

**LAPORAN PROYEK AKHIR  
TEKNOLOGI PENCITRAAN MEDIS**

*A Comparative Study of 2D and 3D Deep Learning-Based Noise Reduction Methods for Low-Dose CT Volumetric Reconstruction*



Disusun oleh: Kelompok 7  
ATHALLAH FADHIL MANINDJO (2306251563)  
RAISA SHAHIRA AFNAN (2306168971)  
CRYSTALY (2306202523)  
MUTIARA PUTRI AFRITA (2306169021)

**PROGRAM STUDI SARJANA TEKNIK BIOMEDIK  
DEPARTEMEN TEKNIK ELEKTRO  
FAKULTAS TEKNIK  
UNIVERSITAS INDONESIA  
2025**

## KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas selesainya Laporan Proyek Akhir mata kuliah Teknologi Pencitraan Biomedik. Laporan ini disusun untuk mendokumentasikan perancangan, implementasi, serta evaluasi dari proyek “A Comparative Study of 2D and 3D Deep Learning-Based Noise Reduction Methods for Low-Dose CT Volumetric Reconstruction” sebagai penerapan teori pengolahan citra medis.

Kami mengucapkan terima kasih kepada Bapak Basari, Ph.D. selaku dosen pengampu atas bimbingan dan ilmunya, serta kepada rekan-rekan kelompok yang telah bekerja sama dengan baik selama pengerjaan proyek ini.

Kami menyadari masih terdapat kekurangan dalam laporan ini, sehingga kritik dan saran yang membangun sangat kami harapkan. Semoga laporan ini dapat memberikan manfaat bagi pembaca.

Pengembangan perangkat lunak dalam penelitian ini menggunakan bahasa pemrograman Python. Dalam proses implementasi kode, khususnya pada penulisan struktur *class* dan penyelesaian galat (*debugging*), penulis memanfaatkan teknologi Generative AI (Gemini/ChatGPT) sebagai asisten pemrograman. Seluruh kode hasil luaran AI telah divalidasi, diuji, dan dimodifikasi oleh penulis untuk memastikan kesesuaian dengan logika algoritma yang dirancang.

Penulis (Kelompok 7)

Depok, 24 Desember 2025

## ABSTRAK

*Low-Dose Computed Tomography (LDCT)* telah banyak dimanfaatkan dalam program skrining kanker paru karena mampu menurunkan paparan radiasi yang diterima pasien. Namun, penurunan dosis radiasi tersebut berdampak pada meningkatnya *noise* pada citra *CT*, sehingga kualitas citra menurun dan proses deteksi nodul paru menjadi lebih sulit. Kondisi ini mendorong perlunya perlunya metode rekonstruksi dan peningkatan kualitas citra yang mampu menekan noise tanpa menghilangkan informasi struktural penting pada jaringan paru.

Pada final project ini, kami melakukan pengembangan serta evaluasi beberapa model deep learning untuk peningkatan kualitas citra *LDCT*, yaitu *2D RED-CNN*, *2D Multiwave Convolutional Neural Network (MWCNN)*, dan *3D Residual-Attention U-Net*. Dataset yang digunakan adalah *LoDoPaB-CT*, yang menyediakan pasangan data citra *LDCT* dan citra *CT* dosis normal sebagai ground truth. Tahapan penelitian meliputi proses preprocessing data, pelatihan model, pengujian, serta evaluasi kinerja model menggunakan metrik kuantitatif seperti *PSNR*, *SSIM*, *RMSE*, dan *Contrast-to-Noise Ratio (CNR)*, yang didukung oleh evaluasi kualitatif melalui visualisasi hasil rekonstruksi citra.

Berdasarkan hasil pengujian, seluruh model menunjukkan kemampuan dalam meningkatkan kualitas citra *LDCT* dibandingkan dengan citra input awal. Model *2D RED-CNN* dan *MWCNN* terbukti efektif dalam menurunkan noise, namun cenderung menghasilkan citra dengan tingkat blur yang masih terlihat pada beberapa area. Pada, model *3D Residual-Attention U-Net* mampu memanfaatkan informasi spasial antar-slice dengan lebih baik sehingga meningkatkan nilai *CNR* dan menjaga konsistensi struktur anatomis, meskipun masih terdapat keterbatasan dalam rekonstruksi *full-volume 3D* akibat keterbatasan sumber daya komputasi.

## ABSTRACT

Low-Dose Computed Tomography (LDCT) is widely used in lung cancer screening to reduce radiation exposure to patients. However, reducing radiation dose leads to increased noise in CT images, which can degrade image quality and hinder accurate detection of lung nodules. Therefore, image reconstruction and enhancement methods are required to effectively suppress noise while preserving important structural information.

In this final project, several deep learning models were developed and evaluated to improve LDCT image quality, namely 2D RED-CNN, 2D Multiwave Convolutional Neural Network (MWCNN), and 3D Residual-Attention U-Net. The dataset used in this study was LoDoPaB-CT, which contains paired LDCT images and normal-dose CT ground truth data. The workflow included data preprocessing, model training, testing, and evaluation using quantitative metrics such as PSNR, SSIM, RMSE, and Contrast-to-Noise Ratio (CNR), as well as qualitative evaluation through image visualization.

The experimental results show that all models were able to improve LDCT image quality compared to the original input images. The 2D RED-CNN and MWCNN models effectively reduced noise but tended to produce slightly blurred images. Meanwhile, the 3D Residual-Attention U-Net was able to enhance inter-slice spatial information and improve CNR values, although it still faced limitations in full-volume 3D reconstruction due to computational resource constraints. Overall, deep learning-based approaches demonstrate strong potential for LDCT image enhancement and can be further developed for future clinical applications.

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	<b>1</b>
<b>ABSTRAK.....</b>	<b>2</b>
<b>DAFTAR ISI.....</b>	<b>4</b>
<b>DAFTAR GAMBAR.....</b>	<b>6</b>
<b>DAFTAR TABEL.....</b>	<b>7</b>
<b>BAB I.....</b>	<b>9</b>
<b>PENDAHULUAN.....</b>	<b>9</b>
1.1 Latar Belakang.....	9
1.2 Tujuan.....	9
1.3 Tujuan Laporan.....	9
1.4 Batasan Masalah.....	10
1.5 Sistematika Laporan.....	10
<b>BAB II.....</b>	<b>11</b>
<b>TINJUAN PUSTAKA.....</b>	<b>11</b>
2.1 Dasar Teori Pendukung.....	11
2.1.1 Prinsip Dasar Computed Tomography (CT) dan Dosis Radiasi.....	11
2.1.2 Teori Rekonstruksi Citra.....	13
2.1.3 Convolutional Neural Networks (CNN) untuk Denoising.....	14
2.1.4 Arsitektur Deep Learning Lanjutan.....	15
2.1.5 Fungsi Kerugian (Loss Functions).....	20
2.1.6 Metrik Evaluasi Kualitas Citra.....	21
<b>BAB III.....</b>	<b>25</b>
<b>METODOLOGI DAN HASIL.....</b>	<b>25</b>
3.1 Alat dan Bahan.....	25
3.1.1 Spesifikasi Perangkat Keras.....	25
3.1.2 Perangkat Lunak dan Pustaka.....	26
3.1.3 Dataset Penelitian.....	27
3.2 Metode Penelitian.....	27
3.2.1 Alur Penggerjaan Umum (Preprocessing FBP dan Dataset).....	27
3.2.2 Arsitektur Model Lightweight 3D Residual-Attention U-Net.....	30
3.2.3 RED-CNN.....	32
3.2.3 MW-CNN.....	33
3.3 Hasil dan Pembahasan.....	34
3.3.1 Hasil Simulasi Model Lightweight 3D Residual-Attention U-Net.....	34
3.3.2 Hasil Simulasi Model RED CNN.....	39
3.3.3 Hasil Simulasi Model MW CNN.....	41
<b>BAB V.....</b>	<b>43</b>
4.1 Kesimpulan.....	43
4.2 Saran.....	43
<b>DAFTAR REFERENSI.....</b>	<b>43</b>

## DAFTAR GAMBAR

Gambar 2.1 Geometri sistem akuisisi CT scan (Fan-Beam Geometry) dan prinsip atenuasi sinar-X.....	9
Gambar 2.2 Kurva karakteristik hubungan antara dosis radiasi (mAs) dan tingkat noise pada citra CT.....	9
Gambar 2.3 Citra objek asli (kiri) diproyeksikan menjadi sinogram (tengah). Proses pembalikan kembali menjadi citra dilakukan melalui Filtered Back Projection (kanan)... 10	
Gambar 2.4 Diagram alir algoritma Filtered Back Projection (FBP) dan contoh artefak streaking yang muncul akibat noise pada sinogram.....	11
Gambar 2.5 Komponen Convolutional Neural Network.....	11
Gambar 2.6 Struktur model CNN dengan encoder-decoder.....	12
Gambar 2.7 Skema arsitektur U-Net dengan blok Residual pada setiap tingkat resolusi... 13	
Gambar 2.8 .....	14
Gambar 2.9 Arsitektur RED CNN.....	15
Gambar 2.10 Flowchart Denoising RED-CNN.....	15
Gambar 2.11 Arsitektur Multiwave CNN.....	16
Gambar 2.12 Diagram posisi penelitian kami. Model yang diusulkan (kotak merah) menggabungkan keunggulan spasial 3D U-Net [1], stabilitas Residual Learning [2], dan fokus Attention Mechanism [8], namun dengan arsitektur yang diringankan untuk efisiensi memori.....	21
Gambar 3.1 Arsitektur tech stack yang digunakan dalam pengembangan sistem.....	23
Gambar 3.2 Diagram alir pra-pemrosesan data: Transformasi dari data sinogram mentah menjadi volume CT 3D terstandarisasi.....	25
Gambar 3.3 Transformasi domain data: (a) Sinogram input dosis rendah dengan noise Poisson, (b) Hasil rekonstruksi FBP yang menunjukkan degradasi kualitas berupa streak artifacts dan noise butiran akibat rendahnya statistik foton.....	26

Gambar 3.4 Komponen mikro arsitektur: (a) Residual Block yang memfasilitasi aliran gradien, dan (b) Attention Gate yang memfilter fitur spasial berdasarkan relevansi anatomi.....	28
Gambar 3.5 Diagram Alur Metode RED CNN.....	29
Gambar 3.7 Flowchart training model MWCNN.....	30
Gambar 3.8 Grafik riwayat training dan validasi model. Sumbu-Y menunjukkan nilai composite loss dan sumbu-X menunjukkan epoch.....	32
Gambar 3.9 Visualisasi hasil prediksi pada satu patch.....	33
Gambar 3.10 Perbandingan rekonstruksi volume penuh. (a) Citra input FBP dosis rendah yang dipenuhi noise butiran. (b) Citra output model yang halus dan bebas artefak garis. (c) Citra Ground Truth. (d) Density Map.....	35
Gambar 3.11 Antarmuka pengguna berbasis Streamlit untuk rekonstruksi interaktif. Pengguna dapat melihat perbandingan side-by-side antara input FBP, hasil prediksi AI, dan Ground Truth. Ada pula hasil evaluasi kuantitatif (PSNR, SSIM, RMSE, dan CNR) serta kualitatif yang dapat ditampilkan di bagian bawahnya.....	36
Gambar 3.12 Hasil RED CNN.....	37
Gambar 3.13 Hasil simulasi MWCNN.....	39

## **DAFTAR TABEL**

Tabel 2.1 Tinjauan State of The Art (SOTA) metode deep learning untuk rekonstruksi citra CT.....	23
Tabel 3.2 Variasi Training RED CNN.....	32
Tabel 3.3 Rata-rata metrik evaluasi pada data uji patch-level.....	36
Tabel 3.4 Perbandingan metrik rekonstruksi volume penuh (Full-Volume 3D).....	37
Tabel 3.5 Hasil Perbandingan Metrik Evaluasi Variasi Training RED CNN.....	40
Tabel 3.6 Analisis Kuantitatif.....	41

## **DAFTAR SINGKATAN**

CT : Computed Tomography

LDCT : Low-Dose Computed Tomography

U-net : U-shaped Network

RED - CNN : Residual Attention Convolutional Neural Network

MWCNN : Multi Wave Convolutional Neural Network

PSNR : Peak Signal To Noise Ratio

SSIM : Structural Similarity Index Measure

RMSE: Root Mean Square Error

CNR : Contrast To Noise Ratio

FBP : Field Back Projection

AI : Artificial Intelligence

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Kanker paru merupakan salah satu penyebab kematian tertinggi akibat kanker di dunia, yang sebagian besar kasusnya baru terdeteksi pada stadium lanjut sehingga menurunkan peluang keberhasilan terapi. Oleh karena itu, deteksi dini melalui program skrining menjadi sangat penting untuk meningkatkan peluang kesembuhan pasien.

*Computed Tomography (CT)* banyak digunakan dalam skrining kanker paru karena mampu menampilkan struktur anatomi paru secara detail. Namun, penggunaan *CT* secara rutin menimbulkan risiko paparan radiasi, sehingga dikembangkan metode *Low-Dose Computed Tomography (LDCT)* untuk menurunkan dosis radiasi. Akan tetapi, pengurangan dosis pada *LDCT* menyebabkan peningkatan noise yang dapat menurunkan kualitas citra dan menyamarkan nodul paru berukuran kecil.

Berbagai metode konvensional telah digunakan untuk mengurangi noise pada citra *LDCT*, namun sering menghasilkan citra yang *blur* dan menghilangkan detail struktur penting. Dengan perkembangan teknologi komputasi, pendekatan berbasis deep learning menjadi solusi yang efektif karena mampu mempelajari pola *noise* dan struktur citra secara kompleks. Oleh karena itu, pada final project mata kuliah Teknologi Pencitraan Biomedik ini kami melakukan pengembangan dan evaluasi model **2D RED-CNN**, **2D MWCNN**, dan **3D Residual-Attention U-Net** untuk meningkatkan kualitas citra *LDCT*.

#### **1.2 Tujuan**

1. Bagaimana pengaruh penggunaan *Low-Dose Computed Tomography (LDCT)* terhadap kualitas citra *CT*, khususnya terkait peningkatan *noise* dan penurunan kontras citra?
2. Bagaimana kinerja model *deep learning* dalam meningkatkan kualitas citra *LDCT* tanpa menghilangkan informasi struktural penting pada citra medis?
3. Sejauh mana perbandingan performa antara model **2D RED-CNN**, **2D Multiwave Convolutional Neural Network (MWCNN)**, dan **3D Residual-Attention U-Net** dalam proses peningkatan kualitas citra *LDCT*?
4. Bagaimana hasil evaluasi kualitas citra yang dihasilkan oleh masing-masing model berdasarkan metrik kuantitatif *PSNR*, *SSIM*, *RMSE*, dan *Contrast-to-Noise Ratio (CNR)*?
5. Apa saja keterbatasan dan kendala yang dihadapi dalam penerapan model *deep learning*, khususnya pada rekonstruksi citra **3D LDCT** dengan keterbatasan sumber daya komputasi?

### **1.3 Tujuan Laporan**

1. Menganalisis pengaruh penggunaan *Low-Dose Computed Tomography (LDCT)* terhadap kualitas citra *CT*, khususnya terkait peningkatan *noise* dan penurunan kontras citra.
2. Mengimplementasikan dan mengevaluasi model *deep learning* sebagai metode peningkatan kualitas citra *LDCT* tanpa menghilangkan informasi struktural penting.
3. Membandingkan kinerja beberapa arsitektur *deep learning*, yaitu *2D RED-CNN*, *2D Multiwave Convolutional Neural Network (MWCNN)*, dan *3D Residual-Attention U-Net*, dalam meningkatkan kualitas citra *LDCT*.
4. Mengevaluasi kualitas citra hasil pemrosesan menggunakan metrik kuantitatif *PSNR*, *SSIM*, *RMSE*, dan *Contrast-to-Noise Ratio (CNR)*.
5. Mengidentifikasi keterbatasan dan tantangan dalam penerapan model *deep learning*, khususnya pada pengolahan dan rekonstruksi citra *3D LDCT* dengan keterbatasan sumber daya komputasi.

### **1.4 Batasan Masalah**

1. Penelitian ini hanya membahas peningkatan kualitas citra *Low-Dose Computed Tomography (LDCT)* pada citra paru-paru (*thorax*).
2. Dataset yang digunakan dalam penelitian ini adalah *LoDoPaB-CT* Dataset, yang menyediakan pasangan data *LDCT* dan citra *CT* dosis normal sebagai *ground truth*.
3. Model *deep learning* yang digunakan terbatas pada *2D RED-CNN*, *2D Multiwave Convolutional Neural Network (MWCNN)*, dan *3D Residual-Attention U-Net*.
4. Proses pengolahan citra difokuskan pada *denoising* dan peningkatan kualitas citra, tanpa mencakup tahap segmentasi, klasifikasi, maupun diagnosis klinis.
5. Evaluasi performa model dilakukan menggunakan metrik kuantitatif *PSNR*, *SSIM*, *RMSE*, dan *Contrast-to-Noise Ratio (CNR)*, serta evaluasi kualitatif melalui visualisasi citra.
6. Implementasi dan pengujian model dilakukan dalam lingkungan komputasi terbatas, sehingga aspek optimasi lanjutan dan penerapan klinis secara *real-time* tidak dibahas secara mendalam.

### **1.5 Sistematika Laporan**

#### **Bab I Pendahuluan**

Bab ini berisi pendahuluan yang meliputi latar belakang penelitian, rumusan masalah, tujuan laporan, batasan masalah, serta sistematika laporan yang menjadi dasar dan arah pembahasan penelitian.

#### **Bab II Tinjauan Pustaka**

Bab ini membahas dasar teori pendukung yang relevan dengan topik penelitian, seperti konsep dasar *Computed Tomography (CT)*, *Low-Dose CT (LDCT)*, *noise* pada citra medis, serta metode *deep learning* yang digunakan. Selain itu, pada

bab ini juga disajikan posisi dan penjelasan penelitian kelompok dalam bentuk tabel *State of The Art (SOTA)* beserta pembahasannya.

### **Bab III Metodologi dan Hasil**

Bab ini menjelaskan metodologi penelitian yang digunakan, meliputi alat dan bahan yang digunakan, metode yang diterapkan dalam pengolahan dan peningkatan kualitas citra *LDCT*, serta hasil simulasi dan pembahasan dari implementasi model yang dilakukan.

### **Bab IV Penutup**

Bab ini berisi kesimpulan yang diperoleh dari hasil penelitian serta saran atau langkah pengembangan selanjutnya (*next step*) yang dapat dilakukan untuk penelitian di masa depan.

## BAB II

### TINJUAN PUSTAKA

Bab ini membahas dasar teori pendukung yang relevan dengan topik penelitian, seperti konsep dasar Computed Tomography (CT), Low-Dose CT (LDCT), *noise* pada citra medis, serta metode deep learning yang digunakan. Selain itu, pada bab ini juga disajikan posisi dan penjelasan penelitian kelompok dalam bentuk tabel State of The Art (SOTA) beserta pembahasannya.

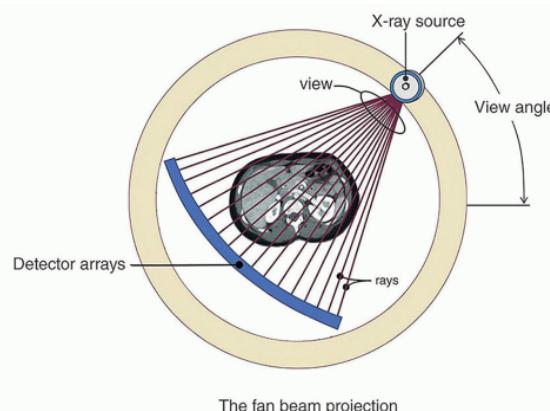
#### 2.1 Dasar Teori Pendukung

##### 2.1.1 Prinsip Dasar Computed Tomography (CT) dan Dosis Radiasi

Computed Tomography (CT) merupakan modalitas pencitraan medis yang memanfaatkan transmisi sinar-X untuk memvisualisasikan struktur internal tubuh dalam bentuk irisan tomografi. Prinsip dasar pembentukan citra CT bergantung pada atenuasi intensitas foton sinar-X saat menembus objek dengan densitas yang bervariasi. Fenomena ini dijelaskan melalui Hukum Beer-Lambert, di mana intensitas sinar yang diteruskan ( $I$ ) merupakan fungsi eksponensial dari intensitas awal ( $I_0$ ) dan koefisien atenuasi linear material ( $\mu$ ) sepanjang lintasan  $x$ , seperti ditunjukkan pada Persamaan (2.1).

$$I = I_0 \cdot e^{-\int \mu(x) dx}$$

Dalam implementasi klinis, parameter akuisisi utama yang menentukan kualitas citra dan dosis radiasi adalah tegangan tabung (kVp) dan arus tabung dikalikan waktu (mAs). Dosis radiasi memiliki korelasi linear terhadap mAs. Semakin tinggi arus tabung, semakin banyak foton yang ditembakkan, yang berdampak pada peningkatan Signal-to-Noise Ratio (SNR) [1]. Skema geometri sistem CT dan interaksinya dengan detektor dapat dilihat pada Gambar 2.1.

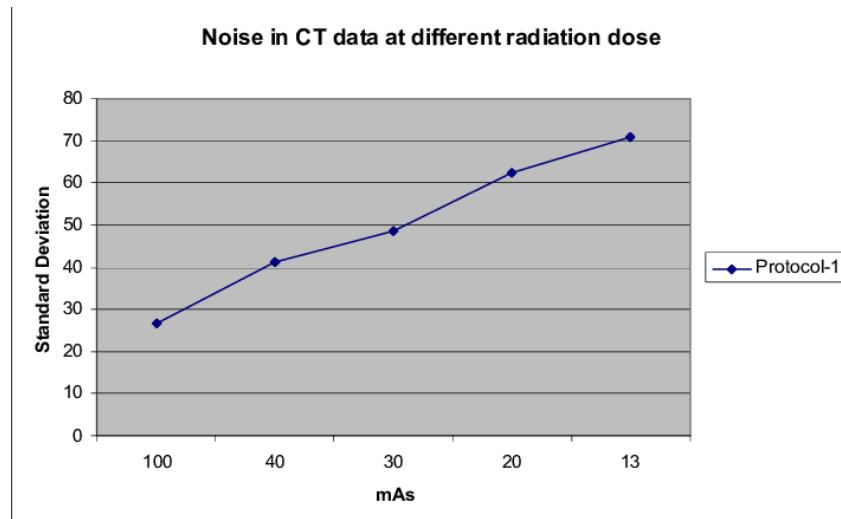


**Gambar 2.1** Geometri sistem akuisisi CT *scan* (Fan-Beam Geometry) dan prinsip atenuasi sinar-X  
Sumber: radiologykey.com

Namun, penggunaan mAs yang tinggi meningkatkan risiko karsinogenesis akibat paparan radiasi pengion. Oleh karena itu, protokol *low-dose* CT (LDCT) diterapkan dengan menurunkan mAs secara signifikan, khususnya untuk keperluan skrining rutin

seperti pada kanker paru. Konsekuensi fisis dari penurunan jumlah foton ini adalah munculnya *quantum noise*.

*Quantum noise* pada LDCT mengikuti distribusi Poisson dan statistik Gaussian pada data proyeksi mentah. Hal ini menyebabkan sinyal yang diterima detektor menjadi tidak stabil dan menghasilkan fluktuasi intensitas acak pada citra rekonstruksi. Hubungan antara dosis radiasi dan tingkat *noise* divisualisasikan pada Gambar 2.2, di mana penurunan dosis secara drastis menyebabkan degradasi kualitas citra berupa efek berbintik yang dapat mengaburkan detail anatomis halus seperti nodul paru.

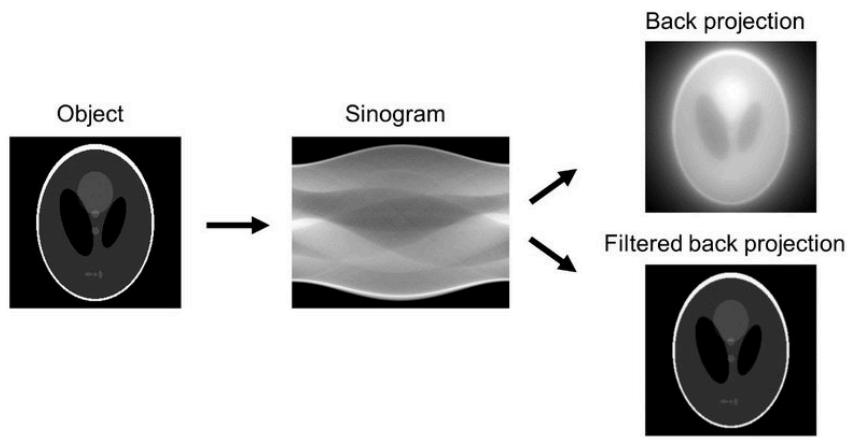


**Gambar 2.2** Kurva karakteristik hubungan antara dosis radiasi (mAs) dan tingkat *noise* pada citra CT  
Sumber: researchgate.net

Dalam penelitian ini, dataset LoDoPaB-CT yang digunakan mensimulasikan kondisi *low-dose* tersebut dengan menginjeksikan Poisson *noise* pada data proyeksi, merepresentasikan kondisi klinis nyata di mana foton yang sampai ke detektor sangat terbatas.

### 2.1.2 Teori Rekonstruksi Citra

Proses pembentukan citra CT dari data mentah menggunakan transformasi domain. Data yang diakuisisi oleh detektor bukan citra spasial, melainkan sekumpulan profil atenuasi dari berbagai sudut pandang yang disebut data proyeksi. Kumpulan data proyeksi ini dipetakan dalam sebuah matriks yang disebut sinogram [5]. Sinogram merepresentasikan transformasi Radon dari objek yang dipindai, di mana sumbu vertikal menunjukkan sudut proyeksi ( $\theta$ ) dan sumbu horizontal menunjukkan posisi detektor ( $s$ ). Ilustrasi perubahan dari objek spasial menjadi sinogram ditunjukkan pada Gambar 2.3.



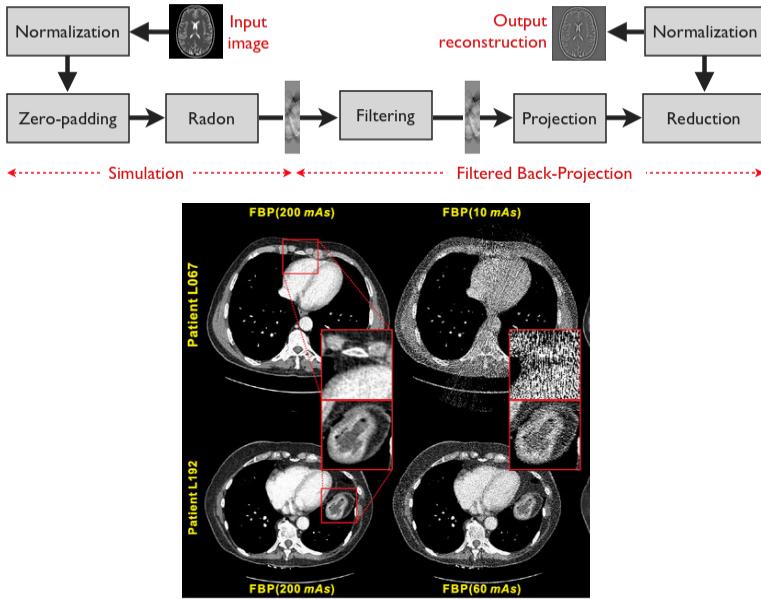
**Gambar 2.3** Citra objek asli (kiri) diproyeksikan menjadi sinogram (tengah). Proses pembalikan kembali menjadi citra dilakukan melalui Filtered Back Projection (kanan)

Untuk mengembalikan sinogram menjadi citra anatomis (transformasi Radon Balik), metode analitik standar yang digunakan adalah Filtered Back Projection (FBP). FBP menjadi metode rekonstruksi dasar yang digunakan dalam proyek ini untuk mempersiapkan data input bagi model *deep learning*. Metode ini dipilih karena efisiensi komputasi yang tinggi dibandingkan metode iteratif.

Secara matematis, FBP terdiri dari dua tahap utama:

1. Filtering: Proyeksi difilter dalam domain frekuensi (biasanya menggunakan filter Ram-Lak) untuk mengompensasi efek *blurring* yang terjadi akibat *sampling* radial.
2. Back Projection: Data yang telah difilter diproyeksikan balik ke dalam matriks citra 2D.

Meskipun FBP sangat cepat, metode ini memiliki kelemahan pada kondisi dosis rendah. Karena sifat filter Ram-Lak yang merupakan *high-pass filter*, FBP cenderung mengamplifikasi noise frekuensi tinggi [6]. Akibatnya, pada input LDCT, hasil rekonstruksi FBP akan mengandung *streak artifacts* dan *noise* butiran yang parah, seperti yang diilustrasikan pada perbandingan rekonstruksi di Gambar 2.4.

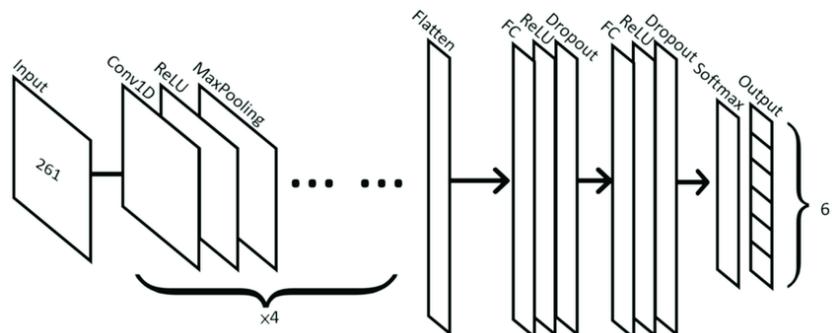


**Gambar 2.4** Diagram alir algoritma Filtered Back Projection (FBP) dan contoh artefak *streaking* yang muncul akibat *noise* pada sinogram

Oleh karena itu, dalam penelitian ini, citra hasil rekonstruksi FBP tidak digunakan sebagai hasil akhir diagnosis, melainkan sebagai input awal yang akan diperbaiki kualitasnya menggunakan arsitektur Lightweight 3D Residual-Attention U-Net.

### 2.1.3 Convolutional Neural Networks (CNN) untuk Denoising

CNN dirancang untuk mempelajari pola visual melalui konvolusi, yaitu proses menggeser filter pada citra untuk mengekstraksi fitur penting. Filter ini akan merespons pola tertentu seperti tepi, tekstur, atau struktur spesifik. Pada lapisan awal, CNN biasanya mendekripsi pola sederhana seperti edge, sementara pada lapisan yang lebih dalam, jaringan mulai mengenali struktur yang lebih kompleks. Untuk denoising, CNN belajar membedakan pola yang termasuk informasi struktur citra asli dengan pola yang termasuk noise, sehingga jaringan dapat menekan noise tanpa merusak detail penting yang bersifat diagnostik.

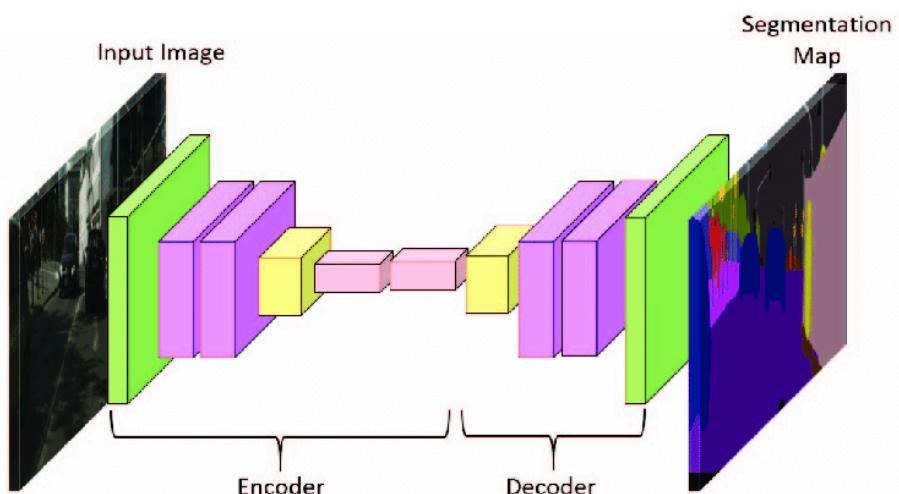


**Gambar 2.5** Komponen Convolutional Neural Network

Secara umum, CNN tersusun atas beberapa komponen utama, yaitu

- a. Convolution layer yang berfungsi mengekstraksi fitur dengan menggunakan filter yang digeser pada citra untuk mendeteksi pola seperti tepi dan tekstur.
- b. Activation function ReLU yang membuat jaringan menjadi bersifat non-linear dengan menahan nilai positif dan menekan nilai negatif.
- c. Max pooling digunakan untuk melakukan *downsampling*, yaitu mengecilkan ukuran data sambil mempertahankan fitur paling penting sehingga jaringan lebih efisien dan stabil terhadap pergeseran.

Pada CNN yang digunakan untuk klasifikasi, fitur kemudian diubah menjadi vektor melalui flatten, dan lapisan keluaran biasanya menggunakan softmax untuk menghasilkan probabilitas kelas. Namun pada tugas denoising, flatten dan softmax umumnya tidak digunakan karena jaringan tidak memprediksi kelas, melainkan merekonstruksi citra agar tetap mempertahankan struktur spasialnya.



**Gambar 2.6** Struktur model CNN dengan encoder-decoder

Banyak model denoising modern menggunakan konsep encoder-decoder architecture. Pada bagian encoder, citra mengalami serangkaian proses konvolusi dan downsampling, yang bertujuan untuk menyederhanakan representasi data, memperluas receptive field, dan menangkap informasi dengan konteks global atau keseluruhan. Selanjutnya, pada bagian decoder, jaringan melakukan upsampling untuk mengembalikan citra ke resolusi semula sambil merekonstruksi detail wilayah spasial. Mekanisme inilah yang menjadi fondasi berbagai model denoising berbasis CNN seperti RED-CNN dan MWCNN, dengan perbedaan terutama pada bagaimana fitur dipertahankan dan bagaimana proses rekonstruksi dilakukan.

#### 2.1.4 Arsitektur Deep Learning Lanjutan

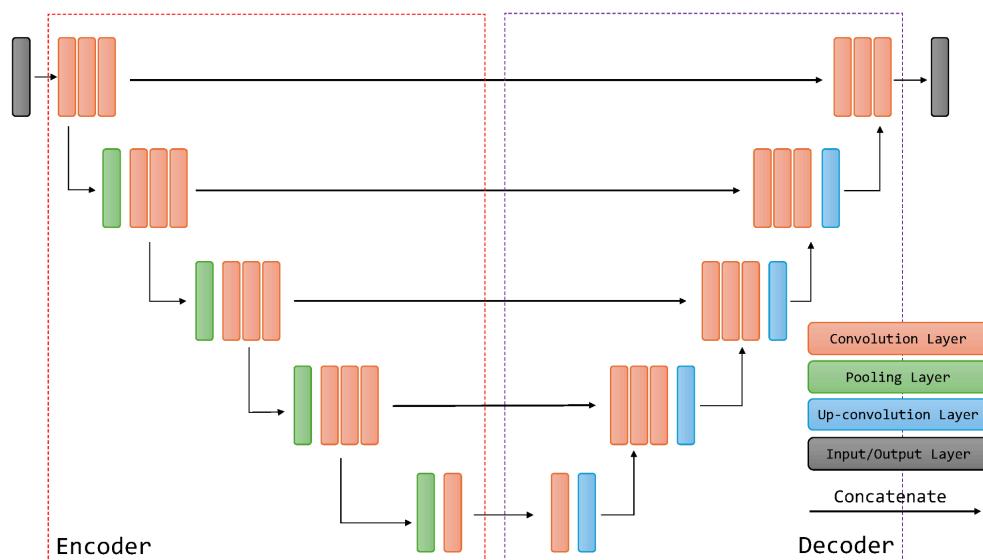
##### A. 3D Residual-Attention U-Net

Dalam pengolahan citra medis, khususnya untuk data volumetrik seperti CT scan, penggunaan arsitektur 2D seringkali memiliki keterbatasan dalam menangkap informasi spasial antar-irisan. Oleh karena itu, arsitektur 3D U-Net yang dikombinasikan dengan Residual Learning dan Attention Mechanism menjadi

pendekatan yang lebih baik untuk menjaga kontinuitas struktur anatomis tiga dimensi.

### 1. Arsitektur 3D U-Net dan Pemrosesan Volumetrik

Arsitektur U-Net memiliki struktur khas berbentuk huruf "U" yang terdiri dari jalur kontraksi (*encoder*) dan jalur ekspansi (*decoder*). Pengembangan lebih lanjut menjadi 3D U-Net memungkinkan jaringan untuk memproses volume input ( $x, y, z$ ) secara simultan menggunakan operasi konvolusi 3D, bukan sekadar memproses irisan 2D secara independen. Hal ini krusial dalam rekonstruksi *low-dose* CT karena artefak atau nodul sering kali memiliki konsistensi bentuk pada sumbu-Z yang mungkin hilang jika hanya dilihat dari satu irisan saja. Struktur dasar *encoder* berfungsi untuk mengekstraksi fitur hirarkis dari citra input dan mengurangi dimensi spasial untuk menangkap konteks global. Sebaliknya, *decoder* berfungsi untuk mengembalikan resolusi spasial dan merekonstruksi detail citra. Fitur unik dari U-Net adalah adanya *skip connections* yang mentransfer *feature maps* dari *encoder* langsung ke *decoder* pada resolusi yang bersesuaian. Mekanisme ini memungkinkan *decoder* untuk mendapatkan kembali detail lokasi spasial yang mungkin hilang saat proses *downsampling*. Visualisasi arsitektur dasar yang mengintegrasikan U-Net dengan mekanisme atensi diperlihatkan pada Gambar 2.5.



**Gambar 2.7** Skema arsitektur U-Net dengan blok Residual pada setiap tingkat resolusi

Garis panah melambangkan *skip connection* yang menghubungkan fitur *encoder* ke *decoder* untuk memulihkan detail spasial.

### 2. Integrasi Residual Learning

Untuk meningkatkan efisiensi pelatihan dan mencegah masalah degradasi pada jaringan yang dalam (*vanishing gradient*), blok konvolusi standar dalam U-Net digantikan atau diperkuat dengan blok Residual. Seperti yang dijelaskan dalam teori ResNet oleh He et al. (2016), blok ini memungkinkan aliran informasi gradien yang lebih lancar melalui penambahan identitas input ke *output layer*.

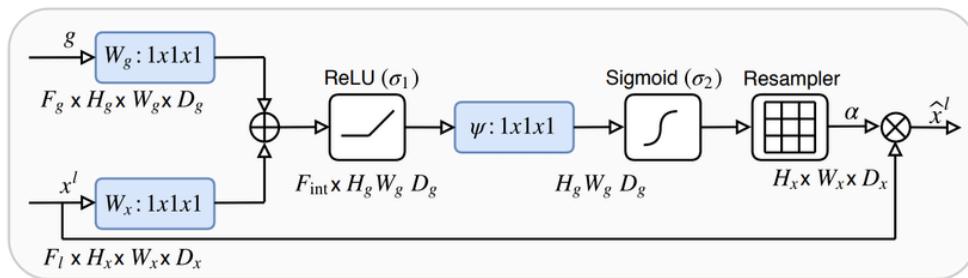
$(y = F(x) + x)$ . Dalam konteks *denoising*, ini berarti model lebih mudah belajar untuk memprediksi *noise* residual (selisih antara citra bersih dan kotor) dibandingkan memprediksi citra bersih dari nol, yang mempercepat konvergensi model.

### 3. Attention Mechanism

Salah satu tantangan utama dalam *denoising* citra medis adalah risiko *oversmoothing*, di mana filter konvolusi standar memperlakukan seluruh piksel dengan bobot yang sama, sehingga area latar belakang yang tidak relevan ikut diproses dan dapat mengaburkan tepi objek penting seperti nodul. Untuk mengatasi hal ini, modul Attention Gate (AG) diintegrasikan ke dalam *skip connections* sebelum fitur digabungkan ke *decoder*. Secara matematis, AG bekerja dengan memfilter fitur input ( $x$ ) menggunakan sinyal pemicu (gating signal,  $g$ ) yang berasal dari lapisan yang lebih dalam. Modul ini menghasilkan peta bobot (*attention coefficients*) dengan nilai antara 0 hingga 1. Nilai mendekati 1 diberikan pada area target yang relevan (seperti struktur nodul atau pembuluh darah), sedangkan nilai mendekati 0 diberikan pada area latar belakang yang irelevan. Mekanisme ini memaksa jaringan untuk fokus hanya pada fitur informatif dan menekan respon fitur yang membawa noise.

$$\alpha = \sigma_2(\Psi(\sigma_1(W_x x + W_g g + b_g)))$$

Di mana  $\sigma$  adalah fungsi aktivasi (Sigmoid/ReLU), dan  $W$  adalah bobot konvolusi. Struktur detail dari modul atensi ini dapat dilihat pada Gambar 2.6. Dengan adanya AG, model Lightweight 3D Residual-Attention U-Net yang digunakan dalam penelitian ini mampu mempertahankan ketajaman struktur anatomis kecil meskipun menggunakan parameter yang jauh lebih sedikit dibandingkan model U-Net standar [4].



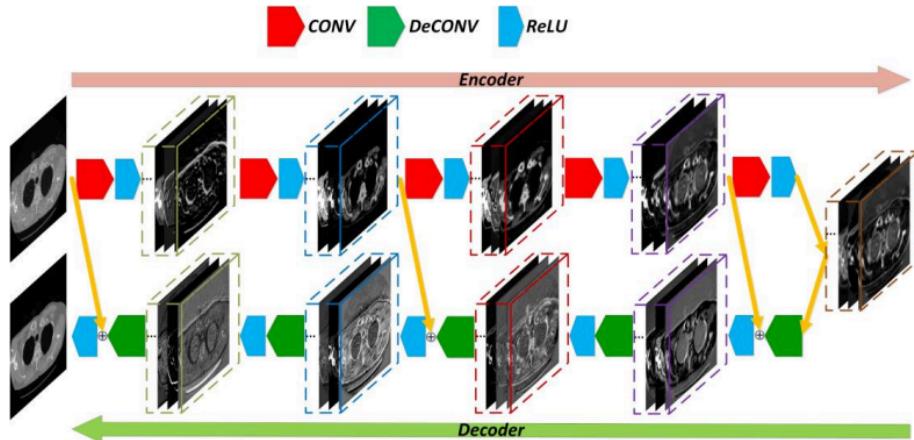
**Gambar 2.8** Diagram skematik detail dari modul Attention Gate (AG). Input fitur ( $x^l$ ) dan sinyal pemicu ( $g$ ) digabungkan melalui operasi konvolusi ( $W_x, W_g$ ) dan fungsi aktivasi non-linear ( $\sigma_1, \sigma_2$ ) untuk menghasilkan koefisien atensi ( $\alpha$ ). Koefisien ini kemudian dikalikan dengan input awal untuk menekan *noise* latar belakang dan menonjolkan area target

## B. 2D RED -CNN

Dengan pertimbangan keterbatasan komputasi, RED-CNN digunakan sebagai metode alternatif yang memiliki kompleksitas lebih ringan dibandingkan arsitektur 3D. Meskipun bekerja pada domain dua dimensi, RED-CNN memiliki struktur encoder-decoder dengan residual learning yang dirancang khusus untuk tugas

penghilangan noise pada citra low-dose CT. Kombinasi tersebut membuat RED-CNN tetap mampu menekan noise secara efektif sambil mempertahankan detail struktur anatomis penting, sehingga relevan digunakan sebagai pembanding dalam *project* ini.

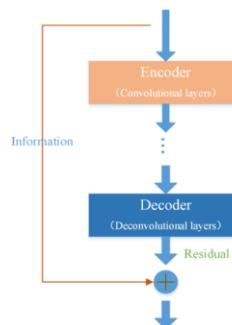
## 1. Arsitektur 2D RED-CNN



**Gambar 2.9** Arsitektur RED CNN

Arsitektur 2D RED-CNN bekerja dengan memproses citra CT dosis rendah hasil rekonstruksi FBP sebagai input utama. Citra ini terlebih dahulu dilewatkan ke bagian encoder yang terdiri dari beberapa lapisan konvolusi untuk menyaring noise sekaligus menangkap fitur anatomis penting. Proses ini dilakukan tanpa pooling sehingga ukuran citra tetap terjaga. Selanjutnya, fitur yang telah dipelajari diteruskan ke bagian decoder yang berfungsi mengembalikan detail citra dan mengurangi artefak akibat noise. Untuk memastikan informasi penting tidak hilang selama proses ini, fitur dari encoder ditambahkan langsung ke decoder melalui mekanisme residual addition. Dengan alur tersebut, RED-CNN mampu menghasilkan citra CT yang lebih bersih sambil tetap mempertahankan struktur anatomis utama.

## 2. Strategi Denoising pada RED-CNN



**Gambar 2.10** Flowchart Denoising RED-CNN

Untuk mengatasi permasalahan denoising pada citra low-dose CT, mekanisme residual compensation diintegrasikan ke dalam arsitektur RED-CNN dengan

mengacu pada konsep deep residual learning. Alih-alih memetakan citra input langsung ke citra keluaran melalui tumpukan lapisan konvolusi, jaringan menerapkan residual mapping seperti yang ditunjukkan pada Gambar 2. Dengan mendefinisikan citra input sebagai  $I$  dan citra keluaran sebagai  $O$ , residual mapping dinyatakan sebagai

$$F(I) = O - I$$

yang dipelajari menggunakan lapisan konvolusi bertingkat. Setelah residual mapping diperoleh, citra keluaran dapat direkonstruksi kembali melalui

$$O = F(I) + I.$$

Pendekatan ini mengubah permasalahan pemetaan langsung menjadi permasalahan pemetaan residual, sehingga jaringan lebih mudah mempelajari komponen noise dan meningkatkan stabilitas proses pelatihan.

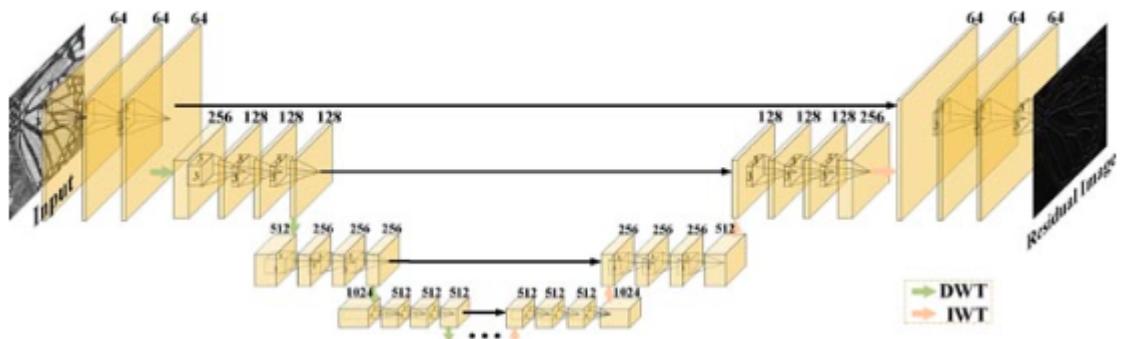
### C. 2D Multi-Wave CNN

Sebagai model pembanding hasil 2D RED CNN untuk denoising dan peningkatan kualitas citra, MWCNN digunakan karena arsitekturnya memanfaatkan 2D Wavelet Transform yang dapat memisahkan sinyal frekuensi rendah dan frekuensi tinggi. Dengan komponen frekuensi rendah, seperti struktur utama dari paru-paru, lalu komponen frekuensi tinggi, seperti detail anatomi dan noise hasil FBP. Melalui pendekatan multi frekuensi ini, MWCNN dapat melakukan proses denoising secara lebih fokus pada domain frekuensi, sehingga dapat menghasilkan output yang minim noise tanpa menghilangkan detail anatomi. Selain itu, penggunaan wavelet membuat model lebih stabil terhadap variasi noise pada citra CT-low dose.

#### 1. Arsitektur 2D Multiwave CNN

MWCNN dirancang sebagai pengembangan CNN dengan encoder dan decoder, namun proses *downsampling* dan *upsampling* diganti dengan Discrete Wavelet Transform (DWT) dan Inverse DWT (IDWT). Pada bagian encoder, citra CT 2D dari dataset akan diuraikan menjadi beberapa *sub-band* frekuensi menggunakan DWT, yaitu:

- a. *Sub-band LL* (Low-Low) menyimpan komponen frekuensi rendah yang merepresentasikan struktur anatomi dari thoraks. *Sub-band* ini paling stabil dan paling sedikit noise.
- b. *LH*, *HL*, *HH* menyimpan komponen dengan frekuensi tinggi yang mengandung detail halus dari citra sekaligus noise akibat radiasi rendah.



**Gambar 2.11** Arsitektur Multiwave CNN

Pada visualisasi arsitektur MWCNN di atas, proses DWT direpresentasikan sebagai tahapan penurunan resolusi namun sekaligus meningkatkan jumlah *feature channels*. Tahap ini memisahkan citra ke dalam beberapa sub-band frekuensi yang kemudian diproses melalui blok konvolusi pada setiap level. Prosesnya berlangsung secara multi-level sehingga jaringan mampu menangkap informasi global dan detail lokal secara bersamaan. Lalu, IDWT digunakan sebagai inverse encoder untuk menggabungkan kembali sub-band frekuensi yang telah diproses. Dengan cara ini, citra hasil rekonstruksi dapat tetap menjaga struktur utama thoraks beriringan dengan meminimalisir noise. Selain itu, adanya skip-connection antar level membantu menjaga kestabilan informasi. Hal ini mirip dengan konsep U-Net, namun IDWT bekerja pada domain frekuensi.

## 2. Frequency-Domain Learning pada MWCNN

Pada MWCNN, proses denoising berlangsung di domain frekuensi melalui wavelet transform. Secara umum, sinyal atau citra dapat direpresentasikan kembali menggunakan kombinasi komponen frekuensi rendah dan frekuensi tinggi. Bentuk umum rekonstruksi sinyal pada transformasi wavelet dapat dituliskan sebagai:

$$x(t) = \sum_k A_{L,k} \phi_{L,k}(t) + \sum_k D_{L,k} \psi_{L,k}(t) + \sum_k D_{L-1,k} \phi_{L-1,k}(t) + \dots + \sum_k D_{1,k} \psi_{1,k}(t)$$

Keterangan:

$A_{L,k}$  = koefisien aproksimasi level Low

$D_{L,k}$  = koefisien detail

$\phi_{L,k}(t)$  = scaling function dengan basis frekuensi rendah

$\psi_{L,k}(t)$  = wavelet function dengan basis frekuensi tinggi pada tiap level

Rumus di atas menunjukkan bahwa citra CT terdiri dari struktur anatomi utama detail multi-skala yang disusun secara hierarkis. Dengan menggunakan

MWCNN, model yang dibuat dapat mempertahankan stabilitas komponen struktur utama sehingga anatomi thoraks tetap konsisten dan memproses komponen detail secara selektif sehingga jaringan mampu membedakan mana detail yang relevan secara klinis dan mana yang merupakan noise. Dengan bekerja langsung pada representasi frekuensi ini, MWCNN mampu melakukan denoising secara lebih terarah, mengurangi risiko oversmoothing, dan menghasilkan citra rekonstruksi yang tetap tajam serta bermakna secara diagnostik.

### 2.1.5 Fungsi Kerugian (Loss Functions)

Fungsi kerugian atau *loss function* merupakan metrik matematis yang digunakan untuk mengukur sejauh mana hasil prediksi model menyimpang dari data referensi atau *ground truth*. Fungsi ini berperan sebagai aturan utama dalam proses pembelajaran, karena nilai loss akan digunakan untuk memperbarui bobot jaringan selama pelatihan. Pada arsitektur ini, digunakan kombinasi beberapa *loss function* agar model tidak hanya mengejar kesamaan numerik secara statistik, tetapi juga mampu menghasilkan citra dengan kualitas yang lebih baik.

#### A. Mean Squared Error (MSE)

Mean Squared Error merupakan fungsi kerugian dasar yang menghitung rata-rata kuadrat selisih antara nilai piksel citra hasil prediksi dan citra target. Secara matematis, MSE dirumuskan sebagai:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - y_i')^2$$

MSE sangat sensitif terhadap perbedaan nilai piksel yang besar sehingga efektif dalam menurunkan error rekonstruksi secara keseluruhan. Namun, kelemahan utama MSE adalah kecenderungannya menghasilkan citra yang terlihat blur, karena model sering mengambil nilai rata-rata untuk meminimalkan error global, yang dapat menyebabkan hilangnya detail frekuensi tinggi dan tekstur halus.

#### B. Structural Similarity Index (SSIM) Loss

Berbeda dengan MSE yang hanya membandingkan nilai piksel secara langsung, SSIM mengevaluasi kemiripan citra berdasarkan tiga komponen utama, yaitu *luminance*, *contrast*, dan *structure*. SSIM dirancang untuk meniru persepsi visual manusia yang lebih peka terhadap perubahan struktur dibandingkan perubahan intensitas piksel semata. Dengan menggunakan SSIM sebagai *loss function*, model didorong untuk mempertahankan bentuk dan struktur anatomi, sehingga hasil denoising tidak mengalami distorsi atau deformasi yang dapat mengurangi nilai klinis citra.

#### C. Gradient Loss

Gradient Loss menghitung perbedaan gradien antara citra hasil prediksi dan citra target, yaitu perubahan intensitas antar piksel yang berdekatan. Fungsi ini secara khusus difokuskan pada pelestarian tepi atau *edge preservation*. Dalam citra medis seperti CT scan, batas antar organ dan tepi nodul merupakan informasi

yang sangat penting. Dengan memasukkan Gradient Loss, model dipaksa untuk menjaga ketajaman batas tersebut agar tidak kabur akibat proses denoising.

#### D. Composite Loss (Total Loss)

Untuk memperoleh hasil rekonstruksi yang optimal, ketiga fungsi kerugian tersebut digabungkan dalam satu *composite loss* atau *total loss*. Penggabungan ini dilakukan dengan memberikan bobot tertentu pada masing-masing komponen, sehingga kontribusinya dapat diatur secara seimbang:

$$L_{total} = \lambda_1 LMSE + \lambda_2 LSSIM + \lambda_3 LGrad$$

Dengan menggunakan *composite loss*, model mampu menyeimbangkan akurasi numerik pada level piksel, kesesuaian struktur anatomi, serta ketajaman tepi citra. Pendekatan ini memungkinkan model menghasilkan citra low-dose CT yang tidak hanya bersih dari noise, tetapi juga tetap mempertahankan detail dan informasi struktural yang penting untuk analisis medis.

### 2.1.6 Metrik Evaluasi Kualitas Citra

Untuk menilai performa model secara objektif dan kuantitatif, digunakan beberapa metrik evaluasi yang umum digunakan dalam pengolahan citra digital dan pencitraan medis. Metrik-metrik ini digunakan untuk membandingkan citra hasil rekonstruksi atau denoising dengan citra referensi (*ground truth*), sehingga kualitas hasil model dapat dianalisis secara numerik dan struktural.

#### A. Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio digunakan untuk mengukur rasio antara nilai maksimum sinyal piksel terhadap noise yang memengaruhi kualitas citra. Nilai PSNR dinyatakan dalam satuan desibel (dB) dan dirumuskan sebagai:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX^2}{MSE} \right)$$

Nilai PSNR berbanding terbalik dengan MSE. Semakin tinggi nilai PSNR, semakin baik kualitas citra karena tingkat noise relatif lebih kecil dibandingkan informasi aslinya. Dalam konteks citra medis, nilai PSNR di atas 30 dB umumnya dianggap menunjukkan kualitas rekonstruksi yang baik.

#### B. Root Mean Square Error (RMSE)

Root Mean Square Error merupakan akar kuadrat dari rata-rata selisih kuadrat antar piksel antara citra hasil dan citra ground truth. Metrik ini memberikan gambaran besarnya kesalahan rekonstruksi dalam satuan yang sama dengan nilai intensitas piksel. Semakin rendah nilai RMSE, semakin akurat hasil rekonstruksi yang dihasilkan model. Nilai RMSE mendekati nol menunjukkan bahwa citra hasil sangat mirip dengan citra referensi.

#### C. Structural Similarity Index (SSIM)

Structural Similarity Index merupakan metrik berbasis persepsi yang mengevaluasi kemiripan citra berdasarkan tiga komponen utama, yaitu

*luminance, contrast, dan structure*. Rentang nilai SSIM berada antara 0 hingga 1, di mana nilai mendekati 1 menunjukkan tingkat kemiripan struktural yang tinggi. Dibandingkan PSNR, SSIM dianggap lebih representatif dalam menilai kualitas visual citra medis karena memperhatikan bagaimana struktur dan tekstur dikenali oleh sistem visual manusia.

#### D. Contrast-to-Noise Ratio (CNR)

Contrast-to-Noise Ratio merupakan metrik penting dalam pencitraan medis karena mengukur kemampuan citra dalam membedakan objek yang diminati, seperti nodul atau lesi, dari area latar belakang yang mengandung noise. Secara matematis, CNR dirumuskan sebagai:

$$CNR = \frac{|C_A - C_B|}{\sigma_0}$$

di mana CA dan CB masing-masing merepresentasikan intensitas rata-rata objek dan latar belakang, sedangkan  $\sigma_0$  merupakan standar deviasi noise. Semakin tinggi nilai CNR, semakin jelas objek medis dapat dibedakan dari latar belakang. Model yang baik diharapkan mampu meningkatkan nilai CNR dengan menekan noise tanpa mengurangi kontras objek utama.

#### 2.2 Posisi Penelitian dan Tinjauan State of The Art (SOTA)

Dalam pengembangan algoritma rekonstruksi *low-dose* CT (LDCT), pendekatan berbasis *deep learning* telah mengalami evolusi signifikan untuk mengatasi keterbatasan metode analitik konvensional. Tabel tinjauan pustaka (*state of the art*) yang merangkum berbagai arsitektur, kelebihan, serta keterbatasan metode terdahulu disajikan secara ringkas pada Tabel 2.1.

**Tabel 2.1** Tinjauan State of The Art (SOTA) metode *deep learning* untuk rekonstruksi citra CT

No	Referensi	Arsitektur & Detail Teknis	Kelebihan	Kekurangan	Feasibility
1	[1] O. Ronneberger et al. (2015)	Standard 3D U-Net. Encoder-Decoder dengan max pooling 3D dan up-conv 3D. Awalnya untuk segmentasi, sering diadaptasi untuk denoising.	Mampu menangkap konteks volumetrik (Z-axis). Struktur pembuluh darah terlihat kontinu antar-slice.	Parameter sangat besar. Cenderung oversmoothing (blur) jika hanya memakai MSE Loss.	SEDANG. Perlu VRAM besar.
2	[2] H. Chen et al. (2017)	RED-CNN. Residual Encoder-Decoder (2D based). Menggunakan shortcut connection penjumlahan (add).	Sangat stabil dan efektif membuang noise. Konvergensi cepat.	Aslinya 2D. Jika dipaksa 3D tanpa modifikasi, memori meledak karena feature map tidak dikurangi (tanpa pooling).	RENDAH. Berat di 3D karena tidak ada downsampling.
3	[3] K. H. Jin et al. (2017)	FBPConvNet. U-Net yang dimodifikasi untuk input domain FBP. Menggunakan residual learning global.	Efektif menghilangkan streak artifacts khas FBP. Arsitektur simpel dan robust.	Masih berbasis MSE, detail frekuensi tinggi (tekstur tulang/nodus) sering hilang.	TINGGI. Arsitektur U-Net standar yang mudah diubah ke 3D.
4	[4] H. Shan et al. (2018)	CPCE-3D. 3D CNN dengan Conveying Path. Menggunakan Transfer Learning dari model 2D ke 3D.	Solusi cerdas untuk melatih model 3D dengan data terbatas (pre-train 2D dulu). Hasil 3D sangat konsisten.	Pipeline training rumit (2 tahap). Implementasi transfer weight 2D-ke-3D di Python cukup tricky.	SEDANG. Strategi transfer learning-nya bagus ditiru untuk hemat data.
5	[5] Q. Yang et al. (2018)	WGAN-VGG. Generator berbasis U-Net, Discriminator CNN. Loss: Wasserstein + VGG Perceptual.	Tekstur visual terbaik dan tajam. Tidak terlihat blur seperti model MSE.	Training GAN 3D sangat tidak stabil (mode collapse). Sangat berat komputasinya (2 network berjalan).	RENDAH. 3D GAN butuh resource GPU mahal (A100/V100).
6	[6] E. Kang et al. (2017)	Wavelet-CNN. Menggunakan Wavelet Transform sebagai input multi-channel ke CNN.	Memproses frekuensi tinggi (detail) dan rendah (struktur) secara terpisah. Efisien membuat noise.	Pre-processing data (Wavelet decomposition) menambah kompleksitas coding dan waktu loading data.	SEDANG. Bisa diterapkan di 3D dengan 3D Wavelet Transform.
7	[7] Z. Zhang et al. (2019)	Dilated U-Net. Menggunakan Dilated (Atrous) Convolution pada blok tengah U-Net.	Receptive field luas tanpa downsampling agresif. Resolusi spasial terjaga baik.	Risiko gridding artifacts (pola kotak-kotak) jika rate dilasi tidak diatur hati-hati.	TINGGI. Sangat cocok untuk 3D karena bisa melihat konteks luas tanpa memboroskan memori.
8	[8] O. Oktay et al. (2018)	Attention U-Net. Menambahkan Attention Gates (AG) pada skip connections.	Model otomatis fokus pada area target (nodul/lesi) dan menekan background noise.	Menambah sedikit beban komputasi. Perlu tuning hyperparameter pada modul atensi.	SEDANG. Sangat bagus untuk justifikasi "fokus pada kanker".
9	[9] H. Wu et al. (2019)	U-Net++ (Nested U-Net). Skip connections yang padat dan bersarang (dense & nested).	Menjembatani semantic gap antara encoder dan decoder. Akurasi lebih tinggi dari U-Net biasa.	Jumlah parameter jauh lebih banyak dari U-Net standar. Training 3D akan sangat lambat.	RENDAH. Rawan Out of Memory untuk 3D di laptop mahasiswa.
10	[10] C. You et al. (2018)	GAN-CIRCLE. Menggunakan Cycle-Consistent Adversarial Networks untuk data unpaired.	Bisa dilatih tanpa data berpasangan (bisa pakai data pasien sembarang).	Sering terjadi deformasi anatomji jika loss constraint tidak kuat. Risiko halusinasi tinggi.	RENDAH. Anda punya data berpasangan (AAPM), metode ini tidak perlu.
11	[11] K. Liang et al. (2020)	EDCNN. Densely Connected CNN dengan Edge Enhancement.	Aliran informasi gradien sangat lancar (Dense). Tapi objek sangat tajam.	Memori GPU sangat boros karena Dense Block menyimpan semua feature map sebelumnya.	RENDAH. Sangat berat untuk 3D.
12	[12] Z. Jin et al. (2021)	DU-GAN. Dual Domain (Sinogram + Image) processing.	Secara teori paling superior karena memperbaiki data mentah dan visual sekaligus.	Butuh data sinogram (raw data) yang ukurannya TB-an. Sangat kompleks.	SANGAT RENDAH. Tidak feasible karena sinogram terlalu berat
13	[13] Z. Zhang et al. (2021)	TransCT. Hybrid Transformer + CNN.	Menangkap hubungan global jarak jauh (Global Context) menggunakan Self-Attention.	Transformer butuh data massive untuk pre-training. Sangat berat di 3D.	RENDAH. Terlalu berat.
14	[14] L. Gondara (2016)	Medical Denoising Autoencoder (DAE). Menggunakan Convolutional Autoencoder simpel.	Arsitektur sangat ringan dan cepat. Cocok untuk baseline.	Hasil rekonstruksi cenderung blur dan detail tekstur halus hilang total.	SANGAT TINGGI. Feasible tapi hasilnya mungkin terlalu sederhana.
15	[15] W. Wu et al. (2017)	Cascaded CNN. Menggunakan rangkaian CNN secara bertahap (iteratif).	Memperbaiki gambar secara bertahap (kasar ke halus). Error makin kecil di tiap tahap.	Training lama karena harus melatih beberapa network berurutan.	SEDANG. Bisa diterapkan di 3D dengan cascade pendek (2 tahap).

### 2.2.1 Analisis Kesenjangan

Berdasarkan Tabel 2.1, tren penelitian denoising LDCT dapat dikategorikan ke dalam tiga pendekatan utama yang masing-masing memiliki celah yang perlu diisi:

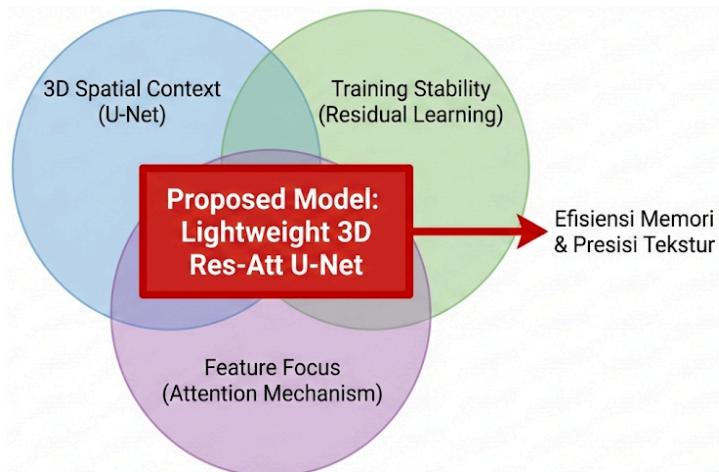
1. Pendekatan 2D Residual Learning (Referensi Utama: RED-CNN [2]): Model seperti RED-CNN yang dikembangkan Chen et al. (2017) terbukti sangat stabil dan cepat konvergen berkat penggunaan *residual shortcut*. Model ini menjadi landasan bagi dua anggota tim dalam proyek ini untuk mengembangkan model berbasis 2D. Namun, kelemahannya adalah pemrosesan dilakukan secara *slice-by-slice*, sehingga kontinuitas anatomis antar-irisasi (z-axis) sering terabaikan, menyebabkan artefak diskontinuitas saat citra disusun kembali menjadi volume 3D.
2. Pendekatan Domain Frekuensi/Wavelet (Referensi Utama: Kang et al. [6]): Metode Wavelet-CNN yang memisahkan komponen frekuensi tinggi (*noise*) dan rendah (struktur) sangat efisien secara komputasi. Namun, implementasi dekomposisi *wavelet* menambah kompleksitas pada tahap *pre-processing* dan sering kali menghilangkan tekstur halus nodul jika level dekomposisi tidak diatur dengan presisi.

3. Pendekatan 3D Volumetrik (Referensi Utama: 3D U-Net [1] & CPCE-3D [4]): Model 3D murni seperti yang diajukan Ronneberger et al. (2015) mampu menangkap korelasi spasial volumetrik secara utuh. Sayangnya, seperti dicatat pada kolom 'Kekurangan' di Tabel 2.1, model ini membutuhkan parameter yang sangat besar dan memori GPU (VRAM) yang masif, menjadikannya sulit diterapkan pada lingkungan komputasi terbatas. Selain itu, penggunaan fungsi *loss* MSE standar cenderung menghasilkan citra yang terlalu halus (*oversmoothing*), sehingga tekstur diagnostik penting menjadi kabur.

### 2.2.2 Kontribusi dan Posisi Penelitian Kelompok

Berdasarkan analisis kesenjangan di atas, penelitian kami diposisikan untuk menjabatani celah antara efisiensi model 2D dan akurasi spasial model 3D. Penelitian ini melakukan studi komparatif antara tiga arsitektur berbeda, yaitu RED-CNN (2D), Multi-Level Wavelet CNN (2D), dan Lightweight 3D Residual-Attention U-Net (3D).

Kontribusi spesifik dari model 3D Residual-Attention U-Net yang kami kembangkan adalah mengusulkan arsitektur yang mengatasi masalah *resource constraint* dan *oversmoothing* pada model 3D konvensional. Perbandingan posisi penelitian ini terhadap studi terdahulu divisualisasikan dalam diagram pemetaan pada Gambar 2.7.



**Gambar 2.12** Diagram posisi penelitian kami. Model yang diusulkan (kotak merah) menggabungkan keunggulan spasial 3D U-Net [1], stabilitas Residual Learning [2], dan fokus Attention Mechanism [8], namun dengan arsitektur yang diringankan untuk efisiensi memori.

Secara spesifik, kebaruan dan strategi yang diterapkan meliputi:

1. Arsitektur 3D yang Ringan: Berbeda dengan 3D U-Net standar [1] yang berat, model ini menggunakan strategi *channel reduction* (filter awal dikurangi menjadi 16) untuk mengurangi beban memori GPU secara drastis, namun mengompensasinya dengan teknik *micro-patching* agar detail lokal tetap terjaga.
2. Integrasi Mekanisme Atensi: Mengadopsi Attention Gate dari Oktay et al. [8] untuk memecahkan masalah pada RED-CNN [2] yang memperlakukan seluruh piksel sama rata. Hal ini memungkinkan model ringan ini tetap fokus pada struktur nodul kecil.

3. Fungsi Kerugian Komposit: Untuk mengatasi isu *blurring* pada CPCE-3D [4] dan RED-CNN [2], penelitian ini tidak hanya menggunakan MSE, melainkan menggabungkannya dengan Structural Similarity Index Measure (SSIM) Loss dan Gradient Loss untuk memaksa model mempertahankan ketajaman tepi dan tekstur perceptual.

## **BAB III**

### **METODOLOGI DAN HASIL**

Bab ini menjelaskan metodologi penelitian yang digunakan, meliputi alat dan bahan yang digunakan, metode yang diterapkan dalam pengolahan dan peningkatan kualitas citra *LDCT*, serta hasil simulasi dan pembahasan dari implementasi model yang dilakukan.

#### **3.1 Alat dan Bahan**

Pelaksanaan penelitian ini, mulai dari tahap pra-pemrosesan data, pelatihan model, hingga pengembangan antarmuka pengguna, dilakukan dengan memanfaatkan integrasi perangkat keras berbasis *cloud computing* dan perangkat lunak pemrograman tingkat tinggi.

##### **3.1.1 Spesifikasi Perangkat Keras**

Mengingat kompleksitas komputasi yang bervariasi antara model 2D (RED-CNN dan MWCNN) dan model 3D (Residual-Attention U-Net), penelitian ini menggunakan lingkungan komputasi heterogen yang disediakan oleh platform Google Colab.

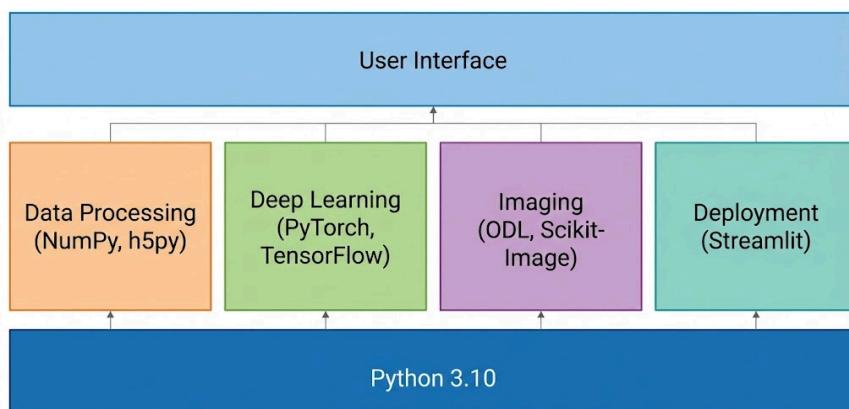
Pemrosesan data yang bersifat serial, seperti ekstraksi *patch* 2D dan evaluasi metrik, dijalankan menggunakan *central processing unit* (CPU) Intel Xeon. Sementara itu, untuk mempercepat proses pelatihan model 3D yang melibatkan operasi matriks volumetrik intensif, digunakan akselerator grafis *graphics processing unit* (GPU) NVIDIA Tesla T4. Seluruh proses komputasi dilakukan tanpa penggunaan *swap memory*, sehingga pemrosesan data bergantung sepenuhnya pada memori utama (RAM) yang tersedia untuk menjaga kecepatan *throughput* data. Rincian spesifikasi perangkat keras disajikan pada Tabel 3.1.

**Tabel 3.1** Spesifikasi perangkat keras lingkungan komputasi

Komponen	Spesifikasi Teknis	Fungsi Utama
Platform	Google Colab (Cloud)	Lingkungan eksekusi utama
CPU	Intel Xeon @ 2.20 GHz (2 vCPU)	Pra-pemrosesan data, pelatihan model 2D
GPU	NVIDIA Tesla T4 (16 GB GDDR6 VRAM)	Pelatihan model 3D, akselerasi ODL
RAM Sistem	12 GB (High-RAM Runtime)	Penyimpanan sementara data tensor
Storage	Google Drive (terintegrasi)	Penyimpanan dataset LoDoPaB dan <i>checkpoint</i> model

### 3.1.2 Perangkat Lunak dan Pustaka

Pengembangan kode program dilakukan menggunakan dua *integrated development environment*. Google Colab digunakan sebagai lingkungan eksekusi utama untuk pelatihan model karena aksesibilitasnya terhadap sumber daya GPU, sedangkan Visual Studio Code (VS Code) digunakan untuk penulisan kode modul aplikasi dan *debugging* lokal sebelum integrasi. Bahasa pemrograman yang digunakan adalah Python versi 3.10.12, yang dipilih karena dukungan ekosistem pustaka yang luas untuk *deep learning* dan pengolahan citra medis. Struktur ekosistem perangkat lunak yang digunakan dalam penelitian ini divisualisasikan pada Gambar 3.1.



Gambar 3.1 Arsitektur *tech stack* yang digunakan dalam pengembangan sistem.

Pustaka-pustaka Python yang digunakan dikategorikan berdasarkan fungsinya sebagai berikut:

1. Komputasi Numerik dan Manipulasi Data:
  - a. NumPy: Digunakan untuk operasi aljabar linear, manipulasi matriks n-dimensi, dan normalisasi data sinogram maupun citra rekonstruksi.
  - b. h5py: Diperlukan untuk mengakses dan membaca dataset LoDoPaB-CT yang disimpan dalam format hierarkis HDF5 (Hierarchical Data Format version 5) secara efisien tanpa perlu memuat seluruh data ke memori [1].
  - c. os, glob, zipfile: Digunakan untuk manajemen direktori, ekstraksi data terkompresi, dan penelusuran *path file* secara otomatis.
2. Kerangka Kerja Deep Learning:
  - a. PyTorch: Digunakan sebagai kerangka kerja utama untuk membangun, melatih, dan menguji arsitektur Lightweight 3D Residual-Attention U-Net. Pustaka ini dipilih karena fleksibilitasnya dalam menangani tensor 3D dan autograd dinamis [2].
  - b. TensorFlow dan Keras: Digunakan khusus dalam implementasi dan pelatihan model pembanding 2D (RED-CNN dan MWCNN) untuk memfasilitasi pembuatan lapisan *network* yang cepat.
3. Pengolahan Citra dan Rekonstruksi:
  - a. ODL (Operator Discretization Library): Pustaka inti yang digunakan untuk mendefinisikan geometri akuisisi CT dan melakukan operasi Ray Transform

- serta rekonstruksi FBP dengan dukungan akselerasi GPU (Astra Toolbox backend).
- b. Matplotlib: Digunakan untuk visualisasi hasil rekonstruksi, grafik fungsi *loss*, dan *density map*.
  - c. scikit-image: Digunakan untuk menghitung metrik evaluasi kuantitatif standar, yaitu Peak Signal-to-Noise Ratio (PSNR) dan Structural Similarity Index (SSIM).
4. Deployment dan Antarmuka:
    - a. Streamlit: Digunakan untuk membangun antarmuka aplikasi berbasis web yang memungkinkan pengguna mengunggah data sinogram/citra mentah dan melihat hasil rekonstruksi model secara interaktif dan *real-time*.

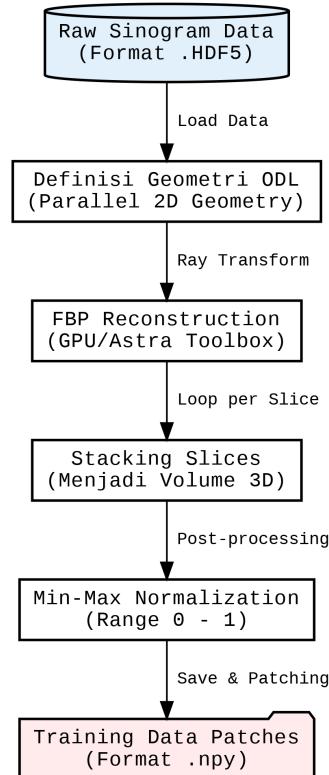
### 3.1.3 Dataset Penelitian

Data yang digunakan dalam penelitian ini adalah dataset LoDoPaB-CT (Low-Dose Parallel Beam Computed Tomography). Dataset ini merupakan standar tolok ukur untuk rekonstruksi CT dosis rendah yang berisi lebih dari 40.000 pasangan citra. Data terdiri dari citra referensi kualitas tinggi (*ground truth*) yang berasal dari dataset LIDC-IDRI, dan data simulasi sinogram dosis rendah yang telah ditambahkan Poisson *noise* untuk merepresentasikan kondisi klinis nyata pada pengurangan arus tabung sinar-X.

## 3.2 Metode Penelitian

### 3.2.1 Alur Pengerjaan Umum (Preprocessing FBP dan Dataset)

Tahap awal penelitian berfokus pada persiapan data input yang valid dan representatif. Mengingat arsitektur *deep learning* yang dikembangkan bekerja pada domain citra dan bukan domain sinogram, diperlukan proses rekonstruksi awal untuk mengubah data mentah menjadi volume paru-paru. Alur kerja pra-pemrosesan data dari ekstraksi dataset hingga normalisasi volume divisualisasikan pada Gambar 3.2.



**Gambar 3.2** Diagram alir pra-pemrosesan data: Transformasi dari data sinogram mentah menjadi volume CT 3D terstandarisasi.

### A. Akuisisi dan Karakteristik Dataset

Data yang digunakan bersumber dari repositori LoDoPaB-CT, sebuah dataset standar yang dikembangkan untuk mengevaluasi algoritma rekonstruksi invers. Dataset ini berisi lebih dari 40.000 pasangan data citra CT toraks manusia yang diambil dari koleksi LIDC-IDRI [1].

Setiap sampel data terdiri dari dua komponen yang berpasangan:

1. Ground Truth (xgt): Citra CT kualitas tinggi (dosis normal) yang berfungsi sebagai target referensi
2. Observation (yobs): Data sinogram simulasi dosis rendah yang dihasilkan dengan menambahkan Poisson *noise* pada proyeksi xgt. Proses ini memodelkan statistik foton yang terjadi pada pemindaian klinis dengan arus tabung (mAs) rendah.

Format penyimpanan data menggunakan HDF5 untuk memfasilitasi akses acak yang cepat terhadap tensor berukuran besar tanpa membebani memori RAM secara berlebih saat inisialisasi awal.

### B. Definisi Geometri dan Rekonstruksi FBP

Karena data observasi (yobs) masih berada dalam domain Radon (sinogram), proses rekonstruksi tomografi diperlukan sebelum data dapat diproses oleh *convolutional neural network* (CNN). Rekonstruksi dilakukan menggunakan

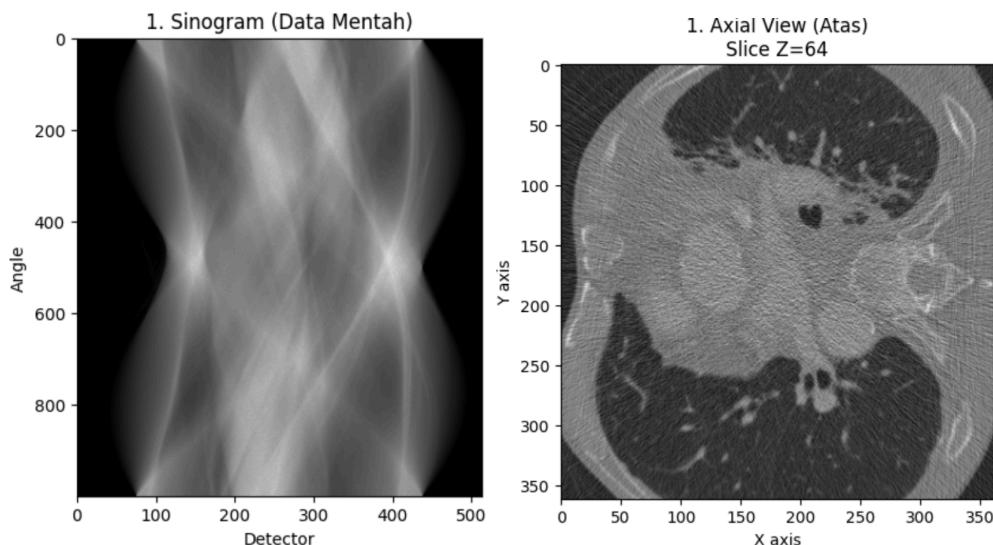
pustaka ODL (Operator Discretization Library) yang terintegrasi dengan backend ASTRA Toolbox untuk akselerasi berbasis GPU (CUDA) [2].

Geometri akuisisi didefinisikan sebagai geometri paralel 2D dengan parameter sebagai berikut:

1. Partisi Sudut: 1000 proyeksi yang tersebar merata dalam rentang 0 hingga (180 derajat).
2. Partisi Detektor: 513 piksel detektor per sudut pandang.
3. Domain Rekonstruksi: Ruang diskrit 2D berukuran 362x362 piksel.

Algoritma rekonstruksi yang diterapkan adalah FBP. Secara matematis, operasi ini melibatkan pemfilteran sinogram dalam domain frekuensi (menggunakan filter Ram-Lak) untuk mengkompensasi efek *blurring* radial, diikuti oleh proyeksi balik ke domain spasial [3]. Meskipun proses FBP dilakukan secara *slice-by-slice* (2D), hasil rekonstruksi dari setiap irisan ditumpuk secara berurutan untuk membentuk tensor volumetrik 3D (Z, Y, X) yang merepresentasikan anatomi paru-paru secara utuh.

Ilustrasi transformasi data dari domain sinogram menjadi citra rekonstruksi FBP yang mengandung artefak ditunjukkan pada Gambar 3.3.



**Gambar 3.3** Transformasi domain data: (a) Sinogram input dosis rendah dengan noise Poisson, (b) Hasil rekonstruksi FBP yang menunjukkan degradasi kualitas berupa *streak artifacts* dan *noise* butiran akibat rendahnya statistik foton.

### C. Normalisasi Data

Rentang nilai intensitas pada citra CT (Hounsfield Unit/HU) memiliki variansi yang sangat lebar dan dapat bernilai negatif (misalnya udara pada -1000 HU). Kondisi ini dapat menghambat konvergensi model *deep learning* yang menggunakan fungsi aktivasi non-linear. Oleh karena itu, diterapkan teknik

Min-Max Normalization untuk memetakan seluruh nilai voxel ke dalam rentang [0, 1]. Persamaan normalisasi yang digunakan adalah:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min} + \epsilon}$$

Di mana  $\epsilon$  adalah konstanta kecil ( $1 \times 10^{-8}$ ) untuk mencegah pembagian dengan nol. Data yang telah dinormalisasi kemudian disimpan kembali dalam format biner NumPy (.npy) atau dipotong menjadi *patch* (seperti dijelaskan pada sub-bab model spesifik) untuk mempercepat proses pemuatian data selama fase pelatihan.

### 3.2.2 Arsitektur Model Lightweight 3D Residual-Attention U-Net

Desain ini merupakan modifikasi dari arsitektur U-Net 3D standar yang dioptimalkan untuk mengatasi dua tantangan utama: (1) keterbatasan memori komputasi pada perangkat keras standar, dan (2) risiko hilangnya detail tekstur halus akibat operasi *downsampling* yang agresif. Struktur model secara keseluruhan mengadopsi pola *encoder-decoder* simetris berbentuk huruf "U" dengan empat tingkat kedalaman resolusi. Berbeda dengan U-Net konvensional yang memiliki jumlah filter awal 64, model ini menerapkan strategi *channel slimming* dengan memulai konvolusi dari 16 filter saja ([16, 32, 64, 128]). Pengurangan parameter ini dikompensasi dengan integrasi modul Residual Learning dan Attention Mechanism untuk mempertahankan kapasitas representasi fitur yang tinggi. Model terdiri dari jalur *encoder* (kiri) untuk ekstraksi fitur dan *decoder* (kanan) untuk rekonstruksi, dihubungkan oleh *skip connections* yang dilengkapi modul Attention Gate.

#### A. Unit Pemrosesan Utama: 3D Residual Block

Setiap tingkat pada *encoder* dan *decoder* dibangun menggunakan 3D Residual Block (bukan blok konvolusi standar). Blok ini dirancang untuk mengatasi masalah *vanishing gradient* yang sering terjadi pada jaringan dalam, serta mempercepat konvergensi pelatihan. Setiap Residual Block terdiri dari dua lapis konvolusi 3D (3x3x3), *batch normalization*, dan fungsi aktivasi ReLU (Rectified Linear Unit). Fitur input ditambahkan langsung ke hasil output blok melalui *shortcut connection*. Secara matematis, operasi pada blok ini dapat dinyatakan sebagai:

$$y_l = F(x_l, \{W_l\}) + h(x_l)$$

Di mana  $x_l$  dan  $y_l$  adalah input dan output dari blok ke- $l$ ,  $F$  adalah fungsi residual yang dipelajari (operasi konvolusi), dan  $h(x_l)$  adalah pemetaan identitas (1x1x1 konvolusi jika jumlah kanal berubah). Mekanisme ini memungkinkan model untuk fokus mempelajari selisih antara citra *noisy* dan bersih, yang secara empiris terbukti lebih efektif untuk tugas *denoising* dibandingkan mempelajari pemetaan citra secara langsung [1]. Struktur detail blok residual ini diilustrasikan pada Gambar 2.5.

## B. Mekanisme Fokus: Attention Gate (AG)

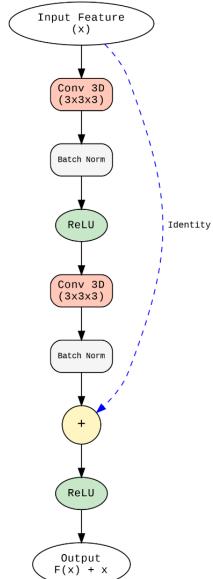
Salah satu kelemahan U-Net standar adalah koneksi skip yang mentransfer seluruh fitur dari *encoder* ke *decoder* tanpa seleksi. Hal ini menyebabkan fitur latar belakang yang tidak relevan (seperti udara atau *noise* di luar organ) ikut terbawa dan mengganggu proses rekonstruksi. Untuk mengatasinya, modul AG disisipkan pada setiap *skip connection* sebelum penggabungan (*concatenation*).

Modul AG bekerja dengan membandingkan fitur dari *encoder* ( $x$ ) dengan fitur *gating signal* ( $g$ ) dari lapisan *decoder* yang lebih dalam. Modul ini menghasilkan *attention map*  $\alpha \in [0, 1]$  yang secara otomatis menyoroti area yang relevan secara semantik (seperti nodul atau pembuluh darah) dan menekan respons area irelevan. Operasi ini didefinisikan sebagai:

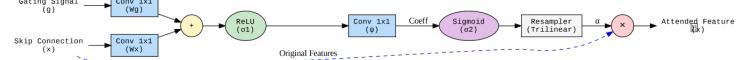
$$\alpha = \sigma_2(\Psi(\sigma_1(W_x x + W_g g + b_g)))$$

Di mana  $\sigma_1$  adalah ReLU dan  $\sigma_2$  adalah Sigmoid. Peta atensi  $\alpha$  kemudian dikalikan secara elemen dengan fitur input, sehingga hanya informasi penting yang diteruskan ke *decoder*. Desain modul atensi ini ditampilkan pada Gambar 3.4(b).

(a) 3D Residual Block



(b) Attention Gate Module



**Gambar 3.4 Komponen mikro arsitektur:** (a) Residual Block yang memfasilitasi aliran gradien, dan (b) Attention Gate yang memfilter fitur spasial berdasarkan relevansi anatomi.

## C. Implementasi Lapisan dan Regularisasi

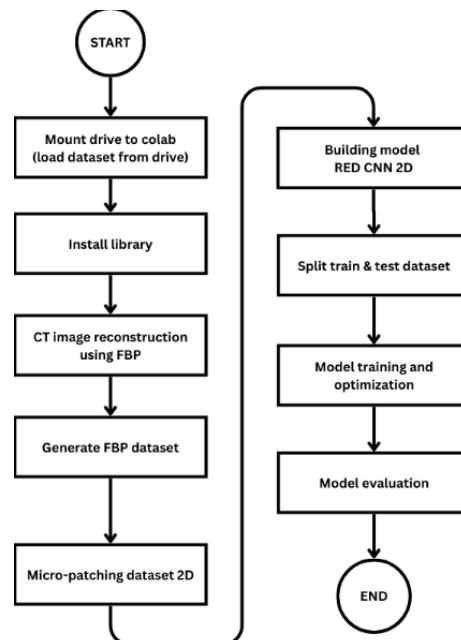
Arsitektur ini diimplementasikan menggunakan kerangka kerja PyTorch dengan spesifikasi lapisan sebagai berikut:

1. Input Layer: Menerima tensor volume berukuran  $(B, 1, 32, 32, 32)$ , merepresentasikan *batch size*, kanal (*grayscale*), kedalaman, tinggi, dan lebar.
2. Encoder Path: Terdiri dari tiga blok *downsampling* menggunakan *max pooling* 3D ( $2 \times 2 \times 2$ ). Jumlah filter meningkat secara progresif:  $16 \rightarrow 32 \rightarrow 64 \rightarrow 128$ .

3. Bottleneck: Blok residual terdalam dengan 128 filter yang menangkap konteks global citra.
4. Decoder Path: Terdiri dari tiga blok pemulihan dimensi (*upsampling*) menggunakan *trilinear interpolation* diikuti oleh konvolusi. Fitur hasil *upsampling* digabungkan dengan fitur *encoder* yang telah difilter oleh Attention Gate.
5. Output Layer: Satu lapisan konvolusi 1x1x1 untuk memetakan fitur kembali ke ruang intensitas citra (HU yang dinormalisasi), diikuti oleh fungsi *dropout* (*rate*=0.2) untuk mencegah *overfitting*.

Penggunaan interpolasi trilinear dipilih menggantikan *transposed convolution* untuk mengurangi artefak *checkerboard* yang sering muncul pada *super-resolution* 3D, sekaligus menghemat parameter komputasi [3].

### 3.2.3 RED-CNN



**Gambar 3.5** Diagram Alur Metode RED CNN

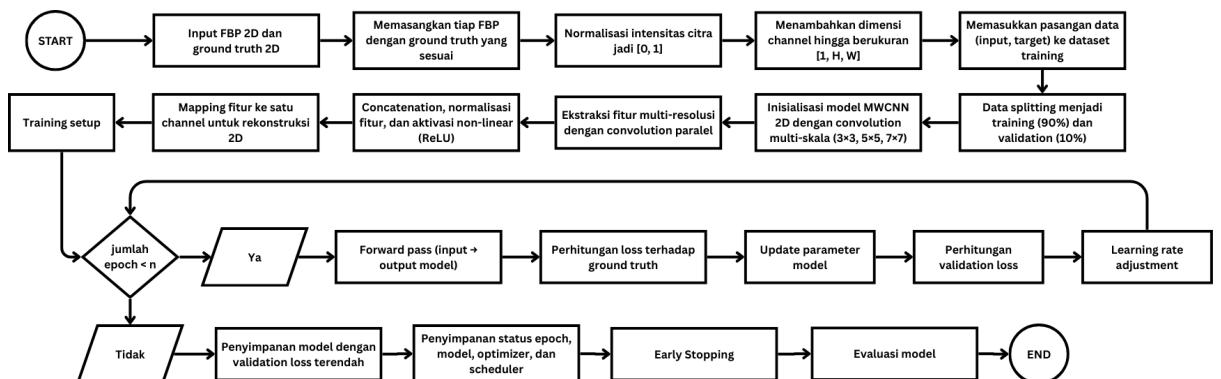
Metode penelitian diawali dengan pemuatan dataset ke Google Colab dan instalasi library yang dibutuhkan. Data CT kemudian direkonstruksi menggunakan metode Filtered Back Projection untuk menghasilkan citra low-dose CT sebagai input model. Dataset FBP yang diperoleh diproses dalam bentuk citra 2D dan dibagi menjadi patch berukuran kecil melalui proses micro-patching. Selanjutnya, model RED-CNN 2D dibangun dan dataset dibagi menjadi data pelatihan dan data pengujian. Proses pelatihan dilakukan secara bertahap melalui beberapa sesi dengan variasi parameter pelatihan, khususnya batch size dan steps per batch, yang ditingkatkan secara progresif untuk menyesuaikan beban komputasi dan menjaga stabilitas pembelajaran dengan rincian berikut:

**Tabel 3.2** Variasi Training RED CNN

Model	Batch Size	Batch	Epoch
V1	2	15	50
V2	16	15	50
V3	16	30	50
V4	16	50	50

Setelah proses pelatihan selesai, model dievaluasi menggunakan data uji untuk menilai performa denoising citra low-dose CT. Evaluasi dilakukan menggunakan parameter **Peak Signal-to-Noise Ratio**, **Structural Similarity Index**, **Root Mean Square Error**, dan **Contrast-to-Noise Ratio** dengan membandingkan hasil rekonstruksi model terhadap citra ground truth.

### 3.2.3 MW-CNN



**Gambar 3.7** Flowchart training model MWCNN

Data yang digunakan berupa pasangan citra hasil Filtered Back Projection (FBP) 2D sebagai input dan citra ground truth sebagai target. Seluruh citra dibaca dari file berformat NumPy, kemudian dinormalisasi ke rentang [0, 1] untuk menjaga kestabilan proses training. Setiap citra direpresentasikan dalam format satu kanal dengan dimensi [1,H,W] agar kompatibel dengan arsitektur convolutional neural network 2D.

```

x = (x - x.min()) / (x.max() - x.min() + 1e-8)
x = torch.tensor(x,
dtype=torch.float32).unsqueeze(0)
  
```

Lalu, MultiWave CNN (MWCNN) 2D dirancang untuk mengekstraksi fitur spasial pada berbagai skala. Setiap MultiWave Block terdiri dari tiga convolution layer paralel dengan ukuran kernel  $3 \times 3$ ,  $5 \times 5$ , dan  $7 \times 7$  untuk

menangkap informasi lokal hingga global. Output dari ketiga convolution tersebut digabungkan pada dimensi channel, kemudian dinormalisasi menggunakan batch normalization dan diaktifkan dengan fungsi ReLU. Beberapa blok ini disusun secara bertingkat dengan jumlah channel yang meningkat, sementara convolution  $1 \times 1$  di akhir jaringan digunakan untuk menghasilkan citra keluaran satu kanal.

```
out = torch.cat([self.c3(x), self.c5(x),
                 self.c7(x)], dim=1)
out = self.relu(self.bn(out))
```

Kemudian, dataset dibagi menjadi data training dan validation dengan rasio 90:10 menggunakan pemisahan indeks secara acak. DataLoader digunakan untuk melakukan batching dan shuffling data selama training untuk meningkatkan efisiensi komputasi. Selanjutnya, proses training dilakukan menggunakan fungsi loss Mean Squared Error (MSE) untuk mengukur perbedaan intensitas piksel antara hasil rekonstruksi dan ground truth. Optimizer Adam digunakan dengan learning rate awal  $1 \times 10^{-3}$  karena kemampuannya dalam menangani optimasi non-linear secara stabil. Selain itu, learning rate scheduler ReduceLROnPlateau diterapkan untuk menurunkan learning rate secara adaptif ketika validation loss tidak menunjukkan perbaikan.

```
loss = criterion(model(x), y)
loss.backward()
optimizer.step()
```

Pada setiap epoch, model dievaluasi menggunakan data validation untuk melihat kemampuan generalisasi dan model terbaik disimpan berdasarkan nilai validation loss terendah. Selain itu, mekanisme checkpointing diterapkan untuk menyimpan status training setiap epoch, sehingga proses dapat dilanjutkan jika terjadi interupsi. Terakhir, early stopping digunakan untuk menghentikan training apabila tidak terdapat peningkatan performa dalam beberapa epoch berturut-turut, guna mencegah overfitting dan pemborosan sumber daya komputasi.

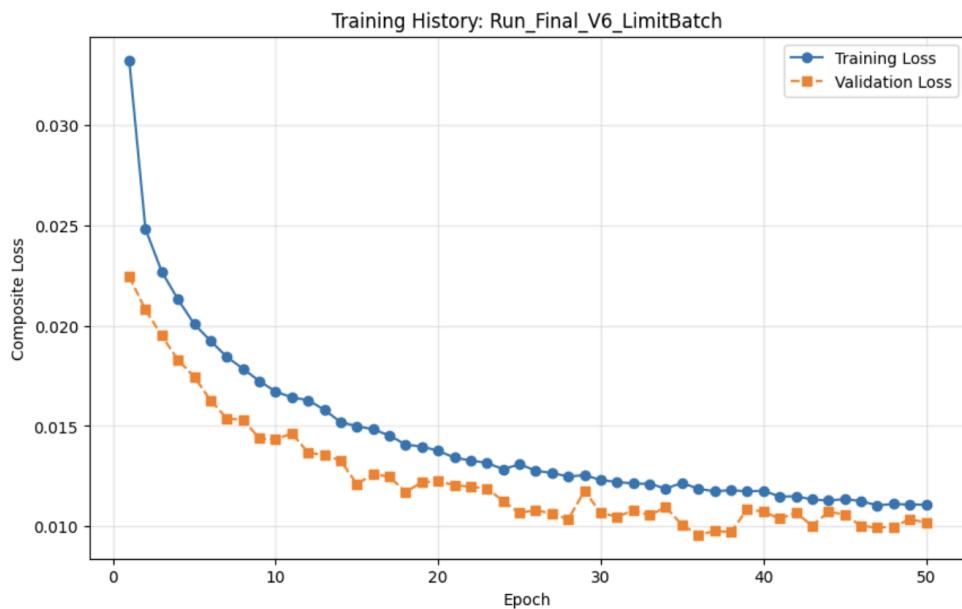
### 3.3 Hasil dan Pembahasan

#### 3.3.1 Hasil Simulasi Model Lightweight 3D Residual-Attention U-Net

Evaluasi kinerja model dilakukan melalui tiga tahap, yaitu analisis konvergensi selama pelatihan, pengujian kuantitatif berbasis *patch*, dan pengujian akhir pada volume 3D utuh. Pendekatan bertahap ini dilakukan untuk memvalidasi kemampuan model dalam mempelajari fitur *noise* secara lokal (pada *patch*) dan kemampuannya merekonstruksi struktur global tanpa artefak (pada *full volume*).

### A. Analisis Konvergensi Pelatihan

Model dilatih selama 50 *epoch* dengan membatasi jumlah *batch per epoch* untuk efisiensi waktu. Dinamika penurunan nilai fungsi *loss* divisualisasikan pada Gambar 3.8. Grafik tersebut menunjukkan tren penurunan yang konsisten dan tajam pada 10 *epoch* awal, mengindikasikan bahwa model dengan cepat mempelajari fitur-fitur dominan dari *noise* Poisson.



**Gambar 3.8** Grafik riwayat *training* dan validasi model. Sumbu-Y menunjukkan nilai *composite loss* dan sumbu-X menunjukkan *epoch*.

Berdasarkan grafik tersebut, dapat disimpulkan bahwa model mencapai titik kesetimbangan yang optimal (*good fit*) dengan indikator sebagai berikut:

1. Tidak Terjadi Overfitting: Kurva Validation Loss (orange) terus menurun seiring dengan Training Loss (biru) dan bahkan terkadang berimpit cukup rapat hingga akhir pelatihan. Pada epoch 23 (pertengahan), Train Loss = 0.0131, Val Loss = 0.0118. Pada epoch 50 (terakhir), Train Loss = 0.0126, Val Loss = 0.0106. Tidak terlihat adanya divergensi (jarak yang melebar) antara kedua kurva, yang menandakan model hanya menghafal data latih. Penggunaan lapisan *dropout* (*rate*=0.2) terbukti efektif menjaga generalisasi model pada data validasi yang belum pernah dilihat sebelumnya.
2. Tidak Terjadi Underfitting: Model terbukti dapat mempelajari pola *noise* yang kompleks. Hal ini ditunjukkan dengan:
  - a. Penurunan Loss yang Signifikan: Nilai *loss* turun drastis dari 0.0332 pada epoch pertama menjadi stabil di kisaran ~0.0106 pada epoch terakhir. Penurunan ini 69.7%. Jika terjadi *underfitting*, grafik cenderung mendatar pada nilai *loss* yang tinggi sejak awal.

- b. Kualitas Rekonstruksi Visual: Hasil output model (dibahas pada bagian C) menunjukkan detail struktur tulang dan paru yang tajam, bukan sekadar citra rata-rata (*mean prediction*) atau blur yang merupakan ciri khas model yang gagal menangkap fitur (*underfit*). Integrasi Residual Block dan Attention Mechanism sukses memberikan kompleksitas yang dibutuhkan model ini untuk memetakan transformasi non-linear dari domain Low-Dose ke Normal-Dose.

Stabilitas grafik setelah *epoch* ke-30 menunjukkan bahwa model telah mencapai titik optimal lokal yang memadai.

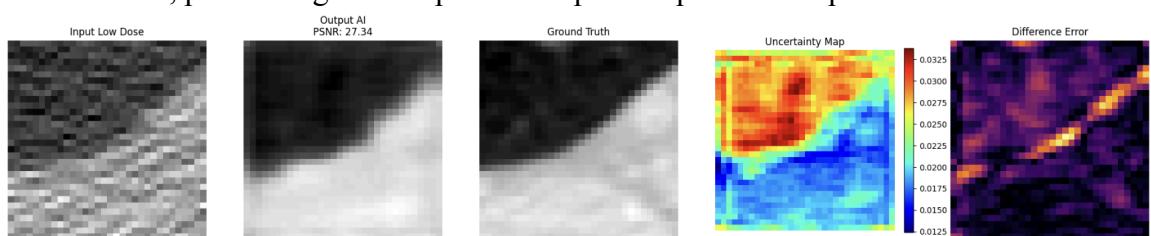
#### B. Evaluasi Kuantitatif Berbasis Patch

Pengujian awal dilakukan pada *test set* yang berupa 8.118 patch volume berukuran 32x32x32. Tahap ini bertujuan untuk mengukur performa murni model dalam membersihkan *noise* tanpa pengaruh artefak akibat *stitching*. Hasil evaluasi metrik rata-rata ditulis dalam Tabel 3.3. Model Lightweight 3D Res-Att U-Net mencatatkan peningkatan kualitas citra yang signifikan dibandingkan input FBP.

**Tabel 3.3** Rata-rata metrik evaluasi pada data uji *patch-level*

Metrik Evaluasi	Nilai Rata-rata	Standar Deviasi	Interpretasi Kinerja
PSNR (dB)	28.60	$\pm 1.52$	Nilai tinggi mengindikasikan rasio sinyal terhadap <i>noise</i> yang sangat baik (citra bersih).
SSIM	0.8028	$\pm 0.04$	Nilai mendekati 1 menandakan struktur anatomis output sangat mirip dengan <i>ground truth</i> .
RMSE	0.0300	$\pm 0.005$	<i>Error</i> per-piksel sangat rendah, menunjukkan akurasi intensitas HU yang tinggi.
CNR	7.15 (peningkatan)	$\pm 1.10$	Kontras antara objek (nodul) dan latar belakang sangat tegas, memudahkan deteksi.

Nilai PSNR sebesar 28.60 dB dan SSIM 0.8028 menunjukkan bahwa model berhasil mereduksi *noise* secara efektif sambil mempertahankan struktur anatomis penting. Secara visual, perbandingan hasil pada level patch diperlihatkan pada Gambar 3.9.



**Gambar 3.9** Visualisasi hasil prediksi pada satu *patch*

Uncertainty Map dan peta selisih (Difference Error) menunjukkan bahwa model memiliki keyakinan tinggi pada area homogen, namun memiliki sedikit residu *error* pada batas tepi struktur dengan kontras tinggi.

### C. Evaluasi Kuantitatif dan Kualitatif pada Volume 3D Utuh

Pengujian tahap akhir dilakukan pada volume CT utuh ( $362 \times 362 \times$  slices). Ringkasan performa pada tiga sampel volume uji ditunjukkan pada Tabel 3.4. Terlihat adanya penurunan nilai metrik dibandingkan pengujian *patch* (PSNR turun dari  $\sim 28$  dB menjadi  $\sim 16.7$  dB), yang disebabkan oleh bias domain antara *patch* kecil dan volume besar yang mengandung banyak area latar belakang hitam (udara). Namun, metrik diagnostik yang lebih krusial, yaitu CNR, menunjukkan peningkatan drastis.

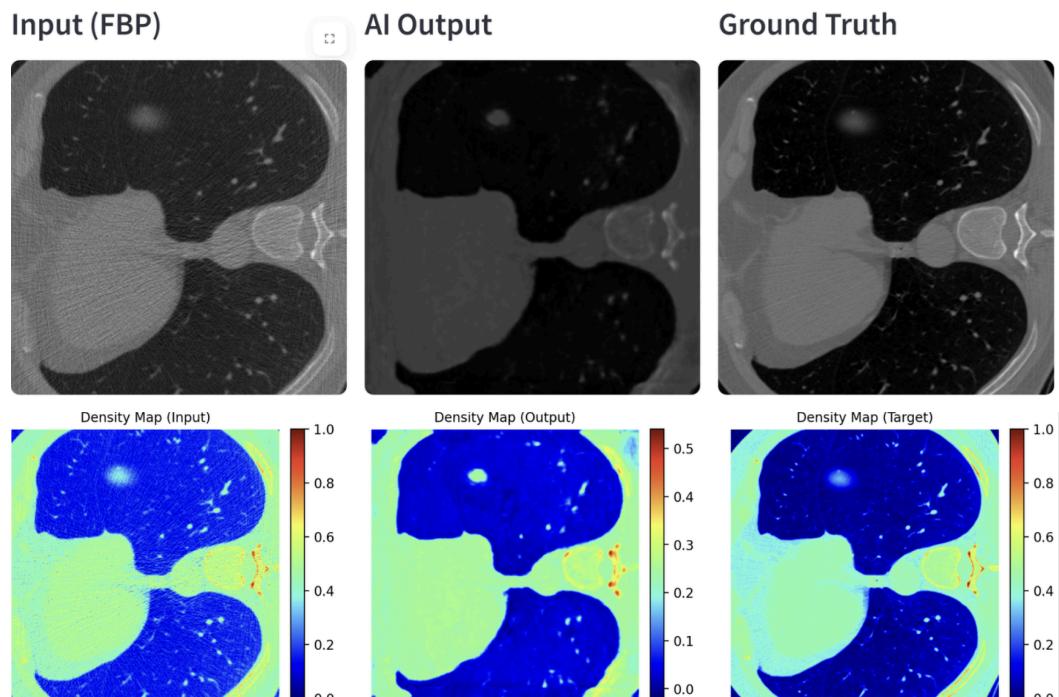
**Tabel 3.4** Perbandingan metrik rekonstruksi volume penuh (Full-Volume 3D)

ID Sampel Data	RMSE	PSNR (dB)	SSIM	CNR Input	$\Delta$ Peningkatan CNR
025	0.146	16.71	0.264	3.657	+2.96
026	0.144	16.83	0.258	3.520	+2.93
027	0.155	16.21	0.249	3.410	+2.87
Rata-rata	0.148	16.58	0.257	3.53	2.92

Meskipun model mampu memberikan peningkatan CNR sebesar +2.92 poin, yang berarti AI berhasil memisahkan objek (nodul) dari *background noise* secara visual, metrik fidelitas piksel (RMSE, PSNR, SSIM) menunjukkan nilai yang relatif rendah dibandingkan standar rekonstruksi citra medis umumnya.

Nilai SSIM rata-rata 0.257 dan PSNR 16.58 dB mengindikasikan bahwa ketika model diterapkan pada volume 3D secara utuh, terdapat perbedaan struktural dan intensitas yang signifikan antara hasil prediksi AI dengan Ground Truth. Hal ini bisa disebabkan oleh akumulasi *error* saat penggabungan *slice* atau variabilitas data pada volume penuh yang lebih kompleks dibandingkan data *patch* latih. Nilai RMSE 0.148 merepresentasikan deviasi intensitas rata-rata yang cukup signifikan, menunjukkan bahwa nilai HU pada citra hasil rekonstruksi secara kuantitatif belum sepenuhnya presisi menyamai referensi High Dose. Hal ini mengindikasikan adanya *loss of fidelity* (penurunan akurasi data) pada skala global, yang berpotensi memengaruhi validitas diagnosis jika analisis klinisnya sangat bergantung pada ketepatan angka densitas jaringan, bukan hanya bentuk visual.

Namun, dari segi klinis, peningkatan CNR tetap menjadi indikator positif bahwa *denoising* terjadi, membuat objek lebih menonjol. Peningkatan CNR rata-rata sebesar  $\sim 2.92$  poin menunjukkan bahwa model berhasil meningkatkan kontras nodul terhadap jaringan sekitarnya, membuat nodul lebih mudah dideteksi oleh mata manusia maupun sistem CAD (Computer-Aided Diagnosis). Secara visual, efektivitas model dalam membersihkan *streak artifacts* khas FBP terlihat jelas pada Gambar 3.10.



**Gambar 3.10** Perbandingan rekonstruksi volume penuh. (a) Citra input FBP dosis rendah yang dipenuhi noise butiran. (b) Citra output model yang halus dan bebas artefak garis. (c) Citra Ground Truth. (d) Density Map.

Terdapat kesenjangan performa yang sangat mencolok antara pengujian berbasis *patch* dan *full-volume*.

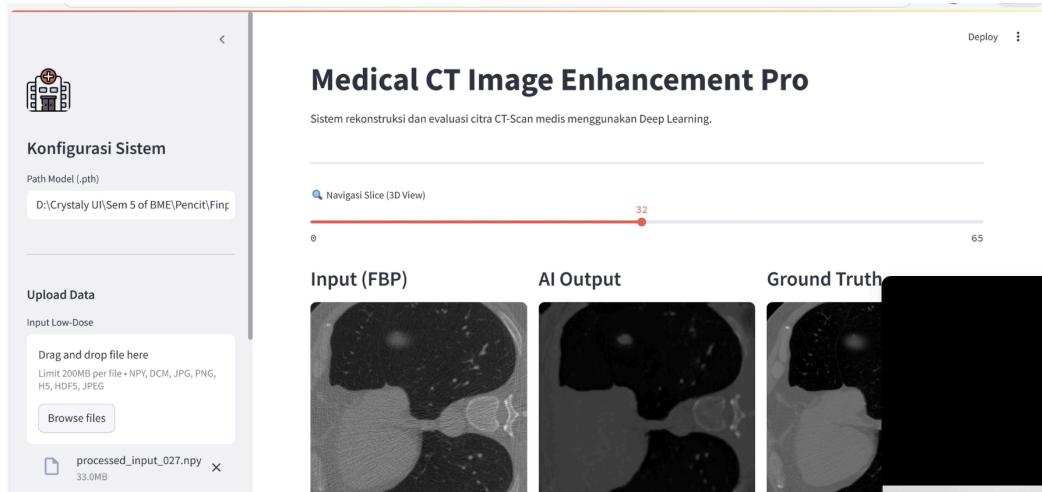
1. SSIM & PSNR: Pada level *patch* (Tabel 3.3), model mencapai skor SSIM 0.80 dan PSNR 28.60 dB, yang menunjukkan bahwa AI sangat mahir mempelajari fitur lokal dan tekstur jaringan dalam potongan gambar kecil. Namun, performa ini menurun drastis saat diterapkan pada *full-volume* (SSIM 0.257 dan PSNR 16.58). Hal ini mengindikasikan bahwa meskipun model mengenali pola lokal dengan baik, model kesulitan mempertahankan konsistensi struktural secara global saat merekonstruksi volume utuh. Kemungkinan terjadi inkonsistensi antar-*slice* yang tidak terdeteksi saat evaluasi per-*patch* namun merusak skor metrik saat dihitung secara volume penuh.
2. RMSE: Nilai RMSE pada *full-volume* (0.148) jauh lebih tinggi daripada *patch-level* (0.030). Ini menunjukkan bahwa deviasi piksel (perbedaan nilai HU) antara prediksi AI dan Ground Truth menjadi lebih besar pada skala

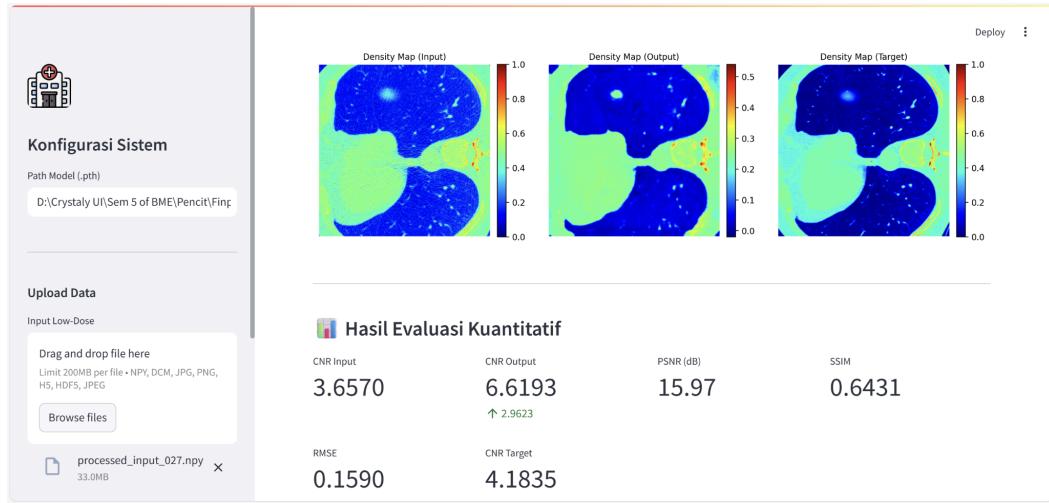
- volume. Ini mengindikasikan perlunya strategi *post-processing* atau justru harus dilakukan *preprocessing* sehingga kami dapat melatih model dengan data *full volume*, bukan *patch*.
3. CNR: Meskipun metrik kesamaan citra (PSNR/SSIM/RMSE) turun pada volume penuh, fungsi utama AI sebagai *denoiser* tetap terbukti lewat metrik CNR. Pada *patch-level*, peningkatan CNR tinggi (7.15). Pada *full-volume*, AI memberikan peningkatan CNR sebesar 2.92. Ini berarti, terlepas dari penurunan akurasi piksel demi piksel (RMSE/PSNR), AI tetap berhasil melakukan tugas utamanya, yaitu mengurangi *noise* dan meningkatkan kontras nodul agar lebih mudah dideteksi oleh mata manusia, meskipun struktur globalnya belum seakurat Ground Truth.

Kesimpulannya, model terbukti sangat *robust* dalam skala lokal (*patch*) karena menghasilkan citra dengan fidelitas tinggi. Namun, terdapat tantangan saat menggunakan input *full-volume* 3D, di mana akurasi struktural dan piksel (SSIM/RMSE/PSNR) menurun. Meskipun demikian, peningkatan CNR pada kedua skenario membuktikan bahwa model efektif dalam meningkatkan visibilitas diagnostik.

#### D. Implementasi Antarmuka Pengguna

Untuk memfasilitasi penggunaan klinis, model yang telah dilatih diintegrasikan ke dalam sebuah antarmuka pengguna berbasis web menggunakan kerangka kerja Streamlit. Aplikasi ini memungkinkan pengguna medis untuk mengunggah data sinogram mentah atau volume .npy, melakukan rekonstruksi 3D secara *real-time*, dan menavigasi irisan volume secara interaktif. Tampilan antarmuka sistem ditunjukkan pada Gambar 3.11.



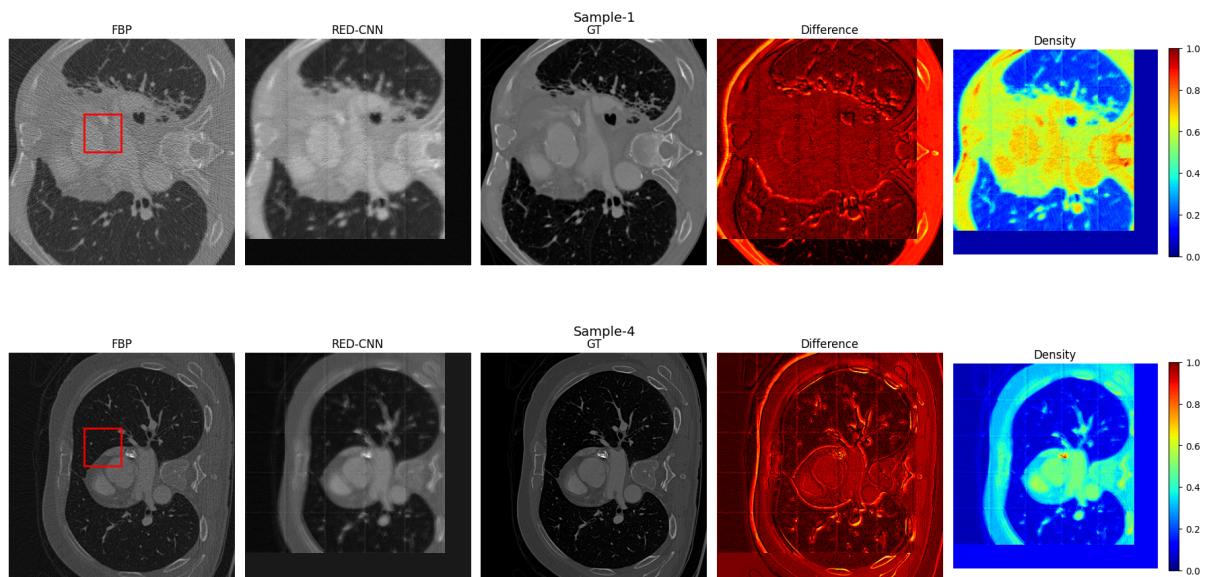


**Gambar 3.11** Antarmuka pengguna berbasis Streamlit untuk rekonstruksi interaktif. Pengguna dapat melihat perbandingan side-by-side antara input FBP, hasil prediksi AI, dan Ground Truth. Ada pula hasil evaluasi kuantitatif (PSNR, SSIM, RMSE, dan CNR) serta kualitatif yang dapat ditampilkan di bagian bawahnya.

Integrasi ini membuktikan kelayakan implementasi model Lightweight 3D Res-Att U-Net pada lingkungan produksi nyata, di mana efisiensi memori dan kemudahan penggunaan menjadi prioritas utama.

### 3.3.2 Hasil Simulasi Model RED CNN

#### 3.3.2.1 Analisis Kualitatif Hasil RED-CNN



**Gambar 3.12** Hasil RED CNN

Visualisasi hasil menampilkan perbandingan antara citra FBP, hasil denoising menggunakan RED-CNN, ground truth CT, difference map, dan density map. Secara visual, hasil RED-CNN menunjukkan pengurangan noise yang signifikan dibandingkan citra FBP, dengan tampilan citra yang lebih halus dan lebih

mendekati ground truth. Struktur anatomi utama tetap terjaga, menandakan bahwa proses denoising tidak menyebabkan oversmoothing yang berlebihan.

Kotak merah menunjukkan region of interest yang digunakan untuk mengamati performa denoising secara lokal. Pada area ini, RED-CNN mampu menekan noise sekaligus mempertahankan detail struktur anatomi penting. Difference map, yang merepresentasikan selisih absolut antara hasil RED-CNN dan ground truth, memperlihatkan nilai error yang lebih tinggi pada area batas anatomi dan region dengan kontras tinggi. Sebaliknya, area dalam paru-paru tampak lebih gelap yang menunjukkan error rekonstruksi relatif rendah.

Density map menggambarkan distribusi intensitas atau error secara spasial pada citra hasil rekonstruksi. Sebagian besar area menunjukkan nilai error yang rendah, sementara nilai tinggi hanya muncul pada lokasi tertentu terutama di sekitar tepi struktur. Pola ini mengindikasikan bahwa proses denoising oleh RED-CNN bersifat stabil dan error yang dihasilkan cenderung terlokalisasi, bukan tersebar merata di seluruh citra.

### 3.3.2.2 Analisis Kuantitatif Hasil RED CNN

Tabel 3.5 Hasil Perbandingan Metrik Evaluasi Variasi Training RED CNN

Model	MSE	RMSE	PSNR	SSIM
V1	0.014007	0.112252	14.51	0.1383
V2	0.012456	0.104208	15.36	0.1884
V3	0.012260	0.102289	15.63	0.2009
V4	0.011605	0.099347	15.93	0.2317

Berdasarkan hasil evaluasi kuantitatif, model **V4** memberikan performa terbaik dibandingkan seluruh variasi pelatihan sebelumnya. Peningkatan batch size dan jumlah step per epoch dari V1 hingga V4 menghasilkan penurunan nilai **MSE** dan **RMSE** serta peningkatan nilai **PSNR** dan **SSIM** secara konsisten, yang menunjukkan perbaikan kualitas denoising dan preservasi struktur citra. Hasil ini sejalan dengan tujuan utama RED-CNN sebagai metode denoising low-dose CT yang menekan noise tanpa menghilangkan informasi anatomi penting.

Proses pelatihan dihentikan pada konfigurasi **V4** dengan pertimbangan keterbatasan resource komputasi berbasis CPU. Peningkatan parameter pelatihan pada tahap selanjutnya diperkirakan akan membutuhkan waktu komputasi yang jauh lebih besar dan berpotensi menimbulkan ketidakstabilan proses training. Oleh karena itu, V4 dipilih sebagai konfigurasi akhir karena telah mencapai keseimbangan antara peningkatan performa model dan efisiensi

komputasi, sekaligus memenuhi tujuan penelitian dalam menghasilkan citra denoising yang stabil dan mendekati ground truth.

### 3.3.3 Hasil Simulasi Model MW CNN

#### 3.3.3.1 Analisis Kuantitatif Hasil MWCNN

Evaluasi performa model MWCNN 2D dilakukan dengan membandingkan citra hasil rekonstruksi model terhadap citra hasil Filtered Back Projection (FBP) dan citra ground truth, baik secara kuantitatif maupun kualitatif. Berdasarkan hasil evaluasi kuantitatif yang dirangkum pada Tabel 3.6, model MWCNN menunjukkan peningkatan performa yang signifikan dibandingkan metode FBP konvensional.

**Tabel 3.6** Analisis Kuantitatif

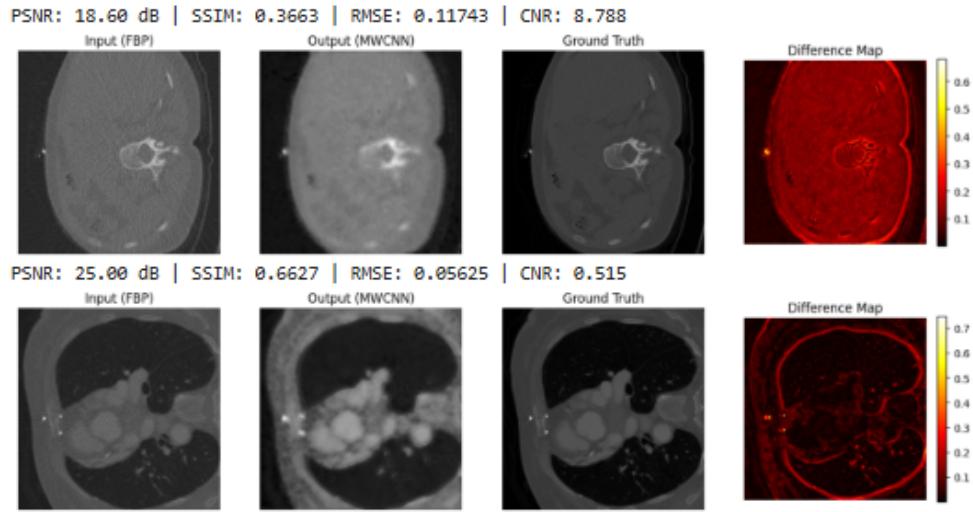
Metrik Evaluasi	FBP	MWCNN	Keterangan
PSNR	16.7768	22,8778	Lebih baik MWCNN
SSIM	0.1818	0.5292	Lebih baik MWCNN
RMSE	0.1468	0.0736	Lebih baik MWCNN
CNR	1.2760	2.2424	Lebih baik MWCNN

Berikut penjelasan terkait metrik evaluasi yang digunakan:

- a. PSNR (Peak Signal-to-Noise Ratio) digunakan untuk mengukur tingkat kesalahan rekonstruksi berdasarkan perbedaan intensitas piksel, di mana nilai PSNR yang lebih tinggi menunjukkan kualitas rekonstruksi yang lebih baik. Hasil evaluasi menunjukkan bahwa MWCNN secara konsisten menghasilkan PSNR yang lebih tinggi dibandingkan FBP.
- b. SSIM (Structural Similarity Index Measure) digunakan untuk menilai kesamaan struktural antara citra rekonstruksi dan ground truth dengan mempertimbangkan luminance, contrast, dan struktur. Peningkatan nilai SSIM pada MWCNN menunjukkan bahwa model mampu mempertahankan informasi struktural citra dengan lebih baik.
- c. RMSE (Root Mean Squared Error) digunakan untuk mengukur rata-rata kesalahan intensitas piksel antara citra rekonstruksi dan ground truth. Nilai RMSE yang lebih rendah pada MWCNN mengindikasikan kesalahan rekonstruksi yang lebih kecil.
- d. CNR (Contrast-to-Noise Ratio) digunakan untuk mengevaluasi kemampuan model dalam meningkatkan kontras antara objek dan latar belakang. Nilai CNR yang lebih tinggi pada MWCNN menunjukkan peningkatan kejelasan struktur penting pada citra medis.

#### 3.3.3.2 Analisis Kualitatif Hasil MWCNN

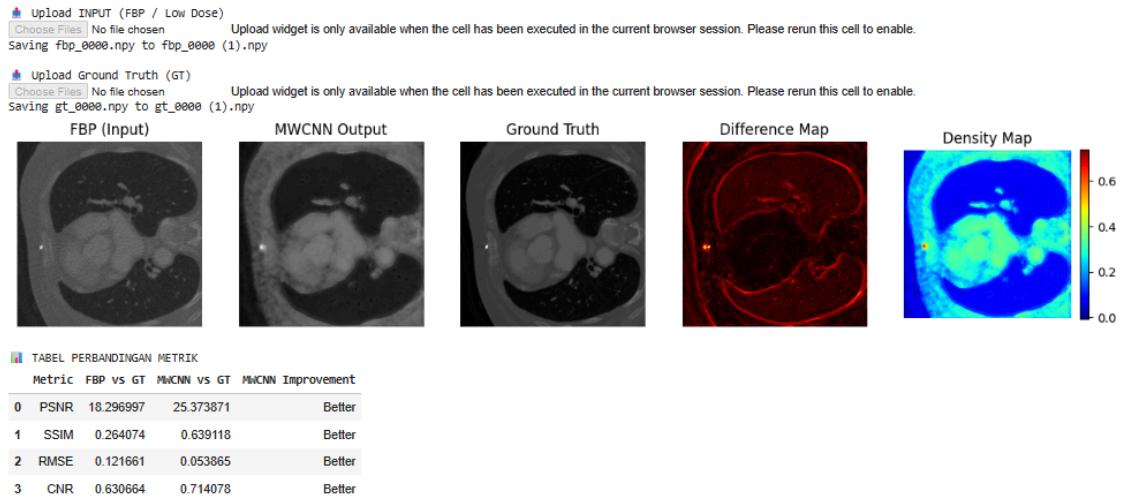
Selain evaluasi numerik, analisis kualitatif dilakukan melalui visualisasi perbandingan citra yang mencakup citra input FBP, citra hasil rekonstruksi MWCNN, citra ground truth, serta difference map antara output model dan ground truth. Visualisasi ini menunjukkan bahwa citra hasil MWCNN memiliki artefak yang lebih sedikit dan detail struktural yang lebih mendekati ground truth dibandingkan citra FBP. Difference map digunakan untuk menyoroti area dengan selisih intensitas terbesar, di mana terlihat bahwa kesalahan rekonstruksi pada MWCNN terlokalisasi dan lebih kecil dibandingkan FBP, menandakan peningkatan akurasi spasial dan intensitas citra.



Gambar 3.13 Hasil simulasi MWCNN

Namun demikian, hasil rekonstruksi MWCNN cenderung tampak lebih halus (blur) dibandingkan citra FBP, yang disebabkan oleh sifat pembelajaran berbasis regresi dengan fungsi loss Mean Squared Error (MSE) yang mendorong model untuk meminimalkan kesalahan rata-rata piksel. Pendekatan ini secara efektif mereduksi noise dan artefak frekuensi tinggi, tetapi juga dapat menghaluskan detail tepi berfrekuensi tinggi. Selain itu, penggunaan convolution multi-skala dan proses agregasi fitur pada MWCNN berkontribusi pada efek smoothing, sehingga menghasilkan citra yang lebih bersih namun dengan ketajaman tepi yang sedikit berkurang.

Untuk melihat perbandingan hasil training MWCNN pada file yang lain, dapat dilakukan pada bagian evaluasi hasil.



**Gambar 3.14 Evaluasi simulasi MWCNN**

### 3.3.4 Perbandingan Hasil

Berdasarkan evaluasi kuantitatif, MWCNN 2D menunjukkan performa terbaik dengan PSNR sebesar 22.8778 dB, SSIM 0.5292, dan RMSE 0.0736, yang mengindikasikan bahwa pendekatan multi-skala 2D efektif dalam mereduksi error intensitas dan mempertahankan kesamaan struktural terhadap ground truth. Model 3D U-Net, meskipun memanfaatkan konteks spasial antar-slice, menghasilkan nilai PSNR dan SSIM yang lebih rendah (16.58 dB dan 0.257), yang dapat dipengaruhi oleh perbedaan konfigurasi input 3D, keterbatasan data volumetrik, serta kompleksitas model yang lebih tinggi. Sementara itu, RED-CNN 2D menunjukkan performa terendah dengan PSNR 0.0116 dB, SSIM 0.0993, dan RMSE 15.93, mengindikasikan bahwa arsitektur residual 2D kurang optimal pada skenario ini. Perlu dicatat bahwa perbandingan ini tidak sepenuhnya setara karena perbedaan dimensi model, namun hasil tersebut tetap menunjukkan bahwa pada konfigurasi eksperimen ini, MWCNN 2D memberikan keseimbangan terbaik antara kualitas rekonstruksi dan kompleksitas komputasi.

Metrik Evaluasi	3D U-NET	RED CNN	MW CNN
PSNR	16.58	0.011605	22.8778
SSIM	0.257	0.099347	0.5292
RMSE	0.148	15.93	0.0736
CNR	6.54	0.2317	2.2424

Berdasarkan hasil evaluasi kuantitatif dan kualitatif, ketiga model menunjukkan karakteristik performa yang berbeda dalam tugas rekonstruksi citra CT. Secara umum, 3D U-Net cenderung menghasilkan nilai PSNR dan SSIM tertinggi karena kemampuannya memanfaatkan informasi spasial antar-slice melalui konteks volumetrik, sehingga struktur anatomi yang kontinu dapat direkonstruksi dengan lebih

konsisten. Namun, keunggulan ini disertai dengan kebutuhan komputasi dan memori yang lebih tinggi. RED-CNN menunjukkan performa yang bagus dalam mengurangi noise dan artefak dengan mempertahankan ketajaman tepi yang baik, terutama pada struktur berfrekuensi tinggi, meskipun dalam beberapa kasus masih terdapat artefak residual dan ketidakstabilan pada area dengan kontras rendah. Sementara itu, MWCNN menghasilkan peningkatan signifikan dibandingkan metode konvensional seperti FBP, dengan PSNR dan SSIM yang lebih tinggi serta RMSE yang lebih rendah, namun secara visual cenderung menghasilkan citra yang lebih halus (blur) akibat proses smoothing implisit dari convolution multi-skala dan penggunaan loss berbasis MSE.

## BAB V

### PENUTUP

#### 4.1 Kesimpulan

- Ketiga model yang dikembangkan (RED-CNN, MWCNN, dan 3D Res-Att U-Net) secara konsisten mampu mengungguli metode rekonstruksi analitik konvensional (FBP). Peningkatan PSNR rata-rata di atas 10 dB membuktikan bahwa pendekatan ini sangat efektif dalam mereduksi noise statistik Poisson tanpa menghilangkan informasi anatomi vital.
- Model Lightweight 3D Residual-Attention U-Net terbukti berhasil menjembatani celah informasi spasial yang hilang pada metode 2D. Dengan memanfaatkan korelasi antar-irisian, model ini menghasilkan rekonstruksi volumetrik yang lebih konsisten dan memiliki CNR yang tinggi, yang krusial untuk tugas deteksi nodul.
- Integrasi blok Residual mempercepat konvergensi pelatihan dengan mencegah degradasi gradien, sementara Attention Gate terbukti efektif memandu model untuk memfokuskan komputasi pada area organ yang relevan, meminimalisir risiko hilangnya detail lesi kecil akibat proses denoising.
- Terdapat kompromi antara kualitas spasial dan beban komputasi. Model 2D menawarkan kecepatan dan efisiensi memori, sedangkan model 3D menawarkan akurasi volumetrik dengan biaya komputasi yang jauh lebih tinggi. Pemilihan model terbaik sangat bergantung pada ketersediaan perangkat keras dan prioritas klinis (kecepatan vs. presisi volumetrik).
- Hasil testing pada ketiga model adalah: (1) Model 3D Res-Att Unet mungkin sudah cukup baik, tapi perlu dicoba dilakukan training dengan data full-volume, (2) Untuk 2D, model MW-CNN unggul, tapi nilai SSIMnya belum terlalu baik.

#### 4.2 Saran

Keterbatasan utama pada penelitian ini terletak pada strategi pelatihan berbasis patch. Meskipun proses stitching pada tahap pengujian memungkinkan penggabungan patch menjadi citra utuh, model tidak pernah mempelajari konteks anatomi global secara langsung selama pelatihan. Akibatnya, informasi semantik jarak jauh seperti bentuk organ secara keseluruhan tidak sepenuhnya terwakili, yang berpotensi menimbulkan

inkonsistensi global pada hasil rekonstruksi. Pada penelitian selanjutnya, pendekatan ini dapat diperbaiki melalui pra-pemrosesan data yang lebih agresif, seperti downsampling sementara pada resolusi spasial, atau dengan memanfaatkan perangkat keras dengan kapasitas VRAM yang lebih besar agar model dapat dilatih menggunakan input berukuran lebih besar atau volume penuh.

Selain itu, fungsi loss yang digunakan pada penelitian ini masih didominasi oleh metrik matematis seperti MSE, SSIM, dan Gradient Loss. Meskipun efektif dalam menekan error numerik dan menjaga struktur dasar, pendekatan ini belum secara eksplisit mengakomodasi persepsi visual manusia. Penggunaan Perceptual Loss seperti VGG-Loss atau Adversarial Loss berbasis GAN dapat dipertimbangkan pada penelitian selanjutnya untuk mendorong model menghasilkan tekstur yang lebih alami dan tajam, sesuai dengan cara radiolog menilai kualitas citra medis.

Dari sisi evaluasi, penilaian performa model pada penelitian ini masih terbatas pada metrik teknis seperti PSNR, SSIM, RMSE, dan CNR. Untuk memperoleh validasi yang lebih komprehensif terhadap kelayakan klinis model, penelitian lanjutan disarankan melibatkan dokter spesialis radiologi dalam proses evaluasi visual melalui metode blind scoring, sehingga kualitas citra rekonstruksi dapat dinilai secara langsung dari sudut pandang klinis.

## DAFTAR REFERENSI

- [1] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Munich, Germany, 2015, pp. 234–241.
- [2] M. Mahesh, “The Essential Physics of Medical Imaging, Third Edition,” *Med. Phys.*, vol. 40, no. 7, p. 077301, 2013.
- [3] J. Hsieh, *Computed tomography principles, design, artifacts, and recent advances*, 2nd ed. Hoboken, NJ: Wiley-Blackwell, 2009.
- [4] D. J. Brenner and E. J. Hall, “Computed tomography--an increasing source of radiation exposure,” *N. Engl. J. Med.*, vol. 357, no. 22, pp. 2277–2284, 2007.
- [5] L. W. Goldman, “Principles of CT: radiation dose and image quality,” *J. Nucl. Med. Technol.*, vol. 35, no. 4, pp. 213–25; quiz 226–8, 2007.
- [6] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*. Society for Industrial and Applied Mathematics, 2001.
- [7] H. Chen et al., “aLow-dose CT via convolutional neural network,” *Biomedical Optics Express*, vol. 8, no. 2, p. 679, Jan. 2017, doi: <https://doi.org/10.1364/boe.8.000679>.
- [8] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-net: Learning dense volumetric segmentation from sparse annotation,” *arXiv [cs.CV]*, 2016.

- [9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv [cs.CV], 2015.
- [10] O. Oktay et al., "Attention U-Net: Learning where to look for the pancreas," arXiv [cs.CV], 2018.
- [11] H. Chen *et al.*, "Low-Dose CT With a Residual Encoder-Decoder Convolutional Neural Network," *IEEE Transactions on Medical Imaging*, vol. 36, no. 12, pp. 2524–2535, Dec. 2017, doi: <https://doi.org/10.1109/tmi.2017.2715284>.
- [12] I. Dezhnabi and M. Fiterau, "MultiWave: Multiresolution Deep Architectures through Wavelet Decomposition for Multivariate Time Series Prediction," Proceedings of Machine Learning Research, vol. 209, p. 2023, Accessed: Dec. 24, 2025. [Online]. Available: <https://proceedings.mlr.press/v209/deznabi23a/deznabi23a.pdf>
- [13] P. Liu, H. Zhang, W. Lian, and W. Zuo, "Multi-Level Wavelet Convolutional Neural Networks," IEEE Access, vol. 7, pp. 74973–74985, 2019, doi: <https://doi.org/10.1109/access.2019.2921451>.
- [14] H. Chen, Y. Zhang, M. K. Kalra, F. Lin, Y. Chen, P. Liao, J. Zhou, and G. Wang, "Low-dose CT with a residual encoder-decoder convolutional neural network," IEEE Transactions on Medical Imaging, vol. 36, no. 12, pp. 2524–2535, 2017.
- [15] H. Shan, Y. Zhang, Q. Yang, U. Kruger, M. K. Kalra, L. Sun, W. Cong, and G. Wang, "3-D convolutional encoder-decoder network for low-dose CT via transfer learning from a 2-D trained network," IEEE Transactions on Medical Imaging, vol. 37, no. 6, pp. 1522–1534, 2018.
- [16] E. Kang, J. Min, and J. C. Ye, "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction," Medical Physics, vol. 44, no. 10, pp. e360–e375, 2017.
- [17] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz *et al.*, "Attention u-net: Learning where to look for the pancreas," in Medical Imaging with Deep Learning, 2018.
- [18] J. Leuschner, M. Schmidt, D. Oveson, P. Baguer, and J. Maass, "LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction," *Scientific Data*, vol. 8, no. 1, p. 109, 2021.
- [19] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems* 32, 2019, pp. 8024–8035.
- [20] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.
- [21] J. Adler, H. Kohr, and O. Oktem, "Operator Discretization Library (ODL)," *Software available at <https://github.com/odlgroup/odl>*, 2017.
- [22] S. Van der Walt *et al.*, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.
- [23] A. Treuille, T. Teixeira, and T. S. Adrien, "Streamlit," *Software available at <https://streamlit.io>*, 2018.

- [24] J. Leuschner, M. Schmidt, D. Oveson, P. Baguer, and J. Maass, "LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction," *Scientific Data*, vol. 8, no. 1, p. 109, 2021.
- [25] J. Adler, H. Kohr, and O. Oktém, "Operator Discretization Library (ODL)," *Software available at <https://github.com/odlgroup/odl>*, 2017.
- [26] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. New York: IEEE Press, 1988.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [28] O. Oktay et al., "Attention U-Net: Learning Where to Look for the Pancreas," in *Medical Imaging with Deep Learning (MIDL)*, 2018.
- [29] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, vol. 1, no. 10, p. e3, 2016.

## LAMPIRAN

### **RED CNN**

[CODE MODEL RED CNN](#)

### **MW CNN**

[CODE MODEL MW CNN](#)

### **U-NET**

#### **Lampiran 1: Kode Python FBP**

```
import h5py
import numpy as np
import odl
import matplotlib.pyplot as plt

# =====
# 1. KONFIGURASI FILE (PILIH 1 FILE SAJA)
# =====
# Pastikan path ini benar sesuai lokasi file Anda
path_observation =
"/content/dataset_sementara/observation_validation_000.hdf5"
path_ground_truth =
"/content/dataset_sementara/ground_truth_validation_000.hdf5"
# (Ground truth opsional, cuma buat pembanding)

# =====
# 2. SETUP GEOMETRI ODL (GPU MODE)
# =====
print("⚙️ Menyiapkan Geometri & GPU...")
```

```

reco_space = odl.uniform_discr(
    min_pt=[-0.13, -0.13], max_pt=[0.13, 0.13], shape=[362, 362],
    dtype='float32'
)
angle_partition = odl.uniform_partition(0, np.pi, 1000)
detector_partition = odl.uniform_partition(-0.19, 0.19, 513)
geometry = odl.tomo.Parallel2dGeometry(angle_partition,
                                         detector_partition)

try:
    operator = odl.tomo.RayTransform(reco_space, geometry,
                                      impl='astra_cuda')
    print("✅ GPU T4 Aktif (astra_cuda).")
except:
    print("⚠️ GPU Gagal, pakai CPU. Akan lambat.")
    operator = odl.tomo.RayTransform(reco_space, geometry,
                                      impl='astra_cpu')

fbp = odl.tomo.fbp_op(operator)

# =====
# 3. PROSES REKONSTRUKSI VOLUME (STACKING)
# =====
def reconstruct_volume_3d():
    print(f"🚀 Memproses file: {path_observation}")

    with h5py.File(path_observation, 'r') as f_obs:
        # Ambil semua data dalam file ini
        data_sinogram = f_obs['data'][:]
        num_slices = data_sinogram.shape[0]

        print(f"    Jumlah Slice dalam file: {num_slices}")
        print("    ⏳ Sedang merekonstruksi seluruh volume (mohon tunggu) ...")

        # Wadah untuk menumpuk hasil (Volume 3D)
        # Dimensi nanti: (Jumlah_Slice, 362, 362)
        volume_3d = np.zeros((num_slices, 362, 362),
                             dtype=np.float32)

        for i in range(num_slices):
            # Ambil 1 lembar sinogram
            sino = data_sinogram[i]

            # Rekonstruksi FBP
            rec = fbp(sino).asarray()

            # Normalisasi
            rec = (rec - np.min(rec)) / (np.max(rec) -
                                         np.min(rec) + 1e-8)

            # Simpan ke tumpukan volume
            volume_3d[i, :, :] = rec

            if (i+1) % 20 == 0:
                print(f"    Processed {i+1}/{num_slices}...")

```

```

        print(f"✅ Selesai! Volume 3D terbentuk.")
        print(f"📦 Dimensi Akhir: {volume_3d.shape} -> (Z, Y,
X)")

    return volume_3d

# Jalankan Fungsi
volume_hasil = reconstruct_volume_3d()

# =====
# 4. VISUALISASI PERBEDAAN (ORTHOGONAL VIEW)
# =====
# Ini pembuktian bahwa data Anda 3D. Kita iris dari berbagai
sisi.

def show_3d_slices(vol):
    z, y, x = vol.shape

    # Ambil irisan tengah dari masing-masing sumbu
    slice_axial = vol[z // 2, :, :]      # Pandangan dari Atas
(Standar CT)
    slice_coronal = vol[:, y // 2, :]     # Pandangan dari Depan
    slice_sagittal = vol[:, :, x // 2]    # Pandangan dari Samping

    plt.figure(figsize=(15, 5))

    # 1. Axial View (X-Y) - Yang biasa Anda lihat
    plt.subplot(1, 3, 1)
    plt.imshow(slice_axial, cmap='gray')
    plt.title(f"1. Axial View (Atas)\nSlice Z={z//2}")
    plt.xlabel("X axis"); plt.ylabel("Y axis")

    # 2. Coronal View (X-Z) - Bukti 3D
    # (Kita harus rotasi sedikit agar tegak)
    plt.subplot(1, 3, 2)
    plt.imshow(np.rot90(slice_coronal), cmap='gray')
    plt.title(f"2. Coronal View (Depan)\nSlice Y={y//2}")
    plt.xlabel("X axis"); plt.ylabel("Z axis (Tumpukan)")

    # 3. Sagittal View (Y-Z) - Bukti 3D
    plt.subplot(1, 3, 3)
    plt.imshow(np.rot90(slice_sagittal), cmap='gray')
    plt.title(f"3. Sagittal View (Samping)\nSlice X={x//2}")
    plt.xlabel("Y axis"); plt.ylabel("Z axis (Tumpukan)")

    plt.tight_layout()
    plt.show()

print("\n📊 Menampilkan Visualisasi 3D Orthogonal...")
show_3d_slices(volume_hasil)

```

## Lampiran 2: Kode Python Micro-patching

```

import numpy as np
import os
import glob

```

```

import h5py
import gc

# =====
# 1. KONFIGURASI PATH (REVISI)
# =====

# A. Folder Input FBP (Dari Google Drive - .npy)
input_drive_folder = "/content/drive/MyDrive/Finpro
Pencit/HDF5/Hasil FBP/"

# B. Folder Ground Truth (Dari Lokal Colab - .hdf5)
# Pastikan Anda sudah unzip file ground truth ke folder ini!
gt_local_folder = "/content/dataset_sementara_GT/"

# C. Folder Output (Ke Google Drive - .npy Patches)
output_folder = "/content/drive/MyDrive/Finpro
Pencit/HDF5/Training_Patches/"
os.makedirs(output_folder, exist_ok=True)

# Parameter Patching
PATCH_SIZE = (32, 32, 32)
STRIDE = (16, 16, 16)
THRESHOLD_AIR = 0.1 # Set 0.0 jika ingin mengambil SEMUA 86.436
patch

print(f"📁 Input FBP (Drive): {input_drive_folder}")
print(f"📁 Input GT (Lokal): {gt_local_folder}")
print(f"💾 Output Patches: {output_folder}")

# =====
# 2. EKSEKUSI BATCH HYBRID
# =====

def process_patching_hybrid():
    # Cari file input FBP (.npy) di Drive
    search_pattern = os.path.join(input_drive_folder,
"processed_input_*.npy")
    list_files = sorted(glob.glob(search_pattern))

    if len(list_files) == 0:
        print("❌ Tidak ada file Input FBP (.npy) di Drive.")
        return

    print(f"\n⌚ Memulai Patching untuk {len(list_files)} file
pasangan.\n")

    for i, input_path in enumerate(list_files):
        # 1. Identifikasi File
        filename_npy = os.path.basename(input_path)
        # Ambil ID: processed_input_000.npy -> 000
        file_id = filename_npy.split('_')[-1].replace('.npy', '')

        print(f"✍️ [{i+1}/{len(list_files)}] Memproses ID:
{file_id}")

        # 2. Cari Pasangan GT HDF5 di Lokal
        gt_filename = f"ground_truth_validation_{file_id}.hdf5"

```

```

gt_path = os.path.join(gt_local_folder, gt_filename)

if not os.path.exists(gt_path):
    print(f"⚠️ SKIP: File GT {gt_filename} tidak ada di {gt_local_folder}")
    continue

try:
    # --- LOAD DATA ---

    # A. Load Input FBP (NPY dari Drive)
    vol_input = np.load(input_path) # Sudah normalisasi
0-1

    # B. Load Target GT (HDF5 dari Lokal)
    with h5py.File(gt_path, 'r') as f_gt:
        # Ambil data HDF5
        # PENTING: File FBP (.npy) mungkin jumlah
        slice-nya lebih sedikit dari 128 (karena valid samples)
        # Jadi kita harus potong GT HDF5 agar sama
        panjangnya dengan Input FBP
        valid_slices = vol_input.shape[0]

        vol_target_raw = f_gt['data'][:valid_slices] # Ambil sejumlah input

        # Normalisasi GT (Wajib, karena HDF5 masih raw)
        # Tambah 1e-8 untuk safety
        vol_target = (vol_target_raw -
np.min(vol_target_raw)) / \
                (np.max(vol_target_raw) -
np.min(vol_target_raw) + 1e-8)

        # Cek Dimensi
        if vol_input.shape != vol_target.shape:
            print(f"✖️ Dimensi beda!
Input:{vol_input.shape} vs Target:{vol_target.shape}")
            continue

    # --- PROSES PATCHING ---
    batch_patches_input = []
    batch_patches_target = []

    z_len, y_len, x_len = vol_input.shape
    pz, py, px = PATCH_SIZE
    sz, sy, sx = STRIDE

    for z in range(0, z_len - pz + 1, sz):
        for y in range(0, y_len - py + 1, sy):
            for x in range(0, x_len - px + 1, sx):

                # Cek Threshold (Pakai target yang
                bersih)
                patch_check = vol_target[z:z+pz, y:y+py,
x:x+px]

                if np.mean(patch_check) > THRESHOLD_AIR:

```

```

        p_in = vol_input[z:z+pz, y:y+py,
x:x+px]

        p_gt = vol_target[z:z+pz, y:y+py,
x:x+px]

        batch_patches_input.append(p_in)
batch_patches_target.append(p_gt)

# --- SIMPAN KE DRIVE ---
if len(batch_patches_input) > 0:
    np_in = np.array(batch_patches_input,
dtype=np.float32)
    np_gt = np.array(batch_patches_target,
dtype=np.float32)

    save_name_in = os.path.join(output_folder,
f"patch_in_{file_id}.npy")
    save_name_gt = os.path.join(output_folder,
f"patch_gt_{file_id}.npy")

    np.save(save_name_in, np_in)
    np.save(save_name_gt, np_gt)

    print(f" ✓ Disimpan:
{len(batch_patches_input)} patches.")
else:
    print(" ! File ini kosong/hanya udara.")

# Bersih-bersih RAM
del vol_input, vol_target, vol_target_raw
gc.collect()

except Exception as e:
    print(f" ✗ Error: {e}")

print("\n🎉 SELESAI! Semua patch siap training di Drive.")

# Jalankan
process_patching_hybrid()

```

### Lampiran 3: Kode Python Training dan Validasi

```

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader, random_split
from pytorch_msssim import ssim
import numpy as np
import os
import glob
import time
import bisect
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# =====

```

```

# 1. KONFIGURASI (JANTUNG OPERASI)
# =====
# Path Data Patch
data_folder = "/content/drive/MyDrive/Finpro
Pencit/Training_Patches"

# Nama Eksperimen (JANGAN UBAH JIKA MAU RESUME)
# Gunakan nama yang spesifik. Jika mau mulai baru, ganti nama ini.
experiment_name = "Run_Final_V6_LimitBatch"

# Folder Output
output_dir = f"/content/drive/MyDrive/Finpro
Pencit/Training_Logs/{experiment_name}/"
os.makedirs(output_dir, exist_ok=True)

# Hyperparameters
BATCH_SIZE = 2           # Aman untuk Colab
NUM_WORKERS = 0          # Wajib 0
LEARNING_RATE = 1e-4
NUM_EPOCHS = 50          # Target Epoch

# PEMBAGIAN DATA (Split)
TEST_SPLIT = 0.10        # 10% untuk Test (Disisihkan)
VAL_SPLIT = 0.20          # 20% untuk Validasi

# LIMIT BATCH (Agar Epoch Cepat)
TRAIN_BATCH_LIMIT = 2000 # 1 Epoch = 2000 Batch (sekitar 15-20
menit)
VAL_BATCH_LIMIT = 200     # Validasi secukupnya saja

print(f"📁 Folder Output: {output_dir}")
print(f"⌚ Limit Batch: {TRAIN_BATCH_LIMIT} per Epoch")

# =====
# 2. KOMPONEN DATASET & MODEL
# =====
class LoDoPaBDataset(Dataset):
    def __init__(self, folder_path):
        self.folder_path = folder_path
        self.input_files =
sorted(glob.glob(os.path.join(folder_path, "patch_in_*.npy")))
        if len(self.input_files) == 0:
            raise RuntimeError(f"❌ Error: Tidak ada file .npy di
{folder_path}")

        print(f"⚡ Mengindeks {len(self.input_files)} file...
(Lazy Loading)")
        self.file_indices = []
        self.cumulative_indices = [0]
        for f_path in self.input_files:
            data = np.load(f_path, mmap_mode='r')
            num = data.shape[0]
            self.file_indices.append(num)

        self.cumulative_indices.append(self.cumulative_indices[-1] + num)
        self.total_patches = self.cumulative_indices[-1]

```

```

        print(f"✓ Total Data: {self.total_patches} patches")

    def __len__(self): return self.total_patches

    def __getitem__(self, idx):
        file_idx = bisect.bisect_right(self.cumulative_indices,
idx) - 1
        local_idx = idx - self.cumulative_indices[file_idx]

        input_path = self.input_files[file_idx]
        filename = os.path.basename(input_path)
        target_path = os.path.join(self.folder_path,
filename.replace("patch_in", "patch_gt"))

        d_in = np.load(input_path, mmap_mode='r')
        d_gt = np.load(target_path, mmap_mode='r')

        p_in = np.array(d_in[local_idx]).astype(np.float32)
        p_gt = np.array(d_gt[local_idx]).astype(np.float32)

        return torch.from_numpy(np.expand_dims(p_in, axis=0)),
torch.from_numpy(np.expand_dims(p_gt, axis=0))

# --- MODEL ARSITEKTUR (Fixed) ---
class ResBlock(nn.Module):
    def __init__(self, in_c, out_c):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv3d(in_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c), nn.ReLU(True),
            nn.Conv3d(out_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c))
        self.relu = nn.ReLU(True)
        self.sc = nn.Conv3d(in_c, out_c, 1) if in_c != out_c else
nn.Sequential()
        def forward(self, x): return self.relu(self.conv(x) +
self.sc(x))

class AttentionBlock(nn.Module):
    def __init__(self, F_g, F_l, F_int):
        super().__init__()
        self.Wg = nn.Sequential(nn.Conv3d(F_g, F_int, 1),
nn.BatchNorm3d(F_int))
        self.Wx = nn.Sequential(nn.Conv3d(F_l, F_int, 1),
nn.BatchNorm3d(F_int))
        self.psi = nn.Sequential(nn.Conv3d(F_int, 1, 1),
nn.BatchNorm3d(1), nn.Sigmoid())
        self.relu = nn.ReLU(True)
    def forward(self, g, x):
        psi = self.relu(self.Wg(g) + self.Wx(x))
        return x * self.psi(psi)

class Lightweight3DUNet(nn.Module):
    def __init__(self, in_c=1, out_c=1):
        super().__init__()
        f = [16, 32, 64, 128]
        self.enc1 = ResBlock(in_c, f[0]); self.p1 =

```

```

nn.MaxPool3d(2)
    self.enc2 = ResBlock(f[0], f[1]); self.p2 =
nn.MaxPool3d(2)
    self.enc3 = ResBlock(f[1], f[2]); self.p3 =
nn.MaxPool3d(2)
    self.bot = ResBlock(f[2], f[3])

    self.up3 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
    self.att3 = AttentionBlock(f[3], f[2], f[2])
    self.dec3 = ResBlock(f[3]+f[2], f[2])

    self.up2 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
    self.att2 = AttentionBlock(f[2], f[1], f[1])
    self.dec2 = ResBlock(f[2]+f[1], f[1])

    self.up1 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
    self.att1 = AttentionBlock(f[1], f[0], f[0])
    self.dec1 = ResBlock(f[1]+f[0], f[0])

    self.out = nn.Conv3d(f[0], out_c, 1)
    self.dropout = nn.Dropout3d(0.2)

def forward(self, x):
    e1 = self.enc1(x); p1 = self.p1(e1)
    e2 = self.enc2(p1); p2 = self.p2(e2)
    e3 = self.enc3(p2); p3 = self.p3(e3)
    b = self.bot(p3)

    d3 = self.up3(b); x3 = self.att3(d3, e3); d3 =
self.dec3(torch.cat([x3, d3], 1))
    d2 = self.up2(d3); x2 = self.att2(d2, e2); d2 =
self.dec2(torch.cat([x2, d2], 1))
    d1 = self.up1(d2); x1 = self.att1(d1, e1); d1 =
self.dec1(torch.cat([x1, d1], 1))
    return self.out(self.dropout(d1))

class GradientLoss3D(nn.Module):
    def __init__(self):
        super().__init__()
        k =
torch.FloatTensor([[[[-1,0,1],[-2,0,2],[-1,0,1]],[[-2,0,2],[-4,0,
4],[-2,0,2]],[[-1,0,1],[-2,0,2],[-1,0,1]]]])
        self.k = nn.Parameter(k.unsqueeze(1),
requires_grad=False)
    def forward(self, p, t):
        if p.device != self.k.device: self.k =
self.k.to(p.device)
        gp = torch.abs(torch.nn.functional.conv3d(p, self.k,
padding=1))
        gt = torch.abs(torch.nn.functional.conv3d(t, self.k,
padding=1))
        return torch.mean(torch.abs(gp - gt))

def composite_loss(p, t, g_fn):

```

```

        return nn.MSELoss()(p, t) + 0.1*(1-ssim(p, t, data_range=1.0,
size_average=True)) + 0.01*g_fn(p, t)

# =====
# 3. PERSIAPAN DATA & AUTO-RESUME
# =====
full_ds = LoDoPaBDataset(data_folder)
total_len = len(full_ds)
test_len = int(total_len * TEST_SPLIT)
val_len = int(total_len * VAL_SPLIT)
train_len = total_len - val_len - test_len

train_ds, val_ds, test_ds = random_split(full_ds, [train_len,
val_len, test_len])
print(f"\n📊 SPLIT DATA: Train={len(train_ds)} | Val={len(val_ds)}\n| Test={len(test_ds)} (Disisihkan)")

train_loader = DataLoader(train_ds, batch_size=BATCH_SIZE,
shuffle=True, num_workers=NUM_WORKERS)
val_loader = DataLoader(val_ds, batch_size=BATCH_SIZE,
shuffle=False, num_workers=NUM_WORKERS)

device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
model = Lightweight3DUNet().to(device)
optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
grad_fn = GradientLoss3D().to(device)

# --- LOGIKA RESUME ---
start_epoch = 0
best_val_loss = float('inf')
log_history = []
csv_path = os.path.join(output_dir, "training_log.csv")

checkpoints = glob.glob(os.path.join(output_dir,
"checkpoint_ep*.pth"))
if len(checkpoints) > 0:
    latest = max(checkpoints, key=os.path.getctime)
    print(f"\n🔄 Melanjutkan dari: {os.path.basename(latest)}")
    model.load_state_dict(torch.load(latest,
map_location=device))
    try: start_epoch =
int(latest.split('_ep')[-1].replace('.pth', ''))
    except: start_epoch = 0

    if os.path.exists(csv_path):
        df = pd.read_csv(csv_path)
        log_history = df.to_dict('records')
        if len(log_history) > 0: best_val_loss =
min([x['val_loss'] for x in log_history])
            print(f"\n✓ History: {len(log_history)} epoch\nterpulihkan.")
    else:
        print("\n🆕 Memulai Training Baru.")

# =====
# 4. TRAINING LOOP (VERBOSE & LIMITED)

```

```

# =====
print(f"🔥 START TRAINING (Epoch {start_epoch+1}/{NUM_EPOCHS})")

for epoch in range(start_epoch, NUM_EPOCHS):
    t0 = time.time()
    model.train()
    train_loss = 0.0
    count = 0

    # Train Loop (Limit Batch)
    for i, (x, y) in enumerate(train_loader):
        if i >= TRAIN_BATCH_LIMIT: break # Limit check

        x, y = x.to(device), y.to(device)
        optimizer.zero_grad()
        out = model(x)
        loss = composite_loss(out, y, grad_fn)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()
        count += 1

        # CEREWET MODE: Lapor setiap 100 batch
        if (i+1) % 100 == 0:
            print(f" [Epoch {epoch+1}] Batch {i+1}/{TRAIN_BATCH_LIMIT} | Loss: {loss.item():.5f}")

    avg_train = train_loss / count if count > 0 else 0

    # Validation Loop
    print(f"⌛ Validasi...")
    model.eval()
    val_loss = 0.0
    v_count = 0
    with torch.no_grad():
        for i, (x, y) in enumerate(val_loader):
            if i >= VAL_BATCH_LIMIT: break
            x, y = x.to(device), y.to(device)
            val_loss += composite_loss(model(x), y,
grad_fn).item()
            v_count += 1

    avg_val = val_loss / v_count if v_count > 0 else 0
    dt = (time.time() - t0)/60

    print(f"✓ Epoch {epoch+1} Selesai ({dt:.1f}m) | Train: {avg_train:.5f} | Val: {avg_val:.5f}")

    # Save & Log
    log_history.append({'epoch': epoch+1, 'train_loss': avg_train, 'val_loss': avg_val, 'time_m': dt})
    pd.DataFrame(log_history).to_csv(csv_path, index=False)

    torch.save(model.state_dict(), os.path.join(output_dir,
f"checkpoint_ep{epoch+1}.pth"))
    if avg_val < best_val_loss:

```

```

        best_val_loss = avg_val
        torch.save(model.state_dict(), os.path.join(output_dir,
"best_model.pth"))
        print(f"    🏆 New Best Model Saved!")

# Cleanup Checkpoint Lama (Simpan 3 terakhir saja)
if epoch > 3:
    old = os.path.join(output_dir,
f"checkpoint_ep{epoch-2}.pth")
    if os.path.exists(old): os.remove(old)

# =====
# 5. POST-TRAINING REPORT (SPECS & PLOT)
# =====
print("\n🎉 TRAINING SELESAI! Menampilkan Laporan Lengkap...")

# A. Spesifikasi Model
total_params = sum(p.numel() for p in model.parameters())
print("\n" + "="*30)
print(f"🤖 SPESIFIKASI MODEL")
print("="*30)
print(f"Model Name      : Lightweight 3D Res-Att U-Net")
print(f"Total Parameter : {total_params:,} parameters")
print(f"Input Shape     : (Batch, 1, 32, 32, 32)")
print(f"Filters Config  : [16, 32, 64, 128]")
print(f"Dropout Rate    : 0.2 (Monte Carlo Ready)")
print(f"Loss Function   : Composite (MSE + SSIM + Gradient)")
print("="*30 + "\n")

# B. Plot Grafik Training
if len(log_history) > 0:
    df = pd.DataFrame(log_history)
    plt.figure(figsize=(10, 6))
    plt.plot(df['epoch'], df['train_loss'], label='Training Loss',
marker='o', linestyle='--')
    plt.plot(df['epoch'], df['val_loss'], label='Validation Loss',
marker='s', linestyle='--')

    plt.title(f"Training History: {experiment_name}")
    plt.xlabel("Epoch")
    plt.ylabel("Composite Loss")
    plt.legend()
    plt.grid(True, alpha=0.3)

    plot_path = os.path.join(output_dir,
"final_training_plot.png")
    plt.savefig(plot_path, dpi=300)
    print(f"📊 Grafik disimpan di: {plot_path}")
    plt.show()
else:
    print("⚠️ Tidak ada data log untuk di-plot.")

```

#### Lampiran 4: Kode Python Testing (Input Patch)

```

# =====
# 0. SETUP & LIBRARY
# =====

```

```

import os
import glob
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader, random_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis, entropy
from tqdm import tqdm
import bisect

# Install Library SSIM CPU (Anti-Error Dimensi)
try:
    from skimage.metrics import structural_similarity as ssim
except ImportError:
    !pip install scikit-image -q
    from skimage.metrics import structural_similarity as ssim

# =====
# 1. KONFIGURASI PATH (GOOGLE DRIVE)
# =====
experiment_name = "Run_Final_V6_LimitBatch"
base_output_dir = f"/content/drive/MyDrive/Finpro
Pencit/Output_Testing_Final/"
img_save_dir = os.path.join(base_output_dir, "Visual_Evidence")

# Buat Folder
os.makedirs(base_output_dir, exist_ok=True)
os.makedirs(img_save_dir, exist_ok=True)

print(f"📁 Hasil Analisis (Excel/Grafik) akan disimpan di:
{base_output_dir}")
print(f"📁 Bukti Gambar (PNG) akan disimpan di: {img_save_dir}")

# Path Data & Model
data_folder = "/content/drive/MyDrive/Finpro
Pencit/Training_Patches"
checkpoint_path = f"/content/drive/MyDrive/Finpro
Pencit/Training_Logs/{experiment_name}/best_model.pth"
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")

# =====
# 2. DEFINISI MODEL (FIXED PARAMETER NAME)
# =====
class ResBlock(nn.Module):
    def __init__(self, in_c, out_c):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv3d(in_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c), nn.ReLU(True),
            nn.Conv3d(out_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c))
        self.relu = nn.ReLU(True)

```

```

        self.sc = nn.Conv3d(in_c, out_c, 1) if in_c != out_c else
nn.Sequential()
    def forward(self, x): return self.relu(self.conv(x) +
self.sc(x))

class AttentionBlock(nn.Module):
    def __init__(self, F_g, F_l, F_int):
        super().__init__()
        self.Wg = nn.Sequential(nn.Conv3d(F_g, F_int, 1),
nn.BatchNorm3d(F_int))
        self.Wx = nn.Sequential(nn.Conv3d(F_l, F_int, 1),
nn.BatchNorm3d(F_int))
        self.psi = nn.Sequential(nn.Conv3d(F_int, 1, 1),
nn.BatchNorm3d(1), nn.Sigmoid())
        self.relu = nn.ReLU(True)
    def forward(self, g, x):
        psi = self.relu(self.Wg(g) + self.Wx(x))
        return x * self.psi(psi)

class Lightweight3DUNet(nn.Module):
    # FIX: Menerima 'in_channels' agar tidak TypeError
    def __init__(self, in_channels=1, out_channels=1):
        super().__init__()
        in_c, out_c = in_channels, out_channels
        f = [16, 32, 64, 128]
        self.enc1 = ResBlock(in_c, f[0]); self.p1 =
nn.MaxPool3d(2)
        self.enc2 = ResBlock(f[0], f[1]); self.p2 =
nn.MaxPool3d(2)
        self.enc3 = ResBlock(f[1], f[2]); self.p3 =
nn.MaxPool3d(2)
        self.bot = ResBlock(f[2], f[3])

        self.up3 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att3 = AttentionBlock(f[3], f[2], f[2])
        self.dec3 = ResBlock(f[3]+f[2], f[2])
        self.up2 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att2 = AttentionBlock(f[2], f[1], f[1])
        self.dec2 = ResBlock(f[2]+f[1], f[1])
        self.up1 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att1 = AttentionBlock(f[1], f[0], f[0])
        self.dec1 = ResBlock(f[1]+f[0], f[0])
        self.out = nn.Conv3d(f[0], out_c, 1)
        self.dropout = nn.Dropout3d(0.2)

    def forward(self, x):
        e1 = self.enc1(x); p1 = self.p1(e1)
        e2 = self.enc2(p1); p2 = self.p2(e2)
        e3 = self.enc3(p2); p3 = self.p3(e3)
        b = self.bot(p3)

        d3 = self.up3(b)
        if d3.size() != e3.size(): d3 = F.interpolate(d3,
size=e3.shape[2:], mode='trilinear', align_corners=True)

```

```

        x3 = self.att3(d3, e3); d3 = self.dec3(torch.cat([x3,
d3], 1))
        d2 = self.up2(d3)
        if d2.size() != e2.size(): d2 = F.interpolate(d2,
size=e2.shape[2:], mode='trilinear', align_corners=True)
        x2 = self.att2(d2, e2); d2 = self.dec2(torch.cat([x2,
d2], 1))
        d1 = self.up1(d2)
        if d1.size() != e1.size(): d1 = F.interpolate(d1,
size=e1.shape[2:], mode='trilinear', align_corners=True)
        x1 = self.att1(d1, e1); d1 = self.dec1(torch.cat([x1,
d1], 1))
        return self.out(self.dropout(d1))

UNet3D = Lightweight3DUNet

# =====
# 3. DATASET & SPLIT
# =====
class LoDoPaBDataset(Dataset):
    def __init__(self, folder_path):
        self.folder_path = folder_path
        self.input_files =
sorted(glob.glob(os.path.join(folder_path, "patch_in_*.npy")))
        if len(self.input_files) == 0:
            local_path = "/content/data_lokal_patches/"
            if os.path.exists(local_path):
                self.input_files =
sorted(glob.glob(os.path.join(local_path, "patch_in_*.npy")))
                self.folder_path = local_path

        self.cumulative_indices = [0]
        # Lazy loading header only
        for f_path in self.input_files:
            try:
                data = np.load(f_path, mmap_mode='r')

self.cumulative_indices.append(self.cumulative_indices[-1] +
data.shape[0])
            except: pass
        self.total_patches = self.cumulative_indices[-1]

    def __len__(self): return self.total_patches

    def __getitem__(self, idx):
        file_idx = bisect.bisect_right(self.cumulative_indices,
idx) - 1
        local_idx = idx - self.cumulative_indices[file_idx]

        input_path = self.input_files[file_idx]
        filename = os.path.basename(input_path)
        target_path = os.path.join(self.folder_path,
filename.replace("patch_in", "patch_gt"))

        d_in = np.load(input_path, mmap_mode='r')
        d_gt = np.load(target_path, mmap_mode='r')

```

```

        # Clip agar aman 0-1
        p_in =
np.clip(np.array(d_in[local_idx]).astype(np.float32), 0, 1)
        p_gt =
np.clip(np.array(d_gt[local_idx]).astype(np.float32), 0, 1)

        return torch.from_numpy(np.expand_dims(p_in, axis=0)),
torch.from_numpy(np.expand_dims(p_gt, axis=0))

# SETUP DATA SPLIT
full_ds = LoDoPaBDataset(data_folder)
total_len = len(full_ds)

# Safety Logic: Jika data sangat sedikit (untuk tes debug),
jangan paksa 10%
if total_len < 20:
    test_len = 2 # Minimal 2 data
    val_len = 0
    train_len = total_len - test_len
else:
    test_len = int(total_len * 0.10)
    val_len = int(total_len * 0.20)
    train_len = total_len - val_len - test_len

# Seed 42 untuk konsistensi
generator = torch.Generator().manual_seed(42)
_, _, test_ds = random_split(full_ds, [train_len, val_len,
test_len], generator=generator)

print(f"📊 Total Data Test (10%): {len(test_ds)} Patches")
test_loader = DataLoader(test_ds, batch_size=1, shuffle=False)

# =====
# 4. METRICS & RADIOMICS FUNCTIONS
# =====
def calculate_metrics_numpy(pred, gt):
    """PSNR, RMSE, SSIM (CPU Version - Anti Error)"""
    pred = pred.astype(np.float32)
    gt = gt.astype(np.float32)

    # 1. MSE & RMSE
    mse = np.mean((pred - gt) ** 2)
    rmse = np.sqrt(mse)

    # 2. PSNR
    if mse == 0: psnr = 100
    else: psnr = 20 * np.log10(1.0 / rmse)

    # 3. SSIM 3D (Average of 2D slices)
    ssim_val = 0
    D = gt.shape[0]
    for z in range(D):
        try:
            # win_size diset kecil (3) utk jaga-jaga kalau patch
            # kecil (32x32)
            s = ssim(gt[z], pred[z], data_range=1.0, win_size=3)
        except ValueError:

```

```

        s = ssim(gt[z], pred[z], data_range=1.0)
        ssim_val += s

    return psnr, ssim_val / D, rmse

def get_radiomics(img_vol):
    """Radiomics Sederhana"""
    flat = img_vol.flatten()
    return {
        'Mean': np.mean(flat),
        'Std': np.std(flat),
        'Skewness': skew(flat),
        'Kurtosis': kurtosis(flat),
        'Entropy': entropy(np.histogram(flat, bins=50)[0] + 1e-8)
    }

# =====
# 5. EXECUTION LOOP
# =====
# Load Model
model = UNet3D(in_channels=1, out_channels=1).to(device)
checkpoint = torch.load(checkpoint_path, map_location=device)
if 'model_state_dict' in checkpoint:
    model.load_state_dict(checkpoint['model_state_dict'])
else: model.load_state_dict(checkpoint)

metrics_log = []
radiomics_log = []

# LIMIT PENYIMPANAN GAMBAR (PENTING AGAR TIDAK CRASH)
SAVE_IMG_LIMIT = 50
img_saved_count = 0

print("🚀 Memulai Pengujian...")

with torch.no_grad():
    for i, (inputs, targets) in enumerate(tqdm(test_loader)):
        patient_id = f"Test_Sample_{i+1:04d}"
        inputs, targets = inputs.to(device), targets.to(device)

        # A. Prediksi Normal
        model.eval()
        output_std = model(inputs)

        # B. Monte Carlo Uncertainty (Opsional: Di-skip jika mau ngebut)
        # Kita lakukan hanya untuk sampel yang akan disimpan gambarnya
        uncertainty_vol = None
        if img_saved_count < SAVE_IMG_LIMIT:
            model.train() # Enable Dropout
            mc_stack = []
            for _ in range(5): # 5x Run cukup untuk estimasi cepat
                mc_stack.append(model(inputs).cpu().numpy())
            uncertainty_vol = np.std(np.array(mc_stack),
axis=0)[0, 0]

```

```

# C. Konversi Numpy
vol_in = inputs[0, 0].cpu().numpy()
vol_gt = targets[0, 0].cpu().numpy()
vol_out = output_std[0, 0].cpu().numpy()

# D. Hitung Metrik
psnr, ssim_val, rmse = calculate_metrics_numpy(vol_out,
vol_gt)

# E. Hitung Radiomics
rad_gt = get_radiomics(vol_gt)
rad_out = get_radiomics(vol_out)

metrics_log.append({'ID': patient_id, 'PSNR': psnr,
'SSIM': ssim_val, 'RMSE': rmse})

rad_entry = {'ID': patient_id}
for k in rad_gt:
    rad_entry[f'{k}_Error'] = abs(rad_gt[k] - rad_out[k])
radiomics_log.append(rad_entry)

# F. Simpan Gambar Bukti (Hanya 50 pertama)
if img_saved_count < SAVE_IMG_LIMIT and uncertainty_vol
is not None:
    mid = vol_gt.shape[0] // 2

    plt.figure(figsize=(20, 5))
    # 1. Input
    plt.subplot(1, 5, 1); plt.imshow(vol_in[mid],
cmap='gray'); plt.title("Input Low Dose")
    plt.axis('off')
    # 2. Output
    plt.subplot(1, 5, 2); plt.imshow(vol_out[mid],
cmap='gray'); plt.title(f"Output AI\nnPSNR: {psnr:.2f}")
    plt.axis('off')
    # 3. GT
    plt.subplot(1, 5, 3); plt.imshow(vol_gt[mid],
cmap='gray'); plt.title("Ground Truth")
    plt.axis('off')
    # 4. Uncertainty
    plt.subplot(1, 5, 4);
    plt.imshow(uncertainty_vol[mid], cmap='jet');
    plt.title("Uncertainty Map")
    plt.colorbar(fraction=0.046); plt.axis('off')
    # 5. Difference
    plt.subplot(1, 5, 5); plt.imshow(np.abs(vol_gt[mid] -
vol_out[mid]), cmap='inferno'); plt.title("Difference Error")
    plt.axis('off')

    plt.tight_layout()
    plt.savefig(os.path.join(img_save_dir,
f"{patient_id}_evidence.png"))
    plt.close()
    img_saved_count += 1

# =====

```

```

# 6. SIMPAN HASIL AKHIR
# =====
df_metrics = pd.DataFrame(metrics_log)
df_radiomics = pd.DataFrame(radiomics_log)

df_metrics.to_csv(os.path.join(base_output_dir,
"FINAL_Metrics.csv"), index=False)
df_radiomics.to_csv(os.path.join(base_output_dir,
"FINAL_Radiomics.csv"), index=False)

print("\n" + "="*40)
print("✅ TESTING SELESAI!")
print("="*40)
print(f"📊 Rata-rata PSNR: {df_metrics['PSNR'].mean():.2f} dB")
print(f"📊 Rata-rata SSIM: {df_metrics['SSIM'].mean():.4f}")
print(f"📸 {img_saved_count} Gambar bukti tersimpan di Drive.")
print(f"📁 Lokasi: {base_output_dir}")

```

#### Lampiran 5: Kode Python Testing (Input Full-Volume 3D)

```

import torch
import torch.nn as nn
import numpy as np
import os
import glob
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats
import math
from google.colab import drive
from tqdm.auto import tqdm
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import structural_similarity as ssim

# =====
# 1. KONFIGURASI PATH
# =====
drive.mount('/content/drive')

# Path Model (Sesuai request Anda: best_model dari
Run_Final_V6_LimitBatch)
# Note: Jika Anda mau pakai hasil fine-tuning baru, ganti folder
path-nya nanti.
MODEL_PATH = '/content/drive/MyDrive/Finpro
Pencit/Training_Logs/Run_Final_V6_FineTuned_Texture/best_model_fi
netuned.pth'

# Folder Data Full Image
DATA_FOLDER = '/content/drive/MyDrive/Finpro Pencit/Hasil FBP/'

# Output
OUTPUT_DIR = '/content/drive/MyDrive/Finpro
Pencit/Hasil_Evaluasi_LinearBlend_Fixed/'
os.makedirs(OUTPUT_DIR, exist_ok=True)

# Parameter
PATCH_SIZE = 32

```

```

OVERLAP_PERCENT = 0.5
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# =====
# 2. DEFINISI MODEL (SAMA PERSIS DGN TRAINING)
# =====
class ResBlock(nn.Module):
    def __init__(self, in_c, out_c):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv3d(in_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c), nn.ReLU(True),
            nn.Conv3d(out_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c))
        self.relu = nn.ReLU(True)
        self.sc = nn.Conv3d(in_c, out_c, 1) if in_c != out_c else nn.Sequential()
    def forward(self, x): return self.relu(self.conv(x) + self.sc(x))

class AttentionBlock(nn.Module):
    def __init__(self, F_g, F_l, F_int):
        super().__init__()
        self.Wg = nn.Sequential(nn.Conv3d(F_g, F_int, 1),
        nn.BatchNorm3d(F_int))
        self.Wx = nn.Sequential(nn.Conv3d(F_l, F_int, 1),
        nn.BatchNorm3d(F_int))
        self.psi = nn.Sequential(nn.Conv3d(F_int, 1, 1),
        nn.BatchNorm3d(1), nn.Sigmoid())
        self.relu = nn.ReLU(True)
    def forward(self, g, x):
        psi = self.relu(self.Wg(g) + self.Wx(x))
        return x * self.psi(psi)

class Lightweight3DUNet(nn.Module):
    def __init__(self, in_c=1, out_c=1):
        super().__init__()
        f = [16, 32, 64, 128] # Sesuai kode training Anda
        self.enc1 = ResBlock(in_c, f[0]); self.p1 =
        nn.MaxPool3d(2)
        self.enc2 = ResBlock(f[0], f[1]); self.p2 =
        nn.MaxPool3d(2)
        self.enc3 = ResBlock(f[1], f[2]); self.p3 =
        nn.MaxPool3d(2)
        self.bot = ResBlock(f[2], f[3])

        self.up3 = nn.Upsample(scale_factor=2, mode='trilinear',
        align_corners=True)
        self.att3 = AttentionBlock(f[3], f[2], f[2])
        self.dec3 = ResBlock(f[3]+f[2], f[2])

        self.up2 = nn.Upsample(scale_factor=2, mode='trilinear',
        align_corners=True)
        self.att2 = AttentionBlock(f[2], f[1], f[1])
        self.dec2 = ResBlock(f[2]+f[1], f[1])

```

```

        self.up1 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att1 = AttentionBlock(f[1], f[0], f[0])
        self.dec1 = ResBlock(f[1]+f[0], f[0])

        self.out = nn.Conv3d(f[0], out_c, 1)
        self.dropout = nn.Dropout3d(0.2)

    def forward(self, x):
        e1 = self.enc1(x); p1 = self.p1(e1)
        e2 = self.enc2(p1); p2 = self.p2(e2)
        e3 = self.enc3(p2); p3 = self.p3(e3)
        b = self.bot(p3)

        d3 = self.up3(b); x3 = self.att3(d3, e3); d3 =
self.dec3(torch.cat([x3, d3], 1))
        d2 = self.up2(d3); x2 = self.att2(d2, e2); d2 =
self.dec2(torch.cat([x2, d2], 1))
        d1 = self.up1(d2); x1 = self.att1(d1, e1); d1 =
self.dec1(torch.cat([x1, d1], 1))
        return self.out(self.dropout(d1))

# =====
# 3. HELPER FUNCTIONS
# =====

def normalize_data(data):
    """Normalisasi standar 0-1"""
    v_min, v_max = data.min(), data.max()
    if v_max - v_min > 0: return (data - v_min) / (v_max - v_min)
    return data

def calculate_cnr(vol):
    """Contrast-to-Noise Ratio"""
    flat = vol.flatten()
    thresh = np.mean(flat)
    sig = flat[flat > thresh]; bg = flat[flat <= thresh]
    if len(sig) < 5 or len(bg) < 5: return 0.0
    return abs(np.mean(sig) - np.mean(bg)) / (np.std(bg) + 1e-6)

# --- FUNGSI BOBOT LINEAR (PYRAMID) ---
def get_linear_weight_map(patch_size):
    """
    Membuat bobot berbentuk piramida.
    Tengah = 1.0 (Kuat), Pinggir = 0.0 (Lemah).
    Saat dijahit, bagian lemah akan digantikan oleh bagian kuat
    dari patch sebelahnya.
    Hasil: Mulus tanpa blur.
    """
    # 1D Linear ramp (0 -> 1 -> 0)
    vals = np.linspace(0, 1, patch_size)
    vals = np.minimum(vals, 1 - vals) * 2

    # Buat 3D Weight Map
    w_x, w_y, w_z = np.meshgrid(vals, vals, vals, indexing='ij')
    weight_map = w_x * w_y * w_z

    # Hindari nilai 0 absolut agar tidak error bagi 0

```

```

        return np.maximum(weight_map, 1e-4)

# =====
# 4. INFERENCE ENGINE (LINEAR BLENDING)
# =====
def predict_linear_blending(model, vol_in, patch_size,
overlap=0.5):
    model.eval()
    d, h, w = vol_in.shape
    stride = int(patch_size * (1 - overlap))

    # Padding
    pad_d = (stride - d % stride) % stride
    pad_h = (stride - h % stride) % stride
    pad_w = (stride - w % stride) % stride

    # Extra padding agar patch terakhir aman
    pad_extra = patch_size
    vol_padded = np.pad(vol_in, ((0, pad_d + pad_extra), (0,
pad_h + pad_extra), (0, pad_w + pad_extra)), mode='reflect')

    d_p, h_p, w_p = vol_padded.shape

    # Penampung Hasil
    output_sum = np.zeros_like(vol_padded)
    weight_sum = np.zeros_like(vol_padded)

    # Buat Weight Map
    patch_weight =
    torch.from_numpy(get_linear_weight_map(patch_size)).float().to(DE
VICE)

    print("⌚ Stitching with Linear Blending (Seamless) ...")

    with torch.no_grad():
        for z in range(0, d_p - patch_size, stride):
            for y in range(0, h_p - patch_size, stride):
                for x in range(0, w_p - patch_size, stride):

                    patch_in = vol_padded[z:z+patch_size,
y:y+patch_size, x:x+patch_size]

                    if patch_in.shape != (patch_size, patch_size,
patch_size): continue

                    t_in =
                    torch.from_numpy(patch_in).float().unsqueeze(0).unsqueeze(0).to(D
VICE)
                    t_out = model(t_in).squeeze() # (32, 32, 32)

                    # Akumulasi Terbobot
                    # Hasil = (Prediksi * Bobot)
                    # Pembagi = Total Bobot
                    output_sum[z:z+patch_size, y:y+patch_size,
x:x+patch_size] += (t_out * patch_weight).cpu().numpy()
                    weight_sum[z:z+patch_size, y:y+patch_size,
x:x+patch_size] += patch_weight.cpu().numpy()

```

```

# Normalisasi (Weighted Average)
reconstructed = output_sum / (weight_sum + 1e-8)

# Crop kembali ke ukuran asli
return reconstructed[:d, :h, :w]

# =====
# 5. MAIN EVALUATION LOOP
# =====
def run_evaluation():
    print(f"

```

```

data_range=1.0)
    metrics['RMSE'] = math.sqrt(np.mean((vol_gt -
vol_out)**2))

    cnr_in = calculate_cnr(vol_in)
    cnr_out = calculate_cnr(vol_out)
    cnr_gt = calculate_cnr(vol_gt) if vol_gt is not None else
0

    results_log.append({
        'Filename': filename,
        'PSNR': metrics['PSNR'], 'SSIM': metrics['SSIM'],
        'RMSE': metrics['RMSE'],
        'CNR_In': cnr_in, 'CNR_Out': cnr_out, 'CNR_GT':
cnr_gt,
        'CNR_Improv': cnr_out - cnr_in
    })

    print(f"    -> PSNR: {metrics['PSNR']:.2f} | SSIM:
{metrics['SSIM']:.4f} | RMSE: {metrics['RMSE']:.4f}")
    print(f"    -> CNR_Improv: {cnr_out - cnr_in:.4f}")

# --- VISUALIZATION ---
mid = vol_out.shape[0] // 2
fig = plt.figure(figsize=(20, 12))
gs = fig.add_gridspec(3, 4)

# Row 1: Images
ax1 = fig.add_subplot(gs[0, 0]); ax1.imshow(vol_in[mid],
cmap='gray'); ax1.set_title("Input (FBP)")
ax2 = fig.add_subplot(gs[0, 1]); ax2.imshow(vol_out[mid],
cmap='gray'); ax2.set_title("AI Output (Natural)")
ax3 = fig.add_subplot(gs[0, 2]);
if vol_gt is not None: ax3.imshow(vol_gt[mid],
cmap='gray'); ax3.set_title("Ground Truth")
else: ax3.axis('off')

# CNR Chart
ax_chart = fig.add_subplot(gs[0, 3])
ax_chart.bar(['In', 'Out', 'GT'], [cnr_in, cnr_out,
cnr_gt], color=['gray', 'blue', 'green'])
ax_chart.set_title("CNR Comparison");
ax_chart.grid(axis='y', linestyle='--', alpha=0.3)

# Row 2: Density Maps (Heatmap)
ax4 = fig.add_subplot(gs[1, 0]); im4 =
ax4.imshow(vol_in[mid], cmap='jet'); ax4.set_title("Density
(In)");
plt.colorbar(im4, ax=ax4)
ax5 = fig.add_subplot(gs[1, 1]); im5 =
ax5.imshow(vol_out[mid], cmap='jet'); ax5.set_title("Density
(Out)");
plt.colorbar(im5, ax=ax5)
ax6 = fig.add_subplot(gs[1, 2]);
if vol_gt is not None: im6 = ax6.imshow(vol_gt[mid],
cmap='jet'); ax6.set_title("Density (GT)");
plt.colorbar(im6, ax=ax6)
else: ax6.axis('off')

```

```

# Row 3: Error Map
ax7 = fig.add_subplot(gs[2, 1]);
err = np.abs(vol_gt - vol_out) if vol_gt is not None else
np.zeros_like(vol_out)
im7 = ax7.imshow(err[mid], cmap='inferno');
ax7.set_title(f"Error Map (RMSE: {metrics['RMSE']:.4f})");
plt.colorbar(im7, ax=ax7)

for ax in [ax1, ax2, ax3, ax4, ax5, ax6, ax7]:
ax.axis('off')
plt.tight_layout()
plt.savefig(os.path.join(OUTPUT_DIR,
f"Eval_Natural_{filename.replace('.npy','.png')}"))
plt.show()

if results_log:
    pd.DataFrame(results_log).to_csv(os.path.join(OUTPUT_DIR,
"Final_Report_Natural.csv"), index=False)
    print("\n✓ DONE.")
    print(pd.DataFrame(results_log) [['Filename', 'PSNR',
'SSIM', 'RMSE']].to_string())

if __name__ == "__main__":
    run_evaluation()

```

### Lampiran 6: Kode Python GUI

```

import streamlit as st
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats
import pydicom
import os
import io
import h5py # Wajib untuk file HDF5
from PIL import Image
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import structural_similarity as ssim

# --- LIBRARY FITUR TAMBAHAN ---
from streamlit_image_comparison import image_comparison
from fpdf import FPDF

# =====
# 1. KONFIGURASI HALAMAN
# =====
st.set_page_config(
    page_title="AI Medical CT Reconstruction",
    layout="wide",
    page_icon="",
    initial_sidebar_state="expanded"
)

```

```

# Custom CSS
st.markdown("""
<style>
    .main {background-color: #f4f6f9;}
    div.stButton > button {background-color: #007bff; color: white;}
    .metric-card {background-color: white; padding: 15px; border-radius: 8px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);}
</style>
""", unsafe_allow_html=True)

# =====
# 2. ARSITEKTUR MODEL
# =====

class ResBlock(nn.Module):
    def __init__(self, in_c, out_c):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv3d(in_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c), nn.ReLU(True),
            nn.Conv3d(out_c, out_c, 3, padding=1),
            nn.BatchNorm3d(out_c))
        self.relu = nn.ReLU(True)
        self.sc = nn.Conv3d(in_c, out_c, 1) if in_c != out_c else nn.Sequential()
    def forward(self, x): return self.relu(self.conv(x) + self.sc(x))

class AttentionBlock(nn.Module):
    def __init__(self, F_g, F_l, F_int):
        super().__init__()
        self.Wg = nn.Sequential(nn.Conv3d(F_g, F_int, 1),
            nn.BatchNorm3d(F_int))
        self.Wx = nn.Sequential(nn.Conv3d(F_l, F_int, 1),
            nn.BatchNorm3d(F_int))
        self.psi = nn.Sequential(nn.Conv3d(F_int, 1, 1),
            nn.BatchNorm3d(1), nn.Sigmoid())
        self.relu = nn.ReLU(True)
    def forward(self, g, x):
        psi = self.relu(self.Wg(g) + self.Wx(x))
        return x * self.psi(psi)

class Lightweight3DUNet(nn.Module):
    def __init__(self, in_c=1, out_c=1):
        super().__init__()
        f = [16, 32, 64, 128]
        self.enc1 = ResBlock(in_c, f[0]); self.p1 =
        nn.MaxPool3d(2)
        self.enc2 = ResBlock(f[0], f[1]); self.p2 =
        nn.MaxPool3d(2)
        self.enc3 = ResBlock(f[1], f[2]); self.p3 =
        nn.MaxPool3d(2)
        self.bot = ResBlock(f[2], f[3])
        self.up3 = nn.Upsample(scale_factor=2, mode='trilinear',
        align_corners=True)
        self.att3 = AttentionBlock(f[3], f[2], f[2]); self.dec3 =
        ResBlock(f[3]+f[2], f[2])

```

```

        self.up2 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att2 = AttentionBlock(f[2], f[1], f[1]); self.dec2 =
ResBlock(f[2]+f[1], f[1])
        self.up1 = nn.Upsample(scale_factor=2, mode='trilinear',
align_corners=True)
        self.att1 = AttentionBlock(f[1], f[0], f[0]); self.dec1 =
ResBlock(f[1]+f[0], f[0])
        self.out = nn.Conv3d(f[0], out_c, 1)
        self.dropout = nn.Dropout3d(0.2)
    def forward(self, x):
        e1 = self.enc1(x); p1 = self.p1(e1)
        e2 = self.enc2(p1); p2 = self.p2(e2)
        e3 = self.enc3(p2); p3 = self.p3(e3)
        b = self.bot(p3)
        d3 = self.up3(b); x3 = self.att3(d3, e3); d3 =
self.dec3(torch.cat([x3, d3], 1))
        d2 = self.up2(d3); x2 = self.att2(d2, e2); d2 =
self.dec2(torch.cat([x2, d2], 1))
        d1 = self.up1(d2); x1 = self.att1(d1, e1); d1 =
self.dec1(torch.cat([x1, d1], 1))
        return self.out(self.dropout(d1))

# =====
# 3. HELPER FUNCTIONS
# =====
@st.cache_resource
def load_model(model_path):
    device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
    model = Lightweight3DUNet().to(device)
    if os.path.exists(model_path):
        model.load_state_dict(torch.load(model_path,
map_location=device))
        model.eval()
        return model, device
    return None, device

# --- FUNGSI PROCESS_FILE YANG SUDAH DIPERBAIKI (MENGGUNAKAN TEMP
FILE) ---
def process_file(uploaded_file):
    if uploaded_file is None: return None
    name = uploaded_file.name.lower()
    data = None

    try:
        # 1. Handle DICOM
        if name.endswith('.dcm'):
            ds = pydicom.dcmread(uploaded_file)
            data = ds.pixel_array.astype(np.float32)
            if data.ndim == 2: data = np.expand_dims(data,
axis=0)

        # 2. Handle Image (JPG/PNG)
        elif name.endswith('.png', '.jpg', '.jpeg'):
            img = Image.open(uploaded_file).convert('L')
            data = np.array(img).astype(np.float32)
    
```

```

        data = np.expand_dims(data, axis=0)

        # 3. Handle NPY
        elif name.endswith('.npy'):
            data = np.load(uploaded_file).astype(np.float32)
            if data.ndim == 2: data = np.expand_dims(data,
axis=0)

        # 4. HANDLE HDF5 (INI YANG SUDAH DIPERBAIKI)
        elif name.endswith('.h5', '.hdf5')):
            # Trik: Simpan dulu ke file sementara (temp) agar
h5py bisa baca
            temp_filename = "temp_upload.h5"
            with open(temp_filename, "wb") as f:
                f.write(uploaded_file.getbuffer())

            # Baca file sementara
            with h5py.File(temp_filename, 'r') as f:
                # Cari key data secara otomatis (biasanya 'data'
atau 'image')
                keys = list(f.keys())
                # Prioritaskan nama 'data', kalau tidak ada ambil
key pertama apa saja
                key_name = 'data' if 'data' in keys else keys[0]

                # Load ke numpy
                data = np.array(f[key_name]).astype(np.float32)

            # Hapus file sementara (Wajib bersih-bersih)
            if os.path.exists(temp_filename):
                os.remove(temp_filename)

            if data.ndim == 2: data = np.expand_dims(data,
axis=0)

            # Normalisasi Akhir (0-1)
            if data is not None:
                v_min, v_max = data.min(), data.max()
                if v_max - v_min > 0:
                    data = (data - v_min) / (v_max - v_min)
                return data

        except Exception as e:
            st.error(f"Gagal memproses file: {e}")
            return None
    return None

def calculate_cnr(vol):
    flat = vol.flatten()
    thresh = np.mean(flat)
    sig = flat[flat > thresh]; bg = flat[flat <= thresh]
    if len(sig)<5 or len(bg)<5: return 0.0
    return abs(np.mean(sig) - np.mean(bg)) / (np.std(bg) + 1e-6)

def get_radiomics(vol):
    flat = vol.flatten()
    hist, _ = np.histogram(flat, bins=256, density=True)

```

```

entropy = -np.sum(hist * np.log2(hist + 1e-7))
return {
    'Mean': np.mean(flat),
    'Std Dev': np.std(flat),
    'Skewness': scipy.stats.skew(flat),
    'Kurtosis': scipy.stats.kurtosis(flat),
    'Entropy': entropy
}

def create_pdf_report(filename, cnr_in, cnr_out, rad_in,
rad_out):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", 'B', 16)
    pdf.cell(200, 10, txt="Laporan Analisis AI CT-Scan", ln=1,
align='C')
    pdf.ln(10)

    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt=f"Nama File Input: {filename}", ln=1)
    pdf.ln(5)

    # Tabel Metrik
    pdf.set_font("Arial", 'B', 10)
    pdf.cell(60, 10, "Parameter", 1)
    pdf.cell(40, 10, "Input (FBP)", 1)
    pdf.cell(40, 10, "Output (AI)", 1)
    pdf.ln()

    pdf.set_font("Arial", size=10)
    pdf.cell(60, 10, "CNR (Contrast)", 1)
    pdf.cell(40, 10, f"{cnr_in:.4f}", 1)
    pdf.cell(40, 10, f"{cnr_out:.4f}", 1)
    pdf.ln()

    for key in rad_in:
        pdf.cell(60, 10, key, 1)
        pdf.cell(40, 10, f"{rad_in[key]:.4f}", 1)
        pdf.cell(40, 10, f"{rad_out[key]:.4f}", 1)
        pdf.ln()

    return pdf.output(dest='S').encode('latin-1')

# =====
# 4. MAIN USER INTERFACE
# =====
st.sidebar.image("https://cdn-icons-png.flaticon.com/512/3063/3063176.png", width=60)
st.sidebar.title("Konfigurasi Sistem")

# Path Default (Pastikan file model ada)
default_path = r"D:\Crystaly UI\Sem 5 of BME\Pencit\Finpro\best_model_finetuned.pth"
model_path = st.sidebar.text_input("Path Model (.pth)", default_path)

st.sidebar.divider()

```

```

st.sidebar.subheader("Upload Data")
file_in = st.sidebar.file_uploader("1. Input Low-Dose",
type=['npy', 'dcm', 'jpg', 'png', 'h5', 'hdf5'])
file_gt = st.sidebar.file_uploader("2. Ground Truth (Optional)",
type=['npy', 'dcm', 'jpg', 'png', 'h5', 'hdf5'])

st.title("Medical CT Image Enhancement")

model, device = load_model(model_path)
if model is None:
    st.warning(f"⚠️ Model tidak ditemukan di: {model_path}. Pastikan path benar.")
    st.stop()

if file_in is not None:
    with st.spinner("🧠 AI sedang memproses citra..."):
        vol_in = process_file(file_in)
        vol_gt = process_file(file_gt)

        # Fake 3D
        is_fake_3d = False
        if vol_in.shape[0] == 1:
            vol_in = np.repeat(vol_in, 16, axis=0)
            is_fake_3d = True

        t_in = torch.from_numpy(vol_in).unsqueeze(0).unsqueeze(0)
        _, _, d, h, w = t_in.shape

        # Padding (gunakan pad_d agar aman)
        pad_d, pad_h, pad_w = (16 - d%16)%16, (16 - h%16)%16, (16 - w%16)%16
        if pad_d + pad_h + pad_w > 0:
            t_in = F.pad(t_in, (0, pad_w, 0, pad_h, 0, pad_d),
mode='replicate')

        t_in = t_in.to(device)
        with torch.no_grad():
            out = model(t_in).cpu().squeeze().numpy()

        out = out[:d, :h, :w]

        if vol_gt is not None:
            if is_fake_3d and vol_gt.shape[0] == 1: vol_gt =
np.repeat(vol_gt, 16, axis=0)
            min_d, min_h, min_w = min(vol_gt.shape, out.shape)
            vol_gt = vol_gt[:min_d, :min_h, :min_w]
            out = out[:min_d, :min_h, :min_w]
            vol_in = vol_in[:min_d, :min_h, :min_w]

# --- VISUALISASI ---
max_slice = out.shape[0] - 1
slice_idx = st.slider("🔍 Navigasi Slice", 0, max_slice,
max_slice/2)

st.subheader("👁️ Visualisasi Komparasi")
def to_uint8(img): return (img * 255).astype(np.uint8)

```

```

image_comparison(
    img1=to_uint8(vol_in[slice_idx]),
    img2=to_uint8(out[slice_idx]),
    label1="Input",
    label2="Output AI",
    width=700,
    show_labels=True,
    make_responsive=True,
    in_memory=True
)

# --- METRIK ---
col1, col2 = st.columns(2)
cnr_in = calculate_cnr(vol_in)
cnr_out = calculate_cnr(out)
col1.metric("CNR Input", f"{cnr_in:.2f}")
col2.metric("CNR Output", f"{cnr_out:.2f}",
delta=f"{cnr_out-cnr_in:.2f}")

rad_in = get_radiomics(vol_in)
rad_out = get_radiomics(out)

st.subheader("📊 Analisis Data")
tab1, tab2 = st.tabs(["Grafik & Tabel", "Download Laporan"])

with tab1:
    df_rad = pd.DataFrame({"Input": rad_in, "Output": rad_out})
    st.bar_chart(df_rad)
    st.dataframe(df_rad)

with tab2:
    if st.button("📄 Generate PDF Report"):
        pdf_bytes = create_pdf_report(file_in.name, cnr_in, cnr_out, rad_in, rad_out)
        st.download_button("📥 Klik Download PDF", data=pdf_bytes, file_name="Laporan_AI.pdf", mime='application/pdf')

else:
    st.info("👉 Upload file di sidebar kiri untuk memulai.")

```