LAB 6: XÂY DỰNG MẠNG NƠ-RON SÂU

Yêu cầu:

- Sinh viên tự thực hiện xây dựng mạng nơ-ron sâu cho bài toán nhận dạng quần áo giày dép thời trang với bộ dữ liệu FASHION-MNIST tương tự Lab 5.

Câu 1:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np

# 1️⃣ Kiểm tra nếu GPU có sẵn
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Câu 2:

```
# 2️⃣ Tải dữ liệu FashionMNIST và chuẩn hóa
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,),
(0.5,))])

trainset = torchvision.datasets.FashionMNIST(root='./data', train=True, download=True,
transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = torchvision.datasets.FashionMNIST(root='./data', train=False, download=True,
transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=False)

# Danh sách tên lớp
classes = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
           'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Câu 3:

```
# 3️⃣ Xây dựng mô hình mạng nơ-ron
class FashionMNIST_NN(nn.Module):
    def __init__(self):
        super(FashionMNIST_NN, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(28 * 28, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
```

```
            nn.Linear(256, 10)
        )

    def forward(self, x):
        x = x.view(-1, 28 * 28)  # Chuyển ảnh 28x28 thành vector 1D
        return self.model(x)

# Khởi tạo mô hình và chuyển sang GPU nếu có
model = FashionMNIST_NN().to(device)
```

Câu 4:

```
# 4️⃣ Định nghĩa hàm mất mát và trình tối ưu hóa
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
```

Câu 5:

```
# 5️⃣ Huấn luyện mô hình
num_epochs = 5
train_losses = []
train_accuracies = []

for epoch in range(num_epochs):
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in trainloader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses.append(running_loss / len(trainloader))
    train_accuracies.append(correct / total)

    print(f'Epoch {epoch+1}/{num_epochs}, Loss: {running_loss / len(trainloader):.4f},
Accuracy: {correct / total:.4f}')

print('✅ Huấn luyện hoàn tất!')
```

Câu 6:

```
# 6️⃣ Đánh giá mô hình
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for images, labels in testloader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = 100 * correct / total
print(f'🎯 Độ chính xác trên tập kiểm tra: {accuracy:.2f}%')
```

Câu 7:

```
# 7️⃣ Vẽ đồ thị Loss và Accuracy
epochs = range(1, num_epochs + 1)

plt.figure(figsize=(12, 5))

# Vẽ loss
plt.subplot(1, 2, 1)
plt.plot(epochs, train_losses, label='Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()

# Vẽ accuracy
plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracies, label='Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training Accuracy')
plt.legend()

# Lưu hình ảnh đồ thị
plt.savefig("lab6.png")
print("📊 Đã lưu ảnh quá trình huấn luyện vào 'lab6.png'")
```

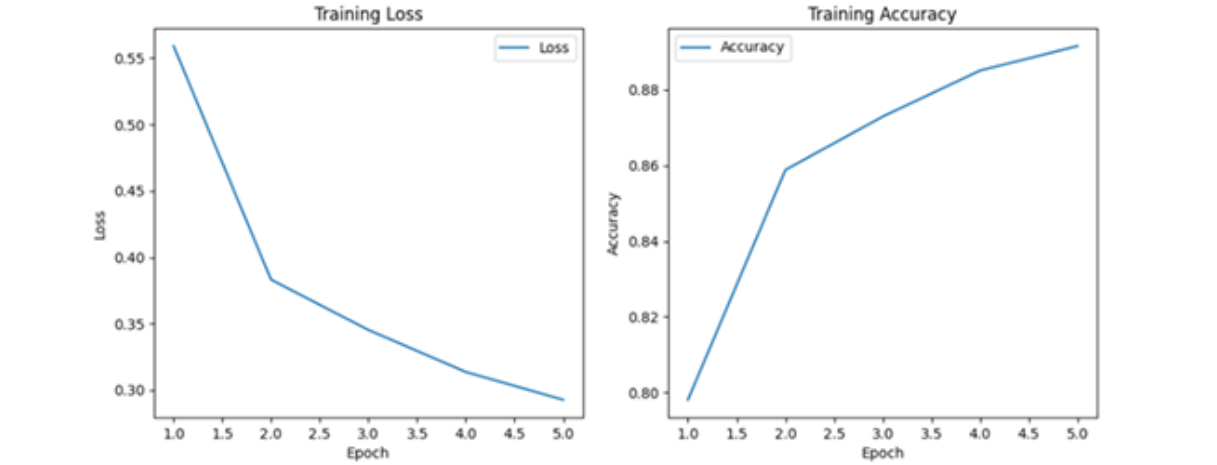
Câu 8:

```
# 8️⃣ Hiển thị một số hình ảnh dự đoán
def imshow(img):
    img = img / 2 + 0.5  # Unnormalize
```

```
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.axis('off')  # Tắt lưới
    plt.show()

# Lấy một batch từ tập kiểm tra
dataiter = iter(testloader)
images, labels = next(dataiter)

# Hiển thị hình ảnh thực tế
imshow(torchvision.utils.make_grid(images))

# Hiển thị nhãn thực tế và dự đoán
outputs = model(images.to(device))
_, predicted = torch.max(outputs, 1)

print('👉 Thực tế: ', ' '.join(classes[labels[j]] for j in range(8)))
print('🤖 Dự đoán: ', ' '.join(classes[predicted[j]] for j in range(8)))

# Lưu ảnh dự đoán
plt.figure(figsize=(8, 4))
plt.imshow(np.transpose(torchvision.utils.make_grid(images).numpy(), (1, 2, 0)))
plt.axis('off')
plt.title('Predictions')
plt.savefig("predictions.png")
print("☑️ Đã lưu ảnh dự đoán vào 'predictions.png'")
```

Kết quả:


```
Epoch 1/5, Loss: 0.5591, Accuracy: 0.7981
Epoch 2/5, Loss: 0.3832, Accuracy: 0.8588
Epoch 3/5, Loss: 0.3453, Accuracy: 0.8729
Epoch 4/5, Loss: 0.3136, Accuracy: 0.8851
Epoch 5/5, Loss: 0.2927, Accuracy: 0.8916
✅ Huấn luyện hoàn tất!
🎯 Độ chính xác trên tập kiểm tra: 87.31%
📊 Đã lưu ảnh quá trình huấn luyện vào 'lab6.png'
👉 Thực tế:  Ankle boot Pullover Trouser Trouser Shirt Trouser Coat Shirt
🤖 Dự đoán:  Ankle boot Pullover Trouser Trouser Shirt Trouser Coat Shirt
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-1.0..1.0].
✅ Đã lưu ảnh dự đoán vào 'predictions.png'
```




```
Epoch 1/5, Loss: 0.5591, Accuracy: 0.7981
Epoch 2/5, Loss: 0.3832, Accuracy: 0.8588
Epoch 3/5, Loss: 0.3453, Accuracy: 0.8729
Epoch 4/5, Loss: 0.3136, Accuracy: 0.8851
Epoch 5/5, Loss: 0.2927, Accuracy: 0.8916
✅ Huấn luyện hoàn tất!
🎯 Độ chính xác trên tập kiểm tra: 87.31%
📊 Đã lưu ảnh quá trình huấn luyện vào 'lab6.png'
🏷️ Thực tế:  Ankle boot Pullover Trouser Trouser Shirt Trouser Coat Shirt
🤖 Dự đoán:  Ankle boot Pullover Trouser Trouser Shirt Trouser Coat Shirt
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-1.0..1.0].
✅ Đã lưu ảnh dự đoán vào 'predictions.png'
```

- Sinh viên chọn 1 đề tài tương tự Lab 5, Lab 6 để thực hiện đồ án môn học.

  Đề tài: Phân loại ảnh động vật

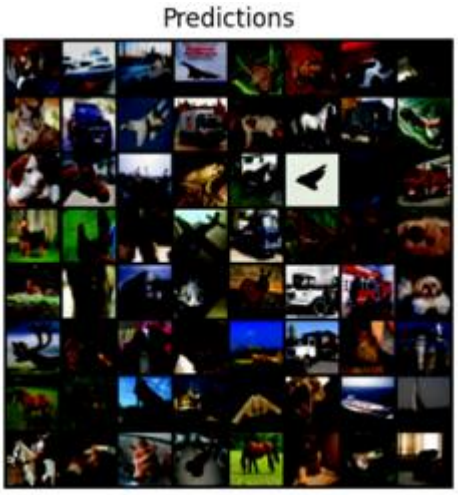
```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np

# Kiểm tra thiết bị (GPU nếu có)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Chuẩn bị dữ liệu CIFAR-10
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=False)

# Danh sách nhãn lớp
classes = ['Plane', 'Car', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship',
'Truck']

# Xây dựng mô hình CNN
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2, 2)
        )
```

```
        self.fc_layers = nn.Sequential(
            nn.Linear(64 * 8 * 8, 512), nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.conv_layers(x)
        x = x.view(x.size(0), -1)
        return self.fc_layers(x)

# Khởi tạo mô hình
model = CNN().to(device)

# Định nghĩa hàm mất mát & tối ưu hóa
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Huấn luyện mô hình
num_epochs = 5
train_losses, train_accuracies = [], []

for epoch in range(num_epochs):
    running_loss, correct, total = 0.0, 0, 0

    for images, labels in trainloader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses.append(running_loss / len(trainloader))
    train_accuracies.append(correct / total)

    print(f'Epoch {epoch+1}/{num_epochs}, Loss: {running_loss / len(trainloader):.4f},
Accuracy: {correct / total:.4f}')

print('Huấn luyện hoàn tất!')
```

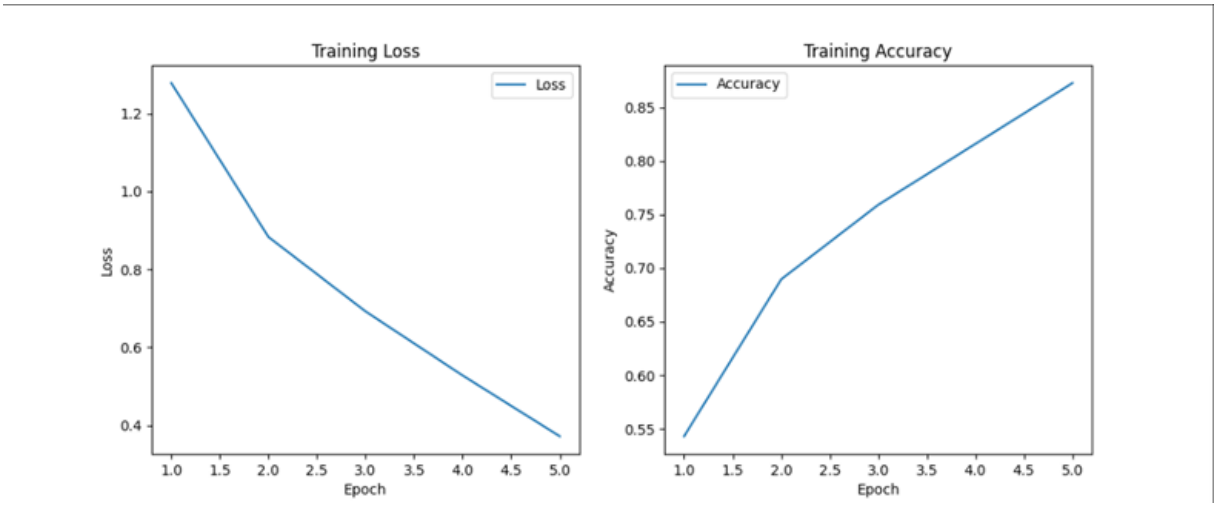
```python
# Đánh giá mô hình
model.eval()
correct, total = 0, 0

with torch.no_grad():
    for images, labels in testloader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = 100 * correct / total
print(f'Độ chính xác trên tập kiểm tra: {accuracy:.2f}%')

# Vẽ đồ thị Loss & Accuracy
epochs = range(1, num_epochs + 1)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, train_losses, label='Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracies, label='Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training Accuracy')
plt.legend()

plt.savefig("cifar10_training.png")
print("Đã lưu ảnh huấn luyện vào 'cifar10_training.png'")

# Hiển thị hình ảnh dự đoán
def imshow(img):
    img = img / 2 + 0.5  # Unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.axis('off')  # Tắt lưới
    plt.show()
```

```python
dataiter = iter(testloader)
images, labels = next(dataiter)

imshow(torchvision.utils.make_grid(images))

outputs = model(images.to(device))
_, predicted = torch.max(outputs, 1)

print('Thực tế: ', ' '.join(classes[labels[j]] for j in range(8)))
print('Dự đoán: ', ' '.join(classes[predicted[j]] for j in range(8)))

# Lưu ảnh dự đoán
plt.figure(figsize=(8, 4))
plt.imshow(np.transpose(torchvision.utils.make_grid(images).numpy(), (1, 2, 0)))
plt.axis('off')
plt.title('Predictions')
plt.savefig("cifar10_predictions.png")
print("Đã lưu ảnh dự đoán vào 'cifar10_predictions.png'")
```

Kết quả:




```
100.0%
Epoch 1/5, Loss: 1.2781, Accuracy: 0.5431
Epoch 2/5, Loss: 0.8831, Accuracy: 0.6895
Epoch 3/5, Loss: 0.6921, Accuracy: 0.7591
Epoch 4/5, Loss: 0.5277, Accuracy: 0.8159
Epoch 5/5, Loss: 0.3716, Accuracy: 0.8726
✅ Huấn luyện hoàn tất!
🎯 Độ chính xác trên tập kiểm tra: 73.01%
📊 Đã lưu ảnh huấn luyện vào 'cifar10_training.png'
🐾 Thực tế:  Cat Ship Ship Plane Frog Frog Car Frog
🤖 Dự đoán:  Cat Ship Ship Plane Cat Frog Truck Frog
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-1.0..1.0].
✅ Đã lưu ảnh dự đoán vào 'cifar10_predictions.png'
```




```
100.0%
Epoch 1/5, Loss: 1.2781, Accuracy: 0.5431
Epoch 2/5, Loss: 0.8831, Accuracy: 0.6895
Epoch 3/5, Loss: 0.6921, Accuracy: 0.7591
Epoch 4/5, Loss: 0.5277, Accuracy: 0.8159
Epoch 5/5, Loss: 0.3716, Accuracy: 0.8726
✅ Huấn luyện hoàn tất!
🎯 Độ chính xác trên tập kiểm tra: 73.01%
📊 Đã lưu ảnh huấn luyện vào 'cifar10_training.png'
🐶 Thực tế:  Cat Ship Ship Plane Frog Frog Car Frog
🚚 Dự đoán:  Cat Ship Ship Plane Cat Frog Truck Frog
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-1.0..1.0].
✅ Đã lưu ảnh dự đoán vào 'cifar10_predictions.png'
```