset)

enhanced_dataset_1=EnhancedDataset(train_dataset_,offset=1)
enhanced_dataset_2=EnhancedDataset(train_dataset_,offset=2)

batch_size=32
# loaders
train_loader=DataLoader(train_dataset,batch_size=batch_size,shuffle=True)
unlabeled_loader=DataLoader(combined_unlabeled_dataset,batch_size=batch_size,shuffle=True)
test_loader=DataLoader(test_dataset,batch_size=256,shuffle=True)
final_test_loader=DataLoader(final_dataset,batch_size=batch_size,shuffle=True)
enhance_loader_1=DataLoader(enhanced_dataset_1,batch_size=batch_size,shuffle=True)
enhance_loader_2=DataLoader(enhanced_dataset_2,batch_size=batch_size,shuffle=True)
```

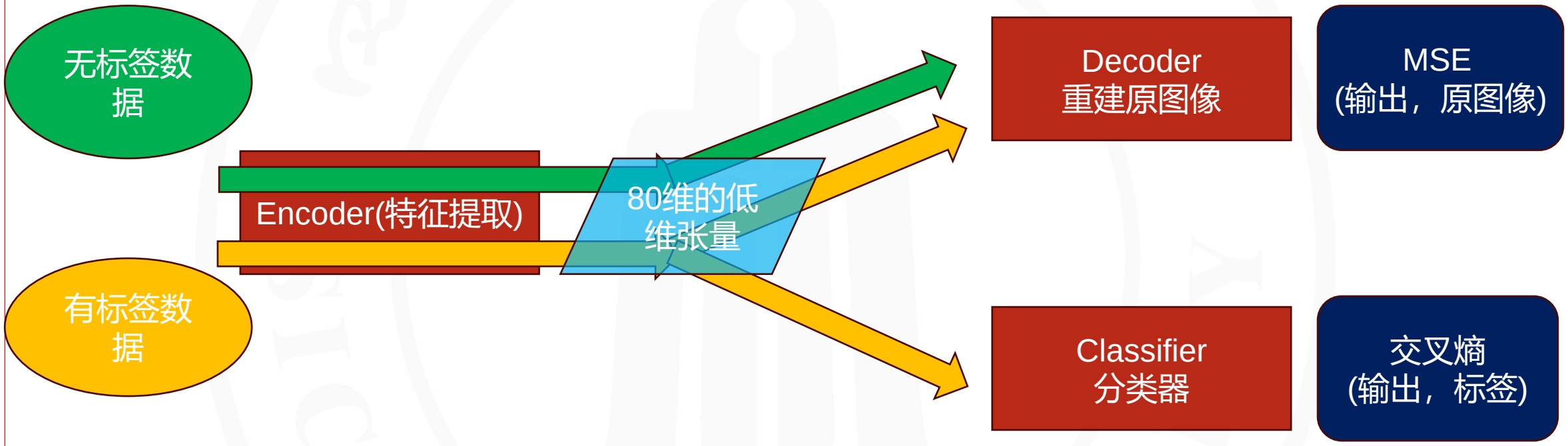
```python
import random
loaders = [enhance_loader_1, enhance_loader_2, train_loader, train_loader]

def train_infty():
    while True:
        loader = random.choice(loaders)
        train_semi_supervised(train_loader=loader, epochs=1)
        update_final_output()
```

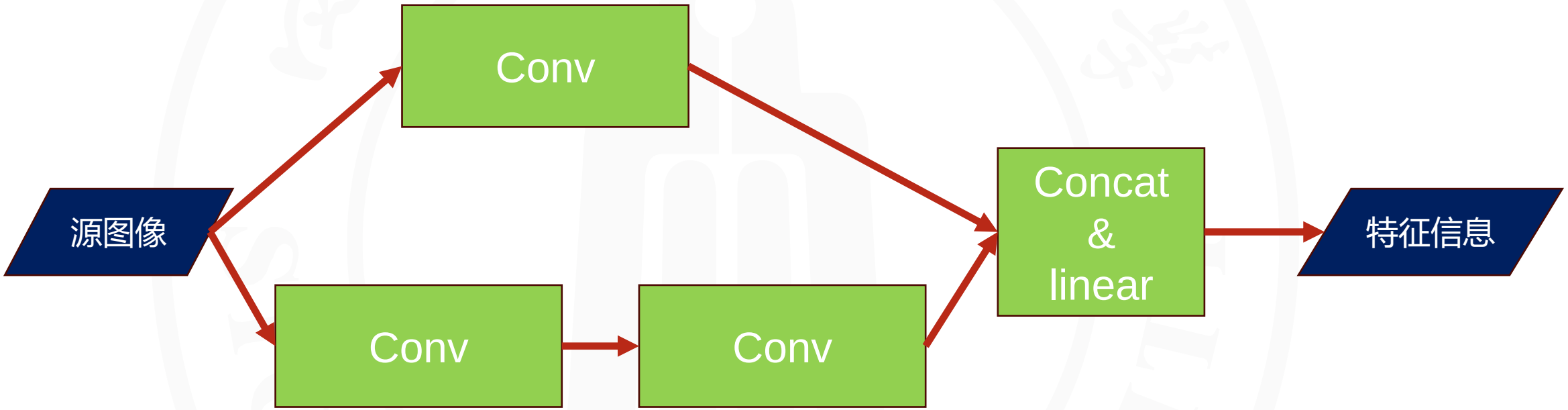
1

模型架构

多任务学习

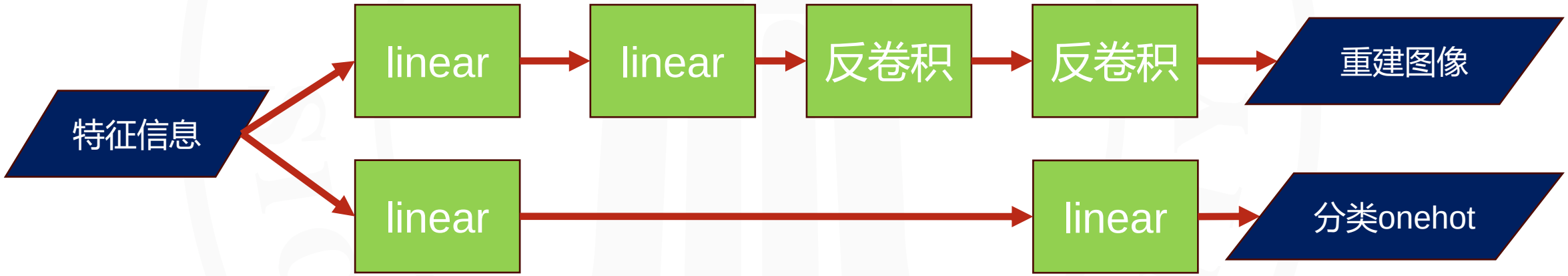

无标签数据
有标签数据
Encoder(特征提取)
80维的低维张量
Decoder
重建原图像
MSE
(输出，原图像)
Classifier
分类器
交叉熵
(输出，标签)

Encoder



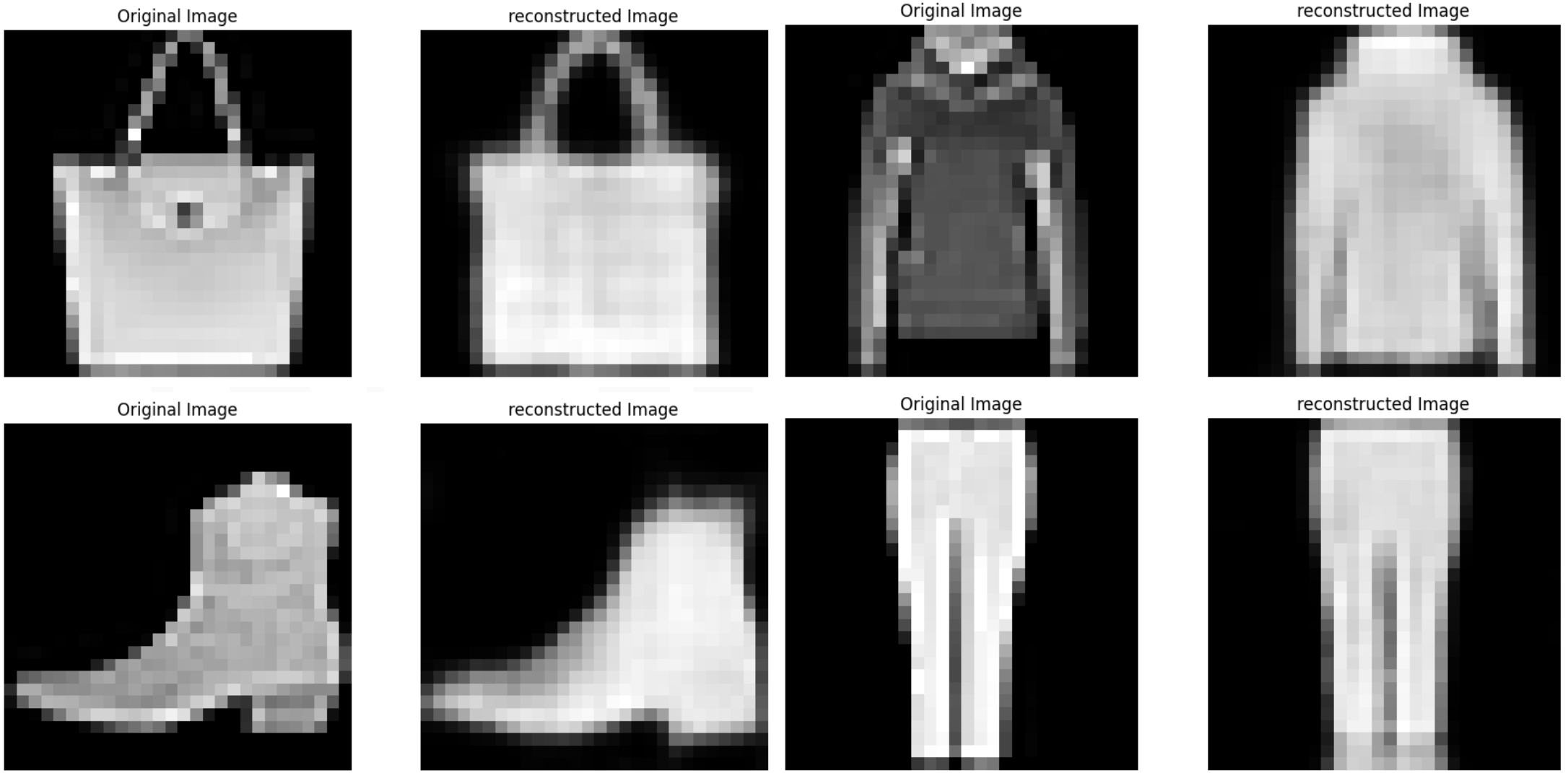
*：激活函数、池化、batchNorm、dropout未画出

Decoder & Classifier



*：激活函数、池化、batchNorm未画出

1 Reconstructed Image



其他优化方法

Batch Norm(标准化):

- **加速训练**：通过减少内部协变量偏移，使得网络可以使用更高的学习率，加速收敛。
- **正则化效果**：由于每个小批量数据的归一化，引入了噪声，类似于数据增强，有一定的正则化效果，可以减少过拟合。
- **减少对初始化的依赖**：批量归一化使得网络对参数的初始值不那么敏感。

Dropout(随机丢包):

- **减少过拟合**：通过随机“丢弃”一些神经元，强制网络学习更加鲁棒的特征，减少了模型对特定训练样本的依赖。
- **模型集成**：Dropout可以看作是在训练时对大量不同网络结构的集成，而测试时则相当于对这些网络的平均。

1

```
102  class Net3(nn.Module):
103      def __init__(self):
104          super().__init__()
105          hidden_size = 80
106          self.conv1 = nn.Conv2d(1, 32, 3, padding=1)
107          self.bn1 = nn.BatchNorm2d(32)
108          self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
109          self.bn2 = nn.BatchNorm2d(64)
110          self.conv12 = nn.Conv2d(1, 4, 3, padding=1)
111          self.bn12 = nn.BatchNorm2d(4)
112          self.maxpool = nn.MaxPool2d(2, 2)
113          self.linear = nn.Linear(hidden_size, 128)
114          self.linear2 = nn.Linear(128, 10)
115          self.encoded_linear = nn.Linear(64 * 7 * 7 + 4 * 14 * 14, hidden_size)
116          self.decoder = nn.Sequential(
117              nn.Linear(hidden_size, 32 * 7 * 7),
118              nn.ReLU(),
119              nn.Linear(32 * 7 * 7, 64 * 7 * 7),
120              nn.ReLU(),
121              nn.Unflatten(1, (64, 7, 7)),
122              nn.ConvTranspose2d(64, 32, 3, stride=2, output_padding=1, padding=1),
123              nn.BatchNorm2d(32),
124              nn.ReLU(),
125              nn.ConvTranspose2d(32, 1, 3, stride=2, output_padding=1, padding=1),
126              nn.Sigmoid()
127          )
128          self.dropout = nn.Dropout(0.2)
129
```

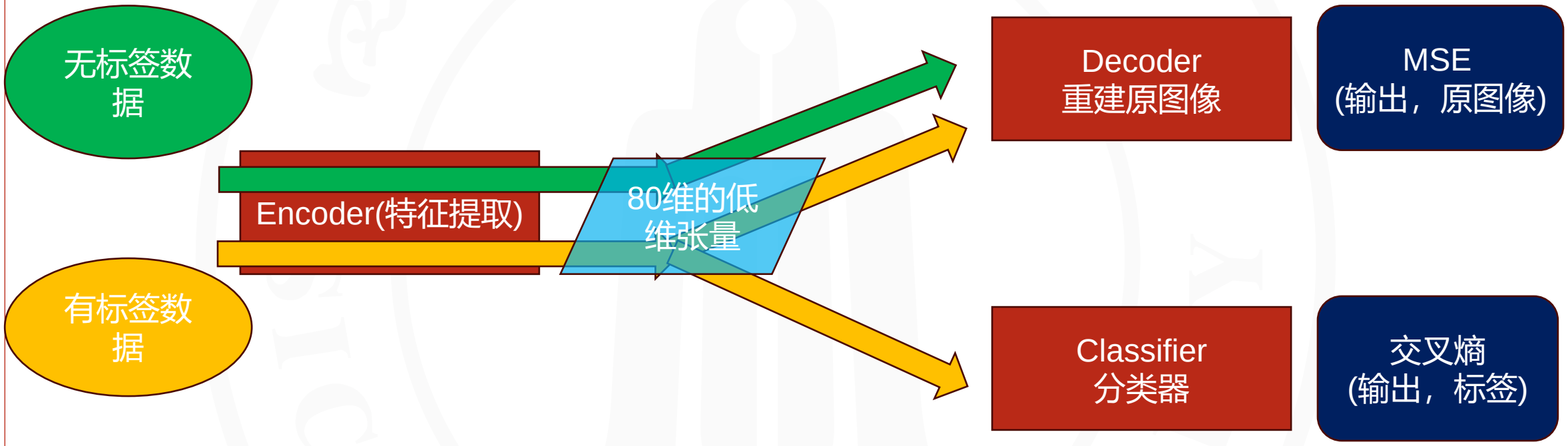
```
130     def encode(self, x: torch.Tensor):
131         x = x.view(-1, 1, 28, 28)
132         x2 = self.conv12(x)
133         x2 = self.bn12(x2)
134         x2 = F.relu(x2)
135         x2 = self.maxpool(x2)  # x2 shape is batch
136         x2 = self.dropout(x2)
137         x2 = x2.flatten(1)
138 
139         x = self.conv1(x)
140         x = self.bn1(x)
141         x = F.relu(x)
142         x = self.dropout(x)
143         x = self.maxpool(x)
144 
145         x = self.conv2(x)
146         x = self.bn2(x)
147         x = F.relu(x)
148         x = self.maxpool(x)
149 
150         encoded = x.flatten(1)
151         encoded = torch.cat((encoded, x2), dim=1)
152         encoded = self.encoded_linear(encoded)
153         encoded = F.relu(encoded)
154         return encoded
155 
156     def decode(self, encoded: torch.Tensor):
157         decoded = self.decoder(encoded)
158         return decoded.flatten(1)
159 
```

```
160     def classify(self, encoded: torch.Tensor):
161         x = self.dropout(encoded)
162         x = self.linear(x)
163         x = F.relu(x)
164         x = self.dropout(x)
165         x = self.linear2(x)
166         return x
167 
168     def forward(self, x: torch.Tensor):
169         encoded = self.encode(x)
170         decoded = self.decode(encoded)
171         classification = self.classify(encoded)
172         return classification, decoded
173 
```

3

训练方法

多任务学习


无标签数据
有标签数据
Encoder(特征提取)
80维的低维张量
Decoder
重建原图像
MSE
(输出，原图像)
Classifier
分类器
交叉熵
(输出，标签)

```python
import torch.nn.functional as F

# Initialize the model
model = Net3().to(device)

# Define loss function and optimizer
criterion_reconstruct = torch.nn.MSELoss()  # Use MSELoss for reconstruction
criterion_class = torch.nn.CrossEntropyLoss()  # Use CrossEntropyLoss for classification
optimizer = optim.Adam(model.parameters())

# Training loop
def train_semi_supervised(model=model, train_loader=enhance_loader_1, unlabeled_loader=unlabeled_loader, epochs=10):
    model.train()
    for epoch in range(epochs):
        train_corrects = 0
        for ((labeled_data, labels), unlabeled_data) in zip(train_loader, unlabeled_loader):
            # Forward pass
            optimizer.zero_grad()
            labeled_class,decoded = model(labeled_data)
            reconstruction_loss=criterion_reconstruct(decoded,labeled_data)
            class_loss = criterion_class(labeled_class, labels)
            total_loss=reconstruction_loss+class_loss
            preds = labeled_class.argmax(1).detach()
            train_corrects += (preds==labels.data).sum()
            # Backward pass
            total_loss.backward()
            optimizer.step()

            # unlabeled
            _,decoded = model(unlabeled_data)
            reconstruction_loss=criterion_reconstruct(decoded,unlabeled_data)
            reconstruction_loss.backward()
            optimizer.step()
        acc=(train_corrects / len(train_loader.dataset)).item()
        print(f'Epoch {epoch+1}/{epochs},total Loss: {total_loss.item():.4f},construction Loss: {reconstruction_loss.item():.4f},class acc:
```

```
40    def test(test_loader=test_loader, net=model,isOffset=True):
41        # set model to eval mode
42        net.eval()
43        corrects = 0
44        with torch.no_grad():
45            for inputs, labels in test_loader:
46                outputs = F.softmax(net(inputs)[0],dim=-1)
47                if isOffset:
48                    for i in transform_offset(inputs):
49                        outputs+=F.softmax(net(i)[0],dim=-1)
50                preds = outputs.argmax(1).detach()
51                corrects += (preds==labels.data).sum()
52        return (corrects / len(test_loader.dataset)).item()
53
```

```python
highest_acc=0
highest_acc_data=None
def update_final_output():
    global highest_acc,highest_acc_data
    model.eval()
    test_acc=test()
    if test_acc>highest_acc:
        print("new higher test_acc",test_acc)
    else:
        print("test_acc not higher",test_acc,"highest_acc",highest_acc)
        return
    with torch.no_grad():
        x=torch.stack([final_dataset[i] for i in range(len(final_dataset))])
        y=F.softmax(model(x)[0],dim=-1)
        for i in transform_offset(x):
            y+=F.softmax(model(i)[0],dim=-1)
        y=y.argmax(1).detach().cpu().unsqueeze(0)
        y=y.numpy()
    highest_acc=test_acc
    highest_acc_data=y
```

```
90  def save_final():
91      import os
92      os.makedirs("out3",exist_ok=True)
93      print("shape",highest_acc_data.shape)
94      np.save(f"out3/output_{highest_acc}.npy",highest_acc_data)
95
96  import random
97  loaders = [enhance_loader_1, enhance_loader_2, train_loader, train_loader]
98
99  def train_infty():
100     while True:
101         loader = random.choice(loaders)
102         train_semi_supervised(train_loader=loader, epochs=1)
103         update_final_output()
104
105 if __name__ == "__main__":
106     train_infty()
```

```
train_infty
```

[11] ✓ 0.0s

```
<function __main__.train_infty()>
```

```
train_infty()
```

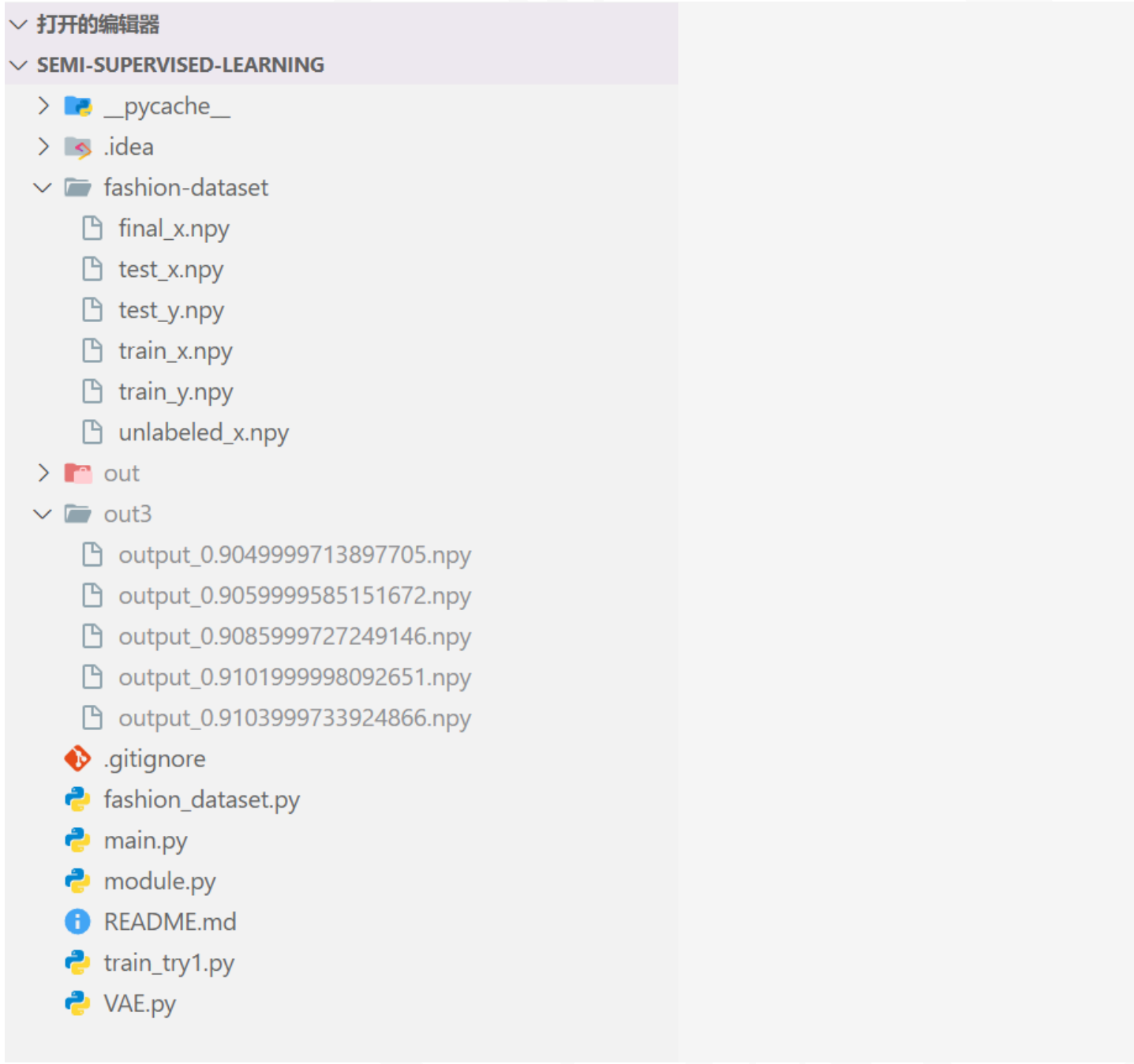
[12] 7m 2.5s

```
Epoch 1/1,total Loss: 0.1095,construction Loss: 0.0220,class acc: 0.9646
test_acc not higher 0.9023999571800232 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.2174,construction Loss: 0.0175,class acc: 0.9820
test_acc not higher 0.8935999870300293 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.1839,construction Loss: 0.0214,class acc: 0.9686
test_acc not higher 0.8953999876976013 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.1508,construction Loss: 0.0184,class acc: 0.9708
test_acc not higher 0.8917999863624573 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0182,construction Loss: 0.0197,class acc: 0.9702
test_acc not higher 0.8939999938011169 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.2152,construction Loss: 0.0252,class acc: 0.9661
test_acc not higher 0.8953999876976013 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0387,construction Loss: 0.0285,class acc: 0.9702
test_acc not higher 0.8675999641418457 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0161,construction Loss: 0.0203,class acc: 0.9775
test_acc not higher 0.8865999579429626 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0151,construction Loss: 0.0187,class acc: 0.9826
test_acc not higher 0.8876000046730042 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.3664,construction Loss: 0.0210,class acc: 0.9795
test_acc not higher 0.8889999985694885 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0362,construction Loss: 0.0167,class acc: 0.9739
test_acc not higher 0.8941999673843384 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.7535,construction Loss: 0.0177,class acc: 0.9787
test_acc not higher 0.8962000012397766 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.3082,construction Loss: 0.0229,class acc: 0.9741
test_acc not higher 0.8779999613761902 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0254,construction Loss: 0.0201,class acc: 0.9657
test_acc not higher 0.8935999870300293 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0901,construction Loss: 0.0213,class acc: 0.9728
test_acc not higher 0.896399974822998 highest_acc 0.9103999733924866
```

按 Enter 执行。

```
test_acc not higher 0.9016000032424927 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0186,construction Loss: 0.0161,class acc: 0.9886
test_acc not higher 0.9043999910354614 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0175,construction Loss: 0.0179,class acc: 0.9898
test_acc not higher 0.9017999768257141 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0222,construction Loss: 0.0165,class acc: 0.9929
test_acc not higher 0.9003999829292297 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0123,construction Loss: 0.0165,class acc: 0.9943
test_acc not higher 0.9057999849319458 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0612,construction Loss: 0.0170,class acc: 0.9824
test_acc not higher 0.894599974155426 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0249,construction Loss: 0.0191,class acc: 0.9807
test_acc not higher 0.8967999815940857 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.6460,construction Loss: 0.0193,class acc: 0.9896
test_acc not higher 0.9041999578475952 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.2330,construction Loss: 0.0239,class acc: 0.9833
test_acc not higher 0.8969999551773071 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.1225,construction Loss: 0.0183,class acc: 0.9937
test_acc not higher 0.8973999619483948 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0211,construction Loss: 0.0153,class acc: 0.9804
test_acc not higher 0.9059999585151672 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0576,construction Loss: 0.0251,class acc: 0.9848
test_acc not higher 0.8967999815940857 highest_acc 0.9101999998092651
Epoch 1/1,total Loss: 0.0145,construction Loss: 0.0162,class acc: 0.9925
hew higher test_acc 0.9103999733924866
Epoch 1/1,total Loss: 2.0172,construction Loss: 0.0246,class acc: 0.9923
test_acc not higher 0.902999997138977 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0154,construction Loss: 0.0186,class acc: 0.9852
test_acc not higher 0.8905999660491943 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0550,construction Loss: 0.0170,class acc: 0.9912
test_acc not higher 0.9016000032424927 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0138,construction Loss: 0.0130,class acc: 0.9882
test_acc not higher 0.9021999835968018 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0145,construction Loss: 0.0166,class acc: 0.9937
test_acc not higher 0.9003999829292297 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.2650,construction Loss: 0.0166,class acc: 0.9893
test_acc not higher 0.8995999693870544 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0263,construction Loss: 0.0177,class acc: 0.9866
test_acc not higher 0.892799973487854 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0313,construction Loss: 0.0184,class acc: 0.9882
test_acc not higher 0.8977999687194824 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0111,construction Loss: 0.0170,class acc: 0.9904
test_acc not higher 0.9009999632835388 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0203,construction Loss: 0.0155,class acc: 0.9777
test_acc not higher 0.896399974822998 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.1616,construction Loss: 0.0188,class acc: 0.9793
test_acc not higher 0.8939999938011169 highest_acc 0.9103999733924866
Epoch 1/1,total Loss: 0.0100,construction Loss: 0.0168,class acc: 0.9932
```

3


打开的编辑器
SEMI-SUPERVISED-LEARNING
__pycache__
.idea
fashion-dataset
final_x.npy
test_x.npy
test_y.npy
train_x.npy
train_y.npy
unlabeled_x.npy
out
out3
output_0.9049999713897705.npy
output_0.9059999585151672.npy
output_0.9085999727249146.npy
output_0.9101999998092651.npy
output_0.9103999733924866.npy
.gitignore
fashion_dataset.py
main.py
module.py
README.md
train_try1.py
VAE.py

谢谢
敬请批评指正

四川大學
SICHUAN UNIVERSITY

Github：The-Brotherhood-of-SCU/semi-supervised-learning