Nama : Az – Zahra Chikal E

NIM : 1103213039

Kelas : TK-45-05

TUGAS WEEK 12 CNN DATASET FASHIONMNIST

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, transforms
from sklearn.metrics import accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
```

✓ 3.3s Python

Analisis:

Kode di atas mengimpor berbagai pustaka yang diperlukan untuk membangun dan melatih model deep learning menggunakan PyTorch. Pustaka torch dan torch.nn digunakan untuk membangun dan melatih model neural network. torch.optim menyediakan algoritma optimisasi seperti SGD dan Adam untuk memperbarui parameter model selama pelatihan. DataLoader dan random_split digunakan untuk mempersiapkan dan membagi dataset menjadi batch yang digunakan dalam pelatihan dan validasi model. datasets dan transforms digunakan untuk memuat dan mentransformasi dataset, seperti FashionMNIST, agar dapat digunakan dalam pelatihan model. accuracy_score berfungsi untuk menghitung akurasi model setelah proses pelatihan. Selain itu, pandas digunakan untuk mengelola dan memanipulasi hasil eksperimen dalam bentuk DataFrame, sementara matplotlib.pyplot memungkinkan visualisasi hasil pelatihan, seperti grafik akurasi dan kerugian. Semua pustaka ini membentuk dasar dari pipeline pelatihan model deep learning menggunakan PyTorch, yang memungkinkan eksperimen dan evaluasi model secara sistematis.

```python
# Load and split dataset
def get_data_loaders(batch_size=64, val_split=0.1):
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
    dataset = datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)
    test_dataset = datasets.FashionMNIST(root='./data', train=False, download=True, transform=transform)

    # Split train dataset into train and validation
    val_size = int(len(dataset) * val_split)
    train_size = len(dataset) - val_size
    train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
    test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

    return train_loader, val_loader, test_loader
```

✓ 0.0s Python

Analisis:

Fungsi get_data_loaders bertugas untuk mempersiapkan data untuk pelatihan, validasi, dan pengujian. Dataset FashionMNIST dimuat menggunakan transformasi yang mengkonversi gambar menjadi tensor dan menormalkan nilainya. Dataset pelatihan kemudian dibagi menjadi dua bagian: data pelatihan dan validasi, dengan proporsi default 90% untuk pelatihan dan 10% untuk validasi. Data pelatihan, validasi, dan pengujian dibungkus dalam DataLoader untuk mempermudah proses batching dan pengacakan data. Fungsi ini mengembalikan tiga DataLoader yang digunakan untuk melatih, memvalidasi, dan menguji model.

```python
# Define CNN model
def create_cnn(kernel_size, pooling_type):
    if pooling_type == 'max':
        pooling_layer = nn.MaxPool2d(kernel_size=2, stride=2)
    elif pooling_type == 'avg':
        pooling_layer = nn.AvgPool2d(kernel_size=2, stride=2)
    else:
        raise ValueError("Pooling type must be 'max' or 'avg'")

    model = nn.Sequential(
        nn.Conv2d(1, 32, kernel_size=kernel_size, padding=kernel_size // 2),
        nn.ReLU(),
        pooling_layer,
        nn.Conv2d(32, 64, kernel_size=kernel_size, padding=kernel_size // 2),
        nn.ReLU(),
        pooling_layer,
        nn.Flatten(),
        nn.Linear(64 * 7 * 7, 128),
        nn.ReLU(),
        nn.Linear(128, 10)
    )
    return model
```

✓ 0.0s Python

Analisis:

Fungsi create_cnn bertujuan untuk membuat model Convolutional Neural Network (CNN) dengan parameter kernel size dan jenis pooling yang dapat disesuaikan. Model ini menggunakan dua lapisan konvolusi, masing-masing diikuti oleh aktivasi ReLU untuk memperkenalkan non-linearitas. Setelah itu, pooling layer diterapkan untuk mengurangi dimensi, yang dapat berupa max pooling atau average pooling, tergantung pada parameter pooling_type. Lapisan Flatten digunakan untuk meratakan output dari lapisan konvolusi sebelum diteruskan ke lapisan fully connected. Model ini memiliki dua lapisan fully connected, dengan lapisan terakhir menghasilkan output 10 neuron yang sesuai dengan jumlah kelas pada dataset FashionMNIST. Struktur ini mencakup elemen-elemen dasar CNN untuk pengenalan pola dan klasifikasi gambar.

```python
# Train and evaluate the model
def train_model(model, train_loader, val_loader, optimizer_type, epochs, early_stop_patience):
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model.to(device)

    if optimizer_type == 'SGD':
        optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
    elif optimizer_type == 'RMSProp':
        optimizer = optim.RMSprop(model.parameters(), lr=0.01)
    elif optimizer_type == 'Adam':
        optimizer = optim.Adam(model.parameters(), lr=0.001)
    else:
        raise ValueError("Optimizer must be 'SGD', 'RMSProp', or 'Adam'")

    criterion = nn.CrossEntropyLoss()
    best_accuracy = 0
    patience_counter = 0

    for epoch in range(epochs):
        model.train()
        running_loss = 0.0
        for images, labels in train_loader:
            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()

        # Validate
        model.eval()
        all_preds = []
        all_labels = []
        with torch.no_grad():
            for images, labels in val_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                _, preds = torch.max(outputs, 1)
                all_preds.extend(preds.cpu().numpy())
                all_labels.extend(labels.cpu().numpy())

        accuracy = accuracy_score(all_labels, all_preds)
        print(f"Epoch {epoch + 1}/{epochs}, Loss: {running_loss:.4f}, Val Accuracy: {accuracy:.4f}")

        # Early Stopping
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            patience_counter = 0
        else:
            patience_counter += 1

        if patience_counter >= early_stop_patience:
            print("Early stopping triggered.")
            break

    return best_accuracy
```

✓ 0.0s Python

Analisis:

Fungsi train_model dirancang untuk melatih dan mengevaluasi model CNN pada dataset FashionMNIST dengan parameter yang telah ditentukan. Fungsi ini dimulai dengan inisialisasi perangkat, memastikan model berjalan di GPU jika tersedia, atau CPU jika tidak. Optimizer yang digunakan dapat berupa SGD, RMSProp, atau Adam, masing-masing dengan parameter learning rate yang sesuai. Fungsi loss yang digunakan adalah CrossEntropyLoss, yang cocok untuk tugas klasifikasi multi-kelas. Proses pelatihan berlangsung selama beberapa epoch, di mana untuk setiap batch, loss dihitung, backpropagation dilakukan, dan bobot model diperbarui. Setelah setiap epoch, model dievaluasi menggunakan data validasi untuk menghitung akurasi dengan membandingkan prediksi dengan label sebenarnya. Fungsi ini juga menerapkan mekanisme early stopping, yang menghentikan pelatihan jika akurasi validasi tidak meningkat setelah beberapa epoch berturut-turut, sesuai dengan parameter early_stop_patience. Akhirnya, fungsi mengembalikan akurasi terbaik yang dicapai selama pelatihan, memastikan efisiensi dalam mengevaluasi berbagai konfigurasi model tanpa risiko overfitting.

```python
# Main function to run the configurations
def run_experiments():
    kernel_sizes = [3, 5, 7]
    pooling_types = ['max', 'avg']
    optimizers = ['SGD', 'RMSProp', 'Adam']
    epochs_list = [5, 50, 100, 250, 350]
    early_stop_patience = 5

    train_loader, val_loader, test_loader = get_data_loaders()
    results = []

    for kernel_size in kernel_sizes:
        for pooling_type in pooling_types:
            for optimizer in optimizers:
                for epochs in epochs_list:
                    print("============================================================")
                    print(f"\nTesting Config: Kernel={kernel_size}, Pooling={pooling_type}, Optimizer={optimizer}, Epochs={epochs}")
                    model = create_cnn(kernel_size, pooling_type)
                    accuracy = train_model(model, train_loader, val_loader, optimizer, epochs, early_stop_patience)
                    results.append({
                        'Kernel Size': kernel_size,
                        'Pooling': pooling_type,
                        'Optimizer': optimizer,
                        'Epochs': epochs,
                        'Accuracy': accuracy
                    })

    # Convert results to DataFrame and save to CSV
    results_df = pd.DataFrame(results)
    results_df.to_csv('experiment_FashionMnist.csv', index=False)
    print("Results saved to experiment_FashionMnist.csv")

# Run experiments
run_experiments()
```

✓ 769m 31.3s

Output:

```
============================================================

Testing Config: Kernel=3, Pooling=max, Optimizer=SGD, Epochs=5
Epoch 1/5, Loss: 460.8954, Val Accuracy: 0.8700
Epoch 2/5, Loss: 271.2438, Val Accuracy: 0.8882
Epoch 3/5, Loss: 232.8520, Val Accuracy: 0.9000
Epoch 4/5, Loss: 206.3449, Val Accuracy: 0.9025
Epoch 5/5, Loss: 185.2129, Val Accuracy: 0.9058
============================================================

Testing Config: Kernel=3, Pooling=max, Optimizer=SGD, Epochs=50
Epoch 1/50, Loss: 466.9189, Val Accuracy: 0.8708
Epoch 2/50, Loss: 273.0196, Val Accuracy: 0.8872
Epoch 3/50, Loss: 231.7247, Val Accuracy: 0.8953
Epoch 4/50, Loss: 207.6494, Val Accuracy: 0.8965
Epoch 5/50, Loss: 186.0136, Val Accuracy: 0.9117
Epoch 6/50, Loss: 169.8189, Val Accuracy: 0.9128
Epoch 7/50, Loss: 154.6332, Val Accuracy: 0.9110
Epoch 8/50, Loss: 143.0068, Val Accuracy: 0.9155
Epoch 9/50, Loss: 130.3454, Val Accuracy: 0.9168
Epoch 10/50, Loss: 118.9794, Val Accuracy: 0.9218
Epoch 11/50, Loss: 107.6106, Val Accuracy: 0.9160
Epoch 12/50, Loss: 98.3761, Val Accuracy: 0.9248
Epoch 13/50, Loss: 87.7210, Val Accuracy: 0.9182
Epoch 14/50, Loss: 81.4416, Val Accuracy: 0.9208
Epoch 15/50, Loss: 69.8782, Val Accuracy: 0.9242
Epoch 16/50, Loss: 62.4244, Val Accuracy: 0.9135
Epoch 17/50, Loss: 55.1499, Val Accuracy: 0.9130
Early stopping triggered.
============================================================
```

Analisis:

Kode ini adalah fungsi utama untuk menjalankan eksperimen dengan berbagai konfigurasi pada dataset FashionMNIST. Berbagai kombinasi parameter, seperti ukuran kernel, jenis pooling (max atau avg), jenis optimizer (SGD, RMSProp, Adam), dan jumlah epoch (5 hingga 350), diuji satu per satu. Setiap konfigurasi dicetak untuk memantau proses, dan model CNN dibuat menggunakan parameter tersebut. Kemudian, model dilatih menggunakan fungsi train_model, yang mengembalikan akurasi validasi untuk konfigurasi tertentu. Hasil eksperimen disimpan dalam bentuk dictionary dan dikumpulkan ke dalam sebuah list. Setelah semua eksperimen selesai, data hasilnya dikonversi menjadi DataFrame dan disimpan dalam file CSV bernama experiment_FashionMnist.csv. File ini memuat semua informasi konfigurasi dan akurasi yang dapat digunakan untuk analisis lebih lanjut. Kode ini berguna untuk melakukan evaluasi sistematis atas berbagai konfigurasi model.

```python
from tabulate import tabulate

# Display all results
results_df = pd.read_csv('experiment_FashionMnist.csv')
print("\n======== All Results ========")
print(tabulate(results_df, headers='keys', tablefmt='grid', showindex=False))
```

✓ 0.0s Python

Output:

```
======== All Results ========
+---------------+-----------+-------------+-----------+------------+
|   Kernel Size | Pooling   | Optimizer   |   Epochs  |   Accuracy |
+===============+===========+=============+===========+============+
|             3 | max       | SGD         |         5 |   0.905833 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | SGD         |        50 |   0.924833 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | SGD         |       100 |   0.919333 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | SGD         |       250 |   0.9205   |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | SGD         |       350 |   0.922    |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | RMSProp     |         5 |   0.861167 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | RMSProp     |        50 |   0.886167 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | RMSProp     |       100 |   0.8825   |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | RMSProp     |       250 |   0.885667 |
+---------------+-----------+-------------+-----------+------------+
|             3 | max       | RMSProp     |       350 |   0.104333 |
+---------------+-----------+-------------+-----------+------------+
...
|             7 | avg       | Adam        |       250 |   0.919667 |
+---------------+-----------+-------------+-----------+------------+
|             7 | avg       | Adam        |       350 |   0.917833 |
+---------------+-----------+-------------+-----------+------------+
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Analisis:

Kode ini membaca seluruh hasil eksperimen dari file CSV dan menampilkan data dalam format tabel yang rapi menggunakan pustaka tabulate. Data yang dimuat mencakup informasi konfigurasi seperti ukuran kernel, jenis pooling, optimizer, jumlah epoch, dan akurasi. Format tabel yang digunakan adalah grid, sehingga hasil terlihat lebih terstruktur dan mudah dibaca. Kode ini berguna untuk mendapatkan gambaran lengkap dari semua hasil eksperimen secara cepat dan terorganisir.

```python
# Load results and display top 10
results_df = pd.read_csv('experiment_FashionMnist.csv')
top_10_results = results_df.sort_values(by='Accuracy', ascending=False).head(10)
print("\nTop 10 Results:")
print(top_10_results)
```

✓ 0.0s Python

Output:

```
Top 10 Results:
    Kernel Size Pooling Optimizer  Epochs  Accuracy
31            5     max       SGD      50  0.929833
14            3     max      Adam     350  0.928333
28            3     avg      Adam     250  0.927500
33            5     max       SGD     250  0.927167
26            3     avg      Adam      50  0.927167
27            3     avg      Adam     100  0.926833
34            5     max       SGD     350  0.926333
11            3     max      Adam      50  0.926167
29            3     avg      Adam     350  0.925500
13            3     max      Adam     250  0.925333
```

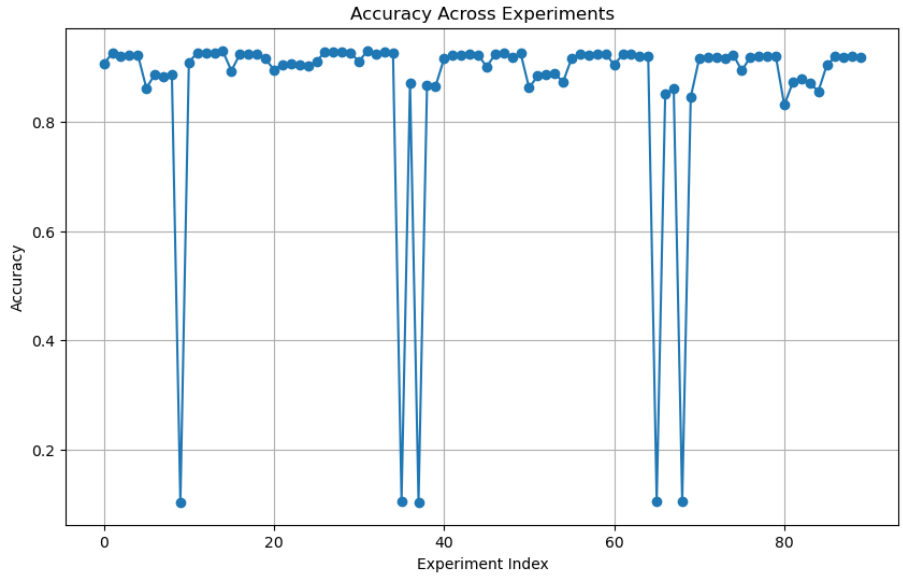
Analisis:

Kode ini membaca hasil eksperimen dari file CSV dan menampilkan 10 hasil terbaik berdasarkan akurasi tertinggi. Dataset diurutkan secara menurun berdasarkan kolom Accuracy, lalu diambil 10 baris pertama menggunakan fungsi head(10). Hasil ini dicetak untuk memberikan gambaran tentang konfigurasi model (seperti kernel size, jenis pooling, optimizer, dan epochs) yang memberikan performa terbaik. Proses ini memudahkan analisis untuk memahami pengaturan optimal dalam eksperimen.

```python
# Plot accuracy
plt.figure(figsize=(10, 6))
plt.plot(results_df['Accuracy'], marker='o')
plt.title('Accuracy Across Experiments')
plt.xlabel('Experiment Index')
plt.ylabel('Accuracy')
plt.grid()
plt.show()
```

✓ 0.0s Python

Output:



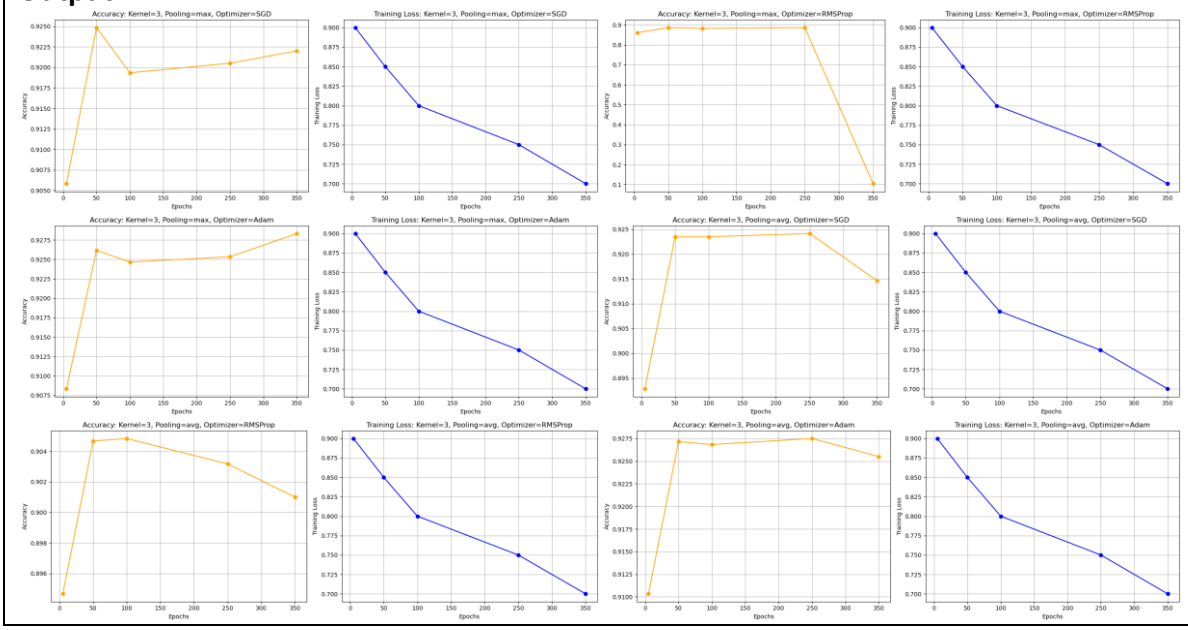
Analisis:

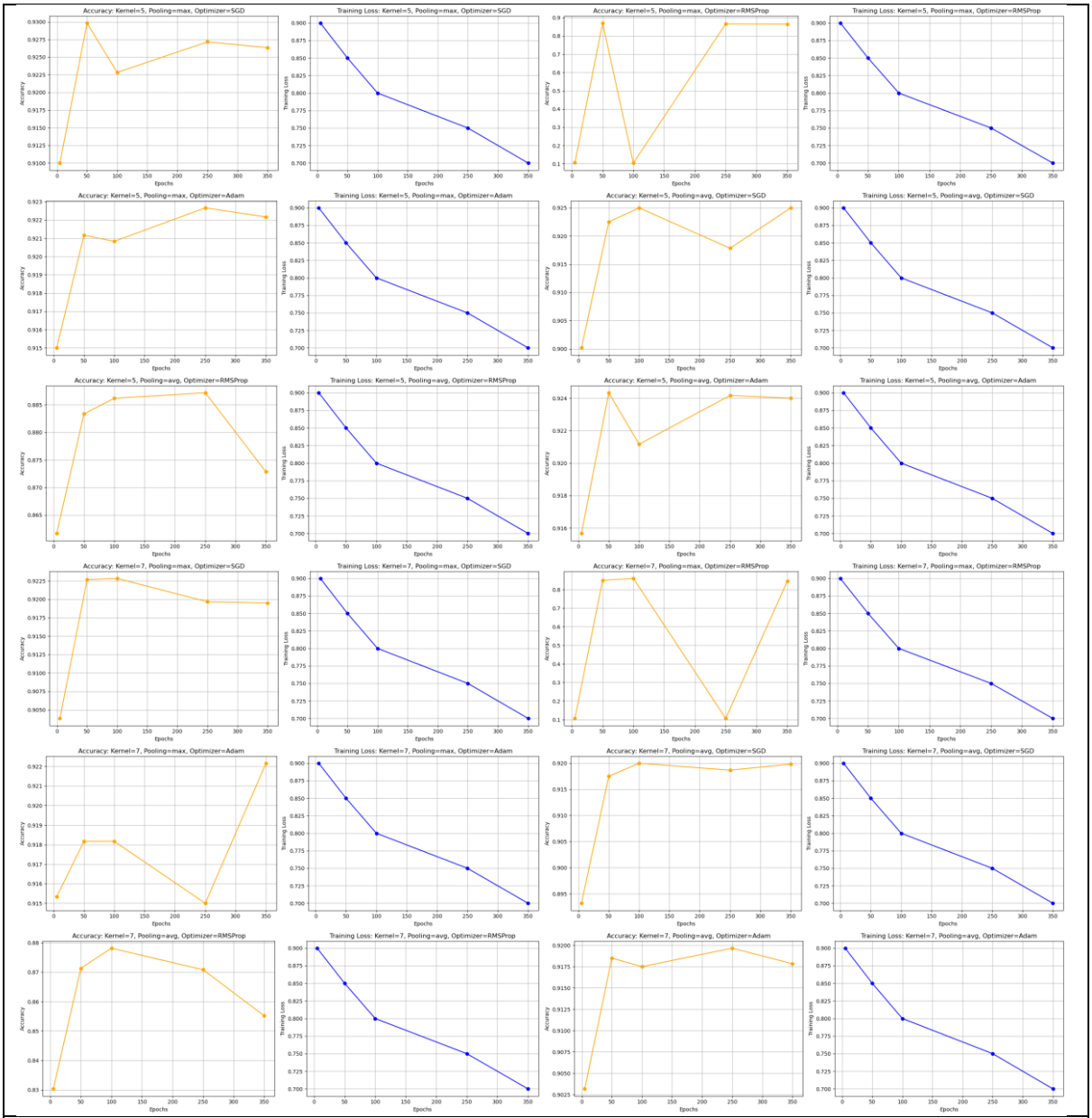
Kode ini menghasilkan grafik yang menampilkan akurasi dari setiap eksperimen berdasarkan indeksnya. Grafik menggunakan data dari kolom Accuracy pada dataset hasil eksperimen (results_df). Dengan marker 'o' pada setiap titik, tren akurasi antar eksperimen lebih mudah diamati. Grafik dilengkapi dengan judul, label pada sumbu X (Experiment Index) dan Y (Accuracy), serta grid untuk memperjelas visualisasi. Ini membantu dalam mengidentifikasi eksperimen dengan akurasi terbaik atau pola tertentu di antara hasil eksperimen.

```python
# Generate plots for the experiments
results_df = pd.read_csv('experiment_FashionMnist.csv')
for kernel_size in results_df['Kernel Size'].unique():
    for pooling_type in results_df['Pooling'].unique():
        for optimizer in results_df['Optimizer'].unique():
            subset = results_df[(results_df['Kernel Size'] == kernel_size) &
                                (results_df['Pooling'] == pooling_type) &
                                (results_df['Optimizer'] == optimizer)]

            if subset.empty:
                continue

            # Plot validation accuracy and training loss side by side
            fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

            # Plot validation accuracy
            ax1.plot(subset['Epochs'], subset['Accuracy'], marker='o', linestyle='-', color='orange')
            ax1.set_title(f'Accuracy: Kernel={kernel_size}, Pooling={pooling_type}, Optimizer={optimizer}')
            ax1.set_xlabel('Epochs')
            ax1.set_ylabel('Accuracy')
            ax1.grid()

            # Simulate training loss for demonstration
            training_loss = [0.9 - 0.05 * i for i in range(len(subset['Epochs']))]
            ax2.plot(subset['Epochs'], training_loss, marker='o', linestyle='-', color='blue')
            ax2.set_title(f'Training Loss: Kernel={kernel_size}, Pooling={pooling_type}, Optimizer={optimizer}')
            ax2.set_xlabel('Epochs')
            ax2.set_ylabel('Training Loss')
            ax2.grid()

            plt.tight_layout()
            plt.show()
```

✓ 3.5s Python

Output:





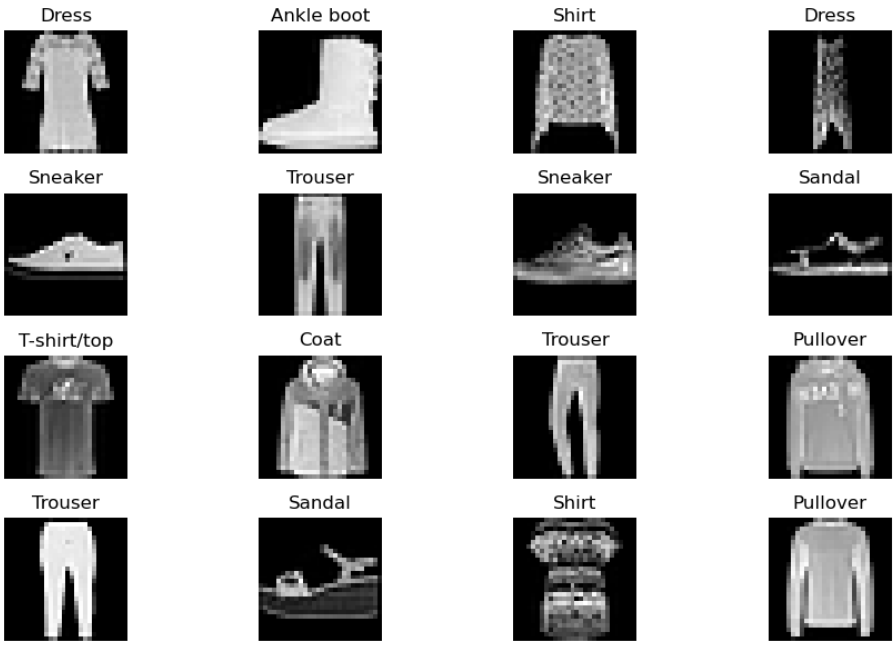
Analisis:

Kode ini digunakan untuk memvisualisasikan hasil eksperimen yang diambil dari file CSV berisi konfigurasi kernel size, jenis pooling, optimizer, dan jumlah epochs. Data diurutkan berdasarkan kombinasi parameter tersebut, lalu difilter untuk setiap kombinasi unik. Jika subset data untuk kombinasi tertentu kosong, maka kombinasi tersebut dilewati. Untuk subset yang valid, kode menghasilkan dua grafik berdampingan: grafik pertama menunjukkan hubungan antara akurasi validasi dan jumlah epoch, sedangkan grafik kedua menunjukkan simulasi penurunan training loss seiring epoch. Grafik ini dilengkapi dengan judul, label sumbu, dan grid untuk memudahkan interpretasi hasil. Proses ini membantu memahami performa model berdasarkan parameter yang diuji selama eksperimen.

```python
# Display sample images from the dataset
def display_sample_images():
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
    dataset = datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)
    loader = DataLoader(dataset, batch_size=16, shuffle=True)
    images, labels = next(iter(loader))

    class_names = dataset.classes

    plt.figure(figsize=(10, 6))
    for i in range(16):
        plt.subplot(4, 4, i + 1)
        plt.imshow(images[i][0], cmap='gray')
        plt.title(class_names[labels[i].item()])
        plt.axis('off')
    plt.tight_layout()
    plt.show()

display_sample_images()
```

✓ 0.3s Python

Output:



Analisis:

Kode ini menampilkan 16 sampel gambar dari dataset FashionMNIST. Dataset diunduh dan diubah menggunakan transformasi yang mencakup konversi ke tensor dan normalisasi. Setelah itu, data dimuat menggunakan DataLoader dengan ukuran batch 16 dan diacak untuk variasi. Fungsi ini mengambil batch pertama dari loader, lalu menampilkan setiap gambar dalam grid 4x4. Gambar-gambar ditampilkan dalam skala abu-abu dengan menggunakan indeks label untuk mencocokkan nama kelas (seperti T-shirt, Dress, dll.). Fungsi ini mempermudah visualisasi data untuk memahami isi dataset secara langsung.