HW06

202401833 신해솔

0. 실행 결과

```
INFO: Using device - cuda:6
INFO: Start Training
Epoch 1, Loss: 1.2421
Epoch 2, Loss: 0.5292
Epoch 3, Loss: 0.3485
Epoch 4, Loss: 0.2712
Epoch 5, Loss: 0.2333
Epoch 6, Loss: 0.2085
Epoch 7, Loss: 0.1870
Epoch 8, Loss: 0.1716
Epoch 9, Loss: 0.1585
Epoch 10, Loss: 0.1485
Epoch 11, Loss: 0.1392
Epoch 12, Loss: 0.1327
Epoch 13, Loss: 0.1255
Epoch 14, Loss: 0.1190
Epoch 15, Loss: 0.1148
INFO: Training Complete
Total: 10000, Correct: 9629, Accuracy: 96.29%
```

1. 필요 라이브러리 import 및 디바이스 설정

```python
import torch
import torch.nn as nn
from torch import optim
from torchvision import transforms, datasets
from torch.utils.data import DataLoader


DEVICE_NUM = 6
if torch.cuda.is_available():
    device = torch.device(f"cuda:{DEVICE_NUM}")
else:
    device = torch.device("cpu")
print(f"INFO: Using device - {device}")
```

- 필요 라이브러리 import

- 학과 서버에서 진행하여 특정 디바이스(cuda) 번호 선택

2. Config 설정

```
BATCH_SIZE = 256
PATCH_SIZE = 7
PATCH_INPUT_SIZE = 49 # 7 * 7
SEQUENCE_LENGTH = 16 # 28*28 / 7*7
HIDDEN_SIZE = 128
NUM_CLASSES = 10
NUM_EPOCHES = 15
LEARNING_RATE = 3e-4
```

- 사용될 상수 값들을 미리 설정

3. 데이터 형태 변환 및 데이터 설정

```
def image_to_patches(image_tensor):  1 usage  new *
    patches = image_tensor.unfold(1, PATCH_SIZE, PATCH_SIZE) # (1, 4, 28, 7)
    patches = patches.unfold(2, PATCH_SIZE, PATCH_SIZE) # (1, 4, 4, 7, 7)
    patches = patches.squeeze().contiguous() # (4, 4, 7, 7)
    patches = patches.view(SEQUENCE_LENGTH, PATCH_INPUT_SIZE) # (16, 49)
    return patches


transform_with_patches = transforms.Compose([
    transforms.ToTensor(),
    transforms.Lambda(image_to_patches)
])

train_dataset = datasets.MNIST(root="./data", train=True, download=True, transform=transform_with_patches)
test_dataset = datasets.MNIST(root="./data", train=False, download=True, transform=transform_with_patches)
train_loader = DataLoader(dataset=train_dataset, batch_size=BATCH_SIZE, shuffle=True)
test_loader = DataLoader(dataset=test_dataset, batch_size=BATCH_SIZE, shuffle=False)
```

- unfold 메소드를 사용하여 block 단위 (7*7)로 이미지를 쪼갬

- squeeze 메소드를 사용하여 크기가 1인 차원을 없애고, contiquous 메소드를 사용해 연속된 메모리 상에 위치시킴 (view 메소드를 사용하기 위함)

- view 메소드를 사용해 (16, 49) shape으로 최종적으로 변환

- transform.Compose class를 사용해 여러 변환을 묶음

- train, test data 별로 데이터셋, 데이터로더 설정

4. 모델 정의 및 하이퍼파라미터 설정

```python
class MNISTRnn(nn.Module):  1 usage  new *
    def __init__(self, input_size, hidden_size, output_size):  new *
        super().__init__()
        self.hidden_size = hidden_size
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):  new *
        batch_size = x.size(0)
        h0 = torch.zeros(1, batch_size, self.hidden_size).to(x.device)
        out, _ = self.rnn(x, h0)
        last_out = out[:, -1, :]
        out = self.linear(last_out)
        return out


model = MNISTRnn(input_size=PATCH_INPUT_SIZE, hidden_size=HIDDEN_SIZE, output_size=NUM_CLASSES)
model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
```

- rnn 모델을 정의

- criterion은 CE를, optimizer는 Adam을 사용

5. 학습 및 평가

```python
print("INFO: Start Training")
for epoch in range(NUM_EPOCHES):
    model.train()
    running_loss = 0.0
    for sequences, labels in train_loader:
        sequences = sequences.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(sequences)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print(f"Epoch {epoch+1}, Loss: {running_loss/len(train_loader):.4f}")
print("INFO: Training Complete")

model.eval()
with torch.no_grad():
    total_prediction = 0
    correct_prediction = 0
    for sequences, labels in test_loader:
        sequences = sequences.to(device)
        labels = labels.to(device)
        outputs = model(sequences)
        _, predicted = torch.max(outputs, dim=1)
        total_prediction += labels.size(0)
        correct_prediction += (predicted == labels).sum().item()
print(f"Total: {total_prediction}, Correct: {correct_prediction}, Accuracy: {correct_prediction/total_prediction:.2%}")
```

- 준비된 데이터와 모델로 학습 및 평가를 진행

- 데이터는 데이터셋을 설정할 때 변환하였으므로 추가 변환이 필요없음