Rizqy Asyraff Athallah

1103210158

Week 12

1) Memuat Dataset

```
# Load dataset
file_path = '/content/drive/MyDrive/week 12/fashion-mnist_test.csv'
data = pd.read_csv(file_path)
```

2) Menyiapkan Dataset

```
# Prepare data
X = data.iloc[:, 1:].values.reshape(-1, 1, 28, 28).astype(np.float32) / 255.0
y = data.iloc[:, 0].values
```

- **X**: Data fitur diambil dari kolom ke-1 sampai akhir, direstrukturisasi menjadi tensor 4D berukuran (jumlah_data, 1, 28, 28). Data ini kemudian dinormalisasi ke rentang [0, 1].
- **y**: Label kelas dari dataset diambil dari kolom pertama.

3) Split Data

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Data dibagi menjadi 80% untuk pelatihan (X_train, y_train) dan 20% untuk pengujian (X_test, y_test).

4) Konversi ke Pytorch Dataset

```
# Convert to PyTorch tensors
train_dataset = TensorDataset(torch.tensor(X_train), torch.tensor(y_train, dtype=torch.long))
test_dataset = TensorDataset(torch.tensor(X_test), torch.tensor(y_test, dtype=torch.long))

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

- Dataset dikonversi ke format TensorDataset yang sesuai untuk digunakan di PyTorch.

5) Klasifikasi

```
# CNN Model
class CNN(nn.Module):
    def __init__(self, kernel_size, pooling_type):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=kernel_size, padding=kernel_size // 2)
        self.pool1 = nn.MaxPool2d(2) if pooling_type == 'max' else nn.AvgPool2d(2)
```

```
        self.conv2 = nn.Conv2d(32, 64, kernel_size=kernel_size, padding=kernel_size // 2)
        self.pool2 = nn.MaxPool2d(2) if pooling_type == 'max' else nn.AvgPool2d(2)
        self.fc1 = nn.Linear(64 * 7 * 7, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = self.pool1(x)
        x = torch.relu(self.conv2(x))
        x = self.pool2(x)
        x = x.view(-1, 64 * 7 * 7)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

- Input berupa gambar Fashion-MNIST (grayscale, ukuran 28×2828 \times 2828×28).
- Output berupa prediksi kelas (10 kelas, misalnya jenis pakaian seperti T-shirt, sepatu, atau tas).

6) Melatih Model CNN

```
# Training function
def train_model(model, optimizer, scheduler, criterion, train_loader, test_loader, epochs,
early_stop_patience):
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model.to(device)

    best_acc = 0
    patience = 0

    for epoch in range(epochs):
        model.train()
        train_loss = 0

        for X_batch, y_batch in train_loader:
            X_batch, y_batch = X_batch.to(device), y_batch.to(device)

            optimizer.zero_grad()
            outputs = model(X_batch)
            loss = criterion(outputs, y_batch)
            loss.backward()
            optimizer.step()

            train_loss += loss.item()

        scheduler.step()
```

```
        # Evaluate
        model.eval()
        correct = 0
        total = 0

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

        acc = correct / total
        print(f"Epoch {epoch+1}/{epochs}, Loss: {train_loss/len(train_loader):.4f}, Accuracy:
{acc:.4f}")

        # Early stopping
        if acc > best_acc:
            best_acc = acc
            patience = 0
        else:
            patience += 1

        if patience >= early_stop_patience:
            print("Early stopping...")
            break
```

- **model**: Arsitektur CNN yang akan dilatih.
- **optimizer**: Algoritma optimasi seperti SGD, RMSProp, atau Adam untuk memperbarui bobot model berdasarkan gradien.
- **scheduler**: Mengubah learning rate selama pelatihan agar pelatihan lebih stabil dan konvergen lebih cepat.
- **criterion**: Fungsi loss untuk menghitung error model terhadap target (dalam kasus ini, CrossEntropyLoss).
- **train_loader & test_loader**: Data loader untuk menyediakan data pelatihan dan pengujian dalam bentuk batch.
- **epochs**: Jumlah iterasi maksimum untuk melatih model.
- **early_stop_patience**: Batas kesabaran untuk early stopping, yaitu menghentikan pelatihan jika akurasi tidak meningkat selama beberapa epoch berturut-turut.

7) Experiment Komparasi Parameter

```
# Experiments
kernel_sizes = [3, 5, 7]
pooling_types = ['max', 'avg']
```

```
epochs_list = [5, 50, 100, 250, 350]
optimizers = {'SGD': optim.SGD, 'RMSProp': optim.RMSprop, 'Adam': optim.Adam}

results = []

for kernel_size in kernel_sizes:
    for pooling_type in pooling_types:
        for optimizer_name, optimizer_class in optimizers.items():
            print(f"\nTesting with kernel size: {kernel_size}, pooling: {pooling_type}, optimizer:
{optimizer_name}")

            model = CNN(kernel_size=kernel_size, pooling_type=pooling_type)
            optimizer = optimizer_class(model.parameters(), lr=0.001)
            scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)
            criterion = nn.CrossEntropyLoss()

            train_model(
                model, optimizer, scheduler, criterion,
                train_loader, test_loader,
                epochs=50,  # Change for specific experiments
                early_stop_patience=10
            )
```

- **kernel_sizes**: Variasi ukuran kernel yang digunakan pada convolutional layers (3x3, 5x5, dan 7x7).
- **epochs_list**: Jumlah epoch yang diuji. Dalam loop ini default-nya adalah 50.
- **SGD (Stochastic Gradient Descent)**: Optimasi klasik berbasis gradien.
- **RMSProp**: Memperhitungkan rata-rata kuadrat gradien untuk penyesuaian learning rate.
- **Adam**: Kombinasi dari momentum dan RMSProp.
- **optimizer**: Dibuat sesuai parameter eksperimen (SGD, RMSProp, atau Adam).
- **scheduler**: Mengatur penurunan learning rate setelah beberapa epoch (step size = 10, gamma = 0.1).
- **criterion**: Fungsi loss yang digunakan untuk klasifikasi multi-kelas (CrossEntropyLoss).