LAPORAN

MACHINE LEARNING

WEEK 12

CNN Cifar10 Dataset

CNN adalah jaringan saraf tiruan yang sangat efektif untuk pengolahan data berbasis gambar. Dengan struktur lapisan konvolusi, pooling, dan fully connected, CNN dapat mengekstraksi fitur secara otomatis dan melakukan klasifikasi dengan akurasi tinggi. CNN telah menjadi fondasi banyak aplikasi modern, seperti mobil otonom, deteksi wajah, dan analisis medis.

```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
```

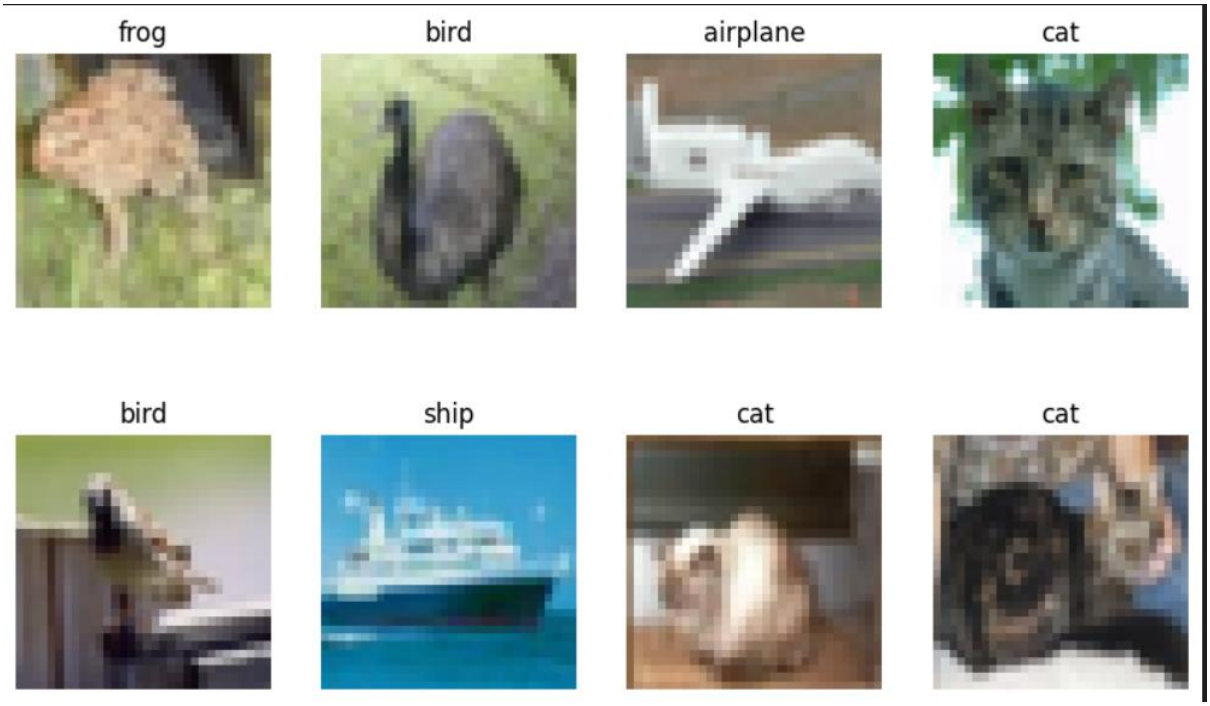
Mempersiapkan dataset CIFAR-10 agar siap digunakan dalam pelatihan model. Pembagian data menjadi train, validation, dan test set memungkinkan pengujian performa model secara terpisah pada data yang tidak terlihat selama pelatihan, sehingga mencegah overfitting dan memastikan evaluasi yang lebih adil. Kombinasi transformasi dan normalisasi membantu mempercepat konvergensi model selama pelatihan.

Exploratory Data Analysis (EDA)

```python
# **Langkah 2: Exploratory Data Analysis (EDA)**
classes = dataset.classes

# Menampilkan beberapa gambar dari dataset
data_iter = iter(train_loader)
images, labels = next(data_iter)

plt.figure(figsize=(10, 6))
for i in range(8):
    plt.subplot(2, 4, i + 1)
    plt.imshow(np.transpose(images[i].numpy(), (1, 2, 0)) * 0.5 + 0.5)  # Denormalisasi
    plt.title(classes[labels[i]])
    plt.axis('off')
plt.show()
```



Membuat arsitektur Convolutional Neural Network (CNN)

```
# **Langkah 3: Membuat Arsitektur CNN**
# 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
```

Fungsi create_cnn adalah sebuah implementasi untuk membangun arsitektur **Convolutional Neural Network (CNN)** yang fleksibel menggunakan library PyTorch. Fungsi ini dirancang agar dapat dikustomisasi melalui dua parameter utama, yaitu kernel_size dan pooling_type. Parameter kernel_size menentukan ukuran kernel atau filter yang digunakan dalam operasi konvolusi, seperti 3x3 atau 5x5, sementara pooling_type memungkinkan pemilihan jenis pooling, yakni **max pooling** untuk mengambil nilai maksimum dari area tertentu atau **average pooling** untuk menghitung rata-rata nilai. Arsitektur CNN yang dihasilkan terdiri dari dua lapisan konvolusi, masing-masing diikuti oleh fungsi aktivasi ReLU untuk menambahkan non-linearitas dan pooling layer untuk mengurangi dimensi data. Setelah fitur diekstraksi melalui lapisan-lapisan tersebut, data diubah menjadi vektor satu dimensi (flattening) dan diteruskan ke dua lapisan fully connected. Lapisan fully connected pertama memetakan fitur ke 128 unit, dan lapisan terakhir menghasilkan output untuk 10 kelas, cocok untuk tugas klasifikasi seperti dataset CIFAR-10. Dengan struktur ini, fungsi create_cnn memberikan fleksibilitas dan efisiensi, mempermudah pengguna untuk mengadaptasi model sesuai kebutuhan dataset atau aplikasi.

Experiment

```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]
    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()
```

run_experiments, yang dirancang untuk menjalankan berbagai konfigurasi eksperimen dalam melatih model Convolutional Neural Network (CNN). Fungsi ini memungkinkan eksplorasi hyperparameter seperti ukuran kernel, jenis pooling, optimizer, dan jumlah epoch.

Parameter Hyperparameter yang Diuji:

1. kernel_sizes:
    - Daftar ukuran kernel untuk lapisan konvolusi, misalnya [3, 5, 7].
    - Kernel menentukan ukuran filter yang digunakan untuk ekstraksi fitur.
2. pooling_types:

- Jenis pooling yang akan digunakan ('max' atau 'avg').
- Pooling membantu mengurangi dimensi data dan mencegah overfitting.

3. optimizers:
   - Daftar optimizer yang akan diuji, seperti 'SGD', 'RMSProp', dan 'Adam'.
4. epochs_list:
   - Pilihan jumlah epoch untuk melatih model, misalnya [5, 50, 100].
5. early_stop_patience:
   - Parameter untuk menerapkan early stopping jika validasi tidak meningkat setelah sejumlah epoch tertentu.

Output

```
# Load best result
best_result = results_df.loc[results_df['Validation Accuracy'].idxmax()]
print("\nBest Result:")
print(tabulate([best_result.to_list()], headers=results_df.columns, tablefmt='grid', showindex=False))
```

```
Best Result:
+---------------+-----------+-------------+----------+----------------------+
|   Kernel Size | Pooling   | Optimizer   |   Epochs |   Validation Accuracy |
+===============+===========+=============+==========+======================+
|             5 | max       | SGD         |       50 |               0.7382 |
+---------------+-----------+-------------+----------+----------------------+
```

Hasil Terbaik

- Kernel Size: 5
  Ukuran kernel terbaik adalah 5x5, yang memungkinkan fitur penting diekstraksi dengan baik tanpa kehilangan informasi.
- Pooling: max
  Max pooling memberikan hasil terbaik, menunjukkan bahwa memilih nilai maksimum dalam area pooling lebih efektif untuk tugas ini.
- Optimizer: SGD
  Optimizer Stochastic Gradient Descent (SGD) menghasilkan akurasi terbaik, mungkin karena stabilitasnya dalam pelatihan.
- Epochs: 50
  Model mencapai akurasi terbaik dengan 50 epoch, menunjukkan bahwa pelatihan lebih lama (tetapi tidak berlebihan) diperlukan untuk konvergensi.
- Validation Accuracy: 0.7382
  Akurasi validasi tertinggi adalah 73.82%, menunjukkan performa model pada data yang tidak terlihat selama pelatihan.

Kesimpulan

Hasil ini mengidentifikasi kombinasi hyperparameter terbaik untuk model CNN, yaitu kernel size 5, max pooling, optimizer SGD, dan 50 epochs, yang memberikan akurasi validasi tertinggi sebesar 73.82%. Pendekatan ini membantu memilih konfigurasi yang optimal untuk pelatihan model, sehingga menghasilkan performa yang lebih baik pada dataset tertentu. Tabel hasil yang rapi juga mempermudah analisis lebih lanjut.