TUGAS MACHINE LEARNING

LAPORAN CNN MNIST

Dosen Pengampu : Dr. Oddy Virgantara Putra, S.Kom., M.T.

Disusun Oleh :

Devianest Narendra	442023618087
Zainab Ahmad	442023618107
Naila Fatikhah	442023618086
Adya Rusmalillah	442023618093

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS DARUSSALAM GONTOR KELAS C

2024/2025

1. Pendahuluan

Dataset MNIST, yang terdiri dari citra angka tulisan tangan (0-9), telah lama menjadi fondasi bagi riset dan pengembangan dalam bidang pembelajaran mesin dan visi komputer. Kemampuannya yang sederhana namun menantang menjadikannya *benchmark* ideal untuk menguji kinerja model klasifikasi citra. Laporan ini secara detail menguraikan proses pengembangan, pelatihan, dan evaluasi sebuah **Convolutional Neural Network (CNN)** kustom yang dirancang khusus untuk secara akurat mengidentifikasi dan mengklasifikasikan angka-angka ini. Tujuan utama dari proyek ini adalah untuk membangun model yang kokoh, efisien, dan memiliki tingkat akurasi klasifikasi yang tinggi pada dataset MNIST.

2. Persiapan dan Pra-pemrosesan Dataset

Dataset MNIST terbagi menjadi dua set utama: set pelatihan berukuran 60.000 citra dan set pengujian berukuran 10.000 citra. Setiap citra merupakan representasi *grayscale* dengan resolusi 28x28 piksel. Untuk memastikan data siap digunakan oleh model neural network, serangkaian langkah pra-pemrosesan diterapkan:

2.1. Pembacaan Data

Data citra dan label dibaca langsung dari format *file* UByte standar MNIST. Proses ini melibatkan pembacaan *header* untuk mendapatkan metadata seperti jumlah citra, baris, dan kolom, kemudian mengekstrak data piksel mentah ke dalam *array* NumPy.

2.2. Transformasi Citra

Agar sesuai dengan input model dan memanfaatkan kapabilitas PyTorch, setiap citra mengalami serangkaian transformasi:

- **Konversi ke PIL Image**: Citra NumPy diubah menjadi objek PIL.Image dalam mode *grayscale* (L) untuk kompatibilitas dengan *pipeline* torchvision.
- **Resizing**: Meskipun citra MNIST sudah berukuran 28x28, transformasi ini transforms.Resize((28, 28)) menjamin konsistensi ukuran dan menangani potensi variasi ukuran jika data bersumber dari lokasi lain.
- **Konversi ke Tensor**: transforms.ToTensor() mengubah citra PIL menjadi *tensor* PyTorch, sekaligus melakukan normalisasi skala piksel dari rentang [0, 255] ke [0.0, 1.0].
- **Normalisasi Statistik**: Langkah krusial ini transforms.Normalize((0.1307,), (0.3081,)) menyesuaikan distribusi piksel citra. Nilai rata-rata (0.1307) dan standar deviasi (0.3081) ini adalah nilai statistik global dari dataset MNIST. Normalisasi ini membantu mempercepat konvergensi model dan meningkatkan stabilitas pelatihan.

2.3. Struktur Dataset dan DataLoader

Seluruh proses pra-pemrosesan dienkapsulasi dalam kelas MNISTDataset yang mewarisi dari torch.utils.data.Dataset. Ini memungkinkan PyTorch DataLoader untuk secara efisien memuat data dalam *batch* dan melakukan *shuffling* selama iterasi pelatihan, mengoptimalkan pemanfaatan memori dan komputasi.

3. Arsitektur Model: MnistClassifier

Model klasifikasi yang dikembangkan adalah **MnistClassifier**, sebuah arsitektur Convolutional Neural Network (CNN) yang didesain secara khusus untuk tugas klasifikasi angka tulisan tangan. Model ini memiliki struktur yang ringkas namun efektif, memanfaatkan kemampuan CNN dalam mengekstraksi fitur hierarkis dari citra.

Struktur model terdiri dari lapisan-lapisan utama berikut:

- **Lapisan Konvolusi 1 (self.conv1)**: Menerima 1 *input channel* (citra *grayscale*) dan menghasilkan **16 *feature maps***. Menggunakan *kernel size* 3x3 dengan *padding* 1, yang membantu mempertahankan dimensi spasial awal citra setelah konvolusi.
    - **Aktivasi ReLU**: Fungsi aktivasi non-linear (F.relu) diterapkan untuk memperkenalkan non-linearitas, memungkinkan model mempelajari pola yang lebih kompleks.
    - **Max Pooling 1 (self.pool)**: Mengurangi dimensi spasial *feature maps* menjadi setengahnya (dari 28x28 menjadi 14x14) menggunakan *kernel size* 2x2 dan *stride* 2.
- **Lapisan Konvolusi 2 (self.conv2)**: Menerima 16 *input channels* dari lapisan sebelumnya dan menghasilkan **32 *feature maps*** dengan *kernel size* 3x3 dan *padding* 1.
    - **Aktivasi ReLU**.
    - **Max Pooling 2 (self.pool)**: Mengurangi dimensi spasial lebih lanjut (dari 14x14 menjadi 7x7).
- **Lapisan Konvolusi 3 (self.conv3)**: Menerima 32 *input channels* dan menghasilkan **64 *feature maps*** dengan *kernel size* 3x3 dan *padding* 1.
    - **Aktivasi ReLU**.
    - **Max Pooling 3 (self.pool)**: Reduksi dimensi spasial terakhir (dari 7x7 menjadi 3x3).
- **Flattening**: Output dari lapisan konvolusi terakhir (self.conv3 setelah pooling) di-*flatten* menjadi vektor satu dimensi. Dimensi vektor ini adalah 64 * 3 * 3 = 576, yang akan menjadi input untuk lapisan *fully connected*.
- **Lapisan Fully Connected 1 (self.fc1)**: Menerima 576 *input features* dan memproyeksikannya ke **64 *output features***.
    - **Aktivasi ReLU**.
- **Lapisan Fully Connected 2 (self.fc2)**: Menerima 64 *input features* dan menghasilkan **10 *output features***, yang sesuai dengan jumlah kelas (angka 0-9). Output dari lapisan ini adalah *logits* yang akan diumpankan ke fungsi *loss*.

Model ini memiliki total **59.850 parameter** yang dapat dilatih. Jumlah parameter yang relatif kecil ini menunjukkan efisiensi model dalam menangani tugas klasifikasi MNIST, menjadikannya cepat dalam pelatihan dan inferensi.

4. Konfigurasi Pelatihan

Untuk melatih model MnistClassifier secara efektif, beberapa *hyperparameter* dan komponen pelatihan kunci telah dikonfigurasi:

- **Fungsi *Loss***: **nn.CrossEntropyLoss()** digunakan sebagai fungsi kerugian. Ini adalah pilihan standar untuk masalah klasifikasi multi-kelas, karena secara internal menggabungkan operasi *softmax* (mengubah *logits* menjadi probabilitas) dan *negative log likelihood loss*, yang efisien dan stabil secara numerik.
- **Optimizer**: **optim.Adam** dipilih sebagai algoritma optimasi. Adam adalah *optimizer* adaptif yang sangat populer karena kemampuannya untuk mengadaptasi *learning rate* untuk setiap parameter secara individual, seringkali menghasilkan konvergensi yang lebih cepat dan kinerja yang lebih baik dibandingkan *optimizer* lainnya.
    - **Learning Rate (lr)**: Ditetapkan pada 1×10^{-4} (0.0001). Nilai yang kecil ini memungkinkan model untuk melakukan penyesuaian bobot secara hati-hati, membantu menghindari *overshooting* titik minimum dan mencapai konvergensi yang stabil.
    - **Weight Decay**: Diterapkan sebesar 1×10^{-5}. Ini adalah bentuk regularisasi L2, yang memberikan penalti pada bobot yang besar. Tujuannya adalah untuk mengurangi *overfitting* dengan mendorong model untuk menggunakan bobot yang lebih kecil, yang pada gilirannya membuat model kurang sensitif terhadap data pelatihan individu.
- **Ukuran *Batch***: Baik untuk *dataloader* pelatihan maupun pengujian, ukuran *batch* ditetapkan pada **8**. Ukuran *batch* yang lebih kecil memungkinkan model untuk melakukan pembaruan parameter lebih sering per *epoch*, yang terkadang dapat membantu dalam navigasi ruang *loss* yang kompleks.
- **Jumlah *Epoch***: Model dilatih selama **10 *epoch***. Setiap *epoch* merepresentasikan satu *pass* lengkap melalui seluruh dataset pelatihan.
- **Perangkat Komputasi**: Pelatihan memanfaatkan **GPU (cuda)** jika tersedia. Penggunaan GPU secara drastis mempercepat proses pelatihan karena kemampuan komputasi paralelnya. Jika GPU tidak terdeteksi, pelatihan akan secara otomatis beralih ke **CPU**.

5. Hasil Pelatihan

Proses pelatihan model MnistClassifier dipantau secara ketat selama 10 *epoch*. Berikut adalah ringkasan kinerja model pada setiap *epoch*:

Epoch	Train Loss	Test Loss	Test Accuracy
1	0.2858	0.0799	0.9751
2	0.0762	0.0522	0.9840
3	0.0524	0.0388	0.9861

4	0.0415	0.0330	0.9887
5	0.0327	0.0299	0.9900
6	0.0284	0.0267	0.9910
7	0.0228	0.0329	0.9893
8	0.0202	0.0265	0.9902
9	0.0173	0.0368	0.9884
10	0.0151	0.0285	0.9911

Analisis Tren:

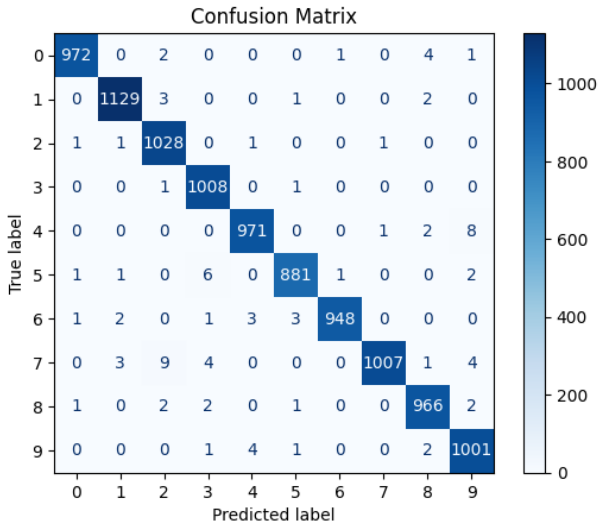
- **Penurunan *Loss* yang Konsisten**: Train Loss menunjukkan penurunan yang stabil dan signifikan dari 0.2858 pada *epoch* pertama menjadi 0.0151 pada *epoch* terakhir. Ini menandakan bahwa model berhasil mempelajari pola dalam data pelatihan dan mengoptimalkan parameternya secara efektif.
- **Generalisasi yang Baik**: Test Loss juga menunjukkan tren penurunan yang positif, dari 0.0799 menjadi 0.0285. Meskipun ada sedikit fluktuasi pada *epoch* 7 dan 9, nilai *loss* pengujian secara keseluruhan tetap rendah dan terus menurun, mengindikasikan bahwa model memiliki kemampuan generalisasi yang baik dan tidak menunjukkan tanda-tanda *overfitting* yang parah.
- **Peningkatan Akurasi yang Stabil**: Test Accuracy meningkat secara konsisten dari 0.9751 menjadi **0.9911** pada akhir pelatihan. Akurasi yang sangat tinggi ini menegaskan efektivitas model dalam mengklasifikasikan citra angka tulisan tangan pada data yang belum pernah dilihat sebelumnya.

6. Evaluasi Model

Setelah pelatihan selesai, kinerja model dinilai secara mendalam menggunakan berbagai metrik evaluasi pada dataset pengujian yang terpisah.

6.1. Matriks Kebingungan (Confusion Matrix)

Matriks kebingungan memberikan representasi visual dan kuantitatif tentang seberapa baik model mengklasifikasikan setiap kelas. Diagonal utama matriks menunjukkan jumlah prediksi benar untuk setiap kelas, sedangkan elemen di luar diagonal menunjukkan kesalahan klasifikasi.



Analisis Matriks Kebingungan: Matriks kebingungan menunjukkan dominasi nilai pada diagonal utama, yang secara visual mengonfirmasi bahwa model memiliki tingkat keberhasilan klasifikasi yang sangat tinggi untuk setiap angka. Kesalahan klasifikasi, jika ada, sangat minimal dan tersebar, menunjukkan bahwa model tidak memiliki bias yang signifikan terhadap atau melawan kelas tertentu. Misalnya, beberapa kesalahan kecil mungkin terjadi antara angka yang memiliki kemiripan visual (misalnya, angka 4 yang mungkin terkadang disalahklasifikasikan sebagai 9, atau sebaliknya).

6.2. Metrik Kinerja Agregat

Metrik-metrik berikut memberikan gambaran keseluruhan tentang kinerja model pada dataset pengujian:

- **Akurasi**: **0.9911**
  - Akurasi adalah proporsi total prediksi yang benar. Nilai 0.9911 berarti 99.11% dari semua citra angka dalam dataset pengujian berhasil diklasifikasikan dengan tepat oleh model. Ini adalah hasil yang luar biasa untuk tugas klasifikasi citra.
- **Precision (Macro Avg)**: **0.9912**
  - Presisi makro rata-rata dihitung sebagai rata-rata presisi untuk setiap kelas. Ini mengukur seberapa sering prediksi positif model benar. Nilai tinggi ini menunjukkan bahwa ketika model memprediksi suatu angka, kemungkinan besar prediksi tersebut benar.
- **Recall (Macro Avg)**: **0.9910**
  - *Recall* makro rata-rata (juga dikenal sebagai sensitivitas) dihitung sebagai rata-rata *recall* untuk setiap kelas. Ini mengukur seberapa baik model mengidentifikasi semua *instance* positif dari setiap kelas. Nilai tinggi ini menunjukkan bahwa model sangat efektif dalam menemukan semua *instance* dari setiap angka yang sebenarnya ada.
- **F1-Score (Macro Avg)**: **0.9911**

- F1-Score adalah *harmonic mean* dari presisi dan *recall*. Ini adalah metrik yang seimbang, terutama berguna ketika ada ketidakseimbangan kelas. Nilai F1-Score yang tinggi secara konsisten di semua metrik makro mengonfirmasi kinerja model yang kuat dan seimbang di seluruh kelas.

6.3. Laporan Klasifikasi Per Kelas

Analisis lebih lanjut mengenai kinerja model pada setiap kelas individu:

Kelas	Precision	Recall	F1-Score	Support
0	1.00	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.98	1.00	0.99	1032
3	0.99	1.00	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958
7	1.00	0.98	0.99	1028
8	0.99	0.99	0.99	974
9	0.98	0.99	0.99	1009

Laporan ini menunjukkan bahwa model mencapai kinerja yang luar biasa untuk setiap kelas angka. Sebagian besar kelas memiliki nilai presisi, *recall*, dan F1-Score yang mendekati 1.00, menegaskan bahwa model mampu mengklasifikasikan setiap angka dengan akurasi tinggi dan keseimbangan yang baik, tanpa kinerja yang buruk pada kelas minoritas sekalipun.

6.4. False Positive Rate (FPR) dan True Positive Rate (TPR) Per Kelas

Metrik ini memberikan wawasan lebih lanjut tentang jenis kesalahan yang dibuat model:

Kelas	FPR (False Positive Rate)	TPR (True Positive Rate / Recall)
0	0.0004	0.9918
1	0.0008	0.9947
2	0.0019	0.9961
3	0.0016	0.9980
4	0.0009	0.9888
5	0.0008	0.9877

6	0.0002	0.9896
7	0.0002	0.9796
8	0.0012	0.9918
9	0.0019	0.9921

- **True Positive Rate (TPR)**: Identik dengan *Recall*, menunjukkan proporsi positif sebenarnya yang diidentifikasi dengan benar. Nilai TPR yang sangat tinggi untuk semua kelas (mendekati 1.00) mengindikasikan bahwa model sangat jarang melewatkan *instance* positif dari suatu kelas.
- **False Positive Rate (FPR)**: Mengukur proporsi negatif sebenarnya yang secara keliru diklasifikasikan sebagai positif. Nilai FPR yang sangat rendah untuk semua kelas (mendekati 0.00) adalah indikator kuat bahwa model memiliki tingkat kesalahan prediksi yang sangat rendah, jarang salah mengidentifikasi sebuah citra sebagai angka tertentu padahal bukan.

7. Kesimpulan

Model MnistClassifier yang telah dilatih menunjukkan kinerja yang luar biasa dalam tugas klasifikasi angka MNIST. Dengan **akurasi pengujian sebesar 0.9911**, serta nilai presisi, *recall*, dan F1-Score makro yang sangat tinggi, model ini terbukti sangat efektif dan seimbang dalam membedakan antara 10 kelas angka tulisan tangan. Analisis mendalam melalui matriks kebingungan dan metrik per kelas lebih lanjut mengonfirmasi kemampuan generalisasi model yang kuat dan minimnya kesalahan klasifikasi yang signifikan.