Laporan Analisis Klasifikasi Angka Tulis Tangan MNIST
Classification with CNN

Machine Learning 2

Anggota Kelompok :

Iqbal Maulana
Ahmad Nugrahadi
Farid Fajar
Abdullah Sukma Jati
Gusti Ahmad
Atha Fatur

1. Pendahuluan

Laporan ini menyajikan analisis klasifikasi angka tulisan tangan menggunakan dataset MNIST dan model Convolutional Neural Network (CNN). Dataset MNIST adalah benchmark standar dalam pembelajaran mesin dan visi komputer, yang terdiri dari 60.000 gambar pelatihan dan 10.000 gambar pengujian angka tulisan tangan (0-9). Tujuan dari proyek ini adalah untuk membangun dan melatih model CNN yang mampu mengklasifikasikan angka tulisan tangan dengan akurasi tinggi.

2. Deskripsi Proyek

Proyek ini bertujuan untuk mengembangkan dan mengevaluasi model **Convolutional Neural Network (CNN)** untuk klasifikasi angka tulisan tangan menggunakan dataset **MNIST**. Model ini dibangun di PyTorch, mengintegrasikan lapisan konvolusi, *pooling*, dan *fully connected* untuk secara efektif mempelajari fitur dari gambar angka. Data MNIST diproses dan dinormalisasi untuk persiapan, lalu model dilatih selama 10 *epoch* menggunakan *optimizer* Adam dan *loss Cross-Entropy*. Hasil menunjukkan akurasi pengujian yang tinggi (lebih dari 98%), menegaskan kemampuan model dalam mengenali angka tulisan tangan dengan presisi.

3. Metodologi

3.1. Persiapan Data

Dataset MNIST dimuat menggunakan kelas **MNISTDataset** kustom. Kelas ini bertanggung jawab untuk:

- Membaca file gambar dan label MNIST dari format idx3-ubyte dan idx1-ubyte masing-masing.
- Melakukan transformasi data yang meliputi:
  - Mengubah ukuran gambar menjadi 28x28 piksel

    `transforms.Resize((self.image_size, self.image_size))`:
  - Mengubah gambar menjadi tensor PyTorch.

    `transforms.ToTensor()`:
  - Menormalisasi piksel gambar menggunakan mean dan standar deviasi umum untuk dataset MNIST

    `transforms.Normalize((0.1307,), (0.3081,))`:
- Data kemudian dibagi menjadi set pelatihan (**train_dataset**) dan set pengujian (**test_dataset**). DataLoader digunakan untuk membuat *batch* data, dengan ukuran *batch* 16 untuk pelatihan dan pengujian, serta *shuffle* data pelatihan untuk setiap *epoch*.

3.2. Arsitektur Model

Model CNN (**MnistClassifier**) dirancang dengan arsitektur sebagai berikut:

- **Lapisan Konvolusi Pertama (conv1):** 1 *channel* input, 16 *output channels*, ukuran *kernel* 3x3, *padding* 1. Diikuti dengan fungsi aktivasi **ReLU** dan **Max Pooling** (ukuran *kernel* 2x2, *stride* 2).
- **Lapisan Konvolusi Kedua (conv2):** 16 *input channels*, 32 *output channels*, ukuran *kernel* 3x3, *padding* 1. Diikuti dengan fungsi aktivasi **ReLU** dan **Max Pooling** (ukuran *kernel* 2x2, *stride* 2).
- **Lapisan Konvolusi Ketiga (conv3):** 32 *input channels*, 64 *output channels*, ukuran *kernel* 3x3, *padding* 1. Diikuti dengan fungsi aktivasi **ReLU** dan **Max Pooling** (ukuran *kernel* 2x2, *stride* 2).
- **Lapisan Fully Connected (fc1):** Lapisan ini menerima *output* dari lapisan konvolusi yang diratakan (**flattened**) dan memproyeksikannya ke 64 *output features*. Diikuti dengan fungsi aktivasi **ReLU**. Ukuran *input* untuk fc1 adalah 64 * 3 * 3, menunjukkan dimensi spasial setelah tiga lapisan konvolusi dan pooling.
- **Lapisan Output (fc2):** 64 *input features*, 10 *output features* (sesuai dengan jumlah kelas angka 0-9).

3.3. Konfigurasi Pelatihan

- **Perangkat (`device`):** Model dilatih menggunakan GPU jika tersedia (CUDA), jika tidak, akan menggunakan CPU.
- **Fungsi *Loss* (`criterion`):** `nn.CrossEntropyLoss()` digunakan sebagai fungsi *loss,* yang cocok untuk masalah klasifikasi multi-kelas.
- **Pengoptimal (`optimizer`):** `optim.Adam()` dipilih sebagai pengoptimal dengan *learning rate* 1e-3 dan *weight decay* 1e-5 untuk regularisasi.
- **Jumlah Epoch:** Model dilatih selama **10 *epoch***.

3.4. Pelatihan dan Evaluasi

Selama setiap *epoch*:

- **Fase Pelatihan:** Model diatur ke mode pelatihan (model.train()). Data diiterasi dalam *batch*, *forward pass* dilakukan, *loss* dihitung, *backward pass* dilakukan untuk menghitung gradien, dan parameter model diperbarui. tqdm digunakan untuk menampilkan progress bar pelatihan.
- **Fase Evaluasi:** Model diatur ke mode evaluasi (model.eval()). *Forward pass* dilakukan tanpa perhitungan gradien (torch.no_grad()). Akurasi dihitung dengan membandingkan prediksi model dengan label sebenarnya. *Loss* pelatihan dan pengujian, serta akurasi pengujian, dicatat dan dicetak di setiap akhir *epoch*.

3.5. Penyimpanan Model

Setelah pelatihan selesai, *state dictionary* model, *optimizer*, *loss* pelatihan, dan *loss* pengujian disimpan ke file modelku.pth untuk penggunaan di masa mendatang.

4. Hasil

Berikut adalah *output* dari proses pelatihan selama 10 *epoch*:

```
Epoch 1: 100%|██████████| 3750/3750 [00:19<00:00, 192.73it/s]
Epoch 1: Train Loss 0.1293, Test Loss 0.0381, Test Acc 0.9871
Epoch 2: 100%|██████████| 3750/3750 [00:18<00:00, 204.56it/s]
Epoch 2: Train Loss 0.0476, Test Loss 0.0427, Test Acc 0.9857
Epoch 3: 100%|██████████| 3750/3750 [00:18<00:00, 207.28it/s]
Epoch 3: Train Loss 0.0341, Test Loss 0.0417, Test Acc 0.9860
Epoch 4: 100%|██████████| 3750/3750 [00:18<00:00, 204.86it/s]
Epoch 4: Train Loss 0.0263, Test Loss 0.0308, Test Acc 0.9903
Epoch 5: 100%|██████████| 3750/3750 [00:18<00:00, 205.56it/s]
Epoch 5: Train Loss 0.0214, Test Loss 0.0296, Test Acc 0.9904
Epoch 6: 100%|██████████| 3750/3750 [00:18<00:00, 205.65it/s]
Epoch 6: Train Loss 0.0191, Test Loss 0.0396, Test Acc 0.9894
Epoch 7: 100%|██████████| 3750/3750 [00:18<00:00, 204.30it/s]
Epoch 7: Train Loss 0.0172, Test Loss 0.0381, Test Acc 0.9887
Epoch 8: 100%|██████████| 3750/3750 [00:17<00:00, 208.35it/s]
Epoch 8: Train Loss 0.0156, Test Loss 0.0325, Test Acc 0.9898
Epoch 9: 100%|██████████| 3750/3750 [00:18<00:00, 206.78it/s]
Epoch 9: Train Loss 0.0120, Test Loss 0.0442, Test Acc 0.9891
Epoch 10: 100%|██████████| 3750/3750 [00:18<00:00, 206.38it/s]
Epoch 10: Train Loss 0.0140, Test Loss 0.0526, Test Acc 0.9869
```

Dari hasil di atas, dapat diamati bahwa:

- Jumlah total parameter dalam model adalah **27.154**, dan semuanya dapat dilatih (*trainable*).
- **Loss pelatihan** secara konsisten menurun sepanjang *epoch*, menunjukkan bahwa model belajar dengan baik dari data pelatihan.
- **Loss pengujian** juga menunjukkan tren penurunan, meskipun ada sedikit fluktuasi.
- **Akurasi pengujian** mencapai puncaknya sekitar **99.04%** pada *epoch* 5 dan tetap tinggi di atas 98% sepanjang pelatihan. Ini menunjukkan kinerja klasifikasi yang sangat baik pada data yang belum pernah dilihat sebelumnya.

4. Kesimpulan

Model CNN yang diimplementasikan berhasil mengklasifikasikan angka tulisan tangan MNIST dengan akurasi yang sangat tinggi, mencapai lebih dari 99% pada set pengujian. Arsitektur CNN dengan beberapa lapisan konvolusi dan *pooling*, diikuti oleh lapisan *fully connected*, terbukti efektif dalam mengekstraksi fitur-fitur penting dari gambar angka tulisan tangan. Normalisasi data dan penggunaan pengoptimal Adam juga berkontribusi pada kinerja model yang kuat.