Nama : Az – Zahra Chikal E

NIM : 1103213039

Kelas : TK-45-05

TUGAS WEEK 12 CNN DATASET CIFAR10

```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
from tabulate import tabulate
```

✓ 4.7s Python

Analisis:

Kode ini mengimpor berbagai pustaka yang diperlukan untuk membangun dan melatih model deep learning menggunakan PyTorch. Pustaka seperti torch dan torch.nn digunakan untuk mendefinisikan dan melatih jaringan saraf, sementara torch.optim menyediakan berbagai algoritma optimisasi seperti SGD, RMSProp, dan Adam. Dengan menggunakan torch.utils.data, dataset dapat dimuat dan dibagi menjadi bagian pelatihan, validasi, dan pengujian. Pustaka torchvision membantu dalam mengakses dataset gambar seperti CIFAR-10 serta melakukan transformasi pada data tersebut. Untuk menghitung akurasi model, digunakan sklearn.metrics, dan pandas mempermudah pengelolaan serta analisis hasil eksperimen, termasuk menyimpan data dalam format CSV. Visualisasi hasil eksperimen dilakukan menggunakan matplotlib.pyplot, sementara tabulate digunakan untuk menampilkan hasil dalam format tabel yang rapi. Semua pustaka ini bekerja bersama-sama untuk mendukung setiap tahap dalam proses pembangunan dan evaluasi model deep learning berbasis dataset gambar.

```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, 0.5), (0.5, 0.5, 0.5))
    ])
    dataset = datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
    test_dataset = datasets.CIFAR10(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:
Kode ini memuat dataset CIFAR-10 dan membagi dataset pelatihan menjadi dua bagian: satu untuk pelatihan dan satu lagi untuk validasi. Dataset CIFAR-10 diunduh dan diubah menjadi tensor dengan normalisasi. Pembagian untuk pelatihan dan validasi dilakukan berdasarkan proporsi val_split (default 10%), dengan menggunakan fungsi random_split. Setelah itu, data dimuat ke dalam DataLoader untuk pelatihan, validasi, dan pengujian, masing-masing dengan ukuran batch yang ditentukan. Kode ini mempersiapkan data untuk digunakan dalam pelatihan dan evaluasi 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(3, 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 * 8 * 8, 128),
        nn.ReLU(),
        nn.Linear(128, 10)
    )
    return model
```

✓ 0.0s Python

Analisis:
Kode ini mendefinisikan model CNN dengan dua lapisan konvolusi dan lapisan pooling yang dapat dipilih antara *max pooling* atau *average pooling*. Berdasarkan jenis pooling yang diberikan, lapisan pooling yang sesuai (MaxPool2d atau AvgPool2d) dipilih. Model dimulai dengan dua lapisan konvolusi yang mengubah gambar input menjadi fitur, diikuti dengan fungsi aktivasi ReLU dan lapisan pooling untuk mereduksi dimensi. Setelah itu, gambar diratakan dan diproses melalui dua lapisan fully connected (Linear) untuk menghasilkan output klasifikasi 10 kelas (sesuai dengan jumlah kelas pada dataset CIFAR-10). Kode ini menghasilkan model CNN yang siap digunakan untuk pelatihan dan evaluasi.

```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:

Kode ini melatih dan mengevaluasi model CNN menggunakan loader data untuk pelatihan dan validasi. Model dilatih di perangkat GPU jika tersedia, dengan pilihan optimizer (SGD, RMSProp, atau Adam) yang ditentukan oleh parameter. Fungsi kehilangan menggunakan CrossEntropyLoss, sementara akurasi validasi dihitung setiap epoch menggunakan accuracy_score. Dalam setiap epoch, model dilatih dengan menghitung *loss* dan memperbarui bobotnya menggunakan backpropagation. Setelah pelatihan, model dievaluasi pada data validasi untuk menghitung akurasi. Proses pelatihan mendukung *early stopping*, menghentikan pelatihan jika akurasi validasi tidak meningkat selama beberapa epoch berturut-turut (ditentukan oleh early_stop_patience). Akurasi terbaik dari validasi dikembalikan sebagai hasil. Kode ini memastikan efisiensi pelatihan dengan pemantauan akurasi dan penggunaan *early stopping*.

```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, pooling_type, optimizer, epochs, accuracy])

    # Convert results to DataFrame and save to CSV
    results_df = pd.DataFrame(results, columns=['Kernel Size', 'Pooling', 'Optimizer', 'Epochs', 'Validation Accuracy'])
    results_df.to_csv('experiment_cifar10.csv', index=False)
    print("Results saved to experiment_cifar10.csv")

    return results_df

# Run experiments
results_df = run_experiments()
```

✓ 666m 16.6s

Output:

```
Files already downloaded and verified
Files already downloaded and verified
================================================================

Testing Config: Kernel=3, Pooling=max, Optimizer=SGD, Epochs=5
Epoch 1/5, Loss: 1093.9939, Val Accuracy: 0.5558
Epoch 2/5, Loss: 781.7136, Val Accuracy: 0.6246
Epoch 3/5, Loss: 652.3111, Val Accuracy: 0.6552
Epoch 4/5, Loss: 552.6174, Val Accuracy: 0.6926
Epoch 5/5, Loss: 474.7717, Val Accuracy: 0.6798
================================================================

Testing Config: Kernel=3, Pooling=max, Optimizer=SGD, Epochs=50
Epoch 1/50, Loss: 1100.5535, Val Accuracy: 0.5604
Epoch 2/50, Loss: 777.6862, Val Accuracy: 0.6432
Epoch 3/50, Loss: 633.2891, Val Accuracy: 0.6634
Epoch 4/50, Loss: 539.3462, Val Accuracy: 0.7106
Epoch 5/50, Loss: 455.2689, Val Accuracy: 0.6874
Epoch 6/50, Loss: 383.9562, Val Accuracy: 0.7186
Epoch 7/50, Loss: 310.3761, Val Accuracy: 0.7168
Epoch 8/50, Loss: 237.1736, Val Accuracy: 0.7274
Epoch 9/50, Loss: 177.3741, Val Accuracy: 0.7130
Epoch 10/50, Loss: 129.8211, Val Accuracy: 0.7300
Epoch 11/50, Loss: 101.0667, Val Accuracy: 0.7234
Epoch 12/50, Loss: 73.8566, Val Accuracy: 0.7166
Epoch 13/50, Loss: 58.2817, Val Accuracy: 0.7004
Epoch 14/50, Loss: 53.3072, Val Accuracy: 0.7228
Epoch 15/50, Loss: 42.1816, Val Accuracy: 0.7160
Early stopping triggered.
================================================================
```

Analisis:

Kode ini menjalankan eksperimen model CNN dengan berbagai kombinasi parameter seperti ukuran kernel, metode pooling, optimizer, dan jumlah epoch. Untuk setiap kombinasi, model CNN dibuat dan dilatih menggunakan fungsi `train_model`, dengan akurasi validasi dicatat. Parameter seperti early stopping juga diterapkan dengan nilai toleransi 5 epoch. Hasil eksperimen disimpan dalam list, kemudian dikonversi menjadi DataFrame `results_df` dan diekspor ke file CSV `experiment_cifar10.csv`. Kode ini mempermudah eksplorasi performa model dengan berbagai konfigurasi secara otomatis.

```python
# Display all results
results_df = pd.read_csv('experiment_cifar10.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 |   Validation Accuracy |
+===============+===========+=============+==========+======================+
|             3 | max       | SGD         |        5 |               0.6926 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | SGD         |       50 |               0.73   |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | SGD         |      100 |               0.7266 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | SGD         |      250 |               0.728  |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | SGD         |      350 |               0.741  |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | RMSProp     |        5 |               0.4974 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | RMSProp     |       50 |               0.6022 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | RMSProp     |      100 |               0.5902 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | RMSProp     |      250 |               0.6168 |
+---------------+-----------+-------------+----------+----------------------+
|             3 | max       | RMSProp     |      350 |               0.5832 |
+---------------+-----------+-------------+----------+----------------------+
...
|             7 | avg       | Adam        |      250 |               0.7098 |
+---------------+-----------+-------------+----------+----------------------+
|             7 | avg       | Adam        |      350 |               0.7084 |
+---------------+-----------+-------------+----------+----------------------+
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Analisis:

Kode ini membaca file `experiment_cifar10.csv` dan menampilkan seluruh hasil eksperimen dalam bentuk tabel yang rapi menggunakan pustaka `tabulate`. Dengan format tabel grid, setiap kolom diberi header, dan indeks baris dihilangkan untuk tampilan yang lebih bersih. Pendekatan ini memudahkan pengguna untuk melihat semua data eksperimen secara menyeluruh dalam satu pandangan.

```python
# Load results and display top 10
top_10_results = results_df.sort_values(by='Validation Accuracy', ascending=False).head(10)
print("\nTop 10 Results:")
print(tabulate(top_10_results, headers='keys', tablefmt='grid', showindex=False))
```

✓ 0.0s Python

Output:

```
Top 10 Results:
```

Kernel Size	Pooling	Optimizer	Epochs	Validation Accuracy
5	max	SGD	250	0.7416
3	max	SGD	350	0.741
5	max	SGD	350	0.7392
5	avg	Adam	100	0.7376
5	max	SGD	50	0.7368
5	max	Adam	250	0.7362
5	max	SGD	100	0.7338
7	max	SGD	250	0.7338
5	max	Adam	50	0.7338
5	max	Adam	100	0.731

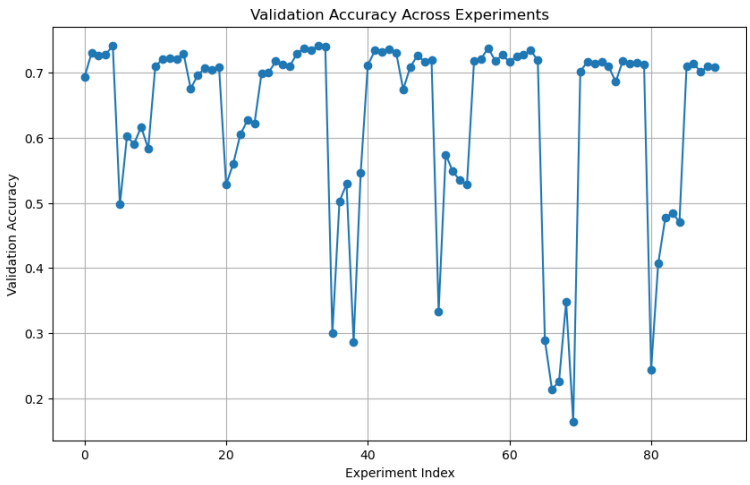
Analisis:

Kode ini digunakan untuk menampilkan 10 hasil eksperimen terbaik berdasarkan akurasi validasi tertinggi. Data diurutkan secara menurun (descending) berdasarkan kolom Validation Accuracy, kemudian 10 baris teratas diambil. Hasilnya ditampilkan dalam format tabel yang rapi menggunakan pustaka tabulate, dengan header kolom, tanpa indeks tambahan, dan menggunakan gaya tabel berbentuk grid. Ini mempermudah identifikasi konfigurasi eksperimen yang menghasilkan performa terbaik.

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

✓ 0.1s Python

Output:



Analisis:

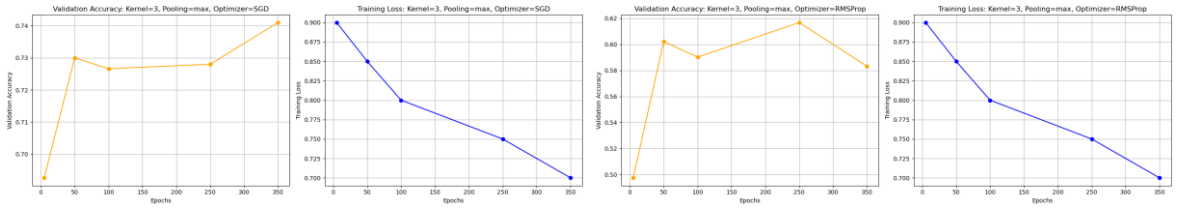
Kode ini membuat grafik sederhana untuk memvisualisasikan akurasi validasi dari berbagai eksperimen. Sumbu x mewakili indeks eksperimen, sedangkan sumbu y menunjukkan nilai akurasi validasi. Titik-titik pada grafik diberi penanda bulat untuk memperjelas setiap data, dan grid ditambahkan untuk membantu pembacaan. Judul dan label sumbu memberikan konteks pada grafik, sehingga mempermudah analisis variasi akurasi di seluruh eksperimen.

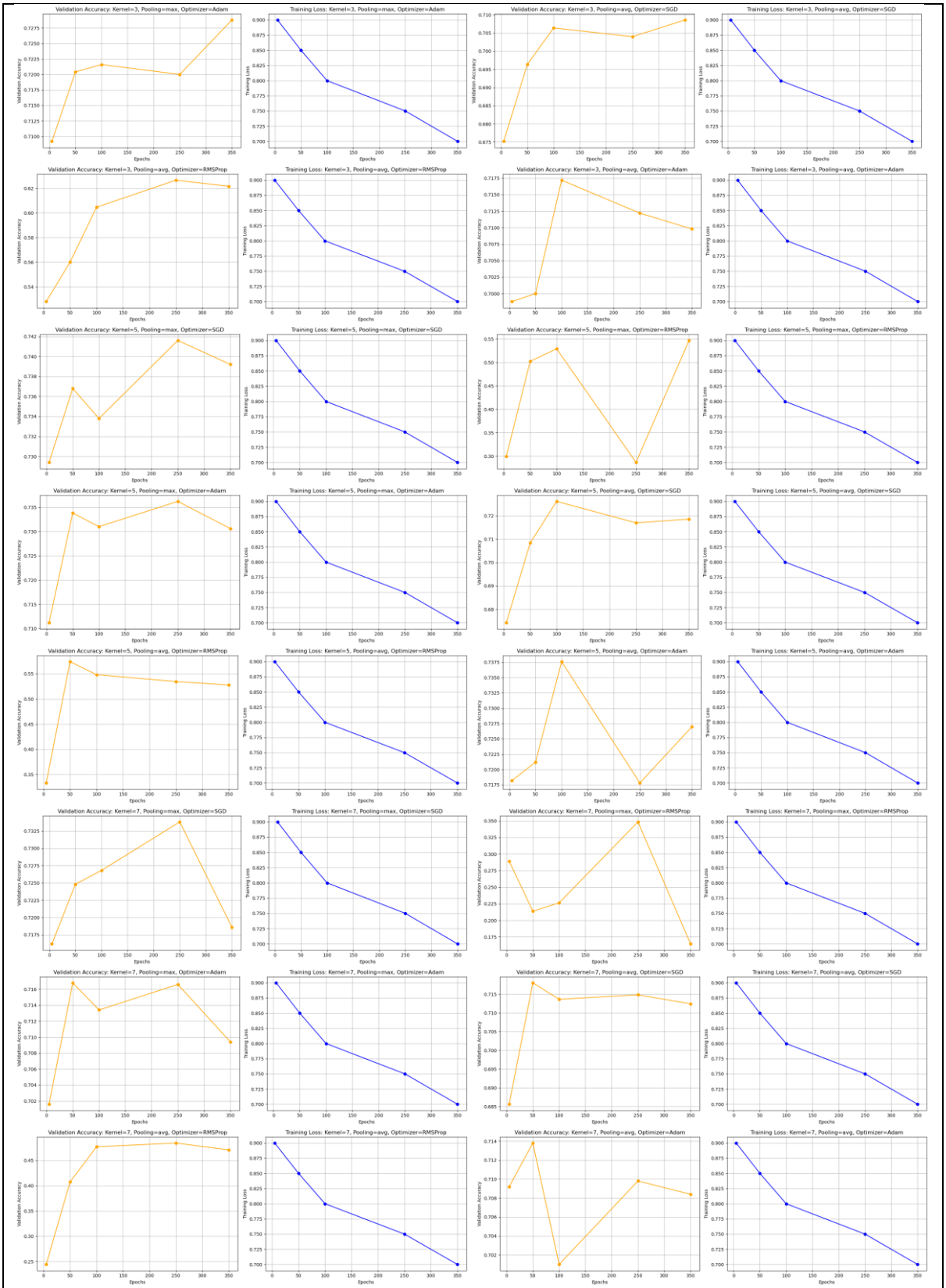
```python
# Generate plots for the experiments
results_df = pd.read_csv('experiment_cifar10.csv')
for kernel_size in results_df['Kernel Size'].unique():
    for pooling_type in results_df['Pooling'].unique():
        for optimizer in results_df['Optimizer'].unique():
            subset = results_df[(results_df['Kernel 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['Validation Accuracy'], marker='o', linestyle='-', color='orange')
            ax1.set_title(f'Validation Accuracy: Kernel={kernel_size}, Pooling={pooling_type}, Optimizer={optimize
            ax1.set_xlabel('Epochs')
            ax1.set_ylabel('Validation Accuracy')
            ax1.grid()

            # Simulate training loss for demonstration
            training_loss = [0.9 - 0.05 * i for i in range(len(sub
            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()
```

(function) linestyle: Any

3.4s Python

Output:





Analisis:

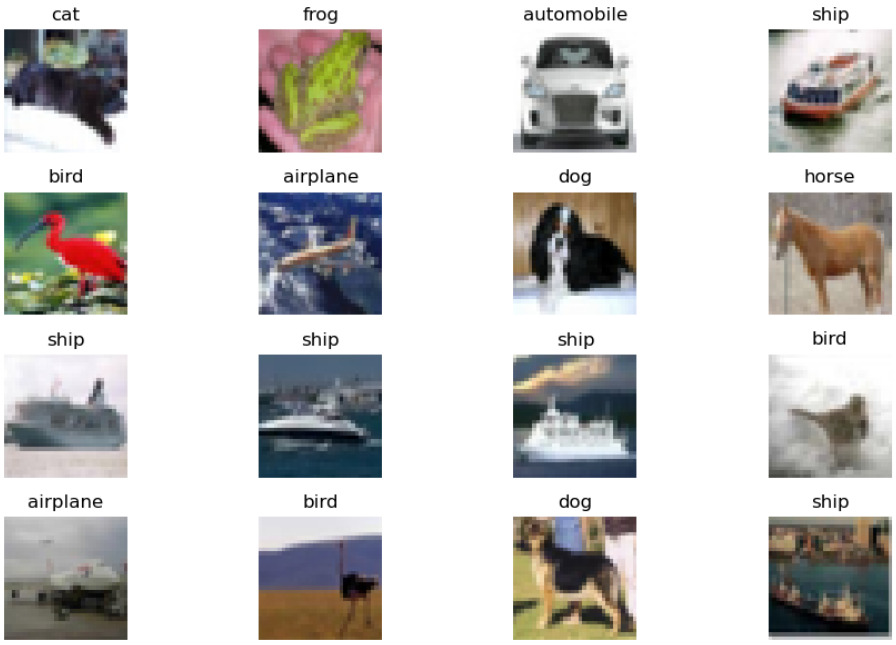
Kode ini bertujuan untuk memvisualisasikan hasil eksperimen model CIFAR-10 dengan membuat grafik akurasi validasi dan *training loss* berdasarkan parameter seperti ukuran kernel, metode pooling, dan optimizer. Data diambil dari file CSV, difilter sesuai kombinasi parameter, lalu divisualisasikan dalam dua grafik berdampingan untuk setiap kombinasi.

Grafik pertama menunjukkan akurasi validasi terhadap epoch, sedangkan *training loss* disimulasikan untuk ilustrasi. Kode ini mempermudah analisis performa model dengan berbagai konfigurasi, meskipun simulasi *training loss* sebaiknya diganti dengan data asli untuk hasil yang lebih akurat.

```python
# Display sample images from the dataset
def display_sample_images():
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])
    dataset = datasets.CIFAR10(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].permute(1, 2, 0) * 0.5 + 0.5)  # Denormalize and permute
        plt.title(class_names[labels[i].item()])
        plt.axis('off')
    plt.tight_layout()
    plt.show()

display_sample_images()
```

✓ 1.4s Python

Output:



Analisis:

Kode di atas bertujuan untuk menampilkan sampel gambar dari dataset CIFAR-10 menggunakan Python dengan pustaka PyTorch. Pertama, transformasi pada gambar diterapkan dengan menggunakan `transforms.Compose`, yang mencakup `transforms.ToTensor` untuk mengubah gambar menjadi tensor dan `transforms.Normalize` untuk menormalisasi nilai piksel ke rentang [-1, 1]. Dataset CIFAR-10 kemudian dimuat melalui `datasets.CIFAR10`, di mana dataset pelatihan akan diunduh jika belum tersedia. Dengan bantuan `DataLoader`, dataset tersebut dipecah menjadi batch berukuran 16 gambar, dan data diacak agar proses pelatihan lebih bervariasi. Batch pertama yang berisi

16 gambar dan labelnya diambil menggunakan `next(iter(loader))`. Nama kelas dataset CIFAR-10 diakses melalui atribut `classes` untuk digunakan sebagai judul pada gambar. Untuk menampilkan gambar, fungsi `plt.imshow` digunakan dengan mendekodenormalisasi gambar kembali ke rentang [0, 1] dan mengubah format tensor agar kompatibel dengan tampilan. Gambar-gambar ini ditampilkan dalam grid 4x4 menggunakan `matplotlib`, dengan judul berupa nama kelas masing-masing gambar. Sumbu gambar dihilangkan untuk menjaga tampilan lebih rapi, dan tata letak grid diatur agar tidak tumpang tindih. Dengan cara ini, kode ini berhasil memvisualisasikan 16 sampel gambar beserta label dari dataset CIFAR-10 secara efisien.