데이터셋 분석 보고서

NURUL HUDA FAIZAH (케이디) 22012889

여기에 있는 모든 코드는 **fashion_mnist\static\data** 폴더에서 가져올 수 있으며, **Fmnist_dataset.ipynb** 파일 안에서 확인할 수 있습니다.

이 코드는 **Fashion MNIST** 데이터셋을 다운로드하고, 전처리하여 학습 및 테스트 데이터를 로드하는 작업을 수행합니다. • 데이터셋의 10 개 클래스에서 각 클래스의 첫 번째 이미지를 시각화하여 Fashion MNIST 에 포함된 다양한 의류 항목을 제공합니다.

1. **Fashion MNIST 데이터셋 다운로드 및 전처리**

- 슬라이드에서 참고한 코드를 실행하면 Fashion MNIST 데이터셋은 ((train-images, train-labels, t10k-images, t10k-labels) 다운로드 받을 수 있습니다. 다운로드 후 파일은 FashionMNIST\raw 디렉토리로 추출됩니다.
- MNIST 와 유사하게 28x28 픽셀 크기의 흑백 이미지가 60,000 개의 학습 세트와 10,000 개의 테스트 셋으로 구성되어 있습니다.

2. **데이터셋의 정보**

- **클래스 구성**: 총 10 개의 카테고리로 나뉘며 각 클래스는 T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot 로 구성됩니다.
- **클래스별 이미지 수**: 각 클래스는 동일한 수의 이미지를 포함하며, 학습 데이터에는 각 클래스당 6,000 개, 테스트 데이터에는 각 클래스당 1,000 개의 이미지가 포함되어 있습니다.
- **데이터 형식**: 각 이미지의 크기는 28x28 픽셀이며, 단일 채널을 가진 흑백 이미지로 각 픽셀은 0~255 의 정수 값으로 표현됩니다. 학습 데이터는 NumPy 배열 또는 PyTorch 텐서로 변환하여 모델 학습에 사용됩니다.

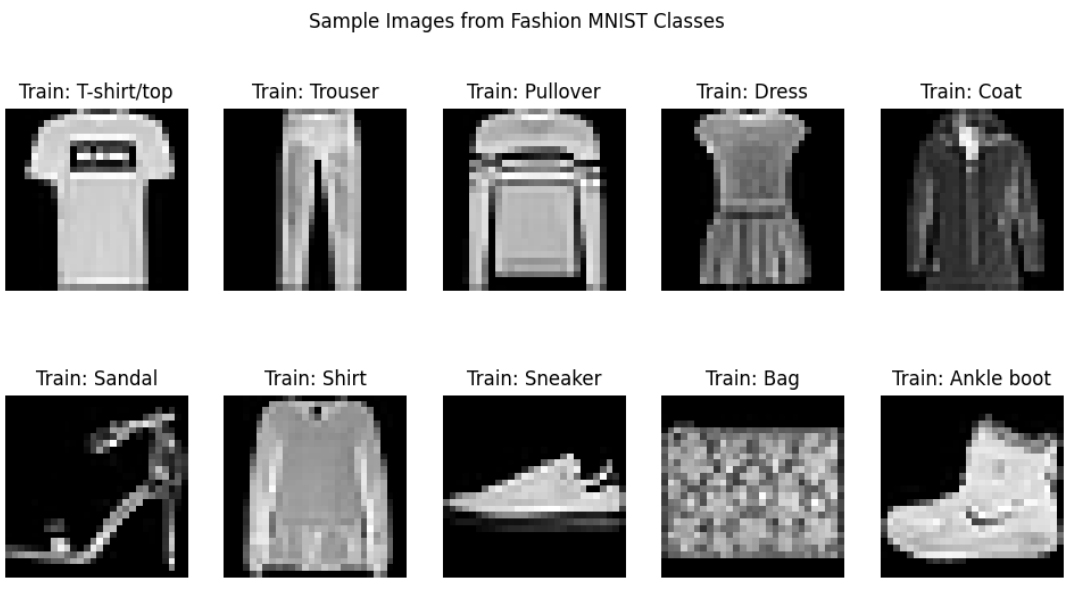
3. **데이터셋 크기 출력:** 학습 세트와 테스트 세트의 크기를 출력합니다.
    - **코드 작성**: 이 코드는 데이터셋을 로드하고 이미지 크기, 클래스 수, 클래스당 이미지 수 등의 구성 요소를 확인합니다.
    - **배치 크기**: 학습과 테스트 데이터를 64 개의 배치 단위로 로드하여 효율적으로 반복 처리할 수 있도록 합니다.
4. **이미지 시각화:**
    - 학습 데이터셋과 테스트 데이터셋에서 각 클래스별 첫 번째 이미지를 2x5 의 그래프 형태로 시각화합니다.
    - 각 이미지 아래에 해당 클래스 이름을 표시하여, Fashion MNIST 데이터셋에 포함된 의류 항목들을 시각적으로 확인할 수 있습니다.

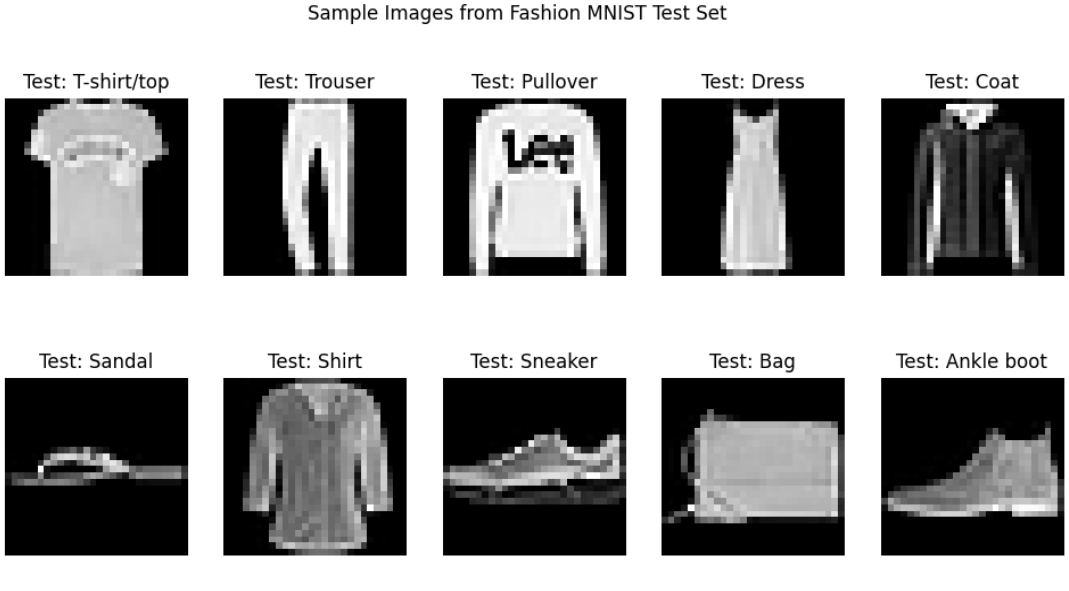
```
import torch

from torchvision import datasets, transforms
import matplotlib.pyplot as plt

# 데이터 전처리: 이미지를 텐서로 변환하고 정규화하기 위한 transform 정의
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))  # Adjusted for single-channel images
])

# 학습 데이터셋 다운로드
trainset = datasets.FashionMNIST('', download=True, train=True,
transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64,
shuffle=True)

# 테스트 데이터셋 다운로드
testset = datasets.FashionMNIST('', download=True, train=False,
transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=True)

print(f"Training set size: {len(trainset)} images")
print(f"Test set size: {len(testset)} images")
```

```python
# Plot graph for the trainset
fig, axes = plt.subplots(2, 5, figsize=(12, 6))
fig.suptitle("Sample Images from Fashion MNIST Classes")

# Define the class labels for Fashion MNIST
class_labels = [
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal',
    'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
]

# Visualize the first image from each class in the trainset
for i in range(10):
    # 각 클래스의 첫 번째 이미지 찾기
    first_idx = next(idx for idx, (_, label) in enumerate(trainset) if label
== i)
    image, label = trainset[first_idx]

    # 이미지 시각화
    ax = axes[i // 5, i % 5]
    ax.imshow(image.squeeze(), cmap='gray')  # 이미지에서 배치 차원 제거
    ax.set_title(f"Train: {class_labels[label]}")
    ax.axis('off')

plt.show()

# Plot graph for the testset
fig, axes = plt.subplots(2, 5, figsize=(12, 6))
fig.suptitle("Sample Images from Fashion MNIST Test Set")

# Visualize the first image from each class in the testset
for i in range(10):
    # 각 클래스의 첫 번째 이미지 찾기
    first_idx = next(idx for idx, (_, label) in enumerate(testset) if label ==
i)
    image, label = testset[first_idx]

    # 이미지 시각화
    ax = axes[i // 5, i % 5]
    ax.imshow(image.squeeze(), cmap='gray')  # 이미지에서 배치 차원 제거
    ax.set_title(f"Test: {class_labels[label]}")
    ax.axis('off')

plt.show()
```

5. **위 코드를 실행하면 다음과 같은 결과가 나옵니다:** 코드 실행 후 각 클래스당 이미지 수, 예시 이미지 시각화 결과 등을 캡처하여 보고서에 포함합니다.

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz to FashionMNIST\raw\train-images-idx3-ubyte.gz
100.0%
Extracting FashionMNIST\raw\train-images-idx3-ubyte.gz to FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz to FashionMNIST\raw\train-labels-idx1-ubyte.gz
100.0%
Extracting FashionMNIST\raw\train-labels-idx1-ubyte.gz to FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to FashionMNIST\raw\t10k-images-idx3-ubyte.gz
100.0%
Extracting FashionMNIST\raw\t10k-images-idx3-ubyte.gz to FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to FashionMNIST\raw\t10k-labels-idx1-ubyte.gz
100.0%
Extracting FashionMNIST\raw\t10k-labels-idx1-ubyte.gz to FashionMNIST\raw

Training set size: 60000 images
Test set size: 10000 images
```





이 코드는 Fashion MNIST 또는 유사한 28x28 픽셀 흑백 이미지 데이터셋을 학습하기 위한 간단한 인공 신경망을 정의하고 학습하는 과정을 보여줍니다. 각 부분을 단계별로 분석해보겠습니다.

1. **신경망 모델 정의 :**

   - SimpleNN 은 28x28 크기의 이미지를 입력으로 받아 10 개 클래스에 대한 예측을 출력하는 단순한 신경망 모델입니다.

   - 모델은 두 개의 완전 연결층과 ReLU 활성화 함수를 사용하여 이미지를 분류합니다.

2. **손실 함수 및 옵티마이저 설정** :

   - CrossEntropyLoss()는 모델이 출력하는 예측 값과 실제 레이블 간의 차이를 계산하여 손실을 평가합니다.

   - SGD 최적화 알고리즘은 학습률과 모멘텀을 사용하여 모델의 파라미터를 조정하며, 경사 하강법을 통해 손실을 최소화하도록 모델을 학습시킵니다.

3. **모델 훈련** : 모델 5 번의 에포크 동안 학습됩니다. 각 배치마다 순전파와 역전파가 수행되며, 손실을 최소화하기 위해 가중치가 업데이트됩니다. 매 에포크가 끝날 때마다 손실 값을 출력하여 훈련 상태를 모니터링할 수 있습니다.

```
from torch import nn, optim

# Define a simple neural network model (신경망 모델 정의)
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(28 * 28, 128)  # 28x28 images flattened into a
vector
        self.fc2 = nn.Linear(128, 10)  # Output layer with 10 classes

    def forward(self, x):
        x = x.view(-1, 28 * 28)  # Flatten the image
        x = torch.relu(self.fc1(x))  # ReLU activation for the first hidden
layer
        x = self.fc2(x)  # Output layer (logits)
        return x

# Initialize the model, loss function, and optimizer (#손실 함수 및 옵티마이저
설정)
model = SimpleNN()
```

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

# Train the model (모델 훈련)
num_epochs = 5
for epoch in range(num_epochs):
    model.train()  # Set the model to training mode
    running_loss = 0.0
    for images, labels in trainloader:
        optimizer.zero_grad()  # Zero the gradients
        outputs = model(images)  # Forward pass
        loss = criterion(outputs, labels)  # Compute the loss
        loss.backward()  # Backward pass
        optimizer.step()  # Update the weights
        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{num_epochs}], Loss:
{running_loss/len(trainloader):.4f}")
```

4. **위 코드를 실행하면 다음과 같은 결과가 나옵니다:**

   코드 실행 후 모델 훈련 중 첫 번째 epoch 에서의 손실(loss)을 나타냅니다.

```
Epoch [1/5], Loss: 0.5350
Epoch [2/5], Loss: 0.3910
Epoch [3/5], Loss: 0.3579
Epoch [4/5], Loss: 0.3309
Epoch [5/5], Loss: 0.3130
```

이 코드는 학습된 모델을 테스트 셋에서 평가하고 예측 결과를 시각화하는 과정입니다. 이 과정은 모델을 테스트하고 시각화하는 데 중요한 단계를 제공합니다.

1. **모델 평가 :** 훈련된 모델의 정확도 (accuracy)를 테스트 데이터셋으로 평가합니다.
2. **예측 결과 시각화 함수 정의 :** 시각화는 모델의 예측 성능을 직관적으로 파악할 수 있도록 해주며, 예측 오류가 발생한 이미지를 찾아볼 수 있는 유용한 방법입니다

```
# Evaluate the model on the test set (모델 평가)
model.eval()  # Set the model to evaluation mode
correct = 0
total = 0

predictions = []
true_labels = []
images_for_visualization = []  # Store the images from the batch for
visualization

# No need to calculate gradients during evaluation
with torch.no_grad():
    for images, labels in testloader:
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)  # Get the predicted class
        total += labels.size(0)  # Total number of images
        correct += (predicted == labels).sum().item()  # Correct predictions
        predictions.extend(predicted.numpy())  # Store predictions
        true_labels.extend(labels.numpy())  # Store true labels
        images_for_visualization.extend(images.numpy())  # Store images for
visualization

accuracy = 100 * correct / total  # Calculate the accuracy percentage
print(f"Test Accuracy: {accuracy:.2f}%")

# Visualization function for predicted images with true and predicted labels
#(예측된 이미지와 레이블 시각화)
def visualize_predictions(images, true_labels, predictions, num=10):
    plt.figure(figsize=(12, 6))
    for i in range(num):
        plt.subplot(2, 5, i + 1)
        plt.imshow(images[i].squeeze(), cmap='gray')  # Display the image
        # Display true and predicted labels under the image
        plt.title(f'True: {class_labels[true_labels[i]]}\nPred:
{class_labels[predictions[i]]}')
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

```
# Visualize predictions from the images stored for visualization
visualize_predictions(images_for_visualization, true_labels, predictions,
num=10)
```

3. **위 코드를 실행하면 다음과 같은 결과가 나옵니다:**

   실행된 코드에서의 결과는 정확도와 예측된 이미지를 표시하는 내용을 담고 있습니다. 각 이미지에 대해 실제 라벨과 예측된 라벨을 표시하는 방식입니다.