LAPORAN

MACHINE LEARNING

WEEK 12

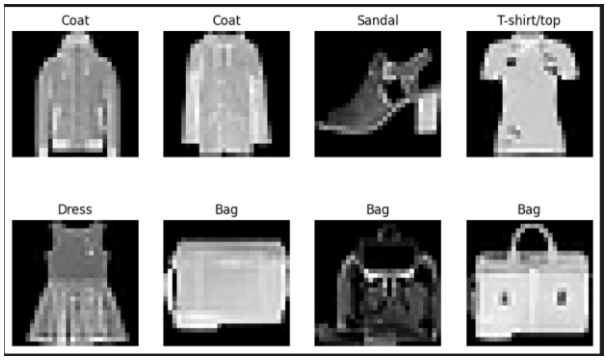
CNN Fashion Mnist Dataset

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.

```
# Dataset preparation
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

# Load Fashion-MNIST
dataset = torchvision.datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.FashionMNIST(root='./data', train=False, download=True, transform=transform)
# Split train set into train and validation
train_idx, val_idx = train_test_split(np.arange(len(dataset)), test_size=0.2, random_state=42)
train_dataset = torch.utils.data.Subset(dataset, train_idx)
val_dataset = torch.utils.data.Subset(dataset, val_idx)

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

Exploratory Data Analysis (EDA)

```
# **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(images[i].squeeze().numpy() * 0.5 + 0.5, cmap='gray')  # Denormalisasi
    plt.title(classes[labels[i]])
    plt.axis('off')
plt.show()
```



Arsitektur CNN

```python
# **Langkah 3: Membuat Arsitektur CNN**
# CNN Model
def create_cnn_model(kernel_size, pooling_type):
    class CNN(nn.Module):
        def __init__(self):
            super(CNN, self).__init__()
            self.conv1 = nn.Conv2d(1, 32, kernel_size=kernel_size, stride=1, padding=kernel_size//2)
            self.pool = pooling_type(kernel_size=2, stride=2)
            self.conv2 = nn.Conv2d(32, 64, kernel_size=kernel_size, stride=1, padding=kernel_size//2)
            self.fc1 = nn.Linear(64 * 7 * 7, 512)
            self.fc2 = nn.Linear(512, 10)

        def forward(self, x):
            x = self.pool(F.relu(self.conv1(x)))
            x = self.pool(F.relu(self.conv2(x)))
            x = x.view(-1, 64 * 7 * 7)
            x = F.relu(self.fc1(x))
            x = self.fc2(x)
            return x

    return CNN()
```

Implementasi model Convolutional Neural Network (CNN) menggunakan kelas nn.Module dari PyTorch. Fungsi create_cnn_model mendefinisikan model CNN dengan dua lapisan konvolusi, jenis pooling yang dapat disesuaikan, dan dua lapisan fully connected. Pada metode __init__, lapisan pertama (conv1) mengambil masukan dengan satu saluran (grayscale) dan menghasilkan 32 fitur, diikuti oleh pooling dengan jenis yang ditentukan (pooling_type, seperti MaxPooling atau AveragePooling). Lapisan kedua (conv2) menerima 32 fitur dari lapisan sebelumnya dan menghasilkan 64 fitur. Output dari lapisan konvolusi kedua diratakan (flattening) sebelum diteruskan ke lapisan fully connected pertama (fc1), yang mengubah fitur menjadi 512 unit, dan kemudian ke lapisan terakhir (fc2) yang menghasilkan output 10 kelas (misalnya, untuk klasifikasi multi-kelas).

Metode forward mendefinisikan jalannya data melalui model. Setiap lapisan konvolusi diikuti oleh fungsi aktivasi ReLU dan pooling. Setelah semua fitur diekstraksi, data diratakan dan diproses oleh lapisan fully connected untuk menghasilkan prediksi akhir. Model ini fleksibel karena mendukung penyesuaian ukuran kernel (kernel_size) dan jenis pooling, memungkinkan penggunaan pada berbagai dataset atau kebutuhan eksperimen. Fungsi ini mengembalikan objek model CNN yang siap dilatih.

Eksperimen

```python
# **Langkah 5: Eksperimen dengan Variasi Parameter**
# Configurations
kernel_sizes = [3, 5, 7]
pooling_types = [nn.MaxPool2d, nn.AvgPool2d]
num_epochs_list = [5, 50, 100]
optimizers = ["SGD", "RMSProp", "Adam"]

best_hyperparameters = None
best_accuracy = 0

# Running experiments
for kernel_size in kernel_sizes:
    for pooling_type in pooling_types:
        for optimizer_name in optimizers:
            for num_epochs in num_epochs_list:
                print(f"\n--- Kernel Size: {kernel_size}, Pooling: {pooling_type.__name__}, Optimizer: {optimizer_name}, Epochs: {num_epochs} ---")
                model = create_cnn_model(kernel_size, pooling_type)
                trained_model, val_loss, accuracy = train_and_evaluate(model, optimizer_name, num_epochs)

                if accuracy > best_accuracy:
                    best_accuracy = accuracy
                    best_hyperparameters = {
                        "kernel_size": kernel_size,
                        "pooling_type": pooling_type.__name__,
                        "optimizer": optimizer_name,
                        "num_epochs": num_epochs
                    }
```

Implementasi untuk eksperimen hyperparameter tuning pada model Convolutional Neural Network (CNN). Tujuannya adalah menguji berbagai kombinasi parameter seperti ukuran kernel, jenis pooling, optimizer, dan jumlah epoch untuk menemukan konfigurasi terbaik.

Output

```
--- Kernel Size: 3, Pooling: MaxPool2d, Optimizer: SGD, Epochs: 5 ---
Epoch 1/5, Loss: 420.1710, Val Loss: 0.3714, Accuracy: 86.29%
Epoch 2/5, Loss: 248.4178, Val Loss: 0.3286, Accuracy: 88.04%
Epoch 3/5, Loss: 210.5945, Val Loss: 0.2903, Accuracy: 89.20%
Epoch 4/5, Loss: 188.8003, Val Loss: 0.2732, Accuracy: 90.04%
Epoch 5/5, Loss: 169.7315, Val Loss: 0.2637, Accuracy: 90.62%

--- Kernel Size: 3, Pooling: MaxPool2d, Optimizer: SGD, Epochs: 50 ---
Epoch 1/50, Loss: 436.2476, Val Loss: 0.3739, Accuracy: 86.74%
Epoch 2/50, Loss: 251.8692, Val Loss: 0.3048, Accuracy: 88.80%
Epoch 3/50, Loss: 212.0351, Val Loss: 0.2808, Accuracy: 89.70%
Epoch 4/50, Loss: 187.7913, Val Loss: 0.2640, Accuracy: 90.22%
Epoch 5/50, Loss: 167.2862, Val Loss: 0.2483, Accuracy: 90.77%
Epoch 6/50, Loss: 154.3108, Val Loss: 0.2493, Accuracy: 90.84%
Epoch 7/50, Loss: 141.2485, Val Loss: 0.2464, Accuracy: 91.06%
Epoch 8/50, Loss: 128.1901, Val Loss: 0.2392, Accuracy: 90.95%
Epoch 9/50, Loss: 117.8433, Val Loss: 0.2438, Accuracy: 91.29%
Epoch 10/50, Loss: 106.8845, Val Loss: 0.2616, Accuracy: 91.22%
Epoch 11/50, Loss: 94.7037, Val Loss: 0.2232, Accuracy: 92.12%
Epoch 12/50, Loss: 84.0698, Val Loss: 0.2284, Accuracy: 92.06%
Epoch 13/50, Loss: 74.2692, Val Loss: 0.2354, Accuracy: 91.65%
Epoch 14/50, Loss: 65.1416, Val Loss: 0.2507, Accuracy: 91.71%
Epoch 15/50, Loss: 57.6537, Val Loss: 0.2416, Accuracy: 92.02%
Epoch 16/50, Loss: 48.7086, Val Loss: 0.2597, Accuracy: 91.84%
...
Epoch 15/100, Loss: 29.2453, Val Loss: 0.3807, Accuracy: 91.47%
Epoch 16/100, Loss: 27.9132, Val Loss: 0.4177, Accuracy: 91.42%
Epoch 17/100, Loss: 25.4034, Val Loss: 0.4209, Accuracy: 91.58%
Early stopping triggered.
```

```python
    # Display best hyperparameters
    print("\nBest Hyperparameters:")
    print(best_hyperparameters)
    print(f"Best Validation Accuracy: {best_accuracy:.2f}%")
```

```
Best Hyperparameters:
{'kernel_size': 3, 'pooling_type': 'AvgPool2d', 'optimizer': 'Adam', 'num_epochs': 50}
Best Validation Accuracy: 92.62%
```

Hasil Hyperparameter Terbaik

- kernel_size: 3
  Ukuran kernel terbaik adalah 3x3, yang memberikan keseimbangan antara kompleksitas fitur dan efisiensi komputasi.
- pooling_type: AvgPool2d
  Average pooling memberikan hasil terbaik, menunjukkan bahwa mengambil rata-rata dalam pooling lebih efektif untuk tugas ini dibandingkan max pooling.
- optimizer: Adam
  Optimizer Adam menghasilkan konvergensi yang baik dan memberikan akurasi tertinggi.
- num_epochs: 50
  Jumlah epoch 50 cukup untuk mencapai performa terbaik tanpa overfitting atau underfitting.
- Best Validation Accuracy: 92.62%
  Model mencapai akurasi validasi tertinggi sebesar 92.62%, yang menunjukkan performa model yang sangat baik pada data validasi.

Kesimpulan

Hasil ini mengidentifikasi kombinasi kernel size 3, average pooling, optimizer Adam, dan 50 epochs sebagai konfigurasi terbaik untuk model CNN. Akurasi validasi sebesar 92.62% menunjukkan bahwa model ini mampu menangani dataset dengan baik. Proses ini menegaskan pentingnya hyperparameter tuning dalam membangun model deep learning yang optimal.