BAB IV. Analisis dan Desain

Bab ini menguraikan rancangan solusi teknis yang diusulkan untuk restorasi dokumen terdegradasi dengan pendekatan Generative Adversarial Network yang dimodifikasi secara khusus untuk meningkatkan keterbacaan teks oleh model pengenalan tulisan tangan. Pembahasan dimulai dari analisis kebutuhan sistem (fungsional dan non-fungsional), dilanjutkan dengan rancangan arsitektur solusi yang mencakup desain Generator (U-Net), Recognizer (Transformer-based yang frozen), Diskriminator Dual-Modal (CNN+LSTM), dan fungsi loss multi-komponen (Adversarial + L1 + CTC). Selanjutnya, detail implementasi solusi diuraikan meliputi lingkungan eksperimen, persiapan dataset (ground truth untuk pengenalan tulisan tangan dan sintetis untuk GAN), implementasi arsitektur model dalam TensorFlow/Keras, prosedur training dengan optimasi alternating, dan justifikasi pemilihan hyperparameter melalui studi ablasi.

IV.1 Analisis Kebutuhan Sistem

Berdasarkan rumusan masalah dan tujuan penelitian, sistem restorasi dokumen yang akan dibangun harus memenuhi serangkaian kebutuhan fungsional dan non-fungsional. Kebutuhan ini menjadi acuan dalam perancangan arsitektur dan tolok ukur dalam evaluasi akhir, sejalan dengan tahapan DSRM.

IV.1.1 Kebutuhan Fungsional

Kebutuhan fungsional mendefinisikan fungsi spesifik yang harus dimiliki oleh sistem GAN-HTR untuk restorasi dokumen terdegradasi secara efektif. Delapan kebutuhan fungsional yang diidentifikasi mencakup tiga kategori:

1. Kemampuan inti pembelajaran mesin untuk restorasi visual dan pengenalan teks,
2. Kemampuan pemrosesan untuk menangani batch inference dan dataset sintetis,
3. Kemampuan integrasi sistem untuk input/output terstandar dan antarmuka fleksibel.

Setiap kebutuhan dirancang untuk memastikan sistem tidak hanya secara teknis mampu melakukan restorasi, tetapi juga praktis untuk digunakan dalam lingkungan produksi dan dapat diintegrasikan dengan sistem lain.

Table 1: Ringkasan Kebutuhan Fungsional Sistem GAN-HTR

Kode	**Nama Kebutuhan**	**Deskripsi dan Detail Spesifikasi**
FR-1	Kemampuan Restorasi Gambar	Menerima input gambar dokumen terdegradasi dan menghasilkan output gambar versi restorasi yang lebih bersih secara visual.
FR-2	Penanganan Berbagai Degradasi	Menangani 4 jenis degradasi utama: tembusan tinta, pemudaran, noda, dan efek buram.
FR-3	Integrasi dengan Proses HTR	Gambar restorasi dapat diproses oleh model HTR untuk menghasilkan transkripsi teks.
FR-4	Inferensi Batch Processing	Memproses dokumen terdegradasi secara batch dengan output: (1) Gambar restorasi penuh, (2) File teks (.txt), (3) File metadata (.json).
FR-5	Dukungan Dataset Sintetis	Menyertakan pipeline pembuatan dataset sintetis terdegradasi dari gambar bersih.
FR-6	Output Terdefinisi	Format output konsisten: (1) Visual: PNG/TIFF 300 DPI, (2) Text: UTF-8 dengan confidence 0.0-1.0, (3) Quality: PSNR/SSIM, (4) Error messages.
FR-7	Input Data Fleksibel	Menangani: (1) Format: JPG, PNG, TIFF, PDF, (2) Validasi: resolusi min 128x1024px, (3) Recovery: skip corrupted files, (4) Preprocessing: grayscale dan noise reduction. *Catatan: Semua input divalidasi sebelum diproses.*
FR-8	Integrasi Sistem	Menyediakan: (1) CLI dengan parameter, (2) Python API, (3) Configuration YAML/JSON, (4) Structured logging.

IV.1.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mendefinisikan kriteria kualitas dan batasan performa sistem GAN-HTR. Empat kebutuhan non-fungsional yang ditetapkan mengukur aspek kualitatif dan kuantitatif performa sistem:

1. Kualitas Visual mengukur kesamaan dengan ground truth menggunakan PSNR dan SSIM,
2. Keterbacaan Teks sebagai metrik utama keberhasilan sistem melalui penurunan CER,
3. Efisiensi Waktu untuk memastikan kepraktisan penggunaan,
4. Reproduktifitas untuk menjamin konsistensi dan keandalan hasil eksperimen.

Target performa ditetapkan berdasarkan studi literatur dan kebutuhan praktis implementasi.

Table 2: Ringkasan Kebutuhan Non-Fungsional Sistem GAN-HTR

Kode	**Nama Requirement**	**Deskripsi dan Target Performa**
NFR-1	Kualitas Visual Restorasi	Kualitas visual gambar hasil restorasi vs ground truth harus mencapai: (1) PSNR rata-rata $\geq$ 35 dB, (2) SSIM rata-rata $\geq$ 0.90 pada dataset validasi. *Target berdasarkan studi literatur dan uji coba awal.*
NFR-2	Peningkatan Keterbacaan Teks	Sistem harus meningkatkan akurasi transkripsi HTR secara signifikan: CER menunjukkan penurunan minimal 25% dibandingkan CER pada gambar terdegradasi asli. *Dihitung sebagai: (CER_asli - CER_restorasi)/CER_asli $\times$ 100%.*
NFR-3	Performa Waktu Inferensi	Waktu inferensi untuk satu gambar dokumen tunggal pada NVIDIA RTX A4000 tidak boleh melebihi 15 detik untuk memastikan kepraktisan penggunaan.
NFR-4	Reproduktifitas	Proses training dan evaluasi harus dapat direproduksi dengan lingkungan perangkat lunak dan dependensi yang didefinisikan secara jelas (melalui file `pyproject.toml`).

Berdasarkan Tabel 1 dan Tabel 2, sistem GAN-HTR yang akan dikembangkan harus memenuhi total 12 kebutuhan (8 fungsional dan 4 non-fungsional) yang mencakup aspek visual, tekstual, performa, dan integrasi sistem.

IV.2 Rancangan Arsitektur

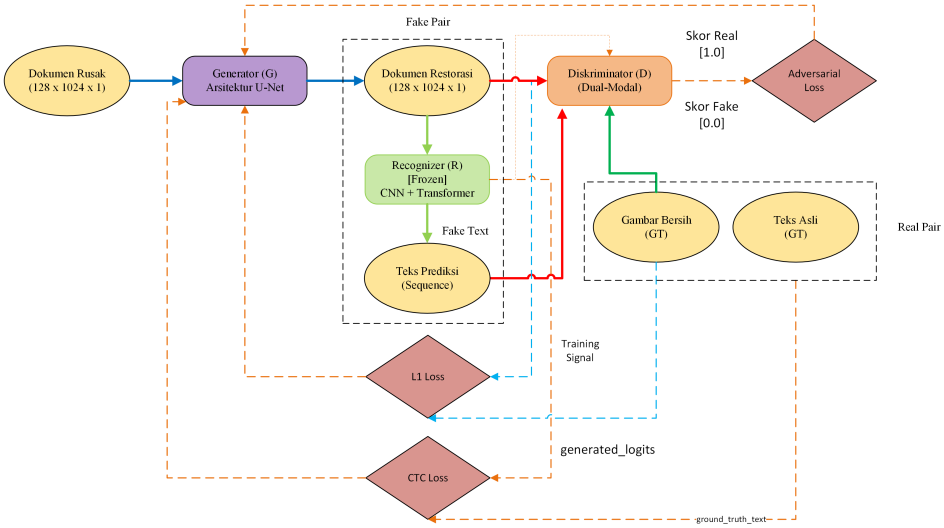
Rancangan solusi bertujuan untuk membangun sebuah model generatif yang tidak hanya mampu membersihkan gambar dari kerusakan visual, tetapi juga secara

eksplisit dioptimalkan untuk menghasilkan gambar yang keterbacaan teksnya maksimal. Untuk mencapai tujuan ini, arsitektur GAN standar dimodifikasi dengan memperkenalkan dua elemen kunci: (1) Diskriminator dual-modal yang menilai koherensi antara gambar dan teks, dan (2) Fungsi loss gabungan yang mencakup loss dari model pengenalan tulisan tangan.

IV.2.1 Gambaran Umum Arsitektur

Arsitektur yang diusulkan terdiri dari tiga komponen utama yang berinteraksi dalam sebuah kerangka adversarial, seperti yang diilustrasikan pada Gambar 1. Ketiga komponen tersebut adalah:

1. Generator (G): Sebuah model dengan arsitektur U-Net yang bertugas menerima gambar dokumen terdegradasi dan menghasilkan versi restorasi dari gambar tersebut.
2. Recognizer (R): Sebuah model HTR berbasis Transformer yang sudah terlatih dan dibekukan (frozen). Tujuannya bukan untuk dilatih, melainkan untuk "membaca" teks dari gambar hasil restorasi dan memberikan sinyal loss terkait keterbacaan.
3. Diskriminator (D): Sebuah model Diskriminator dual-modal yang menjadi inti dari penelitian ini. Tidak seperti diskriminator pada umumnya yang hanya menerima input gambar, diskriminator ini menerima pasangan *(gambar, representasi teks)* untuk menilai apakah sebuah gambar "asli" dan apakah teks di dalamnya koheren.



Gambar 1: Diagram alur arsitektur GAN. Diagram menunjukkan input untuk setiap komponen loss: L1 Loss membandingkan dua gambar, CTC Loss membandingkan dua sekuens teks, dan Adversarial Loss berasal dari Diskriminator.

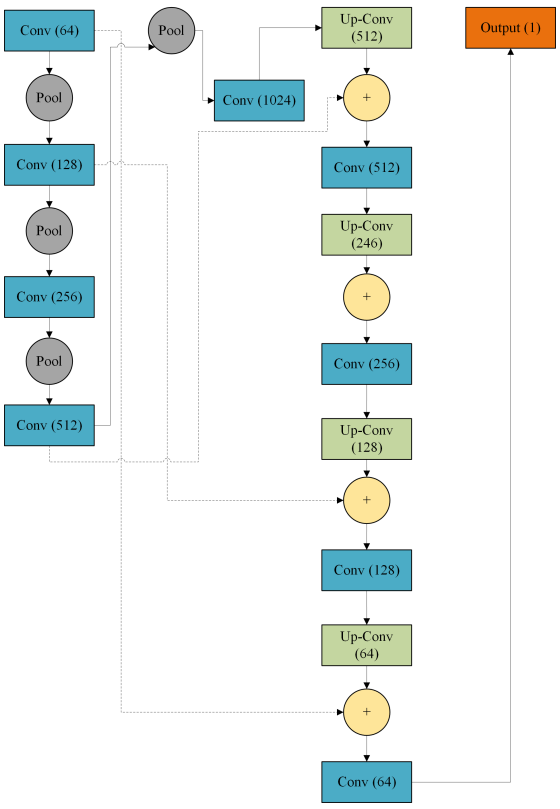
IV.2.2 Desain Generator

- Jalur Encoder: Berfungsi seperti jaringan konvolusi pada umumnya untuk mengekstraksi fitur dari gambar input. Melalui serangkaian blok konvolusi

dan operasi *max-pooling*, resolusi spasial gambar secara bertahap dikurangi (downsampling) sementara kedalaman fitur (jumlah channel) ditingkatkan. Proses ini memungkinkan model untuk menangkap informasi kontekstual dari gambar

- Jalur Decoder: Bertugas untuk merekonstruksi gambar output dari representasi fitur yang telah dipelajari oleh encoder. Melalui serangkaian blok *upsampling* dan konvolusi, resolusi spasial gambar secara bertahap ditingkatkan hingga kembali ke ukuran aslinya.

- Skip Connections: Ini adalah fitur utama dari arsitektur U-Net. Koneksi ini menghubungkan secara langsung feature map dari lapisan di jalur encoder ke lapisan yang bersesuaian di jalur decoder. Dengan meneruskan informasi dari lapisan awal (yang kaya akan detail spasial) ke lapisan akhir, U-Net mampu mengatasi hilangnya informasi detail yang sering terjadi pada arsitektur encoder-decoder standar. Untuk tugas restorasi dokumen, ini sangat penting untuk menjaga ketajaman dan keutuhan gurat-gurat tipis pada karakter tulisan tangan.



Gambar 2: Diagram arsitektur U-Net yang digunakan sebagai Generator.

IV.2.3 Desain Recognizer

Komponen Recognizer (R) memainkan peran khusus dan esensial dalam arsitektur yang diusulkan. Berbeda dengan komponen lain, Recognizer tidak dilatih

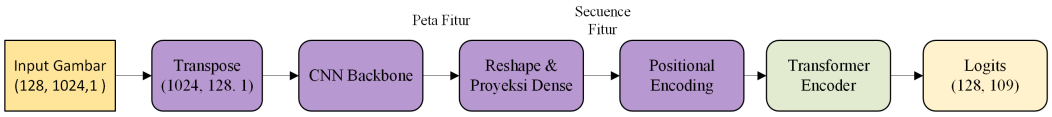
selama proses training GAN. Sebaliknya, komponen ini merupakan sebuah model Handwritten Text Recognition (HTR) berbasis Transformer yang telah dilatih sebelumnya (*pre-trained*) dan bobotnya dibekukan (*frozen*) atau diatur sebagai *non-trainable*.

Fungsi Utama dan Peran dalam Arsitektur Recognizer berfungsi sebagai evaluator keterbacaan teks yang objektif. Komponen ini menerima gambar hasil restorasi dari Generator dan menghasilkan distribusi probabilitas karakter untuk mengukur keterbacaan hasil restorasi. Output dari Recognizer digunakan untuk dua tujuan:

1. Sebagai input teks untuk Diskriminator, yang menilai koherensi antara gambar dan teks yang dikenali.
2. Sebagai input untuk kalkulasi CTC Loss, yang secara langsung mengukur seberapa dapat dibaca gambar hasil restorasi.

Keputusan untuk membekukan bobot Recognizer (frozen) selama training GAN didasarkan pada tiga pertimbangan ilmiah yang krusial:

1. **Objective Evaluator:** Recognizer dengan bobot frozen berfungsi sebagai evaluator objektif dengan metrik yang stabil dan konsisten. Jika bobot recognizer dilatih ulang, metrik keterbacaan akan berubah-ubah seiring training, sehingga kehilangan fungsinya sebagai ground truth evaluasi.
2. **Catastrophic Forgetting Prevention:** Melatih recognizer secara simultan dengan GAN dapat menyebabkan catastrophic forgetting, di mana recognizer kehilangan kemampuan mengenali karakter yang telah dipelajari sebelumnya karena terfokus pada gambar restorasi yang terus berubah.
3. **Focused Visual Optimization:** Dengan membekukan recognizer, optimasi Generator terfokus pada adaptasi visual untuk meningkatkan keterbacaan, bukan melatih recognizer untuk membaca lebih baik. Ini memastikan bahwa peningkatan berasal dari kualitas gambar, bukan dari adaptasi recognizer.



Gambar 3: Diagram arsitektur Recognizer berbasis CNN+Transformer untuk ekstraksi fitur dan pengenalan teks.

Arsitektur High-Level Recognizer menggunakan arsitektur CNN+Transformer yang dirancang khusus untuk dokumen kuno:

- CNN Backbone: Mengekstrak fitur visual hierarkis dari stroke dasar hingga kompleksitas karakter
- Transformer Encoder: Menangkap dependensi jarak jauh dan konteks linguistik dengan multi-head attention

- CTC Output Layer: Menghasilkan probabilitas karakter untuk setiap timestep

Detail justifikasi pemilihan arsitektur CNN+Transformer untuk kasus paleografi telah dibahas secara komprehensif pada Bab 2.3.4.

IV.2.4 Detail Arsitektur Recognizer

Detail arsitektur recognizer mencakup implementasi CNN backbone dengan 7 lapisan untuk ekstraksi fitur hierarkis dari stroke dasar hingga kompleksitas karakter, diikuti oleh transformer encoder dengan 6 layer untuk menangkap dependensi jarak jauh dan konteks linguistik menggunakan multi-head attention. Positional encoding sinusoidal digunakan untuk menyimpan informasi spasial, dan CTC output layer menghasilkan probabilitas karakter untuk setiap timestep. Teknik regularisasi seperti spatial dropout, stochastic depth, attention dropout, dan embedding dropout diterapkan untuk meningkatkan robustness dan mencegah overfitting.

Table 3: Spesifikasi detail arsitektur Recognizer

Komponen	**Spesifikasi**
CNN Backbone	7 convolutional layers dengan residual connections
Transformer Encoder	6 layers, 8 attention heads per layer
Model Dimension	512 untuk representasi fitur
Feed-Forward Dimension	2048 untuk transformer layers
Positional Encoding	Sinusoidal encoding untuk informasi posisi
CTC Output Layer	Softmax atas vocabulary + blank token
Dropout Rate	0.20 pada transformer layers
Weight Decay	1e-4 untuk L2 regularization
Target CER	$< 15\%$ pada validation set
Target WER	$< 25\%$ pada validation set
Inference Speed	$<$ 100ms per image (128×1024) pada GPU
Model Size	$<$ 50MB untuk deployment efficiency

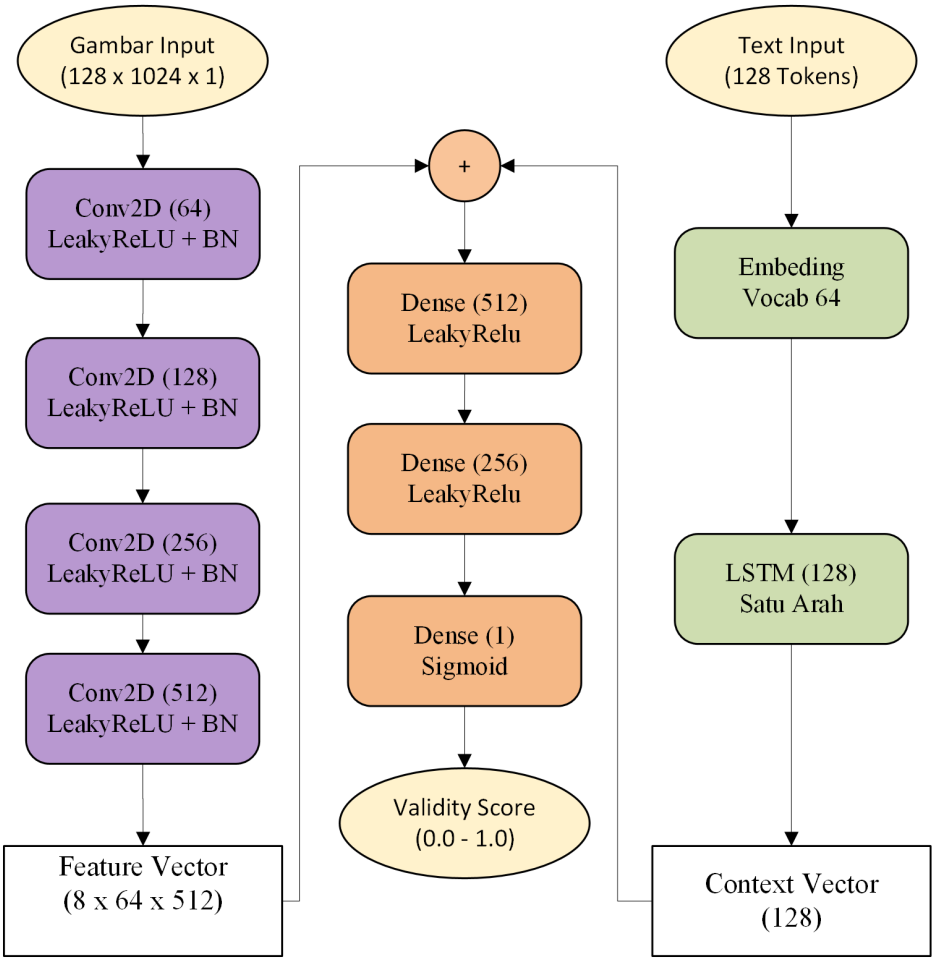
IV.2.5 Desain Diskriminator Dual-Modal

Inovasi utama dalam rancangan solusi ini terletak pada arsitektur Diskriminator (D). Berbeda dengan diskriminator pada GAN konvensional yang hanya menilai realisme sebuah gambar (unimodal), diskriminator yang diusulkan bersifat dual-modal. Tujuannya tidak hanya untuk membedakan gambar asli dari gambar palsu, tetapi juga untuk menilai koherensi antara konten visual sebuah gambar dengan konten semantik dari teks yang diekstraksi dari gambar tersebut.

Diskriminator ini menerima sepasang input: sebuah gambar dan sebuah representasi teks. Tugasnya adalah menghasilkan skor probabilitas tunggal yang menyatakan apakah pasangan tersebut adalah pasangan *(gambar bersih, teks asli)* yang koheren, atau pasangan *(gambar restorasi, teks prediksi)* yang kemungkinan besar tidak koheren.

Arsitektur ini terdiri dari dua jalur pemrosesan paralel yang kemudian digabungkan:

- Jalur Input Gambar: Serangkaian lapisan Conv2D dengan LeakyReLU dan BatchNormalization untuk mengekstrak fitur visual
- Jalur Input Teks: Layer Embedding dan LSTM untuk memproses representasi teks dan menangkap konteks
- Fusi dan Klasifikasi: Konkatenasi fitur visual dan tekstual, diikuti Dense layers untuk klasifikasi akhir



Gambar 4: Diagram arsitektur Diskriminator Dual-Modal.

IV.2.6 Desain Fungsi Loss

Optimalisasi arsitektur GAN yang diusulkan mengandalkan dua fungsi loss yang bekerja secara simultan. Desain fungsi loss untuk Generator menjadi salah satu kontribusi utama karena secara eksplisit memasukkan metrik keterbacaan teks.

Fungsi Loss Diskriminator Fungsi loss untuk Diskriminator (L_D) menggunakan Binary Cross-Entropy (BCE) standar untuk membedakan pasangan data asli dan palsu:

$$L_D = -\mathbb{E}_{x,y}[\log D(x,y)] - \mathbb{E}_{z,y'}[\log(1 - D(G(z),y'))] \tag{1}$$

Fungsi Loss Generator Fungsi loss untuk Generator (L_G) dirancang untuk mencapai dua tujuan: menghasilkan gambar realistis dan gambar yang teksnya dapat dibaca dengan baik. L_G disusun sebagai kombinasi tiga komponen loss:

1. Adversarial Loss (L_{adv}) Mengukur kemampuan Generator dalam "menipu" Diskriminator:

$$L_{adv} = -\mathbb{E}_{z,y'}[\log D(G(z),y')] \tag{2}$$

2. Reconstruction Loss (L_{L1}) L1 Loss untuk kemiripan piksel dengan ground truth:

$$L_{L1} = \mathbb{E}_{x,G(z)}[\|x - G(z)\|_1] \tag{3}$$

3. CTC Loss (L_{CTC}) Inovasi kunci: CTC loss dari frozen HTR model untuk mengoptimalkan keterbacaan teks:

$$L_{CTC} = \mathbb{E}_{z,t}[\text{CTC}(R(G(z)),t)] \tag{4}$$

di mana $R(G(z))$ adalah output recognizer dari gambar hasil restorasi, dan t adalah ground truth transcription.

Fungsi Loss Gabungan Ketiga komponen loss tersebut digabungkan menjadi satu fungsi loss total untuk Generator:

$$L_G = \lambda_{adv} L_{adv} + \lambda_{L1} L_{L1} + \lambda_{CTC} L_{CTC} \tag{5}$$

Bobot λ_{adv}, λ_{L1}, dan λ_{CTC} adalah hyperparameter yang dapat disesuaikan untuk menyeimbangkan kontribusi dari setiap komponen loss terhadap proses training Generator.

IV.3 Spesifikasi Lingkungan Implementasi

Untuk memastikan bahwa rancangan arsitektur GAN-HTR dapat diimplementasikan secara efektif dan reproduktif, bagian ini merinci spesifikasi teknis lingkungan implementasi. Spesifikasi ini dirancang berdasarkan analisis kebutuhan non-fungsional yang telah diidentifikasi sebelumnya, khususnya aspek reproduktifitas dan performa waktu. Dengan spesifikasi yang jelas, framework dapat dikembangkan dalam lingkungan terkontrol yang mendukung eksperimen berulang dan deployment yang konsisten.

IV.3.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi framework GAN-HTR telah didefinisikan secara lengkap pada Bab III (Metodologi Penelitian), khususnya pada Tabel 3.6 yang merinci komponen GPU (NVIDIA RTX A4000 dengan 16 GB GDDR6), CPU (Intel Xeon atau AMD Ryzen multi-core), RAM (32 GB DDR4), dan storage (1 TB NVMe SSD). Spesifikasi ini dipilih untuk memenuhi kebutuhan komputasi intensif training GAN dan memastikan reproduktifitas eksperimen.

IV.3.2 Konfigurasi Perangkat Lunak dan Library

Konfigurasi perangkat lunak dan library yang digunakan dalam implementasi juga telah diuraikan secara komprehensif pada Bab III, Tabel 3.7. Framework utama meliputi Python 3.10, TensorFlow/Keras 2.x untuk deep learning, Poetry untuk dependency management, serta library pendukung seperti NumPy, OpenCV, MLflow, dan Matplotlib. Konfigurasi ini memastikan konsistensi lingkungan development dan memfasilitasi reproduktifitas melalui file `pyproject.toml` yang terdefinisi dengan baik.

IV.3.3 Konfigurasi Docker dan Reproduktifitas

Untuk memastikan reproduktifitas, implementasi menggunakan Docker dengan konfigurasi yang mencakup base image TensorFlow 2.15.0 GPU, instalasi Poetry untuk dependency management, dan konfigurasi GPU NVIDIA dengan environment variables untuk akses penuh GPU dan compute capabilities.

IV.4 Implementasi Pipeline Data

Implementasi pipeline data merupakan langkah krusial dalam mewujudkan rancangan arsitektur GAN-HTR yang telah diuraikan sebelumnya. Pipeline ini mengintegrasikan kebutuhan fungsional terkait dataset sintetis dan dukungan untuk berbagai format input, seperti yang ditentukan dalam analisis kebutuhan. Dengan pipeline yang modular, sistem dapat menangani persiapan data untuk training Recognizer dan pembuatan dataset terdegradasi sintetis, memastikan kualitas data yang konsisten untuk eksperimen yang reproduktif.

IV.4.1 Pipeline Persiapan Dataset HTR

Fase pertama implementasi adalah persiapan dataset untuk training model Recognizer. Pipeline ini diimplementasikan dengan struktur modular:

Algorithm 1 Pipeline Persiapan Dataset HTR

Input: Halaman dokumen bersih, file XML transkripsi
Output: TFRecord dataset (image, transcription)
1: Segmentasi halaman menjadi baris teks menggunakan bounding box dari XML
2: Ekstrak image patch untuk setiap baris teks
3: Preprocessing: normalisasi intensitas, resize ke 128×1024
4: Binarisasi: adaptive threshold untuk contrast enhancement
5: Validasi: verifikasi kesesuaian image-text pairs
6: Konversi ke format TFRecord untuk efisiensi I/O
7: Split: pelatihan (80%), validasi (10%), pengujian (10%)

IV.4.2 Pipeline Degradasi Sintetis

Karena keterbatasan dataset dokumen terdegradasi, dikembangkan pipeline sintetis:

Algorithm 2 Pipeline Synthetic Degradation

Input: Dataset HTR bersih
Output: Dataset triplet (degraded, clean, transcription)
1: Load clean images dan transcriptions
2: Apply degradations dengan probabilitas random:
    a. Bleed-through (30%): overlay mirrored text dengan opacity variabel
    b. Fading (40%): reduce intensity dengan exponential decay
    c. Stains (25%): add random brown spots dengan varying intensity
    d. Blur (20%): Gaussian filter dengan $\sigma \in [0.5, 2.0]$
3: Add noise: salt & pepper (10% pixels) untuk simulasi sensor noise
4: Validasi quality metrics (PSNR > 20 dB) untuk memastikan degradasi tidak ekstrem
5: Simpan sebagai TFRecord triplet dengan struktur (degraded, clean, transcription)

IV.4.3 Format Data dan Preprocessing

Format TFRecord: Setiap sample disimpan dalam TFRecord dengan struktur:

- `degraded_image`: Tensor uint8 [128, 1024, 1]
- `clean_image`: Tensor uint8 [128, 1024, 1]
- `transcription`: String teks ground truth
- `length`: Int panjang sequence

IV.5 Detail Implementasi Training

Bagian ini menguraikan detail implementasi proses training yang mengoperasionalkan rancangan arsitektur GAN adversarial dengan dual-modal

discriminator. Implementasi ini mempertimbangkan kebutuhan non-fungsional seperti performa waktu dan reproduktifitas, dengan konfigurasi yang disesuaikan untuk mencapai target PSNR, SSIM, dan penurunan CER/WER. Teknik seperti multi-component loss dan precision policy diintegrasikan untuk memastikan stabilitas training dan optimasi keterbacaan teks.

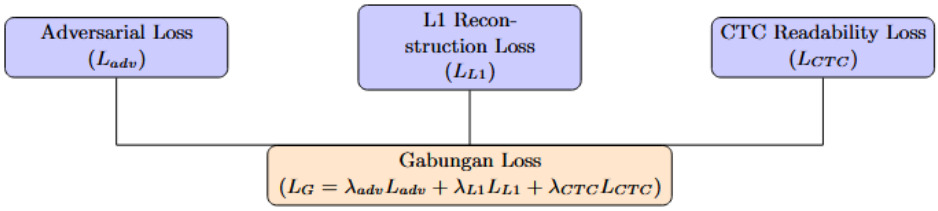
IV.5.1 Konfigurasi Pelatihan Generator

Table 4: Konfigurasi training Generator

Parameter	**Nilai**
Optimizer	Adam
Learning Rate	1×10^{-4}
Batch Size	16
Loss Weights	$\lambda_{adv} = 1.0$, $\lambda_{L1} = 100.0$, $\lambda_{CTC} = 1.0$
Gradient Clipping	clipnorm=1.0
Epochs	100 (dengan early stopping)

IV.5.2 Implementasi Multi-Component Loss Function

Fungsi loss multi-komponen diimplementasikan untuk mengoptimalkan Generator secara simultan pada tiga aspek utama: adversarial loss untuk menipu diskriminator, L1 reconstruction loss untuk kesamaan piksel dengan ground truth, dan CTC readability loss untuk meningkatkan keterbacaan teks. Pendekatan ini memastikan keseimbangan antara kualitas visual dan tekstual melalui bobot yang dapat dikonfigurasi, dengan clipping pada CTC loss untuk stabilitas numerik.



Gambar 5: Diagram komponen multi-component loss untuk Generator.

IV.5.3 Discriminator Mode Implementation

Dual-modal discriminator mendukung dua mode operasi yang dapat dikonfigurasi sesuai dengan desain eksperimen yang telah diuraikan pada Bab III.4.3:

Ground Truth Mode: Mode ini menggunakan ground truth text sebagai input teks untuk diskriminator. Karakteristik:

- **Stabilitas:** Training lebih stabil karena teks konsisten dan bebas error
- **Kelebihan:** Konvergensi cepat, loss smooth, cocok untuk initial training
- **Kekurangan:** Kurang realistis untuk deployment karena tidak mensimulasikan kondisi inference
- **Use Case:** Ideal untuk warmup phase dan debugging architecture

Predicted Mode: Mode ini menggunakan predicted text dari frozen recognizer sebagai input. Karakteristik:

- **Realisme:** Mensimulasikan kondisi deployment dengan error recognition
- **Kelebihan:** Model belajar handle imperfect text, generalisasi lebih baik
- **Kekurangan:** Training lebih challenging, memerlukan recognizer berkualitas tinggi (CER < 15%)
- **Use Case:** Ideal untuk fine-tuning dan evaluation yang realistis

Eksperimen Perbandingan: Perbandingan kuantitatif kedua mode akan dilakukan melalui eksperimen terkontrol dengan metrik CER, WER, PSNR, dan SSIM. Hasil lengkap dan analisis statistical significance akan dibahas pada Bab V (Hasil dan Pembahasan), sesuai dengan desain eksperimen yang dirancang pada Bab V.

IV.5.4 Pipeline Pelatihan Recognizer

Pipeline pelatihan recognizer mengintegrasikan augmentasi data untuk meningkatkan generalisasi, termasuk penyesuaian brightness, contrast, dan penambahan noise secara acak. Learning rate schedule menggunakan cosine annealing dengan warmup untuk stabilisasi awal, sementara CTC loss dengan label smoothing memfasilitasi generalisasi yang lebih baik. Training loop dioptimalkan dengan gradient accumulation, dan metrik validasi multi-metrik (CER dan WER) digunakan untuk monitoring performa. Early stopping dan checkpointing diimplementasikan untuk mencegah overfitting dan menyimpan model terbaik berdasarkan validation CER.

Table 5: Hyperparameter Training Recognizer

Parameter	**Nilai**
Optimizer	AdamW
Learning Rate	3e-4 (dengan cosine annealing)
Batch Size	32
Epochs	Dengan early stopping (patience 10)
Label Smoothing	0.1
Dropout Rate	0.20
Weight Decay	1e-4

IV.5.5 Implementasi Kebijakan Presisi

Kebijakan presisi pure FP32 diimplementasikan secara eksplisit untuk memastikan stabilitas numerik dalam training, terutama untuk kalkulasi CTC loss yang memerlukan presisi penuh. Pendekatan ini menghindari underflow/overflow pada gradient calculations, memastikan konsistensi multi-component loss, dan mendukung stabilitas keseluruhan proses adversarial.

IV.6 Implementasi Pelatihan dan Evaluasi

Implementasi pelatihan dan evaluasi melengkapi siklus pengembangan framework GAN-HTR, dengan fokus pada konfigurasi training diskriminator dan sistem

evaluasi multi-metrik. Bagian ini menjembatani rancangan arsitektur dengan praktik implementasi, memastikan bahwa metrik evaluasi seperti CER dan WER dapat diukur secara konsisten sesuai target non-fungsional. Penggunaan MLflow untuk tracking eksperimen mendukung reproduktifitas dan analisis performa yang mendalam.

IV.6.1 Konfigurasi Pelatihan Diskriminator

Konfigurasi training diskriminator dirancang untuk mencapai Nash equilibrium dalam adversarial training, sesuai dengan strategi yang telah dijelaskan pada Bab III.4.3. Target accuracy 60-80% menunjukkan keseimbangan optimal antara generator dan discriminator.

Table 6: Konfigurasi training Diskriminator

Parameter	**Nilai**
Optimizer	Adam
Learning Rate	5×10^{-4} (2.5x learning rate Generator)
Batch Size	16 (sama dengan Generator)
Loss	Binary Cross-Entropy dengan label smoothing (0.9)
Target Accuracy	60-80% (Nash equilibrium zone)
Gradient Penalty	Tidak digunakan (diganti label smoothing)

Rasio learning rate 2.5:1 (Discriminator:Generator) dipilih untuk mempercepat konvergensi discriminator tanpa menyebabkan mode collapse, berdasarkan best practice GAN training.

IV.6.2 Implementasi MLflow Tracking

MLflow diintegrasikan untuk pelacakan eksperimen secara sistematis, memungkinkan dokumentasi parameter, metrik, dan artefak model dalam lingkungan terstruktur. Pendekatan ini memfasilitasi reproduktifitas eksperimen melalui pencatatan hyperparameter, evolusi metrik per epoch, dan penyimpanan model, sehingga mendukung analisis komparatif dan optimasi berbasis data historis.

IV.6.3 Sistem Evaluasi Multi-Metrik

Sistem evaluasi mengimplementasikan metrik yang dijelaskan pada Bab III.5, dengan fokus pada pengukuran kualitas visual dan tekstual secara simultan. Metrik PSNR dan SSIM digunakan untuk menilai kesamaan visual dengan target di atas 35 dB dan 0.90, sementara CER dan WER mengukur keterbacaan teks dengan target di bawah 5% dan 15%. Pendekatan multi-metrik ini memastikan evaluasi komprehensif yang seimbang antara aspek visual dan tekstual dalam konteks restorasi dokumen.

IV.7 Sistem Monitoring dan Tracking

Sistem monitoring dan tracking diimplementasikan untuk mendukung evaluasi kontinyu selama proses training, sesuai dengan kebutuhan non-fungsional reproduktifitas. Dengan real-time monitoring loss components dan metrics

evolution, sistem ini memungkinkan deteksi dini masalah seperti mode collapse atau imbalance, memastikan bahwa training GAN adversarial berjalan stabil dan efisien. Checkpoint management dan early stopping diintegrasikan untuk optimasi sumber daya komputasi.

IV.7.1 Real-time Training Monitoring

Sistem monitoring real-time diimplementasikan untuk tracking progress training, dengan fokus pada komponen loss generator (adversarial, L1, CTC, total), loss diskriminator (klasifikasi real/fake), dan rasio loss untuk deteksi imbalance. Evolusi metrik seperti PSNR/SSIM per epoch, tren peningkatan CER/WER, serta indikator stabilitas training memungkinkan pengawasan kontinyu dan intervensi dini terhadap masalah seperti mode collapse atau ketidakseimbangan dalam proses adversarial.

IV.7.2 Checkpoint Management Strategy

Strategi manajemen checkpoint dirancang untuk efisiensi penyimpanan, dengan fokus pada penyimpanan model terbaik saja berdasarkan validation loss, pemantauan otomatis, dan pembersihan checkpoint lama. Pendekatan ini mengoptimalkan penggunaan sumber daya komputasi sambil memastikan ketersediaan model optimal untuk evaluasi dan deployment.

IV.7.3 Implementasi Early Stopping

Pemberhentian dini (early stopping) diimplementasikan untuk mencegah overfitting dan menghemat sumber daya, dengan kriteria patience 15 epoch tanpa improvement pada validation total generator loss, serta pemulihan otomatis bobot terbaik. Strategi ini memastikan efisiensi training sambil mempertahankan performa model yang optimal.

IV.8 Optimasi dan Fine-Tuning

Optimasi dan fine-tuning merupakan tahap akhir dalam desain implementasi, yang bertujuan untuk meningkatkan performa framework GAN-HTR melampaui baseline awal. Strategi ini mengintegrasikan teknik regularisasi, hyperparameter tuning, dan debugging untuk mencapai target kualitas visual dan tekstual yang ditetapkan dalam analisis kebutuhan. Dengan pendekatan ini, sistem dapat dioptimalkan untuk skala produksi sambil mempertahankan stabilitas training.

IV.8.1 Strategi Optimasi Hyperparameter

Strategi optimasi hyperparameter mencakup penjadwalan learning rate dengan ReduceLROnPlateau dan patience 5, tuning bobot loss melalui grid search untuk nilai lambda, optimasi batch size dengan eksperimen pada ukuran 8, 16, dan 32, serta pencarian arsitektur dengan variasi jumlah filter dan lapisan. Pendekatan ini memfasilitasi pencapaian performa optimal melalui eksplorasi sistematis parameter kunci.

IV.8.2 Teknik Regularisasi dan Stabilisasi

Teknik regularisasi dan stabilisasi mencakup gradient penalties untuk stabilitas training GAN, spectral normalization pada diskriminator, label smoothing untuk menghindari overconfidence, serta experience replay dengan buffer dari sampel

yang dihasilkan. Pendekatan ini memastikan konvergensi yang stabil dan performa yang konsisten dalam proses adversarial.

IV.8.3 Monitoring dan Debugging

Sistem monitoring mencakup tracking loss real-time dengan TensorBoard, visualisasi sampel per epoch, pemantauan aliran gradien, dan deteksi mode collapse. Strategi debugging meliputi smoke test dengan dataset kecil, pengujian komponen terpisah, ablation loss untuk setiap komponen, serta pencarian learning rate optimal. Pendekatan ini memfasilitasi identifikasi dan resolusi masalah secara sistematis selama fase optimasi.

IV.9 Framework Modular dan Reusable

Framework modular dan reusable dirancang untuk memfasilitasi ekstensi dan penggunaan ulang komponen, sesuai dengan prinsip desain yang efisien dalam konteks analisis dan implementasi. Dengan arsitektur yang loosely coupled, sistem ini dapat diadaptasi untuk berbagai skenario restorasi dokumen, mendukung plugin architecture dan testing framework yang komprehensif. Deployment framework memastikan transisi mulus dari eksperimen ke penggunaan produksi.

IV.9.1 Arsitektur Modular

Framework GAN-HTR dirancang dengan arsitektur modular untuk memastikan reusability dan extensibility, di mana setiap komponen seperti generator (U-Net), diskriminator (dual-modal), recognizer (frozen HTR), loss (multi-component), data pipeline, dan training loop beroperasi sebagai modul independen dengan interface terdefinisi. Prinsip desain mencakup loose coupling untuk dependensi minimal, high cohesion untuk pengelompokan fungsi terkait, abstraction layers untuk implementasi berbeda, serta dependency injection untuk konfigurasi runtime. Pendekatan ini memfasilitasi adaptasi framework terhadap berbagai skenario restorasi dokumen.

IV.9.2 Konfigurasi Sistem

Konfigurasi sistem dirancang untuk mengelola parameter model dan eksperimen secara efisien, mendukung kebutuhan reproduktifitas (NFR-4) melalui mekanisme inheritance dan override. Dengan pendekatan ini, konfigurasi dapat disesuaikan untuk berbagai eksperimen tanpa mengubah kode inti, memastikan konsistensi dalam analisis dan desain framework.

Sistem konfigurasi mengadopsi pendekatan hierarkis yang memungkinkan pewarisan parameter dari konfigurasi dasar ke konfigurasi spesifik eksperimen. Mekanisme override memfasilitasi penyesuaian parameter tertentu sesuai kebutuhan eksperimen, sambil mempertahankan struktur dasar yang konsisten. Pendekatan ini mengoptimalkan efisiensi pengelolaan konfigurasi, mengurangi redundansi, dan memastikan bahwa setiap eksperimen dapat direproduksi dengan parameter yang terdokumentasi secara jelas.

Table 7: Kategori parameter konfigurasi

Kategori	**Parameter**
Model Architecture	Filters, layers, activation functions
Training Hyperparameters	Learning rate, batch size, epochs
Loss Configuration	Loss weights, clipping values
Data Pipeline	Augmentation, preprocessing, loading
Experiment Tracking	MLflow settings, logging frequency

IV.9.3 Arsitektur Plugin

Arsitektur plugin dikembangkan untuk meningkatkan ekstensi framework tanpa modifikasi inti, sesuai dengan prinsip modularitas yang dirancang untuk mendukung kebutuhan integrasi sistem. Dengan mekanisme registrasi komponen kustom, sistem dapat diperluas untuk berbagai skenario restorasi dokumen, memastikan fleksibilitas dan kemudahan maintenance dalam konteks analisis desain.

Arsitektur ini menyediakan dukungan untuk komponen kustom melalui mekanisme registrasi yang terstruktur, memungkinkan pengembang untuk menambahkan lapisan baru, fungsi loss, metrik evaluasi, dan sumber data tanpa mengganggu arsitektur inti. Pendekatan ini memfasilitasi ekstensi framework secara dinamis, di mana komponen kustom dapat didaftarkan dan diintegrasikan pada saat runtime, sehingga meningkatkan adaptabilitas sistem terhadap kebutuhan spesifik aplikasi restorasi dokumen.

IV.9.4 Testing Framework

Framework menyediakan comprehensive testing suite untuk memastikan kualitas dan reusability.

Strategi pengujian unit mencakup validasi berbagai aspek komponen, mulai dari bentuk output model, jumlah parameter, hingga aliran gradien, untuk memastikan integritas setiap modul. Pengujian data memverifikasi proses pemuatan, preprocessing, dan augmentasi, sementara pengujian training mengevaluasi komputasi loss dan pembaruan optimizer. Pengujian integrasi memvalidasi pipeline end-to-end secara keseluruhan, memastikan bahwa semua komponen berinteraksi dengan benar dalam konteks desain framework.

Pipeline pengujian terotomatisasi mengintegrasikan berbagai jenis pengujian dalam urutan yang sistematis, dimulai dari pengujian unit untuk komponen individual, dilanjutkan dengan pengujian integrasi untuk interaksi antar-komponen, dan diakhiri dengan pengujian performa serta validasi untuk memastikan kesesuaian dengan spesifikasi desain. Pendekatan ini memfasilitasi deteksi dini masalah dan pembuatan laporan komprehensif untuk mendukung kualitas framework secara keseluruhan.

IV.9.5 Deployment Framework

Deployment Framework dirancang untuk memenuhi kebutuhan integrasi sistem (FR-8) yang telah diidentifikasi dalam analisis kebutuhan, memungkinkan framework GAN-HTR untuk dioperasikan dalam lingkungan produksi. Dengan pendekatan ini, sistem dapat diekspor dalam berbagai format untuk mendukung deployment skala besar, sambil mempertahankan performa dan reproduktifitas yang telah dirancang.

Framework deployment ini mengintegrasikan komponen-komponen utama melalui mekanisme ekspor model yang fleksibel, yang memungkinkan adaptasi terhadap berbagai platform komputasi. Pendekatan ini memfasilitasi transisi dari lingkungan pengembangan ke produksi dengan meminimalkan overhead integrasi, sekaligus menjaga konsistensi performa yang telah dioptimalkan selama fase pelatihan. Selain itu, antarmuka aplikasi yang terstruktur menyediakan kanal komunikasi standar untuk interaksi dengan sistem eksternal, memungkinkan integrasi seamless dalam alur kerja operasional yang lebih luas.