Tugas 12

Nama : muhammad makhlufi makbullah

Kelas : TK 45 01

NIM : 1103210171

1. Import Library

import torch

import torch.nn as nn

import torch.optim as optim

import torchvision

import torchvision.transforms as transforms

from torch.utils.data import DataLoader, random_split

from torch.optim.lr_scheduler import StepLR

import torch.nn.functional as F

- **torch dan torch.nn**: Digunakan untuk mendefinisikan arsitektur jaringan saraf dan fungsi pelatihan.
- **torch.optim**: Berisi optimizer seperti **SGD**, **RMSProp**, dan **Adam**.
- **torchvision**: Menyediakan dataset seperti CIFAR-10 dan alat transformasi data.
- **transforms**: Untuk normalisasi gambar (seperti skala piksel ke antara [-1, 1]).
- **DataLoader**: Membantu memuat data secara efisien dalam batch.
- **StepLR**: Scheduler untuk mengubah learning rate secara bertahap selama pelatihan.

---

2. Early Stopping

class EarlyStopping:

def __init__(self, patience=5, delta=0):

...

def __call__(self, val_loss):

...

- **Apa Itu Early Stopping?**
  Callback ini menghentikan pelatihan secara otomatis jika validasi loss tidak membaik selama beberapa epoch berturut-turut (ditentukan oleh parameter patience).
- **Fungsi Penting**:
  - Menghindari overfitting.
  - Menghemat waktu pelatihan dengan menghentikan proses lebih awal.

---

3. Dataset CIFAR-10

```
def load_data(batch_size=64):
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])
    ...
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    ...
    return train_loader, val_loader, test_loader
```

- **CIFAR-10**: Dataset berisi 60.000 gambar (10 kelas) berukuran 32x32 piksel.
- **Transformasi Data**:
  - **ToTensor()**: Mengubah gambar menjadi tensor PyTorch.
  - **Normalize()**: Normalisasi agar piksel berada di rentang [-1, 1].
- **Splitting**:
  - Dataset dilatih dipecah menjadi **training (80%)** dan **validation (20%)** menggunakan fungsi random_split.

---

4. Arsitektur CNN

```
class CNN(nn.Module):
    def __init__(self, kernel_size=3, pooling='max'):
```

super(CNN, self).__init__()

...

def forward(self, x):

...

- **Conv2D Layers**:
  - self.conv1: Layer pertama dengan 32 filter.
  - self.conv2: Layer kedua dengan 64 filter.
    Setiap layer convolution menggunakan ReLU (Rectified Linear Unit) sebagai fungsi aktivasi.
- **Pooling**:
  - Tersedia dua pilihan: **MaxPooling** atau **AveragePooling**.
- **Fully Connected Layers**:
  - fc1: Menghubungkan fitur hasil convolution ke dimensi 128.
  - fc2: Layer output (10 unit) untuk klasifikasi 10 kelas.

---

5. Fungsi Pelatihan

def train_model(model, train_loader, val_loader, optimizer, scheduler, criterion, num_epochs=50, patience=5):

...

- **Parameter**:
  - **model**: CNN yang akan dilatih.
  - **optimizer**: Salah satu dari **SGD**, **RMSProp**, atau **Adam**.
  - **scheduler**: Mengubah learning rate berdasarkan strategi tertentu.
  - **criterion**: Fungsi loss (CrossEntropy digunakan karena tugas klasifikasi).
  - **num_epochs**: Jumlah maksimum epoch pelatihan.
- **Early Stopping**:
  - Dihentikan jika validasi loss tidak membaik selama sejumlah epoch.

Proses Pelatihan:

1. **Training**: Model dilatih menggunakan batch data dari training set.

    - Optimizer menghitung gradien dan memperbarui bobot.
2. **Validation**: Model dievaluasi menggunakan validation set.
    - Validation loss dihitung untuk memantau performa dan memicu early stopping.

---

6. Evaluasi Model

def evaluate_model(model, test_loader):

...

- **Evaluasi Performa**:
    - Model diuji pada data test set yang tidak pernah dilihat sebelumnya.
    - Mengukur **akurasi** sebagai persentase prediksi benar.

---

7. Optimizer (SGD, RMSProp, Adam)

1. SGD:

optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

- **Keunggulan**: Cepat dan sederhana.
- **Kekurangan**: Bisa terjebak di local minima.

2. RMSProp:

optimizer = optim.RMSprop(model.parameters(), lr=0.01, alpha=0.9)

- **Keunggulan**: Menangani learning rate yang dinamis pada setiap parameter.
- **Kekurangan**: Sensitif terhadap hyperparameter.

3. Adam:

optimizer = optim.Adam(model.parameters(), lr=0.001)

- **Keunggulan**: Kombinasi terbaik dari momentum (SGD) dan learning rate adaptif (RMSProp).
- **Kekurangan**: Kadang terlalu cepat konvergen ke solusi suboptimal.

---

8. Learning Rate Scheduler

scheduler = StepLR(optimizer, step_size=10, gamma=0.1)

- **Fungsi**: Mengurangi learning rate setiap 10 epoch dengan faktor 0.1.

- **Tujuan**: Membantu model mencapai konvergensi stabil setelah pembelajaran awal.

---

9. Performa dan Analisis

Setelah menjalankan kode untuk **SGD**, **RMSProp**, dan **Adam**, catat hasil berikut:

1. **Akurasi pada Test Set**: Bandingkan akurasi di ketiga optimizer.
2. **Grafik Loss**: Visualisasi loss (train vs validation) untuk melihat pola konvergensi.
3. **Durasi Pelatihan**: Bandingkan waktu pelatihan untuk melihat efisiensi.

Kesimpulan untuk Laporan

1. **Arsitektur CNN**:
    - Dua convolutional layer dengan ReLU dan pooling.
    - Fully connected layer untuk klasifikasi.
2. **Eksperimen Hyperparameter**:
    - **Kernel Size**: Mencoba 3x3 (default), ulangi eksperimen untuk 5x5 dan 7x7.
    - **Pooling**: MaxPooling vs AvgPooling.
    - **Optimizer**: SGD, RMSProp, Adam.
    - **Epoch**: Bandingkan performa pada 5, 50, 100, 250, dan 350 epoch.
3. **Hasil**:
    - Visualisasi akurasi dan loss untuk semua kombinasi hyperparameter.
    - Laporan waktu pelatihan dan konvergensi tiap optimizer.