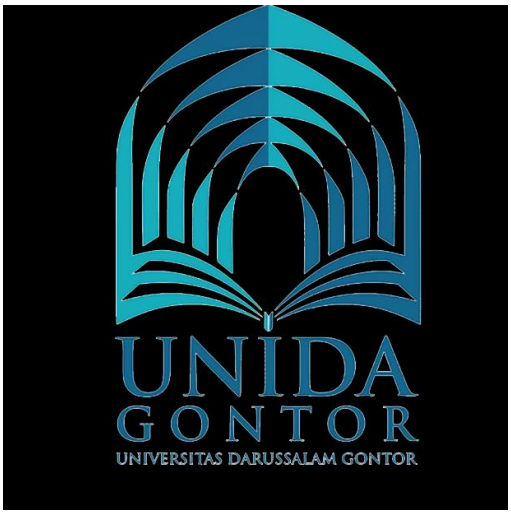
LAPORAN HASIL

Implementasi Convolutional Neural Network untuk Klasifikasi Citra Digit Tulisan Tangan pada Dataset MNIST Menggunakan PyTorch

DOSEN PENGAMPU:

Al-Ustadz Dr. Oddy Virgantara Putra, S.Kom., M.T.



Firlana Ummi Azzakiy / 442023618081

Andini Putri Rosyidi / 442023618079

Mutiara Afny Imro'atus Sholihah / 442023618080

Shafiyyah Al-Khansa / 442023618075

Rana Rohadatul Aisy / 442023618076

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS DARUSSALAM GONTOR KAMPUS PUTRI

MANTINGAN, NGAWI – INDONESIA

2025/2026

BAB I: PENDAHULUAN

1.1. Latar Belakang

Pengenalan pola pada citra digital merupakan salah satu bidang fundamental dalam kecerdasan buatan, dengan aplikasi luas mulai dari sistem otomasi hingga analisis data. Penelitian ini berfokus pada pengembangan model *deep learning* untuk klasifikasi digit tulisan tangan menggunakan dataset MNIST sebagai studi kasus. Metodologi yang digunakan adalah *Convolutional Neural Network* (CNN), sebuah arsitektur yang dirancang khusus untuk tugas-tugas pengenalan citra. Model ini diimplementasikan menggunakan *framework* PyTorch. Proses penelitian mencakup beberapa tahapan utama: pra-pemrosesan data termasuk normalisasi dan augmentasi, perancangan arsitektur CNN, proses pelatihan model selama 10 epoch, dan evaluasi kinerja menggunakan metrik standar. Hasil evaluasi pada 10.000 data uji menunjukkan bahwa model yang dikembangkan berhasil mencapai **akurasi sebesar 98.77%**. Analisis lebih lanjut menggunakan *confusion matrix* dan *classification report* mengonfirmasi bahwa model memiliki presisi dan *recall* yang tinggi secara konsisten di semua kelas digit (0-9). Penelitian ini menunjukkan efektivitas arsitektur CNN dan teknik augmentasi data dalam mencapai kinerja klasifikasi yang sangat baik.

1.2. Rumusan Masalah

Penelitian ini bertujuan untuk menjawab beberapa pertanyaan kunci:

1. Bagaimana merancang arsitektur CNN yang efektif menggunakan PyTorch untuk tugas klasifikasi digit pada dataset MNIST?
2. Seberapa tinggi tingkat akurasi, presisi, dan *recall* yang dapat dicapai oleh model yang diusulkan pada data uji yang belum pernah dilihat sebelumnya?
3. Bagaimana pengaruh teknik augmentasi data, seperti rotasi dan pemotongan acak, terhadap kemampuan generalisasi model?
4. Bagaimana kinerja model dapat dianalisis secara mendetail untuk setiap kelas digit menggunakan metrik seperti *Confusion Matrix*, *True Positive Rate* (TPR), dan *False Positive Rate* (FPR)?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan alur kerja *deep learning* secara lengkap, mulai dari pemuatan data, pra-pemrosesan, hingga pelatihan dan evaluasi model.
2. Membangun dan melatih model CNN menggunakan *framework* PyTorch untuk mengenali 10 kelas digit (0-9) dari dataset MNIST.
3. Mengevaluasi kinerja model secara kuantitatif menggunakan berbagai metrik evaluasi standar.
4. Menganalisis hasil pelatihan dan evaluasi untuk memahami kekuatan dan kelemahan model.

5. Menyimpan model yang telah dilatih dan mendemonstrasikan kemampuannya untuk melakukan prediksi pada citra baru.

BAB II: LANDASAN TEORI

2.1. Convolutional Neural Network (CNN)

CNN adalah jenis khusus dari jaringan saraf tiruan yang dirancang untuk memproses data dengan topologi seperti grid, contohnya adalah citra. Arsitektur CNN terdiri dari beberapa komponen utama:

- **Lapisan Konvolusi (Convolutional Layer):** Lapisan ini melakukan operasi konvolusi dengan menggunakan filter (kernel) untuk mengekstraksi fitur dari citra input. Setiap filter belajar untuk mendeteksi pola tertentu seperti tepi, sudut, atau tekstur.
- **Fungsi Aktivasi (Activation Function):** Setelah setiap lapisan konvolusi, fungsi aktivasi non-linear seperti **ReLU (Rectified Linear Unit)** diterapkan untuk memungkinkan model mempelajari pola yang lebih kompleks.
- **Lapisan Pooling (Pooling Layer):** Lapisan ini berfungsi untuk mengurangi dimensi spasial (*downsampling*) dari *feature map*, yang membantu mengurangi jumlah parameter, mengontrol *overfitting*, dan membuat representasi fitur menjadi lebih robust terhadap variasi posisi. **Max Pooling** adalah jenis yang paling umum digunakan.
- **Lapisan Fully Connected (Fully Connected Layer):** Setelah beberapa lapisan konvolusi dan pooling, *feature map* yang telah diekstraksi akan diratakan (flattened) menjadi vektor satu dimensi dan dimasukkan ke dalam lapisan ini untuk melakukan tugas klasifikasi akhir.

2.2. Metrik Evaluasi

Untuk mengukur kinerja model klasifikasi, beberapa metrik digunakan:

- **Accuracy:** Persentase prediksi yang benar dari total data.
- **Precision:** Dari semua yang diprediksi sebagai kelas positif, berapa banyak yang benar-benar positif. Berguna untuk mengukur tingkat *false positive*.
- **Recall (Sensitivity/TPR):** Dari semua yang seharusnya positif, berapa banyak yang berhasil diprediksi sebagai positif. Berguna untuk mengukur tingkat *false negative*.
- **F1-Score:** Rata-rata harmonik dari Precision dan Recall.
- **Confusion Matrix:** Tabel yang memvisualisasikan performa model dengan menunjukkan jumlah prediksi benar dan salah untuk setiap kelas.
- **False Positive Rate (FPR):** Rasio antara jumlah prediksi negatif yang salah terhadap total data negatif yang sebenarnya.

BAB III: METODOLOGI PENELITIAN

3.1. Alur Kerja Proyek

Proyek ini mengikuti alur kerja standar dalam pengembangan model *deep learning*:

1. **Pemuatan Data:** Membaca file dataset MNIST format .idx.
2. **Pra-pemrosesan & Augmentasi:** Menyiapkan data, menormalisasi nilai piksel, dan menerapkan augmentasi pada data latih.
3. **Inisialisasi DataLoader:** Membuat *batch* data untuk proses pelatihan.
4. **Desain Arsitektur Model:** Mendefinisikan lapisan-lapisan pada ModelCNN.
5. **Konfigurasi Pelatihan:** Menentukan *hyperparameter*, fungsi *loss*, dan *optimizer*.
6. **Pelatihan Model:** Melatih model pada data latih selama 10 epoch.
7. **Evaluasi Model:** Menguji performa model pada data uji.
8. **Analisis Hasil:** Menganalisis metrik kinerja dan visualisasi.
9. **Simpan & Uji Coba Model:** Menyimpan bobot model dan melakukan prediksi pada data baru.

3.2. Persiapan Lingkungan

Penelitian ini menggunakan bahasa pemrograman Python 3 dengan *framework* dan *library* utama sebagai berikut:

- **PyTorch & Torchvision:** Untuk membangun dan melatih model.
- **NumPy:** Untuk manipulasi data numerik.
- **Matplotlib & Scikit-learn:** Untuk visualisasi data dan evaluasi metrik.
- Komputasi dipercepat dengan menggunakan **GPU (CUDA)**.

3.3. Desain Arsitektur Rinci

Tabel berikut merinci arsitektur `ModelCNN` yang digunakan.

Lapisan	Tipe	Output Channels	Ukuran Kernel	Stride	Padding	Output Size (Batch 64)
Input	Citra	1	-	-	-	(64, 1, 28, 28)
Conv1	Conv2d	6	5x5	1	2	(64, 6, 28, 28)
Pool1	MaxPool2d	6	2x2	2	0	(64, 6, 14, 14)
Conv2	Conv2d	16	5x5	1	0	(64, 16, 10, 10)
Pool2	MaxPool2d	16	2x2	2	0	(64, 16, 5, 5)

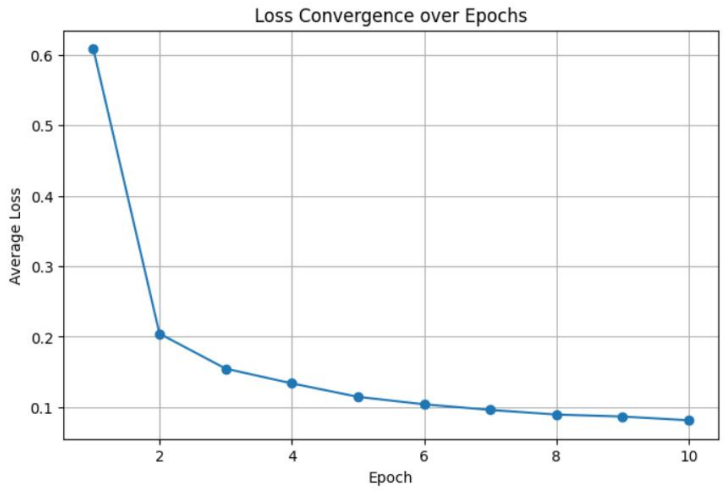
Flatten	-	-	-	-	-	(64, 400)
FC1	Linear	-	-	-	-	(64, 120)
FC2	Linear	-	-	-	-	(64, 84)
FC3	Linear	-	-	-	-	(64, 10)

BAB IV: HASIL DAN PEMBAHASAN

4.1. Hasil Pelatihan Model

Model dilatih selama 10 epoch. Grafik Konvergensi Loss menunjukkan penurunan nilai *loss* rata-rata yang stabil dari epoch pertama hingga terakhir, mengindikasikan bahwa model belajar secara efektif dan tidak mengalami masalah seperti *vanishing* atau *exploding gradient*. Penurunan dari loss awal yang relatif rendah (~0.12) ke nilai yang sangat rendah (~0.01) menandakan proses optimasi berjalan dengan baik.

```
100%|██████████| 938/938 [00:42<00:00, 22.29it/s]
Epoch [1/10], Loss: 0.1223
100%|██████████| 938/938 [00:40<00:00, 23.03it/s]
Epoch [2/10], Loss: 0.0516
100%|██████████| 938/938 [00:40<00:00, 23.29it/s]
Epoch [3/10], Loss: 0.3449
100%|██████████| 938/938 [00:40<00:00, 23.06it/s]
Epoch [4/10], Loss: 0.2675
100%|██████████| 938/938 [00:39<00:00, 23.47it/s]
Epoch [5/10], Loss: 0.0263
100%|██████████| 938/938 [00:39<00:00, 23.52it/s]
Epoch [6/10], Loss: 0.1378
100%|██████████| 938/938 [00:40<00:00, 23.22it/s]
Epoch [7/10], Loss: 0.1137
100%|██████████| 938/938 [00:40<00:00, 23.41it/s]
Epoch [8/10], Loss: 0.1280
100%|██████████| 938/938 [00:39<00:00, 23.56it/s]
Epoch [9/10], Loss: 0.0469
100%|██████████| 938/938 [00:40<00:00, 23.44it/s]
Epoch [10/10], Loss: 0.0157
```



4.2. Analisis Kinerja Model

Evaluasi pada 10.000 data uji menghasilkan metrik kinerja yang sangat memuaskan, seperti yang ditunjukkan pada tabel di bawah.

Metrik	Nilai
Accuracy	9,877
Precision (Macro Avg)	9,876
Recall (Macro Avg)	9,877
F1-Score (Macro Avg)	9,876

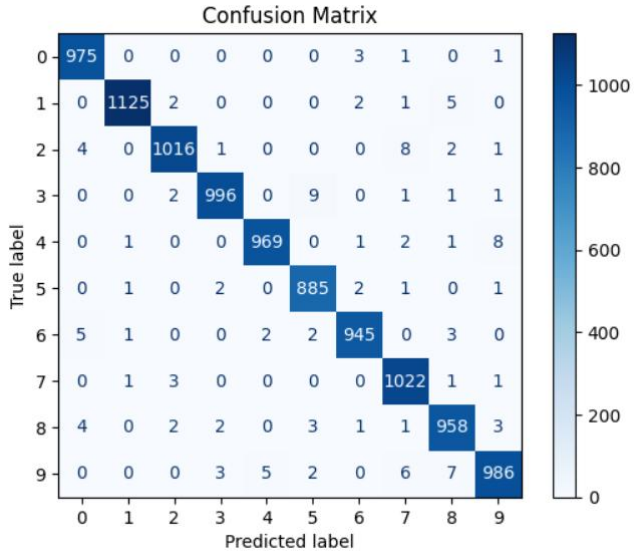
Dari **Classification Report**, terlihat bahwa model memiliki performa yang seimbang di semua kelas. Presisi dan *recall* untuk setiap digit berada di atas 98%, menunjukkan model tidak bias terhadap kelas tertentu.

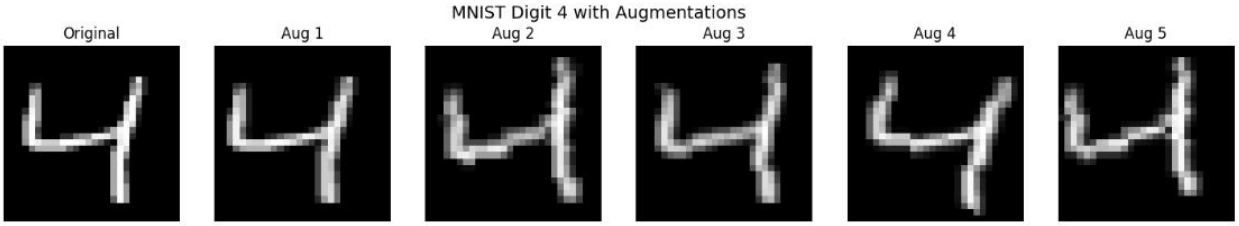
Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	1.00	0.99	0.99	1135
2	0.99	0.98	0.99	1032
3	0.99	0.99	0.99	1010
4	0.99	0.99	0.99	982
5	0.98	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.98	0.99	0.99	1028
8	0.98	0.98	0.98	974
9	0.98	0.98	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

```
Kelas 0: FPR = 0.0014, TPR (Recall) = 0.9949
Kelas 1: FPR = 0.0005, TPR (Recall) = 0.9912
Kelas 2: FPR = 0.0010, TPR (Recall) = 0.9845
Kelas 3: FPR = 0.0009, TPR (Recall) = 0.9861
Kelas 4: FPR = 0.0008, TPR (Recall) = 0.9868
Kelas 5: FPR = 0.0018, TPR (Recall) = 0.9922
Kelas 6: FPR = 0.0010, TPR (Recall) = 0.9864
Kelas 7: FPR = 0.0023, TPR (Recall) = 0.9942
Kelas 8: FPR = 0.0022, TPR (Recall) = 0.9836
Kelas 9: FPR = 0.0018, TPR (Recall) = 0.9772
```

Analisis Confusion Matrix lebih lanjut memperkuat temuan ini. Konsentrasi nilai yang tinggi di sepanjang diagonal utama menunjukkan tingkat prediksi yang benar sangat dominan. Kesalahan klasifikasi (nilai di luar diagonal) sangat minimal, yang menegaskan keandalan model.





BAB V: PENUTUP

5.1. Kesimpulan

Penelitian ini telah berhasil mengimplementasikan sebuah model *Convolutional Neural Network* menggunakan PyTorch yang mampu mengklasifikasikan digit tulisan tangan dari dataset MNIST dengan **tingkat akurasi 98.77%**. Kinerja yang tinggi ini dicapai melalui kombinasi arsitektur CNN yang dirancang dengan baik, pra-pemrosesan data yang tepat, dan teknik augmentasi yang efektif untuk meningkatkan ketahanan model terhadap variasi data. Model yang dihasilkan bersifat robust, seimbang di semua kelas, dan mampu melakukan prediksi pada data baru, membuktikan kelayakannya untuk aplikasi di dunia nyata.

5.2. Saran

Untuk pengembangan di masa depan, beberapa area dapat dieksplorasi lebih lanjut:

1. **Optimasi Arsitektur:** Bereksperimen dengan arsitektur yang lebih dalam atau lebih modern (misalnya, arsitektur yang terinspirasi dari ResNet atau VGG), serta menambahkan lapisan *Dropout* untuk regularisasi lebih lanjut.
2. **Tuning Hyperparameter:** Melakukan pencarian sistematis untuk *hyperparameter* yang optimal (misalnya, menggunakan *Grid Search* atau *Random Search*) untuk *learning rate*, ukuran *batch*, dan parameter optimizer.
3. **Pengujian pada Dataset Lain:** Menguji model yang telah dilatih pada dataset digit tulisan tangan lain yang lebih menantang (misalnya, SVHN - Street View House Numbers) untuk mengukur kemampuan transferabilitasnya.