```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
import matplotlib.pyplot as plt

# データセットの準備
# Fashion-MNISTデータセットをダウンロードして前処理
training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor(),
)

test_data = datasets.FashionMNIST(
    root="data",
    train=False,
    download=True,
    transform=ToTensor(),
)

# データローダーの設定
batch_size = 64
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

# クラスラベルの定義
classes = [
    "Tシャツ/トップ", "ズボン", "プルオーバー", "ドレス", "コート",
    "サンダル", "シャツ", "スニーカー", "バッグ", "アンクルブーツ"
]

# デバイスの設定 (GPU/MPS/CPU)
device = "cuda" if torch.cuda.is_available() else "mps" if torch.backends.mps.is_available() else "cpu"

# ニューラルネットワークモデルの定義
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

# 学習用の関数定義
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)
        pred = model(X)
        loss = loss_fn(pred, y)

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), (batch + 1) * len(X)
            print(f"損失: {loss:>7f}  [{current:>5d}/{size:>5d}]")

# テスト用の関数定義
def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
```

```
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
    print(f"テスト結果: \n 精度: {(100*correct):>0.1f}%, 平均損失: {test_loss:>8f} \n")
```

```
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 data/FashionMNIST/raw/train-images-i
100%|████████████████████| 26.4M/26.4M [00:03<00:00, 7.33MB/s]
Extracting data/FashionMNIST/raw/train-images-idx3-ubyte.gz to data/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 data/FashionMNIST/raw/train-labels-i
100%|████████████████████| 29.5k/29.5k [00:00<00:00, 125kB/s]
Extracting data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to data/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 data/FashionMNIST/raw/t10k-images-idx
100%|████████████████████| 4.42M/4.42M [00:01<00:00, 2.21MB/s]
Extracting data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to data/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 data/FashionMNIST/raw/t10k-labels-idx
100%|████████████████████| 5.15k/5.15k [00:00<00:00, 19.1MB/s]Extracting data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to data/FashionM
```

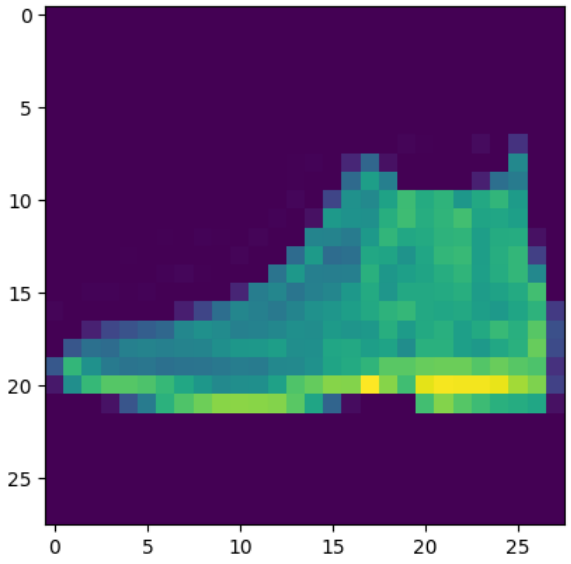
```
# モデルのインスタンス化
model = NeuralNetwork().to(device)

# 損失関数とオプティマイザの設定
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)

# 学習の実行
epochs = 5
for t in range(epochs):
    print(f"エポック {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
```

```
損失: 0.606996  [   64/60000]
損失: 0.700258  [ 6464/60000]
損失: 0.469797  [12864/60000]
損失: 0.764616  [19264/60000]
損失: 0.660035  [25664/60000]
損失: 0.637179  [32064/60000]
損失: 0.621846  [38464/60000]
損失: 0.655116  [44864/60000]
損失: 0.674920  [51264/60000]
損失: 0.611851  [57664/60000]
テスト結果:
 精度: 81.2%, 平均損失: 0.555209

エポック 3
-------------------------------
損失: 0.487262  [   64/60000]
損失: 0.566450  [ 6464/60000]
```

```
損失: 0.524899  [25664/60000]
損失: 0.521741  [32064/60000]
損失: 0.470988  [38464/60000]
損失: 0.585637  [44864/60000]
損失: 0.591835  [51264/60000]
損失: 0.469559  [57664/60000]
テスト結果:
 精度: 83.4%, 平均損失: 0.470676

エポック 5
-------------------------------
損失: 0.394976  [   64/60000]
損失: 0.455883  [ 6464/60000]
損失: 0.285121  [12864/60000]
損失: 0.542552  [19264/60000]
損失: 0.484817  [25664/60000]
損失: 0.494773  [32064/60000]
損失: 0.431160  [38464/60000]
損失: 0.560612  [44864/60000]
損失: 0.560976  [51264/60000]
損失: 0.435021  [57664/60000]
テスト結果:
 精度: 84.0%, 平均損失: 0.449186
```

```
test_iter = iter(test_dataloader)
testX, testy = next(test_iter)
pred = model(testX.to(device))
plt.imshow(testX[0].numpy().reshape(28, 28))
print('True      Label:', classes[testy[0].item()])
print('Estimated Label:', classes[pred.argmax(1)[0]])
```

```
True      Label: アンクルブーツ
Estimated Label: アンクルブーツ
```




```
# モデルの保存
torch.save(model.state_dict(), "model.pth")

# モデルの読み込み
model = NeuralNetwork().to(device)
model.load_state_dict(torch.load("model.pth", weights_only=True))
```