In [ ]:
```
import torch
import torchvision
import torchvision.transforms as transforms
```

Download dữ liệu chữ số viết tay MNIST

In [ ]:
```
# Chuẩn bị dữ liệu
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
# Chuyển đổi sang định dạng float32.
x_train, x_test = np.array(x_train, np.float32), np.array(x_test, np.float3
# Chuẩn hóa ảnh từ from [0, 255] to [0, 1].
x_train, x_test = x_train / 255., x_test / 255.
x_train, x_test, y_train, y_test = torch.from_numpy(x_train), torch.from_nu
```

In [ ]:
```
batch_size = 16
trainloader = []
for (i,j) in zip(None, None):
    trainloader.append([i,j])
trainloader = torch.utils.data.DataLoader(None, shuffle=True, batch_size=No

testloader = []
for (i,j) in zip(None, None):
    testloader.append([i,j])
testloader = torch.utils.data.DataLoader(None, shuffle=True, batch_size=Non
```

In [ ]:
```
num_features = 784
n_hidden_1 = 512
n_hidden_2 = 128
n_hidden_3 = 32
num_classes = 10
```

Sử dụng các tham số ở trên để xây dựng mô hình

In [ ]:
```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        """
        code
        """

    def forward(self, x):
        """
        code
        """

net = Net()
```

In [ ]:
```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(None, lr=0.001, momentum=0.9)
```

In [ ]:
```
for epoch in range(1):

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # load input và labels
        inputs, labels = None

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(None)
        # print(labels.shape)
        # print(outputs.shape)
        loss = criterion(None, None)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:    # print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

```
[1,  2000] loss: 0.598
Finished Training
```

In [ ]:
```
correct = 0
total = 0
# do đang thực hiện việc dự đoán nên ko cần tính đạo hàm
with torch.no_grad():
    for data in testloader:
        inputs, labels = None
        # chạy hàm dự đoán
        outputs = net(None)
        # the class với giá trị xác suất cao nhất là đâu ra dự đoán
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == None).sum().item()

print('Accuracy of the network on the 10000 test images: %d %%' % (
    100 * correct / total))
```

Save and load model

Trình bày 1 trong các các lưu model và load model trong PyTorch

1. Lưu model

```
torch.save(model.state_dict(), PATH)
```

   trong đó PATH là đường dẫn tự định nghĩa
2. Load model

- Trước tiên phải định nghĩa model trước. Model được định nghĩa phải giống hệt với model đã được lưu lại. Như ví dụ trong bài này, thì sẽ thực hiện như sau:

```
model = Net()
```

- Load trọng số đã được học vào mô hình

```
model.load_state_dict(torch.load(PATH))
# vô hiệu hóa các layer như Dropout hay BatchNorm
model.eval()
```

3. Có thể tham khảo thêm các phương pháp lưu và load model tại: https://pytorch.org/tutorials/beginner/basics/saveloadrun_tutorial.html (https://pytorch.org/tutorials/beginner/basics/saveloadrun_tutorial.html)

In [ ]: