

**TUGAS AKHIR - SM234801**

**KLASIFIKASI TUBERKULOSIS MENGGUNAKAN  
*FOCAL TRANSFORMER* PADA CITRA X-RAY  
DADA**

**MUHAMMAD ARIEF RACHMAN**

NRP 5002211164

Dosen Pembimbing

**Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

NIP 19690405 199403 2 003

**Program Studi Sarjana**

Departemen Matematika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2025





**TUGAS AKHIR - SM234801**

**KLASIFIKASI TUBERKULOSIS MENGGUNAKAN  
*FOCAL TRANSFORMER* PADA CITRA X-RAY  
DADA**

**MUHAMMAD ARIEF RACHMAN**

NRP 5002211164

Dosen Pembimbing

**Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

NIP 19690405 199403 2 003

**Program Sarjana**

Departemen Matematika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2025





**FINAL PROJECT - SM234801**

# **CLASSIFICATION OF TUBERCULOSIS USING FOCAL TRANSFORMER ON CHEST X-RAY IMAGES**

**MUHAMMAD ARIEF RACHMAN**

NRP 5002211164

Supervisor

**Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

NIP 19690405 199403 2 003

**Bachelor Program**

Department of Mathematics

Faculty of Sciences

Institut Teknologi Sepuluh Nopember

Surabaya

2025



# LEMBAR PENGESAHAN

## KLASIFIKASI TUBERKULOSIS MENGGUNAKAN FOCAL TRANSFORMER PADA CITRA X-RAY DADA

### TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat  
memperoleh gelar Sarjana Matematika pada  
Program Studi S-1 Matematika

Departemen Matematika  
Fakultas Sains dan Analitika Data  
Institut Teknologi Sepuluh Nopember

Oleh: MUHAMMAD ARIEF RACHMAN  
NRP. 5002211164

Surabaya, Juli 2025

Disetujui oleh Tim Penguji Tugas Akhir:

Pembimbing

1. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT  
NIP. 19690405 199403 2 003

Penguji

1. Dr. Budi Setiyono, S.Si, MT  
NIP. 19720207 199702 1 001

2. Dr. Dieky Adzkiya, S.Si, M.Si  
NIP. 19830517 200812 1 003

3. Mohammad Iqbal, S.Si, M.Si, Ph.D  
NIP. 19880105 201903 1 007



Mengetahui  
Dr. Didik Khusnul Arif, S.Si, M.Si  
NIP. 19730930 199702 1 001



## PERNYATAAN ORISINALITAS

Yang bertanda tangan disini:

Nama Mahasiswa / NRP : Muhammad Arief Rachman / 5002211164

Departemen : Matematika

Dosen Pembimbing / NIP : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT /  
19690405 199403 2 003

dengan ini menyatakan bahwa Tugas Akhir dengan judul "**KLASIFIKASI TUBERKULOSIS MENGGUNAKAN FOCAL TRANSFORMER PADA CITRA X-RAY DADA**" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 31 Juli 2025

Mengetahui  
Dosen Pembimbing

Dr. Dwi Ratna Sulistyaningrum, S.Si, MT  
NIP. 19690405 199403 2 003

Mahasiswa



Muhammad Arief Rachman  
NRP. 5002211164



# **ABSTRAK**

## **KLASIFIKASI TUBERKULOSIS MENGGUNAKAN *FOCAL TRANSFORMER* PADA CITRA X-RAY DADA**

Nama Mahasiswa / NRP : Muhammad Arief Rachman / 5002211164

Departemen : Matematika FSAD -ITS

Dosen Pembimbing : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

### **Abstrak**

Tuberkulosis (TB) merupakan salah satu penyakit menular paling mematikan di dunia, dan Indonesia termasuk negara dengan jumlah kasus tertinggi setiap tahunnya. Citra *X-ray* dada merupakan metode umum dalam diagnosis tuberkulosis. Namun, kemiripan visual antara paru-paru yang terinfeksi dan yang normal sering kali menyebabkan kesalahan klasifikasi. Untuk mengatasi tantangan tersebut, penelitian ini mengusulkan sistem diagnosis otomatis berbasis kecerdasan buatan untuk melakukan klasifikasi TB dari citra *X-ray* dada menggunakan model *Focal Transformer*. Model ini digunakan karena kemampuannya dalam menangkap hubungan spasial lokal dan global melalui mekanisme *focal self-attention*, yang bekerja pada dua tingkat perhatian: tingkat halus untuk area terdekat dan tingkat kasar untuk area yang lebih jauh. Pendekatan ini membantu model membedakan fitur yang merupakan karakteristik utama TB dari kondisi paru normal yang secara visual dapat tampak serupa, dengan mengenali detail-detail lokal pada citra serta memahami konteks spasial secara menyeluruh. Penelitian ini menggunakan tiga varian *Focal Transformer*, yakni *Focal-Tiny*, *Focal-Small*, dan *Focal-Base*, yang dilatih pada dataset *Kaggle Combined Unknown Pneumonia and Tuberculosis* dengan dua kategori: tuberkulosis dan normal. Pengujian dilakukan pada citra tanpa dan dengan gangguan untuk mengevaluasi ketahanan model. *Focal-Base* menunjukkan performa terbaik di antara ketiganya dengan akurasi 93,47% pada citra tanpa gangguan; 90,09% pada citra dengan *Gaussian Noise*; 91,08% pada *Salt and Pepper*; 92,57% pada *Motion Blur*; 80,67% pada *Brightness and Contrast Distortion*; dan 91,20% pada *Partial Occlusion*.

**Kata kunci:** *Tuberkulosis, Focal Transformer, X-Ray dada.*



## ABSTRACT

### CLASSIFICATION OF TUBERCULOSIS USING FOCAL TRANSFORMER ON CHEST X-RAY IMAGES

Student Name / NRP : Muhammad Arief Rachman / 5002211164

Department : Mathematics SCIENTICS - ITS

Advisor : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

#### **Abstract**

Tuberculosis (TB) is one of the most deadly infectious diseases in the world, and Indonesia is among the countries with the highest number of cases each year. Chest X-ray imaging is a common method used for diagnosing tuberculosis. However, the visual similarity between infected and normal lungs often leads to classification errors. To address this challenge, this study proposes an automated diagnosis system based on artificial intelligence to classify TB from chest X-ray images using the Focal Transformer model. This model was chosen for its ability to capture both local and global spatial relationships through a focal self-attention mechanism, which operates on two levels of attention: fine-grained for nearby regions and coarse-grained for distant regions. This approach enables the model to distinguish features that are key characteristics of TB from visually similar normal lung conditions by identifying local image details and understanding the overall spatial context. This study utilizes three variants of the Focal Transformer—Focal-Tiny, Focal-Small, and Focal-Base—which were trained on the Kaggle Combined Unknown Pneumonia and Tuberculosis dataset with two categories: tuberculosis and normal. The models were tested on both clean and distorted images to evaluate their robustness. Among the three, Focal-Base achieved the highest performance, with an accuracy of 93.47% on clean images; 90.09% on images with Gaussian Noise; 91.08% on Salt and Pepper noise; 92.57% on Motion Blur; 80.67% on Brightness and Contrast Distortion; and 91.20% on Partial Occlusion.

**Keywords:** *Tuberculosis, Focal Transformer, Chest X-Ray.*



## KATA PENGANTAR

Puji syukur kehadirat Allah SWT karena atas berkah, rahmat dan ridho-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul :

### "KLASIFIKASI TUBERKULOSIS MENGGUNAKAN *FOCAL TRANSFORMER* PADA CITRA X-RAY DADA"

sebagai salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember Surabaya. Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis ingin mengucapkan terima kasih dan penghargaan kepada :

1. Orang tua yang telah memberikan dukungan, nasehat, dan motivasi sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Bapak Kepala Departemen Matematika Institut Teknologi Sepuluh Nopember (ITS) yang telah membantu dan mendukung kami semasa berkuliahan di departemen Matematika ITS.
3. Bapak Amirul Hakam, S.Si, M.Si selaku dosen wali yang telah memberikan arahan kepada penulis selama masa perkuliahan.
4. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku dosen pembimbing yang telah memberikan arahan kepada penulis.
5. Bapak Dr. Budi Setiyono, S.Si, MT, Bapak Dr. Dieky Adzkiya, S.Si, M.Si, dan Bapak Mohammad Iqbal, S.Si, M.Si, Ph.D. selaku dosen penguji yang telah memberikan arahan, saran, dan masukan dalam penyusunan Tugas Akhir ini.
6. Bapak/ibu dosen pengajar yang tidak bisa penulis sebutkan satu per satu, yang telah memberikan ilmu bermanfaat kepada penulis, serta segenap karyawan, tendik, dan keluarga besar Departemen Matematika Institut Teknologi Sepuluh Nopember atas dukungan dan bantuannya.
7. Teman-teman Matematika ITS angkatan 2021 yang telah banyak memberikan pengalaman, cerita, dan kekeluargaan kepada penulis mulai dari awal perkuliahan hingga penulis dapat menyelesaikan Tugas Akhir.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Juli 2025

Muhammad Arief Rachman



## DAFTAR ISI

LEMBAR PENGESAHAN	i
PERNYATAAN ORISINALITAS	iii
ABSTRAK	v
ABSTRACT	vii
KATA PENGANTAR	ix
DAFTAR ISI	xi
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xv
BAB I PENDAHULUAN	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Batasan Masalah .....	3
1.4 Tujuan .....	3
1.5 Manfaat .....	3
BAB II TINJAUAN PUSTAKA	5
2.1 Penelitian Terdahulu .....	5
2.2 Penyakit Tuberkulosis .....	6
2.3 Pengolahan Citra Digital .....	7
2.4 <i>Deep Learning</i> .....	10
2.5 Transformer .....	11
2.6 Focal Transformer .....	14
2.6.1 <i>Focal Self-Attention</i> .....	15
2.6.2 <i>Window-wise Attention</i> .....	17
2.7 <i>Global Average Pooling</i> .....	18
2.8 Metrik Evaluasi .....	19
BAB III METODOLOGI	21
3.1 Dataset .....	21
3.2 Pra-Pemrosesan Citra .....	22
3.2.1 <i>Cropping</i> .....	22
3.2.2 Augmentasi .....	22
3.2.3 <i>Resize</i> .....	22
3.2.4 Normalisasi .....	23
3.3 Pembagian Data .....	23
3.4 Pelatihan Model .....	23
3.5 Validasi Model .....	24
3.6 Pengujian dan Analisis Hasil Model .....	24
3.7 Penarikan Kesimpulan dan Saran .....	24

BAB IV	PERANCANGAN DAN IMPLEMENTASI	25
4.1	Pengambilan Data .....	25
4.2	Pra-pemrosesan Citra .....	26
4.3	Pelatihan Model .....	36
4.4	Pengujian Model .....	48
4.5	Perancangan Sistem .....	50
4.5.1	Perancangan Pelatihan Model .....	50
4.5.2	Perancangan Pengujian Model .....	50
4.6	Implementasi Program .....	51
4.6.1	Implementasi Pelatihan Model .....	51
4.6.2	Implementasi Pengujian Model .....	52
BAB V	HASIL DAN PEMBAHASAN	53
5.1	Hasil Pelatihan .....	53
5.1.1	Pelatihan Pertama Model <i>Focal-Tiny</i> .....	53
5.1.2	Pelatihan Kedua Model <i>Focal-Tiny</i> .....	54
5.1.3	Pelatihan Pertama Model <i>Focal-Small</i> .....	56
5.1.4	Pelatihan Kedua Model <i>Focal-Small</i> .....	57
5.1.5	Pelatihan Pertama Model <i>Focal-Base</i> .....	58
5.1.6	Pelatihan Kedua Model <i>Focal-Base</i> .....	59
5.2	Hasil Pengujian .....	60
5.2.1	Hasil Pengujian Model <i>Focal Transformer</i> pada Citra Tanpa Gangguan .....	60
5.2.2	Hasil Pengujian Model dengan Kinerja Tertinggi pada Citra Gangguan .....	63
BAB VI	KESIMPULAN DAN SARAN	67
6.1	Kesimpulan .....	67
6.2	Saran .....	67
	UCAPAN TERIMA KASIH	73
	BIODATA PENULIS	75

## DAFTAR GAMBAR

Gambar 2.1	Perbedaan Karakteristik Citra <i>X-ray</i> : (a) Normal, (b) Tuberkulosis	7
Gambar 2.2	Representasi Piksel dalam Citra . . . . .	8
Gambar 2.3	Struktur Model <i>Transformer</i> . . . . .	12
Gambar 2.4	Arsitektur <i>Focal Transformer</i> . . . . .	14
Gambar 2.5	Ukuran Lapangan Reseptif . . . . .	15
Gambar 2.6	Ilustrasi <i>Focal Self-Attention</i> . . . . .	16
Gambar 3.1	Blok Diagram Penelitian . . . . .	21
Gambar 3.2	Alur Pelatihan Model . . . . .	23
Gambar 4.1	Contoh Citra Dataset: (a) dan (b) Citra Kategori Normal, (c) dan (d) Citra Kategori Tuberkulosis . . . . .	25
Gambar 4.2	Contoh Proses <i>Cropping</i> : (a) Citra Sebelum <i>Cropping</i> , (b) Citra Sesudah <i>Cropping</i> . . . . .	28
Gambar 4.3	Contoh Proses <i>Horizontal Flip</i> : (a) Citra Sebelum <i>Horizontal Flip</i> , (b) Citra Sesudah <i>Horizontal Flip</i> . . . . .	29
Gambar 4.4	Contoh Proses Rotasi: (a) Citra Sebelum Rotasi, (b) Citra Sesudah Rotasi . . . . .	30
Gambar 4.5	Contoh Proses Pergeseran: (a) Citra Sebelum Pergeseran, (b) Citra Sesudah Pergeseran . . . . .	31
Gambar 4.6	Contoh Proses Skalasi: (a) Citra Sebelum Skalasi, (b) Citra Sesudah Skalasi . . . . .	32
Gambar 4.7	Contoh Proses <i>Resize</i> : (a) Citra Sebelum <i>Resize</i> , (b) Citra Sesudah <i>Resize</i> . . . . .	35
Gambar 5.1	Grafik Pelatihan Pertama Model <i>Focal-Tiny</i> . . . . .	54
Gambar 5.2	Grafik Pelatihan Kedua Model <i>Focal-Tiny</i> . . . . .	55
Gambar 5.3	Grafik Pelatihan Pertama Model <i>Focal-Small</i> . . . . .	56
Gambar 5.4	Grafik Pelatihan Kedua Model <i>Focal-Small</i> . . . . .	57
Gambar 5.5	Grafik Pelatihan Pertama Model <i>Focal-Base</i> . . . . .	58
Gambar 5.6	Grafik Pelatihan Kedua Model <i>Focal-Base</i> . . . . .	59
Gambar 5.7	<i>Confusion Matrix Focal Transformer</i> pada Citra Tanpa Gangguan: (a) <i>Focal-Tiny</i> , (b) <i>Focal-Small</i> , (c) <i>Focal-Base</i> . . . . .	61
Gambar 5.8	<i>Confusion Matrix Focal-Base</i> pada Citra dengan Gangguan: (a) <i>Gaussian Noise</i> , (b) <i>Salt and Pepper Noise</i> , (c) <i>Motion Blur</i> , (d) <i>Brightness and Contrast Distortion</i> , (e) <i>Partial Occlusion</i> . . . . .	64



## DAFTAR TABEL

Tabel 2.1	Penelitian Terdahulu .....	6
Tabel 2.2	Matriks Konfusi .....	19
Tabel 4.1	Deskripsi dan Contoh Kondisi Citra .....	49
Tabel 4.2	Variasi Model <i>Focal Transformer</i> .....	50
Tabel 4.3	<i>Hyperparameter</i> Pelatihan .....	50
Tabel 4.4	Pengaturan Kondisi Citra .....	51
Tabel 5.1	<i>Hyperparameter</i> Pelatihan Pertama Model <i>Focal-Tiny</i> .....	53
Tabel 5.2	<i>Hyperparameter</i> Pelatihan Kedua Model <i>Focal-Tiny</i> .....	55
Tabel 5.3	<i>Hyperparameter</i> Pelatihan Pertama Model <i>Focal-Small</i> .....	56
Tabel 5.4	<i>Hyperparameter</i> Pelatihan Kedua Model <i>Focal-Small</i> .....	57
Tabel 5.5	<i>Hyperparameter</i> Pelatihan Pertama Model <i>Focal-Base</i> .....	58
Tabel 5.6	<i>Hyperparameter</i> Pelatihan Kedua Model <i>Focal-Base</i> .....	59
Tabel 5.7	Kinerja <i>Focal Transformer</i> pada Citra Tanpa Gangguan .....	60
Tabel 5.8	Kinerja <i>Focal-Base</i> pada Citra dengan Gangguan .....	63



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Penyakit tuberkulosis sampai saat ini masih menjadi permasalahan kesehatan global yang penting, terutama di negara berkembang dengan keterbatasan sistem pelayanan kesehatan. Berdasarkan laporan terbaru dari *World Health Organization* (WHO), Penyakit ini menempati posisi ke-13 sebagai penyebab utama kematian di dunia dan menjadi penyakit menular dengan angka kematian tertinggi setelah penyakit *COVID-19*. Pada tahun 2021, WHO melaporkan bahwa adanya 10,6 juta kasus tuberkulosis dengan angka kematian mencapai 1,3 juta jiwa di seluruh dunia. Indonesia menempati posisi kedua sebagai negara dengan jumlah kasus tuberkulosis tertinggi setelah India, dengan kecenderungan peningkatan jumlah penderita setiap tahunnya (World Health Organization, 2022). Tingginya angka kasus tersebut menunjukkan bahwa penyakit ini masih menjadi permasalahan kesehatan yang memerlukan perhatian khusus. Oleh karena itu, diagnosis dini merupakan langkah yang penting dalam upaya pengendalian penyebaran penyakit tuberkulosis serta meningkatkan keberhasilan pengobatan. Diagnosis yang cepat dan akurat memungkinkan pasien mendapatkan penanganan yang tepat lebih awal, sehingga dapat mengurangi risiko komplikasi serta mengurangi tingkat penularan di masyarakat (Abd El-Ghany et al., 2024).

Diagnosis tuberkulosis dapat dilakukan dengan beberapa metode, seperti pemeriksaan mikroskopis sputum, uji kultur bakteri, metode molekuler *GeneXpert*, dan citra *X-ray* dada. Pemeriksaan mikroskopis sputum umum digunakan karena biayanya yang terjangkau, tetapi memiliki sensitivitas rendah pada pasien dengan kondisi jumlah bakteri yang sedikit. Uji kultur bakteri, sebagai standar acuan diagnosis tuberkulosis, membutuhkan waktu 2 minggu untuk hasil yang valid. Sementara itu, metode *GeneXpert* lebih cepat dan sensitif, tetapi mahal dan sulit diakses di daerah terpencil (World Health Organization, 2022). *X-ray* dada sering digunakan sebagai metode pendukung karena mampu mengidentifikasi pola kelainan paru akibat tuberkulosis, namun pemindaian *X-ray* yang menunjukkan gejala tuberkulosis sering kali disalahartikan sebagai kondisi paru normal yang memiliki karakteristik visual serupa, yang dapat menyebabkan kesalahan diagnosis dan pengobatan yang salah, sehingga dapat memperburuk kondisi pasien (Abd El-Ghany et al., 2024). Oleh karena itu, pendekatan berbasis *artificial intelligence* diharapkan dapat membantu pembacaan *X-ray* dada secara lebih presisi, sehingga dapat mendukung pengambilan keputusan medis yang lebih tepat.

*Artificial Intelligence* (AI), khususnya *deep learning*, telah berperan penting dalam analisis citra medis menggunakan, khususnya dalam membantu sistem komputer mengenali pola dari data berukuran besar untuk diagnosis yang lebih cepat dan akurat dibandingkan metode konvensional. Salah satu model AI berbasis *deep learning* yang banyak digunakan adalah *Convolutional Neural Network* (CNN), yang telah diterapkan dalam berbagai penelitian untuk mengklasifikasikan pola kelainan tuberkulosis pada *X-ray* dada dengan hasil yang cukup baik (Hansun et al., 2023). Namun, CNN memiliki keterbatasan dalam memahami hubungan spasial global karena hanya mengekstrak fitur lokal melalui lapisan konvolusi, serta kurang efektif dalam menangani variasi ukuran, orientasi, dan kontras rendah pada pola kelainan tuberkulosis (Kumar et al., 2025). Oleh karena itu, diperlukan model yang lebih fleksibel dan mampu menangkap hubungan spasial dalam skala lebih luas untuk meningkatkan akurasi klasifikasi citra *X-ray* dada.

Salah satu arsitektur yang dikembangkan untuk mengatasi keterbatasan tersebut adalah *Focal Transformer*, yang dirancang untuk memahami pola kompleks dan hubungan spasial global secara lebih efisien. Arsitektur ini mengimplementasikan *focal self-attention* untuk menggabungkan interaksi lokal dan global (Yang et al., 2021). Melalui pendekatan ini, setiap token citra memperhatikan token-token terdekatnya secara halus (*fine-grained*) di area lokal, serta token yang lebih jauh dengan perhatian global yang lebih kasar (*coarse-grained*). Mekanisme ini memungkinkan model menangkap ketergantungan jangka pendek dan panjang secara efektif. Berbeda dengan *Vision Transformer* yang menggunakan mekanisme *self-attention* pada seluruh bagian citra, *Focal Transformer* dapat melakukan interaksi antar token dengan berbagai tingkat kehalusan pada setiap lapisan transformernya, sehingga menciptakan keseimbangan antara pemrosesan lokal dan global yang lebih cepat dan akurat. Dengan demikian, *Focal Transformer* dapat memberikan performa yang lebih baik dibandingkan dengan *Vision Transformer* dalam tugas seperti klasifikasi citra (Yang et al., 2021).

Oleh karena itu, pada Tugas Akhir ini, dilakukan penerapan *Focal Transformer* untuk klasifikasi penyakit tuberkulosis berbasis citra *X-ray* dada. Penelitian ini diharapkan dapat memberikan kontribusi dalam menciptakan sistem klasifikasi citra penyakit tuberkulosis berbasis *deep learning* yang efektif, serta mendukung penanganan yang cepat dan akurat di fasilitas kesehatan.

## **1.2 Rumusan Masalah**

Dari latar belakang penelitian Tugas Akhir ini, dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana penerapan *Focal Transformer* untuk klasifikasi penyakit tuberkulosis berdasarkan citra *X-ray* dada?
2. Bagaimana kinerja dari penerapan *Focal Transformer* untuk klasifikasi penyakit tuberkulosis berdasarkan citra *X-ray* dada?

## **1.3 Batasan Masalah**

Adapun batasan masalah dari penelitian ini adalah sebagai berikut:

1. Penelitian ini menggunakan dataset *Chest X-ray* dari Kaggle (*Combined Unknown Pneumonia and Tuberculosis*).
2. Model melakukan klasifikasi dengan dua kelas: Tuberkulosis dan Normal.
3. Klasifikasi hanya dilakukan pada dataset yang ada tanpa memperhatikan faktor klinis lainnya, seperti riwayat pasien atau hasil tes pendukung.

## **1.4 Tujuan**

Berdasarkan permasalahan yang sudah dirumuskan, penilitian ini bertujuan untuk:

1. Menerapkan model *Focal Transformer* untuk klasifikasi penyakit tuberkulosis berdasarkan citra *X-ray* dada.
2. Menganalisis kinerja dari penerapan *Focal Transformer* untuk klasifikasi penyakit tuberkulosis berdasarkan citra *X-ray* dada.

## **1.5 Manfaat**

Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Memberikan kontribusi dalam pengembangan kajian ilmiah terkait penerapan model *Focal Transformer* untuk klasifikasi citra medis, serta memberikan pemahaman mengenai potensi kecerdasan buatan dalam meningkatkan akurasi dan efisiensi proses diagnosis penyakit.
2. Memberikan manfaat bagi masyarakat melalui pengembangan sistem klasifikasi tuberkulosis berbasis *deep learning* yang dapat mendukung proses diagnosis secara lebih cepat dan akurat, sehingga membantu mempercepat penanganan serta menekan penyebaran penyakit di lingkungan sekitar.



## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Penelitian Terdahulu

Penelitian oleh Rahman et al. 2020 yang berjudul "*Reliable Tuberculosis Detection Using Chest X-ray With Deep Learning, Segmentation and Visualization*" bertujuan untuk klasifikasi citra tuberkulosis secara akurat dan cepat melalui citra *X-ray* dada menggunakan model *deep learning*, segmentasi citra, dan visualisasi. Penelitian ini mengumpulkan dataset berisi 3.500 citra *X-ray* dada dari pasien TB dan 3.500 citra *X-ray* normal untuk digunakan dalam pelatihan model. Dalam eksperimen ini, penulis menggunakan sembilan model CNN yang telah dilatih sebelumnya seperti ResNet, ChexNet, dan DenseNet untuk mengidentifikasi pola kelainan TB dari citra *X-ray* dada yang telah disegmentasi dan yang belum disegmentasi. Hasil terbaik diperoleh dengan menggunakan DenseNet201 pada citra yang telah disegmentasi, dengan akurasi mencapai 99,9%, sementara ChexNet menunjukkan performa terbaik untuk citra tanpa segmentasi dengan akurasi 97,07%. Penelitian ini juga menunjukkan bahwa teknik segmentasi paru pada gambar rontgen meningkatkan akurasi klasifikasi secara signifikan.

Penelitian lain yang dilakukan oleh Hamza & Jesser 2023 yang berjudul "*A Hierarchical Vision Approach for Enhanced Medical Diagnostics of Lung Tuberculosis Using Swin Transformer*" juga membahas terkait klasifikasi citra tuberkulosis dan normal. Penelitian ini menggunakan dataset dari National Institutes of Health (NIH), yang terdiri dari 6.635 citra *X-ray*, dengan 1.557 citra kategori normal dan 3.498 citra kategori tuberkulosis. Hasil yang dicapai dalam penelitian ini berupa akurasi sebesar 88%, meskipun hasil akurasinya lebih rendah dibandingkan pendekatan CNN dan Transformer sebelumnya, penelitian ini menunjukkan bahwa Swin Transformer dapat digunakan sebagai pendekatan utama dalam klasifikasi citra tuberkulosis berbasis *X-ray* dada.

Dalam Penelitian oleh Yang et al. 2021, yang berjudul "*Focal Self-attention for Local-Global Interactions in Vision Transformers*", para penulis mengusulkan *Focal Transformer*, yang memperkenalkan mekanisme *self-attention* baru dengan kemampuan menggabungkan interaksi lokal halus dan global kasar. Pendekatan ini mengatasi tantangan komputasi yang biasanya terkait dengan tugas citra beresolusi tinggi seperti deteksi objek. Model *Focal Transformer* telah menunjukkan performa yang lebih baik dalam berbagai pengujian standar klasifikasi citra, khususnya pada dataset ImageNet. Pada dataset ini, model tersebut melampaui model Vision Transformer (ViT) lainnya, seperti Swin Transformer, serta model CNN. Secara khusus, *Focal Transformer* dengan

89,8 juta parameter mencapai Top-1 akurasi sebesar 83,8%, mengalahkan ResNet-152 yang memperoleh akurasi 78,3%, ViT-Base yang memperoleh akurasi 77,9%, dan Swin-Base yang memperoleh skor 83,4%. Hasil ini menunjukkan efisiensi dan efektivitas *Focal Transformer* dalam memodelkan interaksi lokal dan global pada tugas visi komputer beresolusi tinggi. Ringkasan perbandingan terkait penelitian terdahulu ditunjukkan pada Tabel 2.1

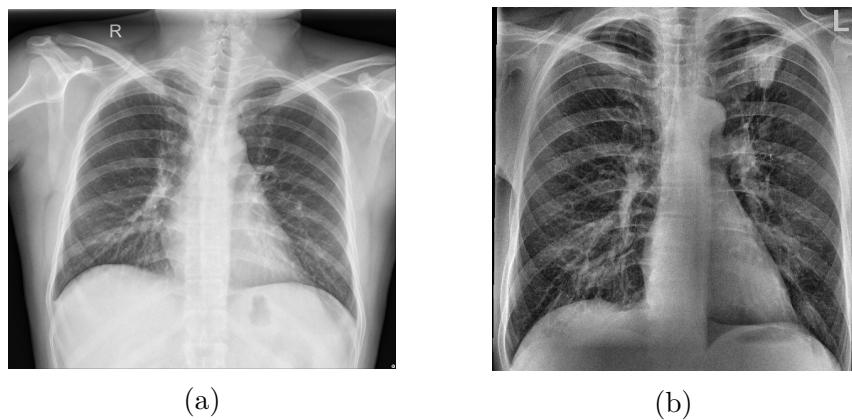
Tabel 2.1 Penelitian Terdahulu

Penulis	Judul	Metode	Dataset	Hasil
Tawsifur Rahman, dkk. (2020)	<i>Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization</i>	ResNet18, ResNet50, ResNet101, DenseNet201, ChexNet, SqueezeNet, InceptionV3, VGG19, MobileNetV2	NLM + Belarus + RSNA Dataset	DenseNet201 mencapai akurasi 99,9% (Segemented), sementara ChexNet 97,07% (Unsegemented)
Syed Amir Hamza dan Alexander Jesser. (2023)	<i>A Hierarchical Vision Approach for Enhanced Medical Diagnostics of Lung Tuberculosis Using Swin Transformer</i>	Swin Transformer	NIH Chest X-ray Dataset	Hasil yang dicapai akurasi sebesar 88%
Jianwei Yang, dkk. (2021)	<i>Focal Self-attention for Local-Global Interactions in Vision Transformers</i>	Focal Transformer	ImageNet-1K	Focal Transformer dengan 89,8 juta parameter mencapai Top-1 akurasi sebesar 83,8%
Rachman, M Arief. (2025)	<i>Classification of Tuberculosis Using Focal Transformer on Chest X-ray Images</i>	Focal Transformer	Kermany + RSNA + NIAID + NLM + Belarus Dataset	Focal-Tiny mencapai akurasi 92,95%, Focal-Small 92,23%, dan Focal-Base 93,47%

## 2.2 Penyakit Tuberkulosis

Tuberkulosis adalah penyakit infeksi menular yang disebabkan oleh bakteri bernama *Mycobacterium tuberculosis*. Penyakit ini sering kali menyerang organ paru-paru melalui udara saat penderita batuk atau bersin (World Health Organization, 2022). Tuberkulosis dapat menyebabkan berbagai komplikasi serius jika tidak terdeteksi dan diobati dengan cepat. Terdapat beberapa gejala tuberkulosis meliputi batuk kronis yang berlangsung lebih dari dua minggu, nyeri dada, penurunan berat badan yang signifikan, kelelahan, dan keringat malam (WHO, 2025). Meskipun pemeriksaan dahak dan kultur bakteri sering digunakan untuk diagnosis, citra medis seperti *X-ray* dada menjadi alat penting dalam mengidentifikasi karakteristik tuberkulosis pada paru-paru.

Paru-paru pada pasien tuberkulosis memiliki beberapa karakteristik khas yang membedakannya dari paru-paru pada pasien normal. Pada pasien tuberkulosis, umumnya ditemukan pola kelainan, penumpukan zat-zat abnormal, endapan kalsium, dan pembentukan lubang abnormal di area paru, terutama pada lobus atas. Citra *X-ray* pada pasien tuberkulosis juga sering menunjukkan penebalan selaput dan peningkatan kepadatan jaringan, yang menunjukkan adanya infeksi aktif atau kronis. Sementara itu, paru-paru yang sehat akan tampak dengan struktur yang lebih homogen, tanpa adanya area putih pekat atau abnormalitas lainnya (Bhalla et al., 2015). Perbedaan karakteristik kondisi tersebut ditunjukkan pada Gambar 2.1.



Gambar 2.1 Perbedaan Karakteristik Citra *X-ray*: (a) Normal, (b) Tuberkulosis  
(Majumder, 2024)

### 2.3 Pengolahan Citra Digital

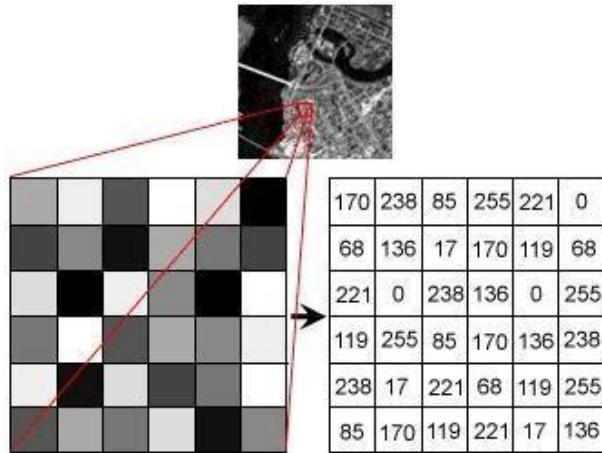
Pengolahan citra digital adalah proses manipulasi citra dalam bentuk digital yang bertujuan untuk meningkatkan kualitas, memperjelas informasi, atau mengekstrak fitur penting yang terdapat pada citra tersebut. Pada bidang medis, pengolahan citra digital memiliki peran penting karena dapat membantu memperbaiki kualitas citra *X-ray* agar model dapat mengenali pola atau kelainan dengan lebih tepat (Gonzalez & Woods, 2018). Beberapa contoh pengolahan citra digital yang akan diterapkan pada penelitian ini mencakup *cropping*, *resize*, dan *normalisasi*.

Citra digital akan direpresentasikan dalam bentuk piksel, yang merupakan unit terkecil dalam sebuah citra. Setiap piksel memiliki nilai intensitas yang menunjukkan warna atau kecerahan pada titik tertentu dalam sebuah citra. Dalam citra *X-ray*, intensitas piksel menggambarkan tingkat kegelapan dan kecerahan dari area paru-paru, dengan nilai piksel lebih tinggi menandakan area yang lebih terang, sementara nilai piksel yang lebih rendah menunjukkan area yang lebih gelap (Gonzalez & Woods, 2008). Dari penjelasan tersebut, maka citra *X-ray* akan direpresentasikan dalam skala *grayscale* dengan nilai intensitas dalam rentang antara 0 hingga 255, di mana 0 menunjukkan hitam dan 255 menunjukkan putih.

Secara matematis, Citra digital dapat didefinisikan sebagai fungsi  $f(x, y)$  yang memiliki ukuran  $H$  baris dan  $N$  kolom, dengan  $x$  dan  $y$  sebagai koordinat spasial citra, dan  $f(x, y)$  sebagai nilai intensitas citra pada koordinat tertentu. Sebagai contoh, untuk citra berukuran  $224 \times 224$  piksel dapat dinyatakan dalam bentuk matriks berukuran  $M \times N$  yang dijelaskan dalam Persamaan 2.1 dan Gambar 2.2 sebagai berikut

$$f(x_i, y_i) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1, 0) & f(M-1, 1) & \cdots & f(M-1, N-1) \end{bmatrix} \quad (2.1)$$

dengan  $i = 0, \dots, M-1, j = 0, \dots, N-1$  (Gonzalez, 2009). Proses pengolahan citra dimulai dengan representasi citra tersebut dalam bentuk matriks, yang kemudian dapat diproses lebih lanjut melalui berbagai cara, seperti *cropping*, *resize* dan normalisasi.



Gambar 2.2 Representasi Piksel dalam Citra  
(Das et al., 2016)

Cropping citra merupakan proses pemotongan citra digital untuk menghilangkan bagian-bagian yang tidak relevan atau kosong, sehingga hanya tersisa area utama yang berisi informasi penting. Pada citra berformat RGB, proses *cropping* dilakukan dengan Persamaan 2.2 sebagai berikut:

$$M(i, j) = \begin{cases} 1, & \text{jika } \max_{c \in \{1, 2, 3\}} I(i, j, c) > 20 \\ 0, & \text{lainnya} \end{cases} \quad (2.2)$$

artinya jika ada minimal satu kanal warna pada piksel  $(i, j)$  bernilai lebih dari 20, maka  $M(i, j) = 1$ . Jika ketiga kanal memiliki nilai rentang 0 hingga 20, maka  $M(i, j) = 0$ .

Tahap berikutnya adalah *resize* citra merupakan proses penting dalam pengolahan citra digital yang digunakan untuk menyesuaikan dimensi citra agar sesuai dengan input yang dibutuhkan dalam model *deep learning*. Proses ini bertujuan untuk mengubah ukuran citra dengan menggunakan metode interpolasi, dengan salah satu metode yang sering digunakan adalah interpolasi bilinear. Metode ini bekerja dengan memperkirakan nilai piksel baru pada citra yang di *resize* berdasarkan nilai empat piksel terdekat dari citra asli. Hal ini sangat penting dalam konteks klasifikasi citra tuberkulosis, di mana model *deep learning* memerlukan citra dengan dimensi yang konsisten, seperti  $224 \times 224$  piksel, agar dapat memproses citra dengan optimal.

Pada interpolasi bilinear, perubahan ukuran citra dilakukan dengan dua langkah interpolasi, baik secara horizontal maupun vertikal. Proses ini melibatkan perhitungan skala perubahan pada sumbu  $x$  dan  $y$  yang disebut sebagai  $x_{\text{scale}}$  dan  $y_{\text{scale}}$ , masing-masing untuk arah horizontal dan vertikal. Setiap posisi baru  $(x', y')$  pada citra yang di *resize* dihitung menggunakan Persamaan 2.3 sebagai berikut (Gonzalez, 2009):

$$A_{\text{resize}}(x', y') = w_{00} \cdot P_{00} + w_{10} \cdot P_{10} + w_{01} \cdot P_{01} + w_{11} \cdot P_{11} \quad (2.3)$$

dengan,

$x_{\text{scale}}$  = Skala perubahan ukuran pada sumbu  $x$

$y_{\text{scale}}$  = Skala perubahan ukuran pada sumbu  $y$

$x$  = Koordinat  $x$  terdekat pada citra asli yang sesuai dengan  $x' = [x' \cdot x_{\text{scale}}]$

$y$  = Koordinat  $y$  terdekat pada citra asli yang sesuai dengan  $y' = [y' \cdot y_{\text{scale}}]$

$\alpha$  = Jarak relatif antara  $x'$  dan  $x = x' - x_{\text{scale}}$

$\beta$  = Jarak relatif antara  $y'$  dan  $y = y' - y_{\text{scale}}$

$P_{00}$  = Nilai piksel pada citra asli di koordinat  $(x, y) = A(x, y)$

$P_{01}$  = Nilai piksel pada citra asli di koordinat  $(x, y + 1) = A(x, y + 1)$

$P_{10}$  = Nilai piksel pada citra asli di koordinat  $(x + 1, y) = A(x + 1, y)$

$P_{11}$  = Nilai piksel pada citra asli di koordinat  $(x + 1, y + 1) = A(x + 1, y + 1)$

$w_{00}$  = Bobot interpolasi untuk  $P_{00} = (1 - \alpha)(1 - \beta)$

$w_{01}$  = Bobot interpolasi untuk  $P_{01} = (1 - \alpha)\beta$

$w_{10}$  = Bobot interpolasi untuk  $P_{10} = \alpha(1 - \beta)$

$w_{11}$  = Bobot interpolasi untuk  $P_{11} = \alpha\beta$

Pada tahap selanjutnya akan diterapkan normalisasi yang bertujuan untuk menstandarkan skala intensitas piksel pada citra agar model dapat melakukan pembelajaran dengan lebih cepat dan efisien. Proses ini mengubah nilai intensitas

piksel dari rentang [0, 255] menjadi rentang yang lebih kecil, seperti [0, 1]. Hal tersebut dilakukan untuk memastikan bahwa model tidak terpengaruh oleh perbedaan skala dalam nilai piksel citra. Secara matematis, proses ini dapat dijelaskan dengan Persamaan 2.4 sebagai berikut:

$$I_{\text{norm}}(x, y) = \frac{I(x, y) - I_{\min}}{I_{\max} - I_{\min}} \quad (2.4)$$

dengan,

$I(x, y)$  = nilai piksel pada posisi  $(x, y)$  pada citra asli

$I_{\min}$  = nilai piksel terkecil dalam citra

$I_{\max}$  = nilai piksel terbesar dalam citra

$I_{\text{norm}}(x, y)$  = nilai piksel setelah normalisasi yang berada dalam rentang [0, 1]

proses ini mengubah nilai piksel ke dalam rentang [0, 1] dengan mengurangkan nilai piksel terkecil dan membaginya dengan rentang nilai piksel, yaitu selisih antara nilai piksel terbesar dan terkecil.

## 2.4 Deep Learning

*Deep Learning* adalah cabang dari *machine learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan untuk memproses data dalam jumlah besar dan kompleks. Model *deep learning* terdiri dari banyak lapisan neuron yang saling terhubung, di mana setiap lapisan memiliki tugas untuk mempelajari representasi data secara bertahap. Setiap lapisan tersebut mengubah input data menjadi fitur yang semakin kompleks, yang memungkinkan model untuk dapat memahami pola yang lebih rumit dalam data. Keunggulan utama dari *deep learning* adalah kemampuannya dalam mempelajari fitur data secara otomatis, tanpa memerlukan ekstraksi fitur manual yang sering kali memakan waktu dan memerlukan keahlian domain. Hal ini menjadikan *deep learning* sangat berguna dalam banyak tugas kompleks seperti klasifikasi citra medis, di mana data yang diproses sangat besar dan variatif, seperti citra *X-ray* dada (Krizhevsky et al., 2012). Dengan menggunakan *deep learning*, model dapat menganalisis dan mengenali pola dalam citra medis untuk mengidentifikasi pola atau kelainan suatu penyakit dengan akurat, seperti yang diperlukan dalam klasifikasi citra penyakit tuberkulosis (Esteva et al., 2017). Dalam penelitian ini, model *Focal Transformer* memanfaatkan kekuatan *deep learning* untuk menganalisis citra *X-ray* dan mengidentifikasi pola-pola yang menunjukkan adanya tuberkulosis.

## 2.5 Transformer

Transformer adalah arsitektur *deep learning* berbasis *self-attention* yang diperkenalkan oleh Vaswani et al. 2017. Model ini mampu menangkap hubungan spasial antar elemen dalam data tanpa ketergantungan sekuensial, menjadikannya lebih efisien dibandingkan model konvensional seperti *Recurrent Neural Networks* (RNNs) dan *Convolutional Neural Networks* (CNNs). Dibandingkan RNN yang memiliki keterbatasan dalam menangani dependensi jangka panjang, Transformer dapat memproses data secara paralel, sehingga lebih cepat dan efektif dalam tugas klasifikasi citra medis seperti klasifikasi penyakit tuberkulosis berbasis *X-ray* (Dosovitskiy et al., 2020).

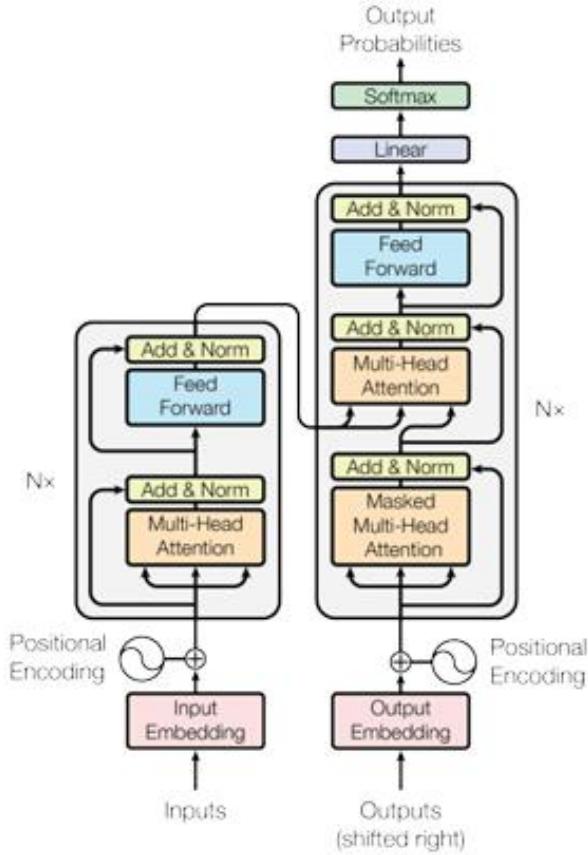
Salah satu komponen utama dalam Transformer adalah *self-attention mechanism*, yang memungkinkan model untuk menentukan bobot perhatian pada setiap elemen dalam input berdasarkan relevansinya dengan elemen lain (Vaswani et al., 2017). Mekanisme ini dihitung menggunakan *scaled dot-product attention*, yang ditunjukkan pada Persamaan 2.5 sebagai berikut:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.5)$$

dengan,

- $Q$  = Query, mewakili vektor yang ingin dicocokkan
- $K$  = Key, referensi untuk pencocokan query
- $V$  = Value, informasi yang ditransformasikan
- $d_k$  = Dimensi vektor key, digunakan untuk normalisasi agar distribusi perhatian lebih stabil
- softmax digunakan untuk menentukan seberapa besar perhatian diberikan pada elemen input lainnya

Arsitektur Transformer terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas mengolah input dengan serangkaian lapisan *self-attention* dan *feed-forward neural networks* (FFN), menghasilkan representasi fitur yang lebih abstrak. Sementara itu, *decoder* menggunakan informasi dari *encoder* dan menerapkannya dalam tugas prediksi, seperti klasifikasi. Dalam konteks klasifikasi citra *X-ray*, encoder lebih relevan karena bertanggung jawab dalam memahami dan mengekstrak fitur dari citra.



Gambar 2.3 Struktur Model *Transformer*  
(Vaswani et al., 2017)

Berdasarkan Gambar 2.3, beberapa komponen utama yang digunakan dalam proses pemrosesan data meliputi *Multi-Head Self-Attention* (MSA), *Feed-Forward Neural Networks* (FFN), *Softmax Function*, *Residual Connection*, dan *Layer Normalization*.

*Multi-Head Self-Attention* (MSA) adalah komponen inti yang memungkinkan model untuk menangkap informasi dari berbagai representasi dalam satu waktu. Berbeda dengan *self-attention* biasa, MSA membagi input menjadi beberapa head sehingga model dapat memahami pola dalam data dari berbagai perspektif. MSA dihitung menggunakan Persamaan 2.6 dan 2.7 sebagai berikut (Vaswani et al., 2017):

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.6)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.7)$$

dengan  $W_i^Q$ ,  $W_i^K$ , dan  $W_i^V$  merupakan matriks bobot untuk *query*, *key*, dan *value* pada head ke- $i$  dan  $W^O$  adalah matriks pembobotan setelah concatenation. Dengan adanya *multi-head attention*, Transformer dapat menangkap berbagai pola dalam input, yang sangat penting dalam klasifikasi tuberkulosis dari citra *X-ray*, di mana area abnormal

seperti penumpukan zat atau endapan kalsium pada jaringan tubuh harus diperhatikan secara detail.

Setelah melewati tahap *self-attention*, hasilnya akan diteruskan ke tahap *Layer Normalization* (LN) yang bertujuan untuk melakukan normalisasi pada setiap fitur di setiap sampel individu dalam *batch*. *Layer Normalization* bekerja dengan cara menghitung rata-rata dan varians untuk setiap fitur data yang terdapat dalam *batch* menggunakan Persamaan 2.8 sebagai berikut:

$$LN = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad (2.8)$$

dengan  $x$  adalah vektor masukan yang akan dinormalisasi,  $\mu$  adalah rata-rata dari  $x$ ,  $\sigma^2$  adalah varians dari  $x$ , serta  $\epsilon$  adalah parameter kecil yang digunakan untuk stabilitas numerik.

Setelah dilakukan normalisasi, hasilnya dikirim ke *Feed-Forward Neural Networks* (FFN) untuk transformasi lebih lanjut. FFN terdiri dari dua lapisan linear dengan fungsi aktivasi ReLU di antaranya, dan didefinisikan pada Persamaan 2.9 sebagai berikut:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.9)$$

dimana  $W_1$  dan  $W_2$  merupakan matriks pembobotan dalam FFN,  $b_1$  dan  $b_2$  adalah bias dalam FFN dan ReLU digunakan untuk menangani non-linearitas dalam data. Peran FFN dalam Transformer adalah meningkatkan kompleksitas fitur yang telah diperoleh dari tahap *self-attention*, sehingga informasi yang diekstrak menjadi lebih representatif sebelum masuk ke tahap klasifikasi akhir.

Softmax digunakan pada tahap akhir untuk menghasilkan probabilitas dari keluaran model, di mana nilai tertinggi menentukan prediksi model. Fungsi ini didefinisikan pada Persamaan 2.10 sebagai berikut:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.10)$$

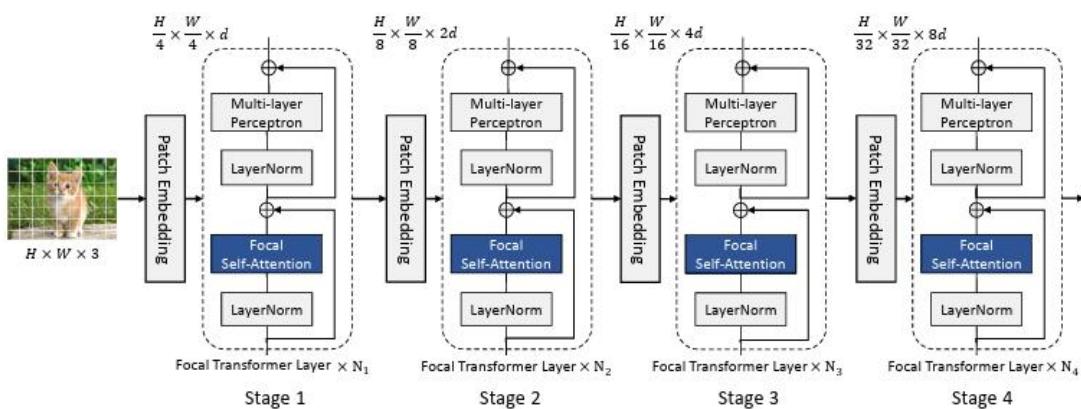
Softmax mengubah nilai numerik menjadi probabilitas yang menjumlahkan 1, sehingga memungkinkan pemilihan kelas dengan probabilitas tertinggi. Dalam konteks penelitian ini, Softmax digunakan untuk menentukan apakah citra *X-ray* termasuk ke dalam kategori tuberkulosis atau normal.

Selain itu, Transformer juga menggunakan *Residual Connection* dan *Layer Normalization* untuk menjaga kestabilan model selama pelatihan. *Residual Connection* memastikan bahwa informasi dari lapisan sebelumnya tetap tersedia, sementara *Layer Normalization* membantu menghindari hilangnya gradien selama proses pembelajaran.

## 2.6 Focal Transformer

*Focal Transformer* adalah varian terbaru dari Vision Transformer (ViT) yang dirancang untuk menangani interaksi lokal dan global dalam tugas visi komputer, terutama pada citra beresolusi tinggi. Perbedaan utama antara *Focal Transformer* dan ViT terletak pada mekanisme attention yang digunakan. *Focal Transformer* mengadopsi *focal self-attention*, yang menggabungkan perhatian lokal halus (*fine-grained*) untuk menangkap ketergantungan visual di area sekitar, dan perhatian global kasar (*coarse-grained*) untuk menangkap ketergantungan visual pada jarak yang lebih jauh. Hal ini memungkinkan *Focal Transformer* untuk menangani citra dengan resolusi tinggi secara lebih efisien, mengurangi biaya komputasi yang tinggi pada metode self-attention yang diterapkan pada ViT. Keunggulan utama dari *Focal Transformer* terletak pada kemampuannya untuk menangkap ketergantungan visual baik yang dekat maupun jauh dengan beban komputasi yang lebih rendah. Berdasarkan hasil eksperimen, *Focal Transformer* terbukti lebih unggul dibandingkan ViT dalam tugas klasifikasi citra. Model Focal-Small dengan 51,1 juta parameter mencapai akurasi Top-1 sebesar 83,5%, sedangkan model Focal-Base dengan 89,8 juta parameter mencapai 83,8%, keduanya mengungguli model ViT lainnya dalam hal akurasi klasifikasi citra pada dataset ImageNet (Yang et al., 2021).

Arsitektur *Focal Transformer* terdiri dari beberapa komponen yang saling bekerja sama untuk menangani tugas visi komputer, terutama pada citra beresolusi tinggi. Komponen tersebut meliputi *Patch Embedding*, *Focal Self-Attention*, *Residual Connection*, *Multi-layer Perceptron* (MLP) dan *Layer Normalization* (LN). Arsitektur ini dibagi menjadi beberapa tahap, yang masing-masing memiliki fungsi spesifik dalam memproses informasi. Ilustrasi dari arsitektur model tersebut ditunjukkan pada Gambar 2.4.

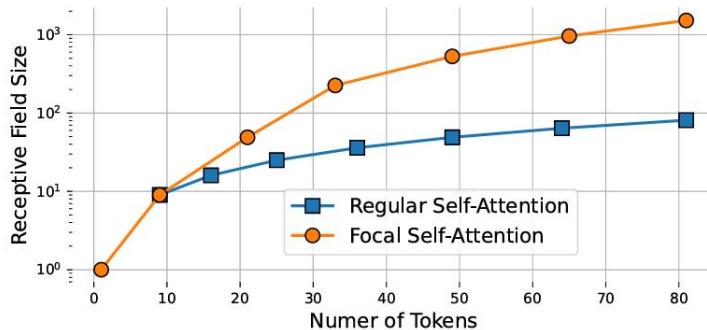


Gambar 2.4 Arsitektur *Focal Transformer*  
(Yang et al., 2021)

*Patch Embedding* bertugas untuk membagi citra input menjadi beberapa *patch* berukuran kecil, yang kemudian diproyeksikan ke dalam representasi ruang fitur dengan dimensi tertentu. Pada tahap pertama, ukuran *patch* adalah  $\frac{H}{8} \times \frac{W}{8} \times 2d$ , di tahap ketiga  $\frac{H}{16} \times \frac{W}{16} \times 4d$ , dan di tahap keempat  $\frac{H}{32} \times \frac{W}{32} \times 8d$ . Setiap tahap menggunakan *focal self-attention*, yang memungkinkan model untuk menangkap ketergantungan visual lokal secara halus dan ketergantungan visual global secara kasar. Dengan cara ini, model dapat mengurangi beban komputasi yang tinggi yang biasa ditemukan pada model ViT, dengan tetap mempertahankan kemampuan untuk menangani ketergantungan pada berbagai skala. Setiap tahap memiliki beberapa lapisan *Focal Transformer*, yang mencakup proses perhatian, lapisan normalisasi, dan pemrosesan lebih lanjut dari informasi yang diperoleh. Lapisan *Focal Transformer* menggabungkan proses perhatian dan *Multi-layer Perceptron* untuk menghasilkan representasi yang lebih kaya untuk tugas visi komputer. *Layer Normalization* digunakan untuk menstabilkan pelatihan dan mempercepat konvergensi dengan menormalkan *output* dari setiap lapisan sebelum melanjutkan ke tahap berikutnya. MLP bertugas untuk mengubah representasi yang dipelajari oleh Transformer menjadi *output* yang dapat digunakan untuk tugas tertentu, seperti klasifikasi citra (Yang et al., 2021).

### 2.6.1 Focal Self-Attention

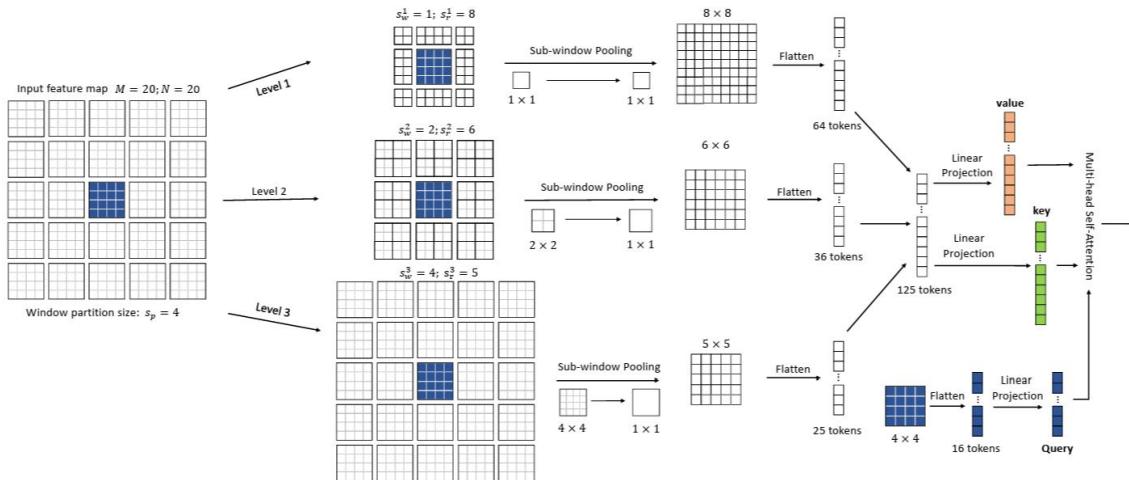
Penggunaan *focal self-attention* untuk membuat lapisan Transformer yang dapat diskalakan untuk input resolusi tinggi. Alih-alih memperhatikan semua token pada tingkat halus (*fine-grain*), model ini hanya memperhatikan token-token halus secara lokal, sementara token lainnya diperhatikan secara global. Dengan cara ini, model dapat mencakup sebanyak mungkin area yang sama seperti *self-attention* standar, namun dengan beban komputasi yang jauh lebih rendah.



Gambar 2.5 Ukuran Lapangan Reseptif  
(Yang et al., 2021)

Gambar 2.5 menunjukkan ukuran area *receptive field* untuk *self-attention* standar dan *focal self-attention* saat jumlah token yang diperhatikan bertambah. Untuk posisi kueri tertentu, dengan menggunakan token yang lebih kasar di sekelilingnya, *focal self-attention* dapat memiliki *receptive field* yang jauh lebih besar dengan biaya yang lebih rendah dalam hal jumlah token visual yang perlu diperhatikan dibandingkan dengan *self-attention* standar.

Mekanisme *focal self-attention* memungkinkan perhatian jarak jauh dengan waktu dan memori yang lebih sedikit, karena hanya memperhatikan token yang lebih sedikit (token yang sudah diringkas). Dalam praktiknya, pengambilan token sekeliling untuk setiap posisi kueri memerlukan memori dan waktu yang lebih sedikit, karena kita hanya perlu mengakses token yang dapat menjangkau *query*. Masalah praktis ini telah dicatat dalam berbagai penelitian sebelumnya, dan solusi umum yang digunakan adalah dengan membagi peta fitur input ke dalam jendela-jendela.



Gambar 2.6 Ilustrasi *Focal Self-Attention*  
(Yang et al., 2021)

Gambar 2.6 menjelaskan *Focal Self-Attention* pada tingkat jendela. Setiap kotak terkecil mewakili token visual yang diambil baik dari peta fitur. Misalnya, jika kita memiliki peta fitur berukuran  $20 \times 20$ , kita pertama-tama membaginya menjadi jendela  $5 \times 5$  dengan ukuran  $4 \times 4$ . Selanjutnya, jendela biru di tengah digunakan sebagai *query*, dan token-token di sekelilingnya diekstrak pada beberapa tingkat granularitas yang berbeda, yang berfungsi sebagai kunci dan nilai.

Pada tingkat pertama, token yang diekstrak berukuran  $8 \times 8$ , yang paling dekat dengan jendela biru pada tingkat halus. Pada tingkat kedua, area perhatian diperluas untuk mencakup wilayah yang lebih besar, yang menghasilkan token-token yang dipadatkan ke dalam jendela bagian berukuran  $2 \times 2$ . Pada tingkat ketiga, perhatian lebih lanjut diperluas untuk mencakup peta fitur seluruhnya, yang menghasilkan lebih banyak token. Akhirnya, tingkat fitur tingkat kehalusan tinggi digunakan untuk menghitung *key*

dan *value* untuk jendela biru dengan total 16 token (*query*). Ini adalah ilustrasi dari cara *focal self-attention* bekerja pada tingkat jendela, dengan memanfaatkan *pooling* token dari berbagai tingkat kehalusan untuk efisiensi dalam menangani token visual.

### 2.6.2 Window-wise Attention

*Window-wise Focal Self-attention* merupakan sebuah mekanisme yang digunakan untuk memperhatikan token-token visual dengan fokus pada jendela-jendela kecil di dalam peta fitur. Ada tiga istilah utama yang digunakan untuk menjelaskan metode ini:

1. **Focal Levels (L)**: Merupakan jumlah tingkat kehalusan yang diekstrak untuk *self-attention*. Dalam ilustrasi pada gambar, ada tiga tingkat focal yang digunakan.
2. **Focal Window Size ( $s_l^w$ )**: Ukuran jendela bagian pada setiap tingkat focal yang digunakan untuk meringkas token-token di tingkat  $l$ . Ukuran jendela bagian ini adalah  $2 \times 2$ ,  $3 \times 3$ , dan  $4 \times 4$  pada tingkat focal pertama hingga ketiga.
3. **Focal Region Size ( $s_l^r$ )**: Jumlah jendela bagian yang digunakan secara horizontal dan vertikal pada wilayah yang diperhatikan di setiap tingkat  $l$ . Pada gambar, ukuran ini adalah 3, 4, dan 4 dari tingkat pertama hingga ketiga.

Langkah pertama dalam *sub-window pooling* adalah membagi peta fitur input  $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$  ke dalam jendela-jendela kecil berukuran  $s_w^l \times s_w^l$ . Kemudian, *pooling* dilakukan menggunakan proyeksi linier  $f_p^l$  untuk setiap tingkat focal, menghasilkan peta fitur yang telah dilakukan pooling:

$$\mathbf{x}^l = f_p^l(\hat{\mathbf{x}}) \in \mathbb{R}^{\frac{M}{s_w^l} \times \frac{N}{s_w^l} \times d}, \quad \hat{\mathbf{x}} = \text{Reshape}(\mathbf{x}) \in \mathbb{R}^{(\frac{M}{s_w^l} \times \frac{N}{s_w^l} \times d) \times (s_w^l \times s_w^l)} \quad (2.11)$$

dengan menggunakan Persamaan 2.11, maka token-token dari wilayah yang lebih besar dapat diringkas dengan cara yang lebih efisien dan komputasi yang lebih ringan. Peta fitur yang telah dilakukan *pooling* kemudian digunakan untuk menghitung perhatian pada masing-masing jendela dengan tingkat kehalusan yang berbeda, seperti yang dijelaskan lebih lanjut dalam perhitungan untuk perhatian.

Setelah mendapatkan peta fitur yang telah dilakukan *pooling* pada semua tingkat kehalusan. Proses perhitungan perhatian dimulai dengan menghitung *query* pada tingkat pertama dan *key* serta *value* pada semua tingkat menggunakan tiga lapisan proyeksi linier  $f_q$ . Untuk setiap *query* pada jendela ke- $i$ , token yang ada di sekitar jendela tersebut diekstrak pada beberapa tingkat kehalusan. Perhitungan untuk mendapatkan *Query* (Q), *Key* (K), dan *Value* (V) didefinisikan dalam Persamaan 2.12 sebagai berikut:

$$Q = f_q(\hat{x}^1), \quad K = \{K^l\}_1^L = f_k(\{x^1, \dots, x^L\}), \quad V = \{V^l\}_1^L = f_v(\{x^1, \dots, x^L\}) \quad (2.12)$$

dengan,  $Q$  adalah proyeksi kueri pada tingkat pertama,  $K$  adalah kumpulan kunci yang diekstrak dari beberapa tingkat kehalusan, dan  $V$  adalah kumpulan nilai yang diambil dari seluruh tingkat.

Untuk melakukan *focal self-attention*, pertama-tama token-token sekitar untuk setiap kueri diekstrak, yang berada dalam jendela dengan ukuran  $s_p \times s_p$ . Proses ini dilakukan untuk semua tingkat  $l$ , dengan mengumpulkan kunci dan nilai dari wilayah sekitar jendela kueri pada tingkat  $l$ . Selanjutnya, untuk menghitung perhatian focal pada Query  $Q_i$ , perhitungan tersebut ditunjukkan dalam Persamaan 2.13 sebagai berikut:

$$\text{Attention}(Q_i, K_l^i, V_l^i) = \text{Softmax} \left( \frac{Q_i K_l^{iT}}{\sqrt{d}} + B \right) V_i \quad (2.13)$$

dengan,  $B$  adalah *learnable relative position bias*, yang mengacu pada bias posisi relatif yang dipelajari untuk setiap tingkat kehalusan, dan  $d$  adalah dimensi fitur dari peta fitur. *Focal self-attention* memungkinkan perhatian jarak jauh untuk dihitung dengan biaya rendah karena hanya memperhatikan token-token yang relevan, bukan seluruh token dalam fitur. Proses ini dilakukan secara paralel di setiap tingkat untuk meningkatkan efisiensi komputasi.

Setelah dihitung bobot *attention* pada tahap sebelumnya, selanjutnya akan dilakukan normalisasi menggunakan *Layer Normalization* sebelum diteruskan ke tahap *Multi-layer Perceptron* (MLP). Tahap tersebut diterapkan untuk menghasilkan representasi akhir dari masukan yang sudah diproses. MLP berfungsi untuk meningkatkan kapasitas model dalam menangani masalah non-linear yang lebih rumit dengan penambahan fungsi aktivasi GELU dan memungkinkan model untuk mempelajari pola yang lebih kompleks menggunakan Persamaan 2.14 sebagai berikut:

$$\begin{aligned} \text{MLP}(x) &= \text{GELU}(xW_1 + b_1)W_2 + b_2 \\ \text{GELU}(x) &= 0.5x \left( 1 + \tanh \left( \sqrt{\frac{2}{\pi}}(x + 0.044715x^3) \right) \right) \end{aligned} \quad (2.14)$$

## 2.7 Global Average Pooling

*Global Average Pooling* (GAP) merupakan teknik yang digunakan untuk menggantikan lapisan *fully connected* (FC) pada *Convolutional Neural Network* (CNN) untuk mengurangi *overfitting* dan meningkatkan generalisasi. Pada CNN konvensional, peta fitur dari lapisan konvolusional terakhir biasanya diubah menjadi vektor dan diteruskan ke lapisan FC dan softmax, namun pendekatan ini rentan *overfitting* karena jumlah parameter yang besar (Lin et al., 2014). Meskipun *dropout* telah digunakan sebagai regularizer untuk mengatasi hal ini (Hinton et al., 2012), GAP menawarkan alternatif yang lebih efisien dengan menghitung rata-rata dari setiap peta fitur dan langsung menghubungkannya ke lapisan softmax tanpa parameter tambahan. Metode

ini mempertahankan keterkaitan spasial antara peta fitur dan kategori, sehingga dapat diinterpretasikan sebagai peta keyakinan (*confidence map*), serta lebih tahan terhadap translasi spasial pada input. GAP dapat dituliskan pada Persamaan 2.15 sebagai berikut:

$$GAP = \frac{1}{L} \sum_{k=1}^L x_k \quad (2.15)$$

dengan  $L$  adalah jumlah peta fitur dan  $x_k$  adalah nilai dari fitur ke- $k$ .

## 2.8 Metrik Evaluasi

Dalam penelitian yang melibatkan model klasifikasi, pemilihan metrik evaluasi yang tepat menjadi penting untuk menilai performa model secara akurat. Metrik berikut digunakan untuk mengevaluasi kemampuan model *deep learning* dalam klasifikasi penyakit tuberkulosis dari citra *X-ray*. Salah satu metrik evaluasi yang akan digunakan pada penelitian ini adalah matriks konfusi. Matriks tersebut merupakan salah satu cara untuk membandingkan *true positive*, *true negative*, *false positive*, dan *false negative* (Swaminathan & Tantri, 2024). Untuk gambaran lebih jelas terkait matriks konfusi ditunjukkan pada Tabel 2.2.

Tabel 2.2 Matriks Konfusi

<b>Tabel Konfusi</b>	<b>Kenyataan</b>	
	<b>Positive</b>	<b>Negative</b>
<b>Prediksi</b>	<b>True Positive</b>	<b>False Positive</b>
	<b>False Negative</b>	<b>True Negative</b>

Istilah *True Positive* merupakan orientasi pada jumlah kasus positif yang terklasifikasi dengan benar. *True Negative* merupakan istilah yang mengarahakan jumlah kasus negatif yang terklasifikasi dengan benar. *False Positive* adalah jumlah kasus negatif yang terklasifikasi sebagai positif. Sedangkan, *False Negative* mendefinisikan jumlah kasus positif yang terklasifikasi sebagai negatif (Swaminathan & Tantri, 2024). Dari matriks konfusi tersebut dapat diuraikan menjadi beberapa metrik evaluasi untuk mengukur kinerja model diantaranya:

### 1. Akurasi

Akurasi merupakan metrik dasar yang digunakan untuk mengukur proporsi prediksi model yang benar terhadap total jumlah sampel. Metrik ini dihitung menggunakan Persamaan 2.16 (Swaminathan & Tantri, 2024):

$$Accuracy = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{False Negative} + \text{True Negative}} \quad (2.16)$$

Akurasi memberikan gambaran umum performa model, namun menjadi kurang representatif pada dataset yang tidak seimbang.

## 2. Presisi

Presisi digunakan untuk mengukur seberapa banyak prediksi positif model sesuai dengan kondisi sebenarnya. Persamaan 2.17 digunakan untuk menghitung presisi yang didefinisikan sebagai berikut (Swaminathan & Tantri, 2024):

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2.17)$$

Presisi penting dalam kasus di mana banyak kesalahan prediksi positif sangat tinggi, seperti mendiagnosis pasien yang sehat sebagai terinfeksi. Model dengan presisi tinggi menunjukkan sedikit kesalahan FP yang rendah.

## 3. Recall

Recall atau dikenal sebagai sensitivitas, mengukur kemampuan model dalam mengklasifikasikan semua kasus positif. Persamaan 2.18 digunakan untuk menghitung recall yang didefinisikan sebagai berikut (Swaminathan & Tantri, 2024):

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2.18)$$

Sensitivitas sangat penting dalam mengklasifikasikan penyakit serius seperti tuberkulosis, di mana kegagalan mengklasifikasikan kasus positif (FN) dapat berdampak serius pada pasien. Model dengan sensitivitas tinggi mampu mengklasifikasikan sebagian besar pasien yang benar-benar terinfeksi.

## 4. F1-Score

F1-Score adalah metrik yang menggabungkan presisi dan recall dalam satu nilai harmoni untuk membantu menilai keseimbangan antara keduanya. Perhitungan tersebut didefinisikan dalam Persamaan 2.19 sebagai berikut (Swaminathan & Tantri, 2024):

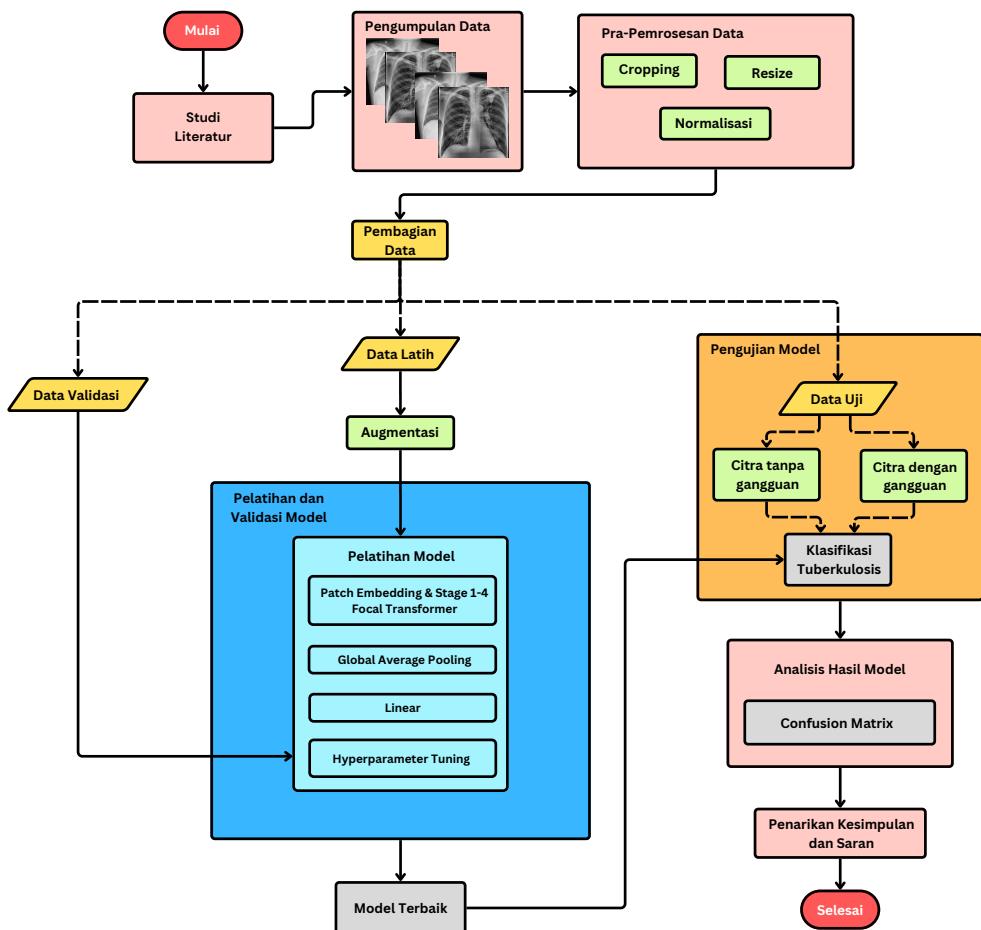
$$F1\text{-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.19)$$

F1-Score didefinisikan sebagai rata-rata dari presisi dan recall. Nilai F1-Score yang tinggi menandakan model yang baik dalam menyeimbangkan antara keduanya, sedangkan nilai yang rendah menunjukkan bahwa model cenderung memiliki kesalahan prediksi yang tinggi.

## BAB III

### METODOLOGI

Penelitian tugas akhir ini terdiri dari beberapa tahapan yang dijelaskan menggunakan blok diagram pada Gambar 3.1.



Gambar 3.1 Blok Diagram Penelitian

#### 3.1 Dataset

Dataset yang digunakan dalam penelitian ini diambil dari sumber terbuka bernama Kaggle, yang diterbitkan pada bulan Desember tahun 2024 oleh Majumder yang bernama *Combined Unknown Pneumonia and Tuberculosis* (<https://www.kaggle.com/datasets/rifatulmajumder23/combined-unknown-pneumonia-and-tuberculosis>). Dataset ini merupakan kumpulan dari berbagai sumber, yaitu Kermany, RSNA, NIAID, NLM, dan Belarus. Dataset tersebut dirancang untuk mendukung pengembangan model pembelajaran mesin dalam mengklasifikasikan berbagai kondisi paru-paru, termasuk

tuberkulosis, pneumonia, dan normal. Dalam penelitian ini, penulis hanya menggunakan citra kelas tuberkulosis dan normal untuk melatih dan mengevaluasi model *Focal Transformer*. Jumlah citra dalam dataset ini terdiri atas 4.197 citra untuk kategori tuberkulosis dan 5.489 citra untuk kategori normal.

### 3.2 Pra-Pemrosesan Citra

Langkah pra-pemrosesan citra dilakukan untuk memastikan kualitas dan konsistensi dataset untuk mendukung pelatihan model *Focal Transformer* dalam klasifikasi tuberkulosis. Tahap pertama yang dilakukan adalah penyaringan citra, di mana hanya citra dengan label tuberkulosis dan normal yang diambil untuk penelitian ini. Penyaringan ini dilakukan untuk memastikan dataset yang digunakan relevan dengan tujuan penelitian. Selain itu, data yang memiliki kualitas rendah, seperti citra yang buram atau mengandung *noise* ekstrem, dikeluarkan dari analisis untuk meminimalkan potensi gangguan pada pelatihan model.

#### 3.2.1 *Cropping*

Langkah kedua dilakukan *cropping* untuk menghilangkan bagian citra yang tidak relevan atau dalam dataset ini adalah bagian hitam yang terdapat pada sekeliling citra, sehingga hanya tersisa bagian yang memiliki informasi penting saja. Pemotongan citra tersebut dilakukan dengan cara mengidentifikasi dan menghapus baris dan kolom matriks fitur yang hanya memiliki nilai intensitas piksel di rentang 0 hingga 20. Sehingga, ketika salah satu nilai intensitas pada baris dan kolom tersebut memiliki nilai diatas 20 dianggap sebagai citra dengan bagian yang memiliki informasi penting.

#### 3.2.2 Augmentasi

Kemudian tahap berikutnya dilakukan augmentasi data citra dengan tujuan untuk memperbanyak dan memperkaya variasi data latih secara artifisial tanpa perlu mengumpulkan data baru secara manual. Augmentasi yang dilakukan hanya perubahan geometri seperti rotasi, pergeseran, skalasi, dan *horizontal flip* dengan pengaturan yang tidak terlalu ekstrim agar tidak mengubah struktur citra *X-ray* dada secara berlebihan. Augmentasi ini hanya diterapkan pada data latih dengan tujuan untuk meningkatkan variasi sudut pandang dan efektivitas proses pembelajaran model.

#### 3.2.3 *Resize*

Setelah dilakukan augmentasi, dilakukan standarisasi ukuran masukan citra untuk menghasilkan ukuran citra yang seragam. Semua citra diubah ukurannya menjadi  $224 \times 224$  piksel menggunakan metode interpolasi bilinear, sehingga memenuhi kebutuhan masukan dari arsitektur *Focal Transformer*. Langkah ini bertujuan untuk memastikan model dapat memproses citra secara efisien tanpa terganggu oleh perbedaan ukuran citra.

### 3.2.4 Normalisasi

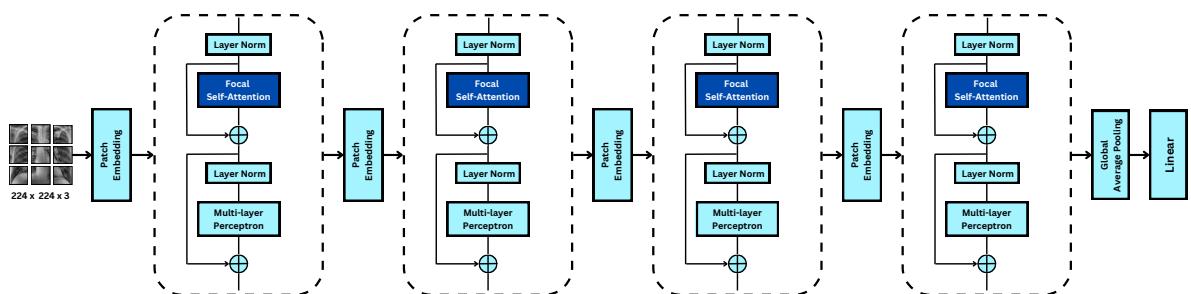
Tahap berikutnya dilakukan normalisasi intensitas piksel. Nilai piksel setiap citra dinormalisasi ke rentang [0, 1] dengan membagi setiap nilai piksel dengan 255. Proses ini bertujuan untuk menyelaraskan distribusi intensitas piksel, sehingga memudahkan model dalam mengenali pola penting pada citra *X-ray* dada. Langkah ini juga berfungsi untuk mempercepat proses pelatihan model dan meningkatkan stabilitas pembelajaran.

## 3.3 Pembagian Data

Pada tahap pembagian data, Citra data yang telah dilakukan pra-pemrosesan dibagi menjadi 3 data yakni data latih, data validasi, dan data uji dengan pembagian yang dilakukan adalah 85% untuk data latih, 5% untuk data validasi, dan 10% untuk data uji. Data latih digunakan untuk melatih model sedangkan data validasi digunakan untuk memvalidasi performa model yang telah dilatih dan data uji digunakan untuk mengevaluasi hasil dari pelatihan model.

## 3.4 Pelatihan Model

Pada tahap pelatihan model, data latih citra yang telah melalui proses pra-pemrosesan menjadi masukan bagi model *Focal Transformer*, yang kemudian diproses melalui tahap *patch partition* serta empat lapisan arsitektur *Focal Transformer*. Keluaran dari tahap *patch partition* digunakan sebagai masukan pada lapisan pertama, yang terdiri dari *Patch Embedding*, *Multi-layer Perceptron* (MLP), *LayerNorm*, dan *Focal Self-attention*. Selanjutnya, keluaran dari lapisan pertama menjadi masukan untuk lapisan kedua, yang mencakup *patch embedding*, *focal self-attention*, MLP, dan lapisan normalisasi. Proses yang sama juga berlaku untuk lapisan ketiga dan lapisan keempat, dengan *patch embedding* dan *focal self-attention* di setiap lapisan, serta MLP dan *LayerNorm* sebagai bagian dari pengolahan informasi lebih lanjut. Keluaran dari lapisan 4 diteruskan ke *global average pooling* dan *linear* dan *dropout*, yang kemudian menghasilkan prediksi kelas untuk tugas klasifikasi. Alur keseluruhan dari proses tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.2 Alur Pelatihan Model

### **3.5 Validasi Model**

Untuk memvalidasi performa model yang telah dilatih, data validasi digunakan untuk mengukur kemampuan model dalam mengklasifikasikan citra yang belum pernah dilihat selama proses pelatihan. Validasi ini dilakukan dengan melihat *validation loss*, yang mencerminkan seberapa baik model dalam mengenali pola pada data validasi. Jika hasil validasi masih belum optimal, maka dilakukan *hyperparameter tuning* untuk mengoptimalkan model hingga diperoleh model dengan parameter yang optimal dalam tugas klasifikasi tersebut.

### **3.6 Pengujian dan Analisis Hasil Model**

Setelah model dengan parameter yang optimal untuk klasifikasi tuberkulosis diperoleh, pengujian dilakukan menggunakan data uji yang belum pernah dilibatkan selama proses pelatihan. Data uji dibagi menjadi dua jenis, yaitu citra tanpa gangguan dan citra dengan gangguan, untuk mengevaluasi kemampuan model meskipun citra yang diproses mengalami perubahan atau gangguan tertentu. Jenis gangguan yang diterapkan meliputi *gaussian noise*, *salt and pepper noise*, *motion blur*, *brightness and contrast distortion*, serta *partial occlusion*. Kedua jenis citra ini kemudian diproses dalam sistem klasifikasi tuberkulosis. Hasil klasifikasi dievaluasi menggunakan matriks konfusi yang membandingkan antara prediksi model dan label sebenarnya. Dari matriks konfusi, diperoleh metrik evaluasi seperti akurasi, presisi, sensitivitas, dan *F1-Score*. Akurasi mengukur persentase prediksi yang benar dibandingkan dengan jumlah total sampel yang diuji, presisi mengukur sejauh mana prediksi positif yang benar dibandingkan dengan total prediksi positif yang dibuat oleh model, sensitivitas menunjukkan kemampuan model dalam mendeteksi seluruh kasus positif yang sebenarnya, dan *F1-Score* memberikan rata-rata harmonis antara presisi dan sensitivitas untuk evaluasi yang lebih seimbang.

### **3.7 Penarikan Kesimpulan dan Saran**

Pada tahap ini dilakukan penarikan kesimpulan dari hasil evaluasi serta analisis yang telah dilakukan. Selain itu, disampaikan saran untuk penelitian selanjutnya.

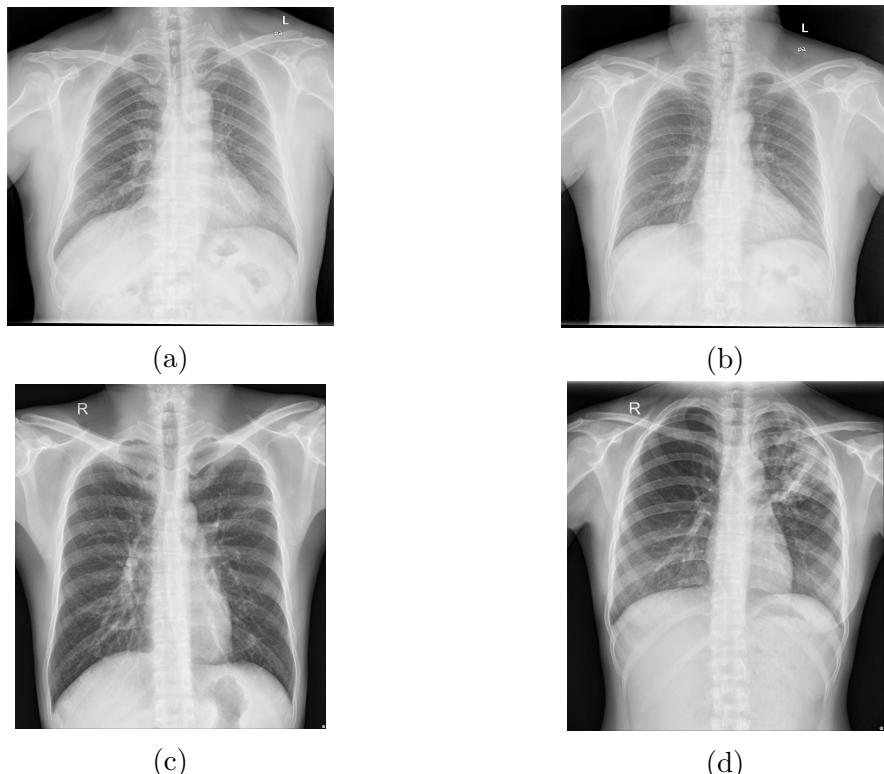
## BAB IV

### PERANCANGAN DAN IMPLEMENTASI

Pada bab ini dibahas mengenai perancangan dan implementasi model *Focal Transformer* untuk klasifikasi tuberkulosis pada citra *X-ray* dada.

#### 4.1 Pengambilan Data

Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari Kaggle dengan judul *Combined Unknown Pneumonia and Tuberculosis*. Dataset tersebut terdiri dari 9.636 citra dengan 5.489 citra kategori normal dan 4.197 citra kategori tuberkulosis dengan 4.227 citra memiliki mode warna *grayscale*, 635 citra mode warna *palette*, dan 4.774 mode warna RGB. Dataset tersebut memiliki ukuran citra yang bervariatif dengan ukuran terkecil adalah  $512 \times 512$  piksel dan ukuran terbesar adalah  $4020 \times 4892$  piksel. Berikut diberikan beberapa contoh citra yang ditunjukkan pada Gambar 4.1.



Gambar 4.1 Contoh Citra Dataset: (a) dan (b) Citra Kategori Normal, (c) dan (d) Citra Kategori Tuberkulosis

(Majumder, 2024)

## 4.2 Pra-pemrosesan Citra

Tahap awal pengolahan data citra dilakukan untuk mempersiapkan citra sebelum memasuki proses pelatihan model klasifikasi tuberkulosis. Langkah pertama yang dilakukan adalah melakukan penyaringan terhadap dataset dengan cara mengidentifikasi serta menghapus citra yang memiliki karakteristik sangat berbeda dari mayoritas data, yang diduga merupakan kesalahan atau pengecualian yang tidak relevan. Dari jumlah dataset awal sebanyak 9.722 citra, setelah proses penyaringan tersisa sebanyak 9.636 citra. Dataset tersebut memiliki variasi mode warna, antara lain *grayscale*, *palette*, dan RGB. Oleh karena itu, tahap pra-pemrosesan selanjutnya adalah melakukan konversi seluruh citra ke dalam mode warna RGB untuk menyelaraskan format *input* sehingga memudahkan pada proses pelatihan model.

Berdasarkan hasil analisis, sebagian besar citra *X-ray* dalam dataset menunjukkan tingkat keseragaman visual yang tinggi, ditandai dengan sudut pandang frontal yang konsisten, komposisi anatomi yang serupa, serta pencahayaan dan kontras yang relatif homogen. Kondisi ini berpotensi menurunkan kemampuan generalisasi model terhadap data baru karena keterbatasan variasi dalam data latih. Untuk mengatasi hal tersebut, dilakukan augmentasi untuk meningkatkan keragaman data latih dan mendukung kinerja model pada data validasi. Jenis augmentasi yang diterapkan meliputi *horizontal flip*, rotasi, translasi, dan skalasi. Oleh karena itu, setelah proses penyeragaman format, tahap pra-pemrosesan dan augmentasi citra diterapkan dengan perhitungan sebagai berikut:

### 1. *Cropping*

Pertama, dilakukan identifikasi terhadap piksel yang memiliki nilai intensitas lebih besar dari 20 pada setidaknya satu kanal warna. Piksel tersebut dikatakan non-kosong dikarenakan pada salah satu kanalnya bernilai lebih dari 20 menggunakan Persamaan 2.2.

Selanjutnya, dilakukan pencarian batas-batas koordinat spasial dari kumpulan piksel non-kosong tersebut. Batas atas diperoleh dengan mencari baris terkecil yang memiliki minimal satu piksel non-kosong, sedangkan batas bawah adalah baris terbesar yang memenuhi kondisi serupa. Demikian pula, batas kiri dan batas kanan dicari berdasarkan kolom terkecil dan terbesar yang memiliki piksel non-kosong. Atau dapat didefinisikan dengan rumus sebagai berikut.

$$\begin{aligned} \text{top} &= \min\{i \mid \exists j \text{ dengan } M(i, j) = 1\} \\ \text{bottom} &= \max\{i \mid \exists j \text{ dengan } M(i, j) = 1\} \\ \text{left} &= \min\{j \mid \exists i \text{ dengan } M(i, j) = 1\} \\ \text{right} &= \max\{j \mid \exists i \text{ dengan } M(i, j) = 1\} \end{aligned}$$

artinya sistem mencari baris terkecil dan terbesar  $i$  kemudian kolom terkecil dan terbesar  $j$  yang memiliki setidaknya satu piksel non-kosong.

Setelah diperoleh koordinat batas atas, bawah, kiri, dan kanan, maka area *cropping* didefinisikan sebagai sub-citra yang mencakup baris dari batas atas sampai batas bawah dan kolom dari batas kiri sampai batas kanan, untuk ketiga kanal warna. Dengan demikian, bagian citra yang hanya berisi piksel kosong di pinggir terpotong dan hanya tersisa area yang mengandung informasi. Maka dari itu, ukuran hasil *cropping* pada arah vertikal dan horizontal dihitung sebagai berikut:

$$H_{\text{crop}} = \text{bottom} - \text{top} + 1$$

$$W_{\text{crop}} = \text{right} - \text{left} + 1$$

sehingga ukuran akhir citra setelah dilakukan *cropping* menjadi

$$H_{\text{crop}} \times W_{\text{crop}} \times 3$$

penambahan angka 1 bertujuan untuk menghitung ukuran secara inklusif, yaitu agar batas atas dan batas bawah serta batas kiri dan kanan termasuk dalam perhitungan ukuran *crop*.

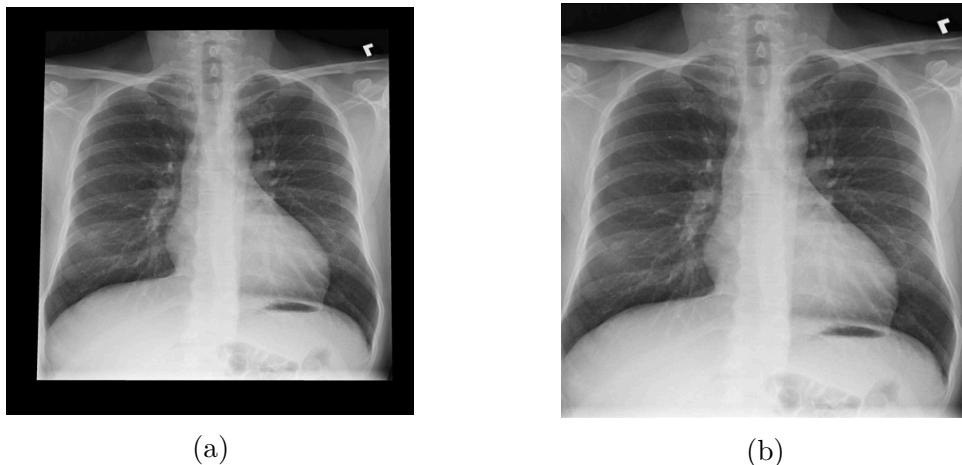
Berikut diberikan contoh penggambaran dari proses *cropping* pada salah satu kanal warna. Misal diberikan matriks nilai intensitas citra  $M$  sebagai berikut:

$$M = \begin{bmatrix} 1 & 17 & 18 & 16 & 14 & 11 & 7 \\ 0 & 30 & 35 & \cdots & 44 & 45 & 4 \\ 2 & 33 & 33 & \cdots & 49 & 42 & 19 \\ 6 & \vdots & \vdots & \ddots & \vdots & \vdots & 10 \\ 5 & 93 & 242 & \cdots & 22 & 26 & 4 \\ 8 & 29 & 74 & \cdots & 32 & 39 & 5 \\ 15 & 9 & 13 & 12 & 3 & 6 & 0 \end{bmatrix}$$

maka apabila dilakukan *cropping* pada baris dan kolom yang hanya berisi piksel kosong dibagian pinggir citra, matriks nilai intesitas citra membentuk matriks nilai intesitas yang baru sebagai berikut:

$$M_{\text{crop}} = \begin{bmatrix} 30 & 35 & \cdots & 44 & 45 \\ 33 & 33 & \cdots & 49 & 42 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 93 & 242 & \cdots & 22 & 26 \\ 29 & 74 & \cdots & 32 & 39 \end{bmatrix}$$

Berikut ditampilkan citra sebelum dan sesudah dilakukan proses *cropping* pada Gambar 4.2.



Gambar 4.2 Contoh Proses *Cropping*: (a) Citra Sebelum *Cropping*, (b) Citra Sesudah *Cropping*

## 2. Horizontal Flip

Teknik ini adalah transformasi yang membalikkan citra pada sumbu vertikal. Sehingga, koordinat  $x$  dari setiap titik citra dibalik, tetapi koordinat  $y$  tetap sama.

Diberikan permisalan matriks awal berupa:

$$C = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

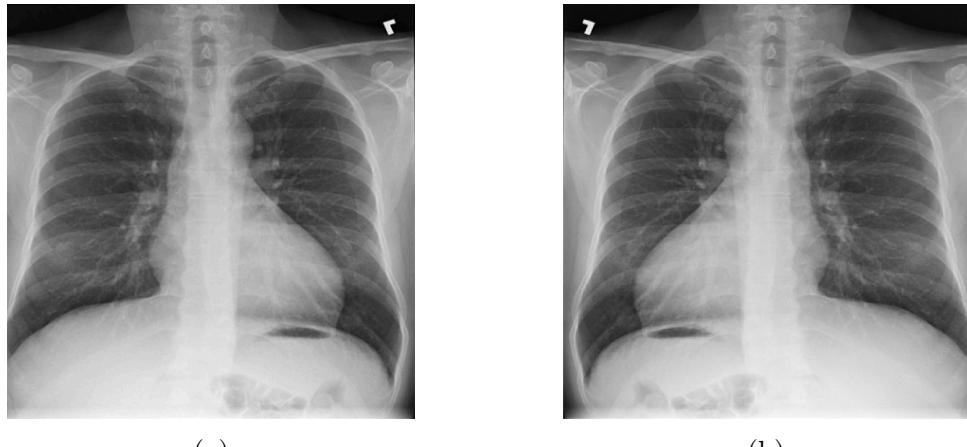
dengan matrik  $C_H$ :

$$C_H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

kemudian dilakukan perhitungan sebagai berikut  $C_{\text{flip}} = C \cdot C_H$  diperoleh matriks setelah dilakukan *horizontal flip* sebagai berikut:

$$C_{\text{flip}} = \begin{bmatrix} 10 & 5 \\ 20 & 15 \end{bmatrix}$$

Berikut ditampilkan citra sebelum dan sesudah dilakukan proses *Horizontal FLip* pada Gambar 4.3.



Gambar 4.3 Contoh Proses *Horizontal Flip*: (a) Citra Sebelum *Horizontal Flip*, (b) Citra Sesudah *Horizontal Flip*

### 3. Rotasi

Rotasi merupakan transformasi yang memutar citra sebesar  $\theta$  derajat yang ditentukan. Rotasi dapat dihitung menggunakan matriks rotasi dan dikalikan dengan matriks citra awal.

Diberikan matriks awal sebagai berikut:

$$C = \begin{bmatrix} 10 & 5 \\ 20 & 15 \end{bmatrix}$$

Untuk rotasi sebesar 90 derajat searah jarum jam, matriks rotasi  $R(90^\circ)$  adalah sebagai berikut:

$$R(90^\circ) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

kemudian dilakukan perhitungan  $C_{rotasi} = C \cdot R(90^\circ)$  dan diperoleh matriks akhir setelah rotasi sebagai berikut:

$$C_{rotasi} = \begin{bmatrix} -20 & -15 \\ 10 & 5 \end{bmatrix}$$

Berikut ditampilkan citra sebelum dan sesudah dilakukan proses rotasi dengan menggunakan derajat rotasi sebesar 10 pada Gambar 4.4.



(a)



(b)

Gambar 4.4 Contoh Proses Rotasi: (a) Citra Sebelum Rotasi, (b) Citra Sesudah Rotasi

#### 4. Translasi

Translasi adalah pergeseran titik citra dalam ruang. Dalam implementasi ini, kita menerapkan translasi pada koordinat  $x$  dan  $y$  dengan nilai  $T_x$  dan  $T_y$ . Translasi dapat dihitung dengan menambahkan nilai  $T_x$  dan  $T_y$  secara langsung ke elemen matriks citra. Operasi translasi untuk setiap elemen matriks  $C$  adalah sebagai berikut:

$$C_{translasi} = \begin{bmatrix} C_{11} + T_x & C_{12} + T_x \\ C_{21} + T_y & C_{21} + T_y \end{bmatrix}$$

dengan menggunakan permisalan yang didapatkan dari proses rotasi dan nilai  $T_x = 5$  dan  $T_y = -3$  maka diperoleh hasil matriks dari proses translasi sebagai berikut:

$$C = \begin{bmatrix} -20 & -15 \\ 10 & 5 \end{bmatrix}, \quad T = \begin{bmatrix} 5 & -3 \\ 5 & -3 \end{bmatrix}$$

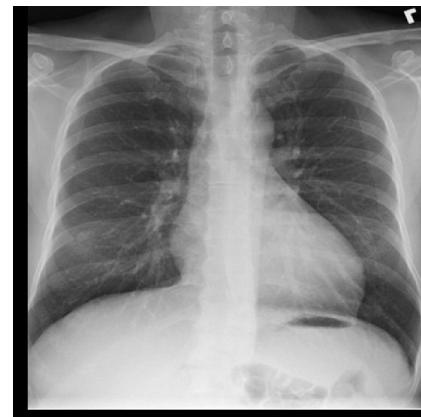
$$C_{translasi} = \begin{bmatrix} -20 + 5 & -15 + 5 \\ 10 + (-3) & 5 + (-3) \end{bmatrix}$$

$$C_{translasi} = \begin{bmatrix} -15 & -10 \\ 7 & 2 \end{bmatrix}$$

Berikut ditampilkan citra sebelum dan sesudah dilakukan proses pergeseran dengan besar pergeseran sebesar 0.05 pada sumbu  $x$  dan  $y$  ditunjukkan pada Gambar 4.5.



(a)



(b)

Gambar 4.5 Contoh Proses Pergeseran: (a) Citra Sebelum Pergeseran, (b) Citra Sesudah Pergeseran

## 5. Skala

Skalasi adalah transformasi yang mengubah ukuran citra dengan memperbesar atau memperkecil titik berdasarkan faktor skala  $s_x$  dan  $s_y$  yang diterapkan secara terpisah pada sumbu  $x$  dan  $y$ . Matriks skala tersebut dapat direpresentasikan sebagai berikut:

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

dengan menggunakan permisalan matriks yang diperoleh dari tahap sebelumnya dan dilakukan skala pada matriks  $C$  dengan faktor skala  $s_x = 2$  dan  $s_y = 0.5$ , maka dituliskan perhitungan sebagai berikut:

$$S = \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad C = \begin{bmatrix} -15 & -10 \\ 7 & 2 \end{bmatrix}$$

$$C_{\text{skala}} = S \cdot C = \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} -15 & -10 \\ 7 & 2 \end{bmatrix}$$

$$C_{\text{skala}} = \begin{bmatrix} -30 & -20 \\ 3.5 & 1 \end{bmatrix}$$

Berikut ditampilkan citra sebelum dan sesudah dilakukan proses skalasi ditunjukkan pada Gambar 4.5.



(a)



(b)

Gambar 4.6 Contoh Proses Skalasi: (a) Citra Sebelum Skalasi, (b) Citra Sesudah Skalasi

## 6. **Resize**

Citra yang telah melalui proses *cropping* untuk menghilangkan bagian kosong atau tidak relevan kemudian diproses lebih lanjut dengan melakukan *resize*. Tujuan *resize* adalah menyelaraskan ukuran citra menjadi  $224 \times 224$  piksel agar sesuai dengan kebutuhan *input* model *deep learning*. Proses ini dilakukan menggunakan metode interpolasi bilinear seperti yang tertera pada Persamaan 2.3, yaitu dengan memperkirakan nilai piksel baru berdasarkan bobot kombinasi linier dari empat piksel terdekat pada citra asli. Pada setiap posisi piksel baru, koordinatnya dihitung terlebih dahulu dengan mengalikan indeks piksel baru dengan faktor skala. Selanjutnya, nilai interpolasi dihitung menggunakan koefisien jarak horizontal dan vertikal antara piksel baru dengan piksel-piksel asli di sekitarnya. Dengan demikian, interpolasi bilinear menghasilkan citra hasil *resize* yang halus dan mempertahankan kontur serta detail visual secara lebih baik dibandingkan metode interpolasi sederhana seperti *nearest neighbor*. Sebagai contoh sederhana diberikan matriks  $A$  berukuran  $4 \times 5$  yang dilakukan *resize* ke matriks ukuran  $2 \times 2$ :

$$\mathbf{B} = \begin{bmatrix} 10 & 20 & 30 & 40 & 50 \\ 15 & 25 & 35 & 45 & 55 \\ 20 & 30 & 40 & 50 & 60 \\ 25 & 35 & 45 & 55 & 65 \end{bmatrix}$$

kemudian sebagai langkah pertama dilakukan perhitungan faktor skala dari dimensi citra asli terhadap dimensi citra baru menggunakan perhitungan sebagai berikut:

$$W_{\text{original}} = 5, \quad H_{\text{original}} = 4$$

$$W_{\text{new}} = 2, \quad H_{\text{new}} = 2$$

$$x_{\text{scale}} = \frac{5}{2} = 2.5, \quad y_{\text{scale}} = \frac{4}{2} = 2$$

dengan menggunakan interpolasi bilinear  $2 \times 2$ , setiap koordinat piksel baru  $A_{\text{resize}}(x', y')$  dihitung dengan persamaan sebagai berikut:

$$A(x', y') = w_{00}P_{00} + w_{10}P_{10} + w_{01}P_{01} + w_{11}P_{11}$$

dengan koordinat, jarak relatif, bobot dan nilai piksel sebagai berikut:

$$x = \lfloor x' \times x_{\text{scale}} \rfloor, \quad y = \lfloor y' \times y_{\text{scale}} \rfloor$$

$$\alpha = (x' \times x_{\text{scale}}) - x, \quad \beta = (y' \times y_{\text{scale}}) - y$$

$$w_{00} = (1 - \alpha)(1 - \beta), \quad w_{10} = \alpha(1 - \beta), \quad w_{01} = (1 - \alpha)\beta, \quad w_{11} = \alpha\beta$$

$$P_{00} = A(x, y), \quad P_{10} = A(x + 1, y), \quad P_{01} = A(x, y + 1), \quad P_{11} = A(x + 1, y + 1)$$

maka, **Piksel B(1, 0)**:

$$x' = 1, \quad y' = 0$$

$$x = \lfloor 1 \times 2.5 \rfloor = 2, \quad y = \lfloor 0 \times 2 \rfloor = 0$$

$$\alpha = 1 \times 2.5 - 2 = 0.5, \quad \beta = 0$$

$$P_{00} = A(2, 0) = 30, \quad P_{10} = A(3, 0) = 40$$

$$P_{01} = A(2, 1) = 35, \quad P_{11} = A(3, 1) = 45$$

$$w_{00} = (1 - 0.5)(1 - 0) = 0.5, \quad w_{10} = 0.5 \times 1 = 0.5, \quad w_{01} = 0, \quad w_{11} = 0$$

$$\begin{aligned}
 B(1,0) &= 0.5 \times 30 + 0.5 \times 40 + 0 + 0 \\
 &= 15 + 20 = 35
 \end{aligned}$$

Dengan melakukan perhitungan yang sama pada koordinat  $B(0,0)$ ,  $B(0,1)$ , dan  $B(1,1)$ , didapatkan piksel baru dari citra yang telah diresize sebagai berikut:

$$\mathbf{B}_{\text{resize}} = \begin{bmatrix} 10 & 35 \\ 20 & 45 \end{bmatrix}$$

Misalkan diambil salah satu citra dengan nilai piksel citra tersebut dalam matriks  $a$  yakni sebagai berikut:

$$a = \begin{bmatrix} 35 & 35 & 35 & \cdots & 48 & 50 & 53 \\ 34 & 34 & 34 & \cdots & 46 & 48 & 49 \\ 33 & 33 & 33 & \cdots & 46 & 47 & 47 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 19 & 18 & 18 & \cdots & 27 & 29 & 29 \\ 19 & 18 & 18 & \cdots & 28 & 30 & 30 \\ 19 & 18 & 18 & \cdots & 29 & 30 & 31 \end{bmatrix}$$

Setelah dilakukan *resize* didapatkan nilai piksel yang baru yakni matriks  $b$  dari citra sebagai berikut:

$$b = \begin{bmatrix} 29 & 29 & 29 & \cdots & 30 & 34 & 40 \\ 23 & 23 & 23 & \cdots & 21 & 25 & 32 \\ 15 & 15 & 15 & \cdots & 14 & 18 & 24 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 18 & 18 & 18 & \cdots & 26 & 68 & 52 \\ 18 & 18 & 18 & \cdots & 69 & 146 & 134 \\ 18 & 18 & 18 & \cdots & 54 & 123 & 96 \end{bmatrix}$$

Berikut ditampilkan pada Gambar 4.7 proses *resize* jika diterapkan ke salah satu citra pada dataset.



(a)



(b)

Gambar 4.7 Contoh Proses *Resize*: (a) Citra Sebelum *Resize*, (b) Citra Sesudah *Resize*

## 7. Normalisasi

Citra yang sudah melalui proses *resize* dengan menggunakan metode interpolasi bilinear dan membentuk nilai piksel baru kemudian diproses lebih lanjut ke proses normalisasi piksel. Tujuan dari normalisasi adalah untuk menstandarkan skala intensitas piksel pada citra agar model dapat melakukan pembelajaran dengan lebih cepat dan efisien. Cara kerja proses ini adalah dengan mengubah nilai intensitas piksel dari rentang  $[0, 255]$  menjadi rentang  $[0, 1]$ . Proses tersebut dilakukan dengan mengurangkan nilai piksel terkecil dan membaginya dengan rentang nilai piksel, yaitu selisih antara nilai piksel terbesar dan terkecil. Proses tersebut dihitung menggunakan Persamaan 2.4.

Berikut diberikan contoh dari proses normalisasi pada nilai intensitas piksel dari proses *resize* sebelumnya. Diperoleh matriks nilai intensitas citra sebagai berikut:

$$b = \begin{bmatrix} 29 & 29 & 29 & \cdots & 30 & 34 & 40 \\ 23 & 23 & 23 & \cdots & 21 & 25 & 32 \\ 15 & 15 & 15 & \cdots & 14 & 18 & 24 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 18 & 18 & 18 & \cdots & 26 & 68 & 52 \\ 18 & 18 & 18 & \cdots & 69 & 146 & 134 \\ 18 & 18 & 18 & \cdots & 54 & 123 & 96 \end{bmatrix}$$

maka,  $I_{\text{norm}}(1, 1)$ :

$$I_{\text{norm}}(1, 1) = \frac{29 - 0}{255 - 0} = 0.133$$

dengan melakukan perhitungan yang sama pada semua koordinat nilai intensitas piksel, diperoleh nilai baru yang telah dinormalisasi sebagai berikut:

$$b_{\text{norm}} = \begin{bmatrix} 0.113 & 0.113 & 0.113 & \cdots & 0.117 & 0.133 & 0.156 \\ 0.090 & 0.090 & 0.090 & \cdots & 0.082 & 0.098 & 0.125 \\ 0.058 & 0.058 & 0.058 & \cdots & 0.054 & 0.070 & 0.094 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0.070 & 0.070 & 0.070 & \cdots & 0.101 & 0.266 & 0.203 \\ 0.070 & 0.070 & 0.070 & \cdots & 0.270 & 0.572 & 0.525 \\ 0.070 & 0.070 & 0.070 & \cdots & 0.211 & 0.482 & 0.376 \end{bmatrix}$$

### 4.3 Pelatihan Model

Model *Focal Transformer* untuk klasifikasi citra *X-ray* tuberkulosis terdiri dari beberapa tahap utama, yaitu *Patch Embedding* yang mengubah citra menjadi *patch* fitur, dan serangkaian lapisan *Focal Transformer* yang menerapkan mekanisme self-attention untuk menangkap konteks lokal dan global secara efisien. Hasil ekstraksi fitur tersebut kemudian diproses oleh lapisan *linear* untuk menghasilkan prediksi kelas dua, yakni normal dan tuberkulosis.

#### 1. *Patch Embedding*

Bagian ini adalah proses membagi citra *input* menjadi potongan citra dengan ukuran yang lebih kecil atau *patches* untuk menyesuaikan citra dengan masukan yang diharapkan oleh model. Setiap *patches* dalam model ini dipecah menjadi menjadi ukuran  $4 \times 4$  piksel sehingga masukan citra yang berukuran  $224 \times 224$  memiliki jumlah *patch* sejumlah  $56 \times 56 = 3136$  *patches*. Kemudian *patches* tersebut diproyeksikan menjadi representasi vektor berdimensi  $d$ . Dalam implementasi ini  $d$  bernilai 96 sehingga *patches* yang sudah diproyeksikan berbentuk  $\mathbb{R}^{96 \times 56 \times 56}$ . Kemudian dilakukan *flattening* dan *transpose* untuk mengubah tensor menjadi urutan token yang dapat digunakan oleh model dengan mengubahnya menjadi bentuk [banyak *patch*,  $d$ ] dalam implementasi model ini bentuknya menjadi  $\mathbb{R}^{3136 \times 96}$ . Misalkan diambil contoh sederhana citra dengan ukuran matriks  $h \times w \times c$  yaitu  $16 \times 16 \times 1$ . Matriks tersebut dapat direpresentasikan secara numerik menjadi matriks berikut:

$$I = \begin{bmatrix} 1 & 2 & 3 & \cdots & 14 & 15 & 16 \\ 17 & 18 & 19 & \cdots & 30 & 31 & 32 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 225 & 226 & 227 & \cdots & 238 & 239 & 240 \\ 241 & 242 & 243 & \cdots & 254 & 255 & 256 \end{bmatrix}$$

Maka matriks tersebut dapat dipecah jadi beberapa bagian seperti berikut:

$$P = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ P_5 & P_6 & P_7 & P_8 \\ P_9 & P_{10} & P_{11} & P_{12} \\ P_{13} & P_{14} & P_{15} & P_{16} \end{bmatrix}$$

Sehingga hasil dari potongan bagian tersebut adalah sebagai berikut:

$$P_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 17 & 18 & 19 & 20 \\ 33 & 34 & 35 & 36 \\ 49 & 50 & 51 & 52 \end{bmatrix}$$

⋮

$$P_{16} = \begin{bmatrix} 205 & 206 & 207 & 208 \\ 221 & 222 & 223 & 224 \\ 237 & 238 & 239 & 240 \\ 253 & 254 & 255 & 256 \end{bmatrix}$$

Selanjutnya masing-masing *patches* tersebut diproyeksikan menggunakan Conv2d dengan ukuran kernel  $4 \times 4$  dan dimensi embedding  $d = 4$ . Misalkan diinisialisasi matriks filter Conv2d adalah nilai sebagai berikut:

$$W_{conv} = \left[ \begin{array}{cccc} 0.2429 & 0.1254 & 0.0610 & -0.0623 \\ -0.1085 & -0.0800 & 0.1587 & 0.1289 \\ -0.1131 & -0.0813 & 0.1347 & -0.0102 \\ -0.2440 & -0.0228 & -0.0945 & -0.2077 \end{array} \right],$$

$$\left[ \begin{array}{cccc} 0.0759 & 0.0249 & 0.1243 & -0.1500 \\ 0.0724 & -0.1855 & 0.1219 & -0.1579 \\ 0.1300 & 0.1655 & -0.1579 & -0.2153 \\ -0.0034 & -0.2414 & 0.0784 & 0.0384 \end{array} \right],$$

$$\left[ \begin{array}{cccc} -0.1262 & 0.1786 & 0.0679 & 0.0721 \\ -0.0559 & -0.2202 & 0.0477 & -0.1532 \\ -0.1504 & -0.0125 & 0.0912 & 0.1517 \\ 0.1239 & 0.0609 & -0.0003 & 0.1119 \end{array} \right],$$

$$\left[ \begin{array}{cccc} 0.0543 & 0.1793 & 0.0522 & 0.1471 \\ 0.0576 & -0.2340 & -0.1883 & -0.0223 \\ 0.1786 & -0.0870 & -0.0217 & -0.1077 \\ 0.1637 & -0.2125 & 0.1536 & -0.1309 \end{array} \right]$$

Perhitungan proyeksi untuk *patch* pertama  $P_1$  dilakukan dengan perkalian elemen demi elemen antara  $P_1$  dan  $W_{conv}$ , diikuti dengan penjumlahan seluruh elemen hasil perkalian. Untuk proyeksi *patch* pertama dengan dimensi embedding 1 diperoleh hasil sebagai berikut:

$$\begin{aligned} P_1[1] = & (1 \times 0.2429) + (2 \times 0.1254) + (3 \times 0.0610) + (4 \times (-0.0623)) \\ & + (17 \times (-0.1085)) + (18 \times (-0.0800)) + (19 \times 0.1587) + (20 \times 0.1289) \\ & + (33 \times (-0.1131)) + (34 \times (-0.0813)) + (35 \times 0.1347) + (36 \times (-0.0102)) \\ & + (49 \times (-0.2440))) + (50 \times (-0.0228))) + (51 \times (-0.0945))) \\ & + (52 \times (-0.2077))) = -27.9607 \end{aligned}$$

dengan cara perhitungan yang sama, dihitung nilai-nilai untuk dimensi embedding kedua, ketiga, dan keempat sehingga diperoleh:

$$P_1[2] = -28.6522, \quad P_1[3] = -29.3437, \quad P_1[4] = -30.0352$$

Proses tersebut dilakukan untuk *patch* lainnya, menghasilkan tensor *output* setelah proyeksi *patch* dengan bentuk  $\mathbb{R}^{1 \times 4 \times 4 \times 4}$  sebagai berikut

$$X_{proj} = \begin{bmatrix} \begin{bmatrix} -27.9607 & -28.6522 & -29.3437 & -30.0352 \\ -39.0245 & -39.7160 & -40.4075 & -41.0990 \\ -50.0883 & -50.7798 & -51.4712 & -52.1627 \\ -61.1520 & -61.8435 & -62.5350 & -63.2265 \end{bmatrix}, \\ \begin{bmatrix} -12.6315 & -13.7515 & -14.8715 & -15.9915 \\ -30.5514 & -31.6714 & -32.7914 & -33.9114 \\ -48.4712 & -49.5912 & -50.7112 & -51.8312 \\ -66.3911 & -67.5111 & -68.6311 & -69.7510 \end{bmatrix}, \\ \begin{bmatrix} 11.7401 & 12.4893 & 13.2385 & 13.9877 \\ 23.7274 & 24.4766 & 25.2258 & 25.9750 \\ 35.7147 & 36.4639 & 37.2131 & 37.9623 \\ 47.7020 & 48.4512 & 49.2004 & 49.9496 \end{bmatrix}, \\ \begin{bmatrix} -9.4742 & -9.5457 & -9.6173 & -9.6889 \\ -10.6191 & -10.6907 & -10.7623 & -10.8338 \\ -11.7641 & -11.8357 & -11.9072 & -11.9788 \\ -12.9091 & -12.9806 & -13.0522 & -13.1237 \end{bmatrix} \end{bmatrix}$$

Selanjutnya, dilakukan proses *flattening* dan *transpose* terhadap tensor citra hasil pembagian *patch*. Sebelumnya, data citra telah diubah ke dalam bentuk tensor berdimensi  $\mathbb{R}^{1 \times 4 \times 4 \times 4}$ , di mana dimensi tersebut merepresentasikan 1 kanal warna, ukuran *patch* sebanyak  $4 \times 4$ , serta setiap *patch* memiliki dimensi fitur sebesar 4. Dengan kata lain, tensor ini memiliki 4 baris dan 4 kolom *patch*, masing-masing dengan vektor fitur berdimensi 4.

Proses *flattening* dilakukan untuk menggabungkan dimensi spasial *patch*  $4 \times 4$  menjadi satu dimensi linier, yakni 16 *patch* secara berurutan. Secara matematis, ini mengubah tensor dari  $\mathbb{R}^{1 \times 4 \times 4 \times 4}$  menjadi bentuk  $\mathbb{R}^{1 \times 16 \times 4}$ , di mana 16 menunjukkan jumlah total *patch*  $4 \times 4$ , dan 4 tetap merupakan panjang vektor fitur dari masing-masing *patch*. Proses ini menyederhanakan struktur data sehingga setiap *patch* dianggap sebagai satu berdimensi 4.

Setelah *flattening*, dilakukan operasi *transpose* untuk menyelaraskan format tensor agar sesuai dengan *input* yang dibutuhkan oleh modul transformer. Pada banyak arsitektur transformer, urutan dimensi biasanya adalah ukuran *batch*, jumlah *patch*,

dan dimensi fitur. Maka dari itu, hasil akhir dari proses ini adalah tensor dengan bentuk  $\mathbb{R}^{1 \times 16 \times 4}$ , yang siap untuk dimasukkan kedalam lapisan normalisasi untuk proses selanjutnya.

$$P_{\text{embedded}} = \begin{bmatrix} -27.9607 & -12.6315 & 11.7401 & -9.4742 \\ -28.6522 & -13.7515 & 12.4893 & -9.5457 \\ -29.3437 & -14.8715 & 13.2385 & -9.6173 \\ -30.0352 & -15.9915 & 13.9877 & -9.6889 \\ -39.0245 & -30.5514 & 23.7274 & -10.6191 \\ -39.7160 & -31.6714 & 24.4766 & -10.6907 \\ -40.4075 & -32.7914 & 25.2258 & -10.7623 \\ -41.0990 & -33.9114 & 25.9750 & -10.8338 \\ -50.0883 & -48.4712 & 35.7147 & -11.7641 \\ -50.7798 & -49.5912 & 36.4639 & -11.8357 \\ -51.4712 & -50.7112 & 37.2131 & -11.9072 \\ -52.1627 & -51.8312 & 37.9623 & -11.9788 \\ -61.1520 & -66.3911 & 47.7020 & -12.9091 \\ -61.8435 & -67.5111 & 48.4512 & -12.9806 \\ -62.5350 & -68.6311 & 49.2004 & -13.0522 \\ -63.2265 & -69.7510 & 49.9496 & -13.1237 \end{bmatrix}$$

## 2. *Layer Normalization*

*Layer normalization* adalah teknik normalisasi yang diterapkan pada setiap sampel data secara individu dengan cara menormalkan nilai fitur di sepanjang dimensi fitur pada sampel tersebut. *layer normalization* menghitung rata-rata dan varians dari semua fitur dalam satu baris vektor fitur, kemudian setiap nilai fitur dinormalisasi dengan mengurangkan mean tersebut dan membaginya dengan akar varians yang sudah ditambah nilai kecil (epsilon) untuk menghindari pembagian dengan nol. Hasil normalisasi ini kemudian biasanya di-skala dan digeser menggunakan parameter trainable, yaitu gamma (skala) dan beta (*offset*), sehingga model dapat menyesuaikan distribusi *output* sesuai kebutuhan selama pelatihan. Keluaran yang dihasilkan setelah tahap ini memiliki bentuk yang tetap seperti keluaran tahap *Patch Embedding* yaitu  $\mathbb{R}^{3136 \times 96}$  hanya saja dengan representasi numerik yang sudah dinormalisasi. Berikut diberikan perhitungan *Layer Normalization* menggunakan Persamaan 2.8 dan masukan dari tahap sebelumnya yaitu  $\mathbb{R}^{1 \times 16 \times 4}$ .

Untuk setiap baris, pertama-tama kita menghitung rata-rata dari nilai fiturnya. Sebagai contoh, untuk *patch* pertama, rata-ratanya adalah jumlah elemen dibagi jumlah elemen, yaitu

$$\mu_1 = \frac{1}{4} \sum_{i=1}^4 x_i = \frac{(-27.9607) + (-12.6315) + 11.7401 + (-9.4742)}{4} = -9.5816$$

Setelah mendapatkan rata-rata tiap *patch*, langkah berikutnya adalah menghitung variansi dari fitur-fitur dalam vektor *patch* tersebut. Variansi ini dihitung dengan menjumlahkan kuadrat selisih setiap elemen terhadap rata-rata, lalu dibagi dengan jumlah elemen. Untuk *patch* pertama dapat dihitung dengan rumus sebagai berikut:

$$\begin{aligned}\sigma_1^2 &= \frac{1}{4} \sum_{i=1}^4 (x_i - \mu_1)^2 \\ \sigma_1^2 &= \frac{1}{4} [(-18.3791)^2 + (-3.0499)^2 + (21.3217)^2 + (0.1074)^2] \\ \sigma_1^2 &= \frac{1}{4} [337.7593 + 9.2958 + 454.6594 + 0.0115] \\ \sigma_1^2 &= \frac{801.7259}{4} = 200.4315\end{aligned}$$

Setelah menghitung rata-rata dan variansi, langkah selanjutnya adalah menormalkan setiap elemen dalam *patch* pertama menggunakan rumus berikut:

$$\hat{x}_1 = \frac{x_1 - \mu_1}{\sqrt{\sigma_1^2 + \epsilon}}$$

dimana  $\epsilon = 10^{-5}$  adalah konstanta kecil yang ditambahkan untuk mencegah pembagian dengan nol

$$\begin{aligned}\hat{x}_1 &= \frac{[-27.9607, -12.6315, 11.7401, -9.4742] - (-9.5816)}{\sqrt{200.4315 + 10^{-5}}} \\ \hat{x}_1 &= \frac{[-18.3791, -3.0499, 21.3217, 0.1074]}{\sqrt{200.4315}} = \frac{[-18.3791, -3.0499, 21.3217, 0.1074]}{14.16} \\ \hat{x}_1 &= [-1.2974, -0.2152, 1.5064, 0.0076]\end{aligned}$$

Selanjutnya, hasil normalisasi ini biasanya dikalikan dengan parameter skala  $\gamma$  dan ditambah dengan parameter offset  $\beta$  untuk proyeksi linier, sehingga *output LayerNorm* untuk *patch* pertama dihitung dengan rumus  $y_1 = \gamma \hat{x}_1 + \beta$ . Namun pada umumnya nilai  $\gamma$  dan  $\beta$  diinisialisasi sebagai 1 dan 0 secara default sehingga hasil akhirnya tetap sama seperti hasil normalisasi di atas. Sehingga hasil akhir *Layer Normalization* semua *patch* adalah sebagai berikut:

$$LN(\hat{x}) = \begin{bmatrix} -1.2974 & -0.2152 & 1.5064 & 0.0076 \\ -1.2754 & -0.2638 & 1.5176 & 0.0217 \\ -1.2535 & -0.3084 & 1.5272 & 0.0347 \\ -1.2324 & -0.3495 & 1.5352 & 0.0467 \\ -1.0309 & -0.6802 & 1.5664 & 0.1448 \\ -1.0199 & -0.6958 & 1.5663 & 0.1495 \\ -1.0094 & -0.7105 & 1.5661 & 0.1539 \\ -0.9993 & -0.7245 & 1.5657 & 0.1581 \\ -0.9000 & -0.8537 & 1.5564 & 0.1972 \\ -0.8942 & -0.8608 & 1.5556 & 0.1994 \\ -0.8886 & -0.8676 & 1.5547 & 0.2014 \\ -0.8831 & -0.8742 & 1.5539 & 0.2034 \\ -0.8265 & -0.9406 & 1.5434 & 0.2238 \\ -0.8230 & -0.9446 & 1.5426 & 0.2250 \\ -0.8196 & -0.9485 & 1.5419 & 0.2262 \\ -0.8163 & -0.9522 & 1.5412 & 0.2273 \end{bmatrix}$$

### 3. Focal Self-Attention

Langkah pertama pada tahap ini adalah *Sub-window Pooling*. Proses tersebut membagi matriks fitur menjadi jendela kecil berukuran  $s_{lw} \times s_{lw}$ , dan dilakukan *pooling* untuk setiap *Sub-window* dengan menggunakan proyeksi linear  $f_p^l$  untuk setiap tingkat seperti yang terdapat pada Persamaan (2.11). Diberikan permisalan masukan matriks fitur  $x \in \mathbb{R}^{20 \times 20 \times 1}$  sebagai berikut:

$$x = \begin{bmatrix} 0.8091 & \cdots & -1.6161 & -0.7953 & 1.1050 & 1.5268 & \cdots & 1.9937 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1.5517 & \cdots & 0.7666 & -0.9149 & 0.9196 & 1.2929 & \cdots & -1.5311 \\ 0.2519 & \cdots & -0.9440 & 0.2369 & -1.7269 & -0.0645 & \cdots & -1.8339 \\ 0.4655 & \cdots & -0.0348 & -0.2256 & 0.9864 & -0.4456 & \cdots & -0.5346 \\ -1.3086 & \cdots & 1.1042 & -0.6232 & -0.7784 & 1.6999 & \cdots & -1.9451 \\ \vdots & \vdots \\ 0.1625 & \cdots & 0.7338 & 1.0089 & -1.3323 & -0.5756 & \cdots & -0.3930 \end{bmatrix}$$

dengan matriks fitur *query* sebagai berikut:

$$x_{4 \times 4} = \begin{bmatrix} 0.7666 & -0.9149 & 0.9196 & 1.2929 \\ -0.9440 & 0.2369 & -1.7269 & -0.0645 \\ -0.0348 & -0.2256 & 0.9864 & -0.4456 \\ 1.1042 & -0.6232 & -0.7784 & 1.6999 \end{bmatrix}$$

Selanjutnya untuk *Sub-window pooling* pada tingkat pertama dengan ukuran *Sub-window*  $s_w^1 = 1$  dan jumlah *Sub-window* horizontal dan vertikal  $s_r^1 = 8$  diperoleh matriks fitur  $x^1 = f_p^1(\hat{x}) \in \mathbb{R}^{8 \times 8 \times 1}$  dari sekitar jendela  $4 \times 4$  pusat sebagai berikut:

$$x^1 = \begin{bmatrix} 0.3511 & -0.5572 & -0.6627 & -0.3359 & -1.76 & 1.0448 & 1.5718 & 1.6408 \\ -0.4059 & -0.0419 & 0.042 & -0.6288 & 1.6866 & 1.9648 & 0.1245 & 1.1128 \\ 1.4674 & 1.3588 & 0.7666 & -0.9149 & 0.9196 & 1.2929 & 1.5551 & -0.4354 \\ 1.8113 & 0.0627 & -0.944 & 0.2369 & -1.7269 & -0.0645 & -1.1457 & 1.3114 \\ 1.3781 & 0.4277 & -0.0348 & -0.2256 & 0.9864 & -0.4456 & 0.8767 & -0.1435 \\ -1.9238 & 1.8468 & 1.1042 & -0.6232 & -0.7784 & 1.6999 & 1.1494 & 1.8545 \\ -1.6051 & -0.8954 & -0.586 & -0.4416 & 1.3466 & -0.1485 & 1.5671 & 0.957 \\ 1.8112 & -1.5396 & -0.81 & 1.1828 & -0.8285 & 0.8587 & -1.3392 & -0.9198 \end{bmatrix}$$

dikarenakan pada tingkat pertama ukuran *sub-window*  $1 \times 1$  di *pooling* ke ukuran  $1 \times 1$  pula, maka ukuran matriks fitur tidak berubah yaitu tetap  $\mathbb{R}^{8 \times 8 \times 1}$ .

Selanjutnya untuk *Sub-window pooling* pada tingkat kedua dengan ukuran *sub-window*  $s_w^2 = 2$  dan jumlah *sub-window* horizontal dan vertikal  $s_r^2 = 6$  diperoleh matriks fitur  $x^2 = f_p^2(\hat{x}) \in \mathbb{R}^{12 \times 12 \times 1}$  dari sekitar jendela  $4 \times 4$  pusat sebagai berikut:

$$x^2 = \begin{bmatrix} 1.4312 & \cdots & 1.5945 & 0.2707 & -1.5103 & -1.6415 & \cdots & 1.0218 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.4072 & \cdots & 0.7666 & -0.9149 & 0.9196 & 1.2929 & \cdots & -1.0740 \\ 1.0006 & \cdots & -0.9440 & 0.2369 & -1.7269 & -0.0645 & \cdots & 0.0962 \\ -0.8013 & \cdots & -0.0348 & -0.2256 & 0.9864 & -0.4456 & \cdots & -0.0084 \\ -0.9345 & \cdots & 1.1042 & -0.6232 & -0.7784 & 1.6999 & \cdots & -1.0112 \\ \vdots & \vdots \\ 0.8445 & \cdots & -0.2173 & 0.0360 & 1.3892 & -1.1765 & \cdots & 0.3010 \end{bmatrix}$$

kemudian pada tingkat kedua, ukuran *Sub-window*  $2 \times 2$  di *pooling* ke ukuran  $1 \times 1$  menggunakan *AveragePooling* dengan cara menghitung nilai rata-rata dari setiap sub-jendela. Oleh karena itu diperoleh matriks fitur  $\mathbb{R}^{6 \times 6 \times 1}$  sebagai berikut:

$$x^2 = \begin{bmatrix} 0.2853 & -0.2129 & 0.0049 & -1.1737 & -0.5642 & 0.3859 \\ -0.7655 & -0.1635 & -0.3964 & 0.5467 & 0.9635 & 0.0652 \\ -0.2338 & 1.1751 & -0.2139 & 0.1053 & 0.3214 & -0.0229 \\ -0.793 & 0.4322 & 0.0552 & 0.5884 & 0.0070 & -0.7037 \\ 0.3664 & -0.5572 & -0.0950 & 0.7102 & -0.2101 & 0.3985 \\ -0.0445 & 0.1126 & -0.1458 & 0.8818 & 0.1617 & 0.2471 \end{bmatrix}$$

Kemudian untuk *Sub-window pooling* pada tingkat ketiga dengan ukuran *Sub-window*  $s_w^3 = 4$  dan jumlah *Sub-window* horizontal dan vertikal  $s_r^3 = 5$  diperoleh matriks fitur  $x^3 = f_p^3(\hat{x}) \in \mathbb{R}^{20 \times 20 \times 1}$  seperti masukan awal matriks fitur dinyatakan dalam bentuk sebagai berikut:

$$x^3 = \begin{bmatrix} 0.8091 & \cdots & -1.6161 & -0.7953 & 1.1050 & 1.5268 & \cdots & 1.9937 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1.5517 & \cdots & 0.7666 & -0.9149 & 0.9196 & 1.2929 & \cdots & -1.5311 \\ 0.2519 & \cdots & -0.9440 & 0.2369 & -1.7269 & -0.0645 & \cdots & -1.8339 \\ 0.4655 & \cdots & -0.0348 & -0.2256 & 0.9864 & -0.4456 & \cdots & -0.5346 \\ -1.3086 & \cdots & 1.1042 & -0.6232 & -0.7784 & 1.6999 & \cdots & -1.9451 \\ \vdots & \vdots \\ 0.1625 & \cdots & 0.7338 & 1.0089 & -1.3323 & -0.5756 & \cdots & -0.3930 \end{bmatrix}$$

pada tingkat ketiga, ukuran *Sub-window*  $4 \times 4$  di *pooling* ke ukuran  $1 \times 1$  menggunakan *AveragePooling* dengan cara menghitung nilai rata-rata dari setiap sub-jendela. Sehingga diperoleh matriks fitur  $\mathbb{R}^{5 \times 5 \times 1}$  sebagai berikut:

$$x^3 = \begin{bmatrix} -1.0767 & -0.3691 & -0.4931 & 0.2292 & -0.0967 \\ 0.0109 & 0.4057 & -0.1153 & -0.1478 & -0.2166 \\ -0.4448 & -0.4708 & 0.1003 & 0.5682 & -0.2212 \\ -0.6298 & -0.1257 & -0.0583 & 0.4183 & -0.0465 \\ 0.0695 & 0.0427 & 0.0886 & -0.1489 & -0.2776 \end{bmatrix}$$

Setelah didapatkan semua matriks fitur dari setiap tingkat jendela perhatian, selanjutnya dilakukan *flatten* untuk mengubah bentuknya menjadi vektor satu dimensi. Pada jendela perhatian tingkat pertama matriks fitur yang sudah dilakukan *pooling* menghasilkan 64 token, untuk tingkat kedua menghasilkan 36 token, dan tingkat ketiga menghasilkan 25 token yang sudah di *flatten*.

Sehingga bentuk vektor satu dimensi dari setiap tingkat jendela perhatian dapat ditulis sebagai representasi vektor satu dimensi sebagai berikut:

$$x^1 = [0.3511 \ -0.5572 \ -0.6627 \ \dots \ 0.8587 \ -1.3392 \ -0.9198]$$

$$x^2 = [0.2853 \ -0.21295 \ 0.0049 \ \dots \ 0.8818 \ 0.1617 \ 0.2471]$$

$$x^3 = [-1.0767 \ -0.3691 \ -0.4931 \ \dots \ 0.0886 \ -0.1489 \ -0.2776]$$

Setelah dilakukan *flatten* maka ketiga *set* vektor berdimensi satu tersebut digabungkan sehingga menghasilkan 125 token yang selanjutnya dilakukan *linear projection* dan perhitungan *attention*. Setelah diperoleh matriks fitur yang sudah dilakukan *pooling* pada ketiga tingkat jendela perhatian  $(x^l)_1^L$  dihitung *query* pada tingkat pertama dan *key* serta *value* pada semua tingkat menggunakan ketiga proyeksi linier  $f_q$ ,  $f_k$ , dan  $f_v$  sebagaimana yang tertera pada Persamaan (2.12). Dengan menggunakan permisalan matriks fitur yang sama, maka dihitung setiap  $Q$ ,  $K$  dan  $V$  dengan anggapan  $f_q$ ,  $f_k$ ,  $f_v$  merupakan matriks identitas maka diperoleh:

$$Q = \begin{bmatrix} 0.7666 & -0.9149 & 0.9196 & 1.2929 \\ -0.9440 & 0.2369 & -1.7269 & -0.0645 \\ -0.0348 & -0.2256 & 0.9864 & -0.4456 \\ 1.1042 & -0.6232 & -0.7784 & 1.6999 \end{bmatrix}$$

$$K = [0.3511 \ -0.5572 \ -0.6627 \ \dots \ 0.0886 \ -0.1489 \ -0.2776]$$

$$V = [0.3511 \ -0.5572 \ -0.6627 \ \dots \ 0.0886 \ -0.1489 \ -0.2776]$$

Selanjutnya dilakukan perhitungan *attention* untuk setiap tingkat jendela perhatian menggunakan rumus 2.13 dan permisalan dimensi  $d = 4$  serta bias  $B$  merupakan matriks  $4 \times 4$  yang seluruh nilainya diatur sebesar 0.5. Maka dilakukan perhitungan *attention* pada tingkat pertama jendela perhatian dengan melakukan perkalian *dot product* antara  $Q_1 \cdot K_1^T$  terlebih dahulu menggunakan Persamaan 2.13.

$$\begin{bmatrix} 0.7666 & -0.9149 & 0.9196 & 1.2929 \\ -0.9440 & 0.2369 & -1.7269 & -0.0645 \\ -0.0348 & -0.2256 & 0.9864 & -0.4456 \\ 1.1042 & -0.6232 & -0.7784 & 1.6999 \end{bmatrix} \cdot \begin{bmatrix} 0.3511 & -0.5572 & -0.6627 & -0.3359 \\ -1.7600 & 1.0448 & 1.5718 & 1.6408 \\ -0.4059 & -0.0419 & 0.04200 & -0.6288 \\ 1.6866 & 1.9648 & 0.1245 & 1.1128 \end{bmatrix}$$

$$= \begin{bmatrix} -0.26476849 & 1.26171408 & -1.04718095 & 1.04858136 \\ 0.7026431 & -0.9112199 & 0.34127129 & -1.41346393 \\ -0.3905242 & 0.64482416 & 0.34520004 & -0.87500944 \\ 0.67978093 & -1.02880456 & -1.52367262 & 2.43261828 \end{bmatrix}$$

Kemudian hasil tersebut dibagi dengan  $\sqrt{d} = \sqrt{4}$  dan ditambahkan dengan Bias  $B$ . Sehingga diperoleh hasil:

$$\frac{Q_1 \cdot K_1^T}{\sqrt{4}} + B = \begin{bmatrix} 0.36761576 & 1.13085704 & -0.02359047 & 1.02429068 \\ 0.85132155 & 0.04439005 & 0.67063564 & -0.20673197 \\ 0.3047379 & 0.82241208 & 0.67260002 & 0.06249528 \\ 0.83989047 & -0.01440228 & -0.26183631 & 1.71630914 \end{bmatrix}$$

Hasil matriks tersebut kemudian diterapkan Softmax sehingga dipeoleh matriks

$$\text{Softmax}\left(\frac{Q_1 \cdot K_1^T}{\sqrt{4}} + B\right) = \begin{bmatrix} 0.17391823 & 0.37309255 & 0.11761058 & 0.33537864 \\ 0.38050975 & 0.16979304 & 0.31761054 & 0.13208667 \\ 0.2037644 & 0.34194113 & 0.29436678 & 0.15992769 \\ 0.24037527 & 0.10229989 & 0.07987593 & 0.57744892 \end{bmatrix}$$

Hasil dari Softmax tersebut dilakukan perkalian *dot product* dengan  $V_1$  sehingga menghasilkan keluaran *attention* sebagai berikut:

$$\text{Attention}(Q_1, K_1, V_1) = [-0.3374187 \quad -0.21586012 \quad -0.36778449 \quad -0.21950461]$$

dengan perhitungan yang sama untuk perhitungan *attention* tingkat kedua dan ketiga maka diperoleh keluaran *attention* sebagai berikut:

$$\text{Attention}(Q_2, K_2, V_2) = [1.21796999 \quad 1.34225786 \quad 1.37671225 \quad 1.15203369]$$

$$\text{Attention}(Q_3, K_3, V_3) = [1.77866077 \quad 1.76497099 \quad 1.76935628 \quad 1.78185008]$$

Setelah diperoleh keluaran *attention* dari semua tingkat, selanjutnya digabungkan hasil *attention* dari setiap tingkat menjadi satu representasi tunggal yang kemudian diteruskan ke tahap *Residual Connection* dan *LayerNorm* sebelum akhirnya masuk ke tahap *Multi-layer Perceptron*

#### 4. Multi-layer Perceptron

Tahap ini diterapkan setelah proses *Focal Self-Attention* untuk menghasilkan representasi akhir dari masukan yang sudah diproses. MLP berfungsi untuk meningkatkan kapasitas model dalam menangani masalah non-linear yang lebih rumit dengan penambahan fungsi aktivasi GELU dan memungkinkan model untuk mempelajari pola yang lebih kompleks. *Input* yang digunakan pada tahap ini adalah keluaran dari tahap *Focal Self-Attention* yang sudah dilakukan penambahan *residual* dan dilakukan normalisasi. Pada lapisan pertama *input* tersebut dikalikan dengan matriks bobot ukuran ( $W$ ) lalu ditambahkan dengan vektor bias ( $b$ ). *Output* dari lapisan pertama tersebut kemudian diterapkan fungsi aktivasi GELU dengan menggunakan Persamaan 2.14. Kemudian hasilnya diteruskan ke lapisan kedua untuk dilakukan perhitungan yang sama pada lapisan pertama. Sebagai contoh perhitungan, diberikan permisalan keluaran matriks dari lapisan pertama dan diterapkan aktivasi GELU sebagai berikut:

$$\begin{aligned} MLP(y) &= \text{GELU} \begin{bmatrix} 0.5678 & -1.2345 & 1.2345 & 0.6789 \\ -0.1234 & 2.3456 & -0.8765 & 0.9876 \\ 1.4567 & -0.6789 & 0.2345 & -0.3456 \\ -0.2345 & 0.6789 & -1.1234 & 1.3456 \end{bmatrix} \\ &= \begin{bmatrix} 0.6484 & -0.8944 & 1.0723 & 0.7886 \\ -0.1165 & 2.2773 & -0.8209 & 0.9225 \\ 1.1589 & -0.7110 & 0.2349 & -0.3497 \\ -0.2274 & 0.7734 & -1.1079 & 1.1203 \end{bmatrix} \end{aligned}$$

Keluaran dari tahap MLP ini diteruskan ke tahap *Residual Connection* sebelum masuk ke tahap *Patch Embedding* pada lapisan kedua model *Focal Transformer*.

#### 5. Global Average Pooling

Bagian ini diterapkan untuk mereduksi dimensi spatial dari *input* dengan cara mengambil rata-rata dari sekelompok nilai di dalam suatu area tertentu. Tujuannya adalah untuk menghasilkan representasi global yang lebih ringkas dan lebih stabil dari fitur *input*, sementara tetap mempertahankan informasi penting. *Global Average Pooling* secara dinamis menghitung rata-rata dari setiap peta fitur di sepanjang dimensi *patch* untuk menghasilkan output dengan ukuran yang diinginkan. *Input* untuk tahap ini diperoleh dari matriks fitur yang sudah melewati keempat lapisan dari model *Focal Transformer*, dimana setiap lapisan tersebut menambahkan dimensinya menjadi dua kali lipat lebih besar dan juga mengurangi jumlah *patch* empat kali lebih kecil. Sehingga diperoleh *input* untuk tahap

ini memiliki ukuran  $\mathbb{R}^{64 \times 49 \times 768}$ , *Global Average Pooling* menghitung rata-rata di sepanjang dimensi *patch* dengan panjang 49 untuk setiap dimensi yang memiliki panjang 768. Sehingga diperoleh hasil akhir dari tahap ini berupa  $\mathbb{R}^{64 \times 768 \times 1}$ . Secara matematis, proses perhitungan tahap ini dapat dituliskan dengan Persamaan sebagai berikut:

$$v_{i,k,j} = \frac{1}{49} \sum_{k=1}^{49} x_{i,k,j}$$

dengan,

- $x_{i,k,j}$  = elemen input tensor  $x$  dengan indeks  $i$  (*batch*),  $k$  (*patch*),  $j$  (dimensi).
- $v_{i,k,j}$  = nilai output yang merupakan rata-rata dari seluruh elemen di sepanjang dimensi *patch*.

Sebagai contoh salah satu perhitungan pada batch pertama dan dimensi pertama, diberikan *input* sebagai berikut:

$$v_{1,k,1} = [2.1234 \ 2.5678 \ 2.9101 \ \dots \ 3.2345 \ 2.7654 \ 3.2345]$$

Kemudian dihitung rata-ratanya atau dapat dituliskan dengan Persamaan sebagai berikut:

$$v_{1,1,1} = \frac{1}{49} \sum_{k=1}^{49} x_{1,k,1}$$

$$v_{1,1,1} = \frac{1}{49} (2.1234 + 2.5678 + 2.9101 + \dots + 3.2345 + 2.7654 + 3.2345)$$

$$v_{1,1,1} = 141.9344$$

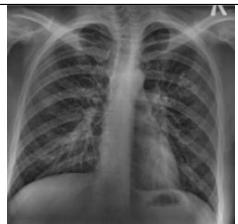
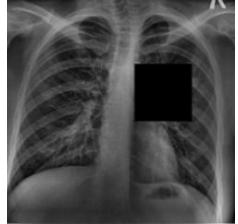
Perhitungan yang sama diterapkan pada setiap dimensi sehingga hasil akhir diperoleh matriks berbentuk  $v \in \mathbb{R}^{768 \times 1}$ . Hasil tersebut diteruskan ke tahap *linear* untuk dihitung nilai probabilitas yang menunjukkan kemungkinan citra tersebut termasuk dalam kelas tuberkulosis atau normal. Proses tersebut dilakukan dengan cara mengoperasikannya dengan bobot  $W$  berukuran [768, 2] dan bias  $B$  berukuran [1, 2] sehingga hasil dari lapisan tersebut memiliki dimensi [1, 2] yang kemudian diterapkan fungsi Softmax untuk mengubah skor tersebut menjadi nilai probabilitas.

#### 4.4 Pengujian Model

Pada tahap ini, model diuji dengan beberapa kondisi citra tanpa gangguan dari dataset dan juga yang telah diberi gangguan untuk mengevaluasi kemampuannya dalam

mempertahankan akurasi klasifikasi. Pengujian ini bertujuan untuk mengamati apakah model yang telah dilatih mampu mengenali pola yang relevan pada citra yang memiliki variasi kondisi. Kondisi citra yang digunakan dalam pengujian mencakup beberapa jenis gangguan atau distorsi, antara lain *Gaussian Noise*, *Salt and Pepper Noise*, *Motion Blur*, *Brightness and Contrast Distortion*, dan *Partial Occlusion*. Kondisi tersebut merupakan contoh gangguan nyata yang sering muncul pada citra medis, seperti gambar *X-ray*. Tabel 4.1 memberikan penjelasan lebih lanjut mengenai masing-masing kondisi citra yang dimaksud, termasuk tujuan pengujian dan efek yang diharapkan dari setiap jenis gangguan tersebut.

Tabel 4.1 Deskripsi dan Contoh Kondisi Citra

Kondisi Citra	Deskripsi	Tujuan	Contoh Citra
<i>Gaussian Noise</i>	Simulasi noise acak akibat sensor scanner	Menguji sensitivitas terhadap <i>noise</i> normal acak yang umum dalam pencitraan X-Ray	
<i>Salt and Pepper Noise</i>	Titik-titik putih dan hitam acak yang bisa muncul akibat kesalahan transmisi atau pengambilan	Menguji ketahanan terhadap gangguan biner ekstrim	
<i>Motion Blur</i>	Meniru efek gerakan pasien selama proses X-Ray	Menguji pengaruh kabur karena artefak gerakan terhadap prediksi model	
<i>Brightness and Contrast Distortion</i>	Perubahan intensitas keseluruhan, simulasi dari variasi mesin atau kondisi pencahayaan	Melihat apakah model tetap mengenali pola penting meskipun intensitas berubah	
<i>Partial Occlusion</i>	Sebagian area gambar ditutupi (simulasi gambar rusak atau sensor tertutup)	Menguji kemampuan model mengenali pola meskipun sebagian citra tertutup	

## 4.5 Perancangan Sistem

Pada tahap ini dijelaskan mengenai perancangan yang dilakukan pada penelitian. Perancangan tersebut meliputi proses pelatihan, proses evaluasi untuk mencari model dengan performa tertinggi, serta perancangan pengujian model klasifikasi tuberkulosis pada citra *X-ray*.

### 4.5.1 Perancangan Pelatihan Model

Pelatihan Model dilakukan menggunakan *Kaggle GPU P100* dan *Visual Studio Code GPU GeForce RTX 400*. Model *Focal Transformer* ini memiliki beberapa versi, dengan setiap versi memiliki jumlah parameter, *Embedding Dim*, dan *Depths* yang berbeda. Pada penelitian ini digunakan 3 variasi versi model untuk eksperimen dengan rincian yang ditunjukkan pada Tabel 4.2 sebagai berikut:

Tabel 4.2 Variasi Model *Focal Transformer*

Variasi	Parameter (Juta)	<i>Embedding Dim</i>	<i>Depths</i>
Focal-Tiny	29.1	96	2,2,6,2
Focal-Small	51.1	96	2,2,18,2
Focal-Base	89.8	128	2,2,18,2

Pelatihan model *Focal Transformer* yang digunakan untuk klasifikasi citra *X-ray* diterapkan *hyperparameter* seperti pada Tabel Tabel 4.3

Tabel 4.3 *Hyperparameter* Pelatihan

<i>Hyperparameter</i>	Nilai
<i>epoch</i>	20, 30
<i>batch size</i>	32, 64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$1e - 5$ , $3e - 5$ , $5e - 5$
<i>weight decay</i>	$3e - 3$ , $1e - 3$ , $3e - 2$ , $1e - 2$
<i>patience</i>	3
<i>factor</i>	0.3
<i>dropout</i>	0.5

### 4.5.2 Perancangan Pengujian Model

Pada Pengujian Model dilakukan seperti yang dijelaskan pada sub bab 4.4 dengan beberapa kondisi citra yang disebutkan. Adapun kondisi citra tersebut diatur sedemikian rupa agar tidak terlalu ekstrim sehingga masih relevan dengan kondisi nyata pada kehidupan sehari-hari dan tidak merusak struktur dari citra itu sendiri. Sehingga pada penelitian ini diterapkan pengaturan kondisi citra yang ditunjukkan pada Tabel 4.4 sebagai berikut.

Tabel 4.4 Pengaturan Kondisi Citra

Kondisi Citra	Keterangan
<i>Gaussian Noise</i>	mean=0, std=0.3
<i>Salt and Pepper</i>	amount=0.08, salt pepper=0.5
<i>Motion Blur</i>	degree=30, angle=90
<i>Brightness and Contrast Distortion</i>	brightness=1.2, contrast=0.8
<i>Partial Occlusion</i>	size=128

## 4.6 Implementasi Program

Pada bagian ini ditunjukkan implementasi program dalam pelatihan model dan evaluasi model.

### 4.6.1 Implementasi Pelatihan Model

Berikut ditampilkan kode program untuk pelatihan model klasifikasi citra menggunakan model *Focal Transformer*.

```

1 model = FocalTransformer(
2     img_size=224, patch_size=4, in_chans=3, embed_dim=96, depths=[2, 2,
3     18, 2],
4     num_heads=[3, 6, 12, 24], window_size=7, mlp_ratio=4.0
5 ).cuda()
6
6 model.head = nn.Sequential(
7     nn.Dropout(p=0.5),
8     nn.Linear(model.num_features, 2)
9 ).cuda()
10
11 criterion = nn.CrossEntropyLoss(weight=class_weights)
12 optimizer = optim.AdamW(model.parameters(), lr=3e-5, weight_decay=1e-2)
13 scheduler = ReduceLROnPlateau(optimizer, mode='max', patience=3, factor
14     =0.3)
14 model = model.to(device)

```

Kode Program 4.1 Pelatihan Model

Kode Program 4.1 merupakan kode untuk melakukan pelatihan model dengan menginisialisasi model yang sudah dibentuk sebelumnya beserta jumlah *embedding dim*, *depths*, dan *num heads*. Kemudian diinisialisasikan juga nilai parameter yang ingin digunakan pada proses pelatihan model seperti nilai *dropout*, *learning rate*, *weight decay*, *patience*, dan juga *factor*. Pada kode diatas juga diinisialisasi penggunaan *criterion*, *optimizer*, dan *scheduler* yang dalam implementasi ini menggunakan *CrossEntropyLoss*, *AdamW*, dan *ReduceLROnPlateau*.

#### 4.6.2 Implementasi Pengujian Model

Berikut ditampilkan kode program untuk pelatihan model klasifikasi citra menggunakan model *Focal Transformer*.

```
1 model.eval()
2 y_true = []
3 y_pred = []
4
5 with torch.no_grad():
6     for inputs, labels in test_loader:
7         inputs, labels = inputs.cuda(), labels.cuda()
8         outputs = model(inputs)
9         _, preds = torch.max(outputs, 1)
10        y_true.extend(labels.cpu().numpy())
11        y_pred.extend(preds.cpu().numpy())
12
13 # Konversi ke NumPy array
14 y_true = np.array(y_true)
15 y_pred = np.array(y_pred)
16 # Confusion Matrix
17 cm = confusion_matrix(y_true, y_pred)
18 classes = ['NORMAL', 'TUBERCULOSIS']
19 plt.figure(figsize=(6, 5))
20 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
21             yticklabels=classes)
22 plt.xlabel('Predicted')
23 plt.ylabel('Actual')
24 plt.title('Confusion Matrix')
25 plt.show()
26 # Classification Report
27 print("Classification Report:")
28 print(classification_report(y_true, y_pred, target_names=classes))
29 # Accuracy
30 acc = accuracy_score(y_true, y_pred)
31 print(f"Accuracy: {acc*100:.2f}%")
```

Kode Program 4.2 Pengujian Model

Kode Program 4.2 diatas merupakan kode untuk melakukan pengujian model dengan cara menyimpan hasil prediksi kelas klasifikasi dan label asli dari data yang sudah di *split* di dalam *test loader*. Kemudian hasil prediksi tersebut dikonversi ke dalam fomat *array* untuk ditampilkan pada *confusion matrix* dan *classification report* untuk dilihat jumlah prediksi yang benar dan salah serta hasil metrik evaluasi dari masing-masing kelas kategori normal dan tuberkulosis.

## BAB V

# HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan terkait hasil dari penelitian yang sesuai dengan perancangan dan implementasi pada bab sebelumnya. Hasil yang didapat selanjutnya dianalisa sehingga memberikan kesimpulan penelitian yang sesuai dengan tujuan penelitian.

### 5.1 Hasil Pelatihan

Pada sub bab ini dibahas terkait hasil dari pelatihan yang telah dilakukan untuk ketiga variasi model *Focal Transformer* yaitu *Focal-Tiny*, *Focal-Small*, dan *Focal-Base*. Pelatihan model tersebut dilakukan dengan menggunakan citra data latih dan dievaluasi menggunakan data validasi yang sudah melalui tahap pra-pemrosesan citra yang dijelaskan pada sub bab 4.2 terkait pra-pemrosesan citra.

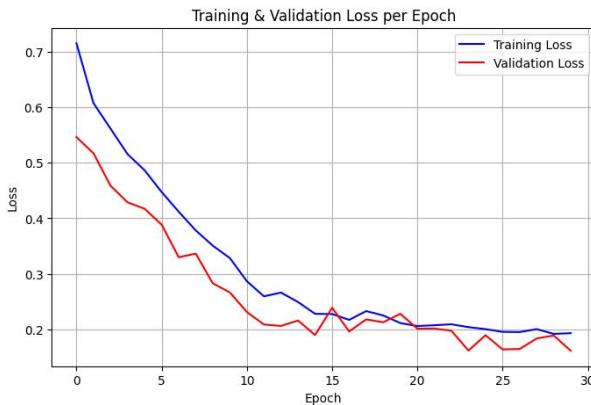
Hasil dari pelatihan ketiga variasi model *Focal Transformer* dilakukan berdasarkan *hyperparameter* yang ditunjukkan pada Tabel 4.3. Pelatihan model dilakukan melalui beberapa percobaan dengan kombinasi *hyperparameter* yang berbeda hingga diperoleh grafik *loss* yang paling optimal dibandingkan percobaan lainnya. Berikut ditampilkan beberapa *hyperparameter* yang digunakan beserta hasil grafik *loss* dari seluruh percobaan yang telah dilakukan untuk setiap variasi model.

#### 5.1.1 Pelatihan Pertama Model *Focal-Tiny*

Nilai *hyperparameter* yang diterapkan tercantum pada Tabel 5.1 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.1.

Tabel 5.1 *Hyperparameter* Pelatihan Pertama Model *Focal-Tiny*

<i>Hyperparameter</i>	Nilai
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$2e - 3$
<i>patience</i>	5
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.1 Grafik Pelatihan Pertama Model *Focal-Tiny*

Berdasarkan grafik yang ditampilkan, dapat dilihat bahwa nilai *loss* pada data pelatihan maupun data validasi mengalami penurunan yang konsisten seiring bertambahnya jumlah *epoch*. Penurunan ini menunjukkan bahwa model mampu mempelajari pola dalam data pelatihan secara bertahap sekaligus mempertahankan kinerja yang baik terhadap data validasi.

Pada tahap awal pelatihan, nilai *training loss* dan *validation loss* berada pada tingkat yang relatif tinggi, yaitu sekitar 0,70 dan 0,55. Seiring berjalannya proses pelatihan, kedua nilai tersebut menurun secara progresif hingga mencapai kisaran 0,20 pada *epoch* akhir. Penurunan yang seimbang pada kedua *loss* ini mengindikasikan bahwa model tidak hanya mampu mengoptimalkan prediksi terhadap data pelatihan, tetapi juga dapat mempertahankan kemampuan generalisasi terhadap data yang tidak digunakan selama pelatihan, yaitu data validasi.

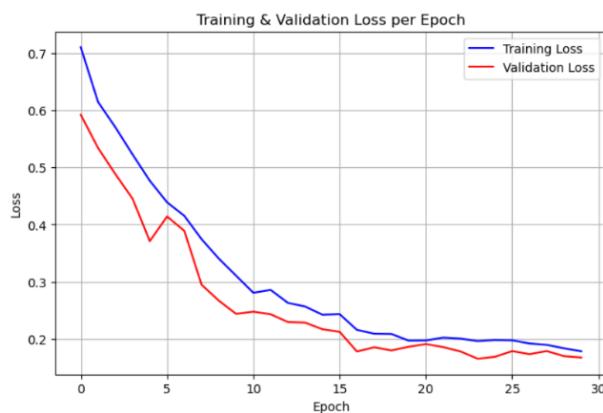
Mulai dari sekitar *epoch* ke-10 hingga menjelang akhir pelatihan, nilai *validation loss* secara konsisten lebih rendah dibandingkan *training loss*. Pada *epoch* terakhir, terlihat adanya indikasi awal *overfitting*, ditandai dengan peningkatan nilai *training loss* yang disertai penurunan nilai *validation loss*. Hal ini dapat mengindikasikan bahwa model mulai kehilangan kemampuan untuk lebih lanjut mengoptimalkan data pelatihan tanpa mengorbankan kinerjanya terhadap data validasi.

### 5.1.2 Pelatihan Kedua Model *Focal-Tiny*

Nilai *hyperparameter* yang diterapkan tercantum pada Tabel 5.2 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.2.

Tabel 5.2 *Hyperparameter* Pelatihan Kedua Model *Focal-Tiny*

<b><i>Hyperparameter</i></b>	<b>Nilai</b>
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$1e - 2$
<i>patience</i>	3
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.2 Grafik Pelatihan Kedua Model *Focal-Tiny*

Berdasarkan grafik yang ditampilkan, dapat dilihat perkembangan *training loss* dan *validation loss* sepanjang *epoch* pelatihan model *Focal-Tiny*. Pada awal pelatihan, kedua nilai *loss* menunjukkan angka yang relatif tinggi. Namun, seiring bertambahnya jumlah *epoch*, *training loss* mengalami penurunan yang signifikan dan cenderung stabil pada nilai sekitar 0.2 setelah *epoch* ke-20.

Sementara itu, *validation loss* menunjukkan pola yang serupa dengan *training loss*, meskipun dengan fluktuasi yang lebih besar pada awal pelatihan. Fluktuasi ini mungkin disebabkan oleh variasi yang terdapat pada data validasi atau model yang belum sepenuhnya stabil pada fase awal pelatihan. Namun, setelah beberapa *epoch*, *validation loss* juga mengalami penurunan yang stabil, meskipun tidak secepat penurunan yang terjadi pada *training loss*. Hal ini menunjukkan bahwa model mulai belajar dengan baik pada data pelatihan, namun masih mengalami kesulitan dalam generalisasi pada data validasi pada tahap awal pelatihan.

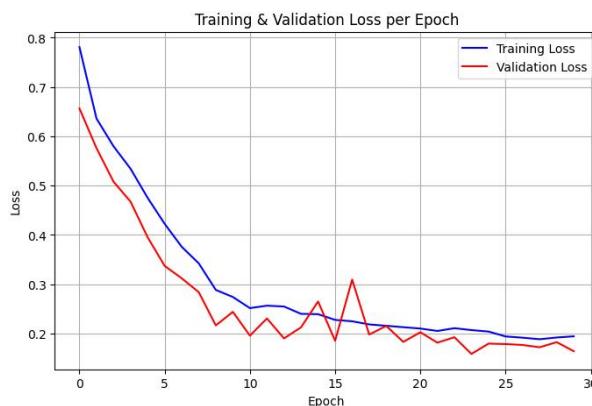
Secara keseluruhan, grafik ini menunjukkan bahwa model *Focal-Tiny* menunjukkan konvergensi yang baik, dengan penurunan *training loss* dan *validation loss* yang semakin mendekati nilai yang lebih rendah seiring berjalannya waktu. Hal ini mengindikasikan bahwa model mampu mempelajari pola dari data pelatihan dan validasi dengan semakin baik.

### 5.1.3 Pelatihan Pertama Model *Focal-Small*

Nilai *hyperparameter* yang diterapkan tercantum pada Tabel 5.3 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.3.

Tabel 5.3 *Hyperparameter* Pelatihan Pertama Model *Focal-Small*

Hyperparameter	Nilai
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$2e - 3$
<i>patience</i>	5
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.3 Grafik Pelatihan Pertama Model *Focal-Small*

Berdasarkan grafik di atas, dapat dilihat bahwa nilai *loss* pada data pelatihan dan data validasi secara umum menunjukkan penurunan seiring bertambahnya jumlah *epoch*. Hal ini mencerminkan bahwa proses pelatihan berjalan secara progresif, dengan model yang secara bertahap mampu meminimalkan kesalahan prediksi baik pada data pelatihan maupun data validasi.

Pada awal proses pelatihan, nilai *loss* untuk data pelatihan berada pada kisaran 0,78, sementara untuk data validasi sekitar 0,66. Kedua nilai tersebut menurun secara konsisten hingga mencapai nilai mendekati 0,2 pada akhir pelatihan, menunjukkan bahwa model berhasil belajar dengan baik dari data yang tersedia. Meskipun terdapat sedikit fluktuasi pada *validation loss* setelah *epoch* ke-10, nilai tersebut tetap berada dalam rentang yang stabil dan tidak menunjukkan pola peningkatan yang signifikan.

Kondisi di mana *validation loss* cenderung lebih rendah dibandingkan *training loss*, khususnya setelah pertengahan pelatihan, mengindikasikan bahwa model memiliki kemampuan generalisasi yang cukup baik. Namun, pada *epoch* terakhir, terdapat

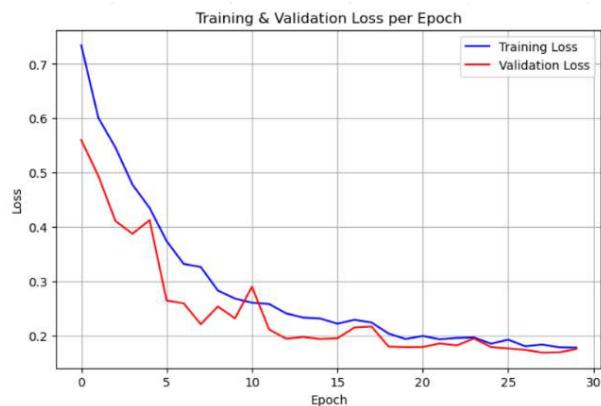
indikasi awal *overfitting* yang ditandai dengan meningkatnya nilai training loss sementara validation loss justru mengalami penurunan. Hal ini mengindikasikan bahwa model mulai kehilangan konsistensi dalam mengoptimalkan kinerja terhadap data pelatihan tanpa disertai peningkatan terhadap data validasi.

#### 5.1.4 Pelatihan Kedua Model *Focal-Small*

Pada pelatihan ini diterapkan nilai *hyperparameter* yang ditunjukkan pada Tabel 5.4 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.4.

Tabel 5.4 *Hyperparameter* Pelatihan Kedua Model *Focal-Small*

<i>Hyperparameter</i>	Nilai
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$1e - 2$
<i>patience</i>	3
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.4 Grafik Pelatihan Kedua Model *Focal-Small*

Pada grafik tersebut, yang menunjukkan *training loss* dan *validation loss* setiap *epoch* untuk model *Focal-Small*, terlihat adanya penurunan yang signifikan pada kedua *loss* sejak *epoch* pertama hingga mencapai titik konvergen. *Training loss* dimulai dengan nilai yang cukup tinggi pada awal pelatihan, namun mengalami penurunan yang cepat setelah beberapa *epoch* pertama, kemudian stabil pada nilai yang lebih rendah di akhir pelatihan. Hal ini mengindikasikan bahwa model dapat mempelajari pola dari data pelatihan dengan efektif.

*Validation loss* menunjukkan pola yang serupa dengan *training loss*, meskipun terdapat fluktuasi yang sedikit lebih besar pada beberapa *epoch* awal. Fluktuasi ini mungkin disebabkan oleh proses adaptasi model terhadap data yang belum pernah dilihat sebelumnya, atau karena adanya variasi pada data validasi yang relatif lebih kecil. Meskipun demikian, *validation loss* menunjukkan penurunan yang konsisten, yang mengindikasikan bahwa model tidak mengalami *overfitting*.

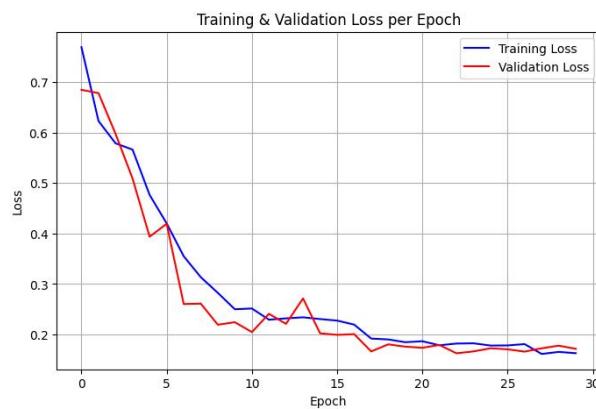
Berdasarkan analisis tersebut, grafik ini menunjukkan bahwa model ini memiliki kinerja yang baik, dengan konvergensi yang stabil dan kemampuan untuk mengurangi tingkat kesalahan baik pada data pelatihan maupun data validasi seiring berjalannya waktu.

### 5.1.5 Pelatihan Pertama Model *Focal-Base*

Pada pelatihan diterapkan nilai *hyperparameter* yang ditunjukkan pada Tabel 5.5 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.5.

Tabel 5.5 *Hyperparameter* Pelatihan Pertama Model *Focal-Base*

<b>Hyperparameter</b>	<b>Nilai</b>
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$2e - 3$
<i>patience</i>	5
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.5 Grafik Pelatihan Pertama Model *Focal-Base*

Berdasarkan grafik yang ditampilkan, terlihat bahwa nilai *loss* pada data pelatihan dan validasi mengalami penurunan seiring dengan bertambahnya jumlah *epoch*. Hal ini menandakan bahwa model berhasil belajar dalam meminimalkan kesalahan prediksi selama proses pelatihan.

Pada awal pelatihan, nilai *loss* berada pada kisaran yang relatif tinggi, yakni sekitar 0,76 untuk data pelatihan dan 0,70 untuk data validasi. Namun, kedua kurva menunjukkan penurunan yang signifikan pada lima *epoch* pertama, kemudian perlahan-lahan menurun hingga mendekati nilai 0,18 pada akhir pelatihan. Hal tersebut menunjukkan bahwa model memiliki kemampuan generalisasi yang baik terhadap data validasi.

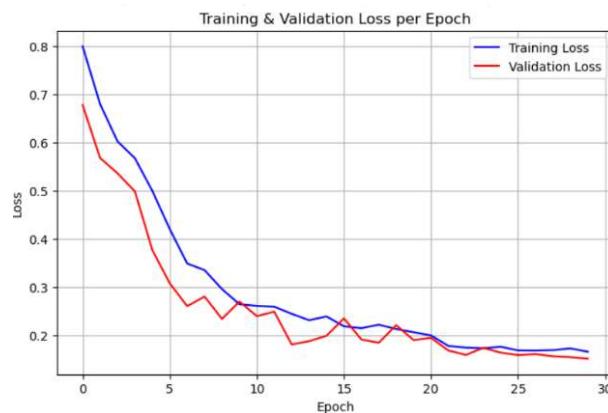
Namun, pada beberapa *epoch* terakhir, nilai *validation loss* cenderung stagnan dan tidak menunjukkan penurunan yang signifikan, sementara *training loss* masih mengalami sedikit penurunan. Pola tersebut menunjukkan bahwa proses pelatihan mulai mencapai tahap akhir, di mana kemampuan model untuk memperbaiki kinerjanya terhadap data validasi menjadi semakin terbatas.

#### 5.1.6 Pelatihan Kedua Model *Focal-Base*

Pada pelatihan diterapkan nilai *hyperparameter* yang ditunjukkan pada Tabel 5.6 dan diperoleh grafik pelatihan yang ditunjukkan pada Gambar 5.6.

Tabel 5.6 *Hyperparameter* Pelatihan Kedua Model *Focal-Base*

<i>Hyperparameter</i>	Nilai
<i>epoch</i>	30
<i>batch size</i>	64
<i>optimizer</i>	AdamW
<i>learning rate</i>	$3e - 5$
<i>weight decay</i>	$1e - 2$
<i>patience</i>	3
<i>factor</i>	0.3
<i>dropout</i>	0.5



Gambar 5.6 Grafik Pelatihan Kedua Model *Focal-Base*

Grafik pelatihan tersebut memperlihatkan perbandingan antara *training loss* dan *validation loss* sepanjang *epoch* selama proses pelatihan model. Pada grafik ini, terlihat bahwa kedua *training loss* dan *validation loss* mengalami penurunan yang konsisten dari awal hingga akhir pelatihan. *Training loss* menurun secara stabil, menunjukkan bahwa model berhasil belajar dari data pelatihan dengan baik.

Sementara itu, *validation loss* juga mengalami penurunan yang serupa, meskipun sedikit lebih lambat, yang mengindikasikan bahwa model mampu menggeneralisasi pola yang dipelajari dari data pelatihan ke data yang tidak terlihat dengan lebih efektif. Tidak terdapat fluktuasi atau lonjakan besar pada *validation loss*, yang menunjukkan bahwa model tidak memiliki indikasi *overfitting* dan mampu mempertahankan kinerjanya dengan baik pada data validasi. Penurunan yang konsisten pada kedua *loss* ini menandakan bahwa model ini tidak hanya belajar dengan baik pada data pelatihan, tetapi juga dapat menggeneralisasi dengan baik pada data yang tidak terlihat sebelumnya, sehingga model ini cenderung stabil dan fit.

## 5.2 Hasil Pengujian

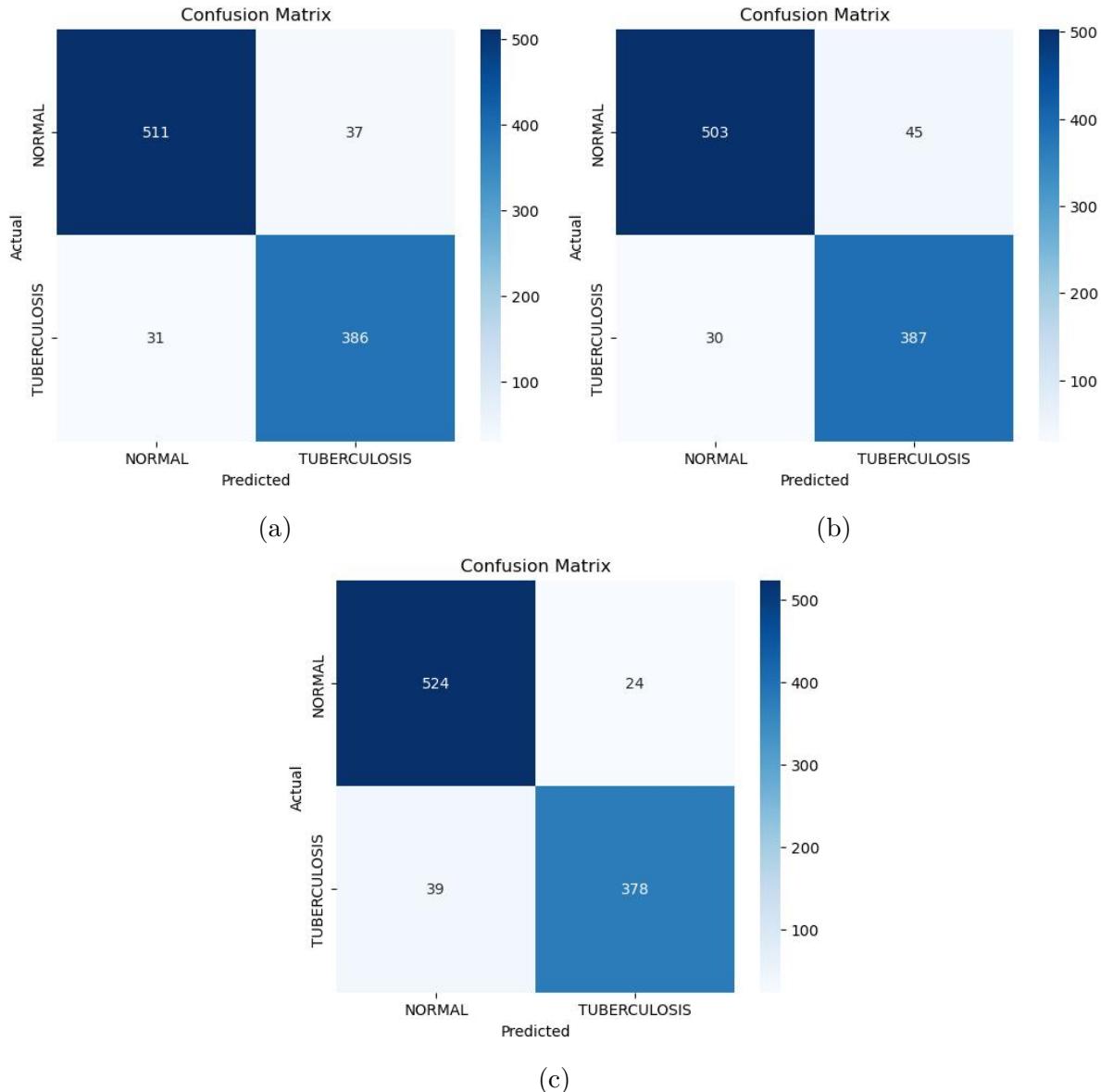
Setelah melakukan pelatihan model dan mendapatkan kinerja dari model, dilakukan tahap pengujian terhadap model tersebut. Pengujian hasil klasifikasi citra untuk penyakit tuberkulosis pada citra *X-ray* dada ini menggunakan metrik evaluasi berupa *Confusion Matrix* sehingga didapatkan hasil akurasi, presisi, sensitivitas, dan *F1-Score*. Perhitungan metrik evaluasi tersebut dihitung satu persatu menggunakan perhitungan yang tercantum pada sub bab 2.8 untuk setiap variasi model *Focal Transformer* dengan jumlah data uji sebanyak 548 citra untuk kategori normal dan 417 citra untuk kategori tuberkulosis.

### 5.2.1 Hasil Pengujian Model *Focal Transformer* pada Citra Tanpa Gangguan

Pada tahap ini, model dilakukan pengujian menggunakan citra dengan kondisi tanpa gangguan untuk melihat kinerja dari masing-masing variasi model *Focal Transformer*. Pengujian tersebut dilakukan untuk mencari variasi model yang terbaik untuk dilakukan pengujian lebih lanjut pada citra kondisi gangguan tertentu.

Tabel 5.7 Kinerja *Focal Transformer* pada Citra Tanpa Gangguan

Metrik Evaluasi	Label	Model		
		Tiny	Small	Base
Presisi	Normal	0.9428	<b>0.9437</b>	0.9307
	TB	0.9125	0.8958	<b>0.9403</b>
Sensitivitas	Normal	0.9325	0.9179	<b>0.9562</b>
	TB	0.9257	<b>0.9281</b>	0.9065
<b>F1-Score</b>	Normal	0.9376	0.9305	<b>0.9433</b>
	TB	0.9189	0.9113	<b>0.9231</b>
<b>Akurasi</b>		92.95%	92.23%	<b>93.47%</b>



Gambar 5.7 *Confusion Matrix Focal Transformer* pada Citra Tanpa Gangguan: (a) *Focal-Tiny*, (b) *Focal-Small*, (c) *Focal-Base*

Berdasarkan Gambar 5.7 dan metrik evaluasi pada Tabel 5.7, kinerja model *Focal Transformer* dalam klasifikasi penyakit tuberkulosis pada citra *X-ray* dada menunjukkan kinerja yang sangat baik secara keseluruhan. Ketiga varian model *Focal-Tiny*, *Focal-Small*, dan *Focal-Base* mampu memberikan akurasi yang tinggi, yaitu di atas 91%.

Pada model *Focal-Tiny* yang ditunjukkan pada Gambar 5.7a, model ini menunjukkan kemampuan yang cukup baik dalam mengklasifikasikan kedua kelas, yaitu normal dan tuberkulosis. Dengan akurasi sebesar 92.95%, model ini menunjukkan hasil yang baik meskipun terdapat kesalahan klasifikasi pada kedua kelas. Model ini berhasil mengklasifikasikan 511 citra normal dan 386 citra tuberkulosis dengan benar, namun terdapat 37 citra normal yang salah diklasifikasikan sebagai tuberkulosis, yang menunjukkan bahwa model ini cenderung sedikit mengklasifikasikan citra tuberkulosis sebagai normal. Model ini juga salah mengklasifikasikan 31 citra tuberkulosis sebagai normal. Meskipun demikian, model ini memiliki presisi sebesar 0.9428 untuk kelas normal dan presisi sebesar 0.9125 untuk kelas tuberkulosis serta sensitivitas sebesar 0.9325 dan 0.9257 untuk masing-masing kelas, dengan *F1-Score* untuk kelas normal sebesar 0.9376 dan untuk tuberkulosis sebesar 0.9189, yang menunjukkan bahwa model ini dapat mempertahankan keseimbangan yang baik antara presisi dan sensitivitas.

Model *Focal-Small* yang ditunjukkan pada Gambar 5.7b menunjukkan kinerja yang hampir serupa dengan model *Tiny*, namun dengan sedikit penurunan. Model ini memiliki akurasi 92.23%, yang sedikit lebih rendah dibandingkan dengan model *Tiny*. Untuk kelas normal, model ini berhasil mengklasifikasikan 503 citra normal dengan benar, tetapi ada 45 citra normal yang salah diklasifikasikan sebagai tuberkulosis. Sementara itu, untuk kelas tuberkulosis, model ini mengidentifikasi 387 citra tuberkulosis dengan benar, namun terdapat 30 citra tuberkulosis yang salah diklasifikasikan sebagai normal. Meskipun model ini sedikit dibawah dalam hal akurasi dibandingkan dengan model *Tiny*, nilai presisi dan sensitivitas untuk kedua kelas tetap tinggi, dengan presisi untuk kelas normal sebesar 0.9437 dan sensitivitas sebesar 0.9179. Untuk kelas tuberkulosis, model ini menunjukkan presisi 0.8958 dan sensitivitas 0.9281, yang tetap sangat baik meskipun ada sedikit penurunan dibandingkan dengan model *Focal-Tiny*.

Model *Focal-Base* yang ditampilkan pada Gambar 5.7c menunjukkan hasil yang paling optimal di antara ketiga varian. Dengan akurasi tertinggi sebesar 93.47%, model ini menunjukkan kinerja terbaik. Model ini berhasil mengklasifikasikan 524 citra normal dengan benar dan hanya 24 citra normal yang salah diklasifikasikan sebagai tuberkulosis. Untuk kelas tuberkulosis, model ini berhasil mengidentifikasi 378 citra tuberkulosis dengan benar, meskipun ada 39 citra tuberkulosis yang salah diklasifikasikan sebagai normal. Model ini memiliki nilai presisi dan sensitivitas yang sangat baik pada kedua kelas, dengan presisi untuk kelas normal sebesar 0.9307 dan sensitivitas sebesar 0.9562, yang menunjukkan bahwa model ini sangat baik dalam mengidentifikasi citra normal dan memiliki tingkat kesalahan yang rendah pada kelas ini. Meskipun ada penurunan pada kelas tuberkulosis, model ini tetap menunjukkan presisi 0.9403 dan sensitivitas 0.9065, yang masih sangat baik.

Dalam konteks klasifikasi penyakit, presisi dan sensitivitas pada kelas tuberkulosis sangat penting untuk meminimalkan kesalahan diagnosis. Kedua metrik ini berperan dalam memastikan bahwa kasus tuberkulosis dapat terkласifikasi dengan benar. Presisi mengukur proporsi hasil klasifikasi yang benar-benar merupakan tuberkulosis, sementara sensitivitas mengukur seberapa banyak kasus tuberkulosis yang berhasil terdeteksi oleh model. Kedua metrik ini sangat relevan untuk penyakit menular seperti tuberkulosis, di mana diagnosis dini sangat penting untuk mencegah penyebaran penyakit.

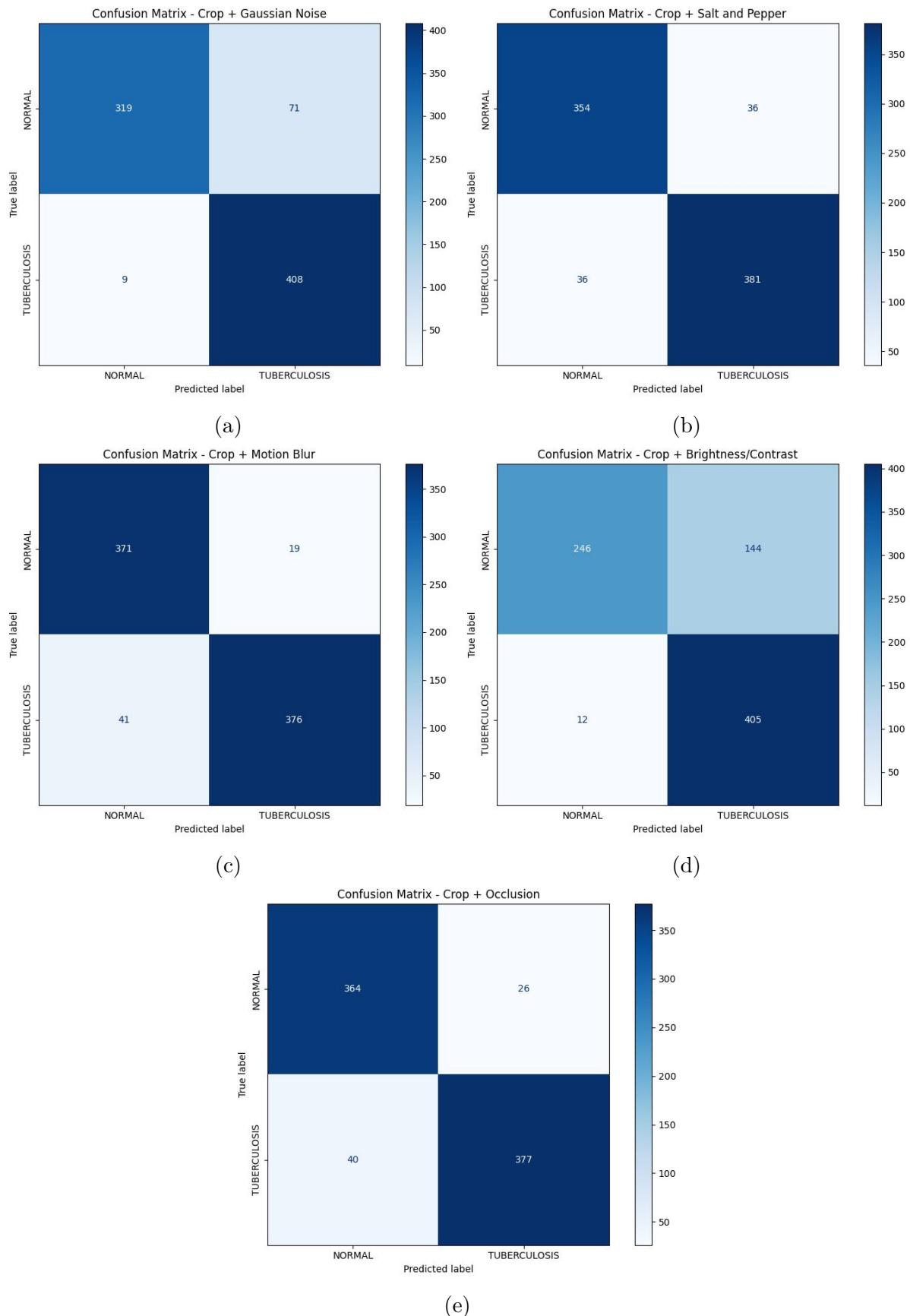
Oleh karena itu, berdasarkan Tabel 5.7, model *Focal-Base* menunjukkan hasil terbaik pada kedua metrik tersebut, dengan nilai presisi tuberkulosis tertinggi dibandingkan variasi lainnya yaitu sebesar 0.9403 dan sensitivitas tuberkulosis sebesar 0.9065. Walaupun nilai sensitivitas *Focal-Base* tersebut lebih rendah dibandingkan *Focal-Tiny* dan *Focal-Small*, nilai tersebut masih tergolong tinggi. Hal ini menjadikan model *Focal-Base* sebagai model dengan kinerja tertinggi diantara varian lainnya karena tidak hanya memberikan akurasi tinggi yaitu sebesar 93.47%, tetapi juga secara konsisten menunjukkan presisi, sensitivitas, dan *F1-Score* yang sangat baik pada kelas tuberkulosis.

### 5.2.2 Hasil Pengujian Model dengan Kinerja Tertinggi pada Citra Gangguan

Pada tahap ini, dilakukan pengujian lanjutan terhadap model dengan kinerja tertinggi pada tahap sebelumnya, yaitu *Focal-Base*. Pengujian ini dilakukan terhadap citra data uji dengan variasi kondisi gangguan seperti yang dijelaskan pada Tabel 4.1. Pengujian tersebut dilakukan dengan tujuan untuk melihat kinerja ketahanan model jika diberikan data baru dengan gangguan tertentu yang belum pernah dilihat sebelumnya. Berikut ditampilkan hasil *confusion matrix* pada Gambar 5.8 dan metrik evaluasi yang diperoleh menggunakan model *Focal-Base* pada pengujian terhadap citra dengan diberi gangguan atau *noise* dicantumkan pada Tabel 5.8.

Tabel 5.8 Kinerja *Focal-Base* pada Citra dengan Gangguan

Kondisi Citra	Presisi		Sensitivitas		F1-Score		Akurasi
	Normal	TB	Normal	TB	Normal	TB	
Tanpa Gangguan	0.9307	0.9403	0.9562	0.9065	0.9433	0.9231	93.47%
<i>Gaussian Noise</i>	<b>0.9784</b>	0.8482	0.8128	<b>0.9832</b>	0.8885	0.9100	90.09%
<i>Salt and Pepper</i>	0.9036	0.9176	0.9128	0.9086	0.9082	0.9129	91.08%
<i>Motion Blur</i>	0.9005	<b>0.9519</b>	<b>0.9513</b>	0.9014	<b>0.9251</b>	<b>0.9252</b>	<b>92.57%</b>
<i>Brightness and Contrast</i>	0.9535	0.7377	0.6308	0.9712	0.7596	0.8379	80.67%
<i>Occlusion</i>	0.8900	0.9347	0.9333	0.8911	0.9110	0.9111	91.20%



Gambar 5.8 *Confusion Matrix Focal-Base* pada Citra dengan Gangguan: (a) *Gaussian Noise*, (b) *Salt and Pepper Noise*, (c) *Motion Blur*, (d) *Brightness and Contrast Distortion*, (e) *Partial Occlusion*.

Pada kondisi *Gaussian Noise*, dapat dilihat pada Gambar 5.8a bahwa model berhasil mengklasifikasikan 319 citra kelas normal secara benar dan 408 citra kelas TB secara benar, dengan masing-masing kesalahan klasifikasi sebanyak 71 untuk kelas normal dan 9 untuk TB. Berdasarkan Tabel 5.8, presisi untuk kelas normal mencapai nilai 0.9784, sedangkan untuk TB sebesar 0.8482. Nilai sensitivitas menunjukkan bahwa model lebih sensitif terhadap kelas TB yaitu sebesar 0.9832 dibandingkan kelas normal sebesar 0.8128. Adapun *F1-score* berada pada angka 0.8885 untuk kelas normal dan 0.9100 untuk kelas TB. Meskipun akurasi keseluruhan tetap tinggi yaitu sebesar 90.09%, terlihat bahwa keberadaan *Gaussian Noise* mulai memengaruhi kemampuan model terutama dalam mengenali citra TB, yang ditunjukkan dari menurunnya nilai presisi untuk kelas tersebut.

Pada pengujian dengan *Salt and Pepper Noise*, model menunjukkan kinerja yang lebih seimbang. Berdasarkan Gambar 5.8b, terdapat 354 citra kelas normal dan 381 citra kelas TB yang diklasifikasikan dengan benar. Kesalahan klasifikasi masing-masing adalah 36 untuk kelas normal dan 36 untuk kelas TB. Presisi untuk kedua kelas cukup tinggi, yaitu sebesar 0.9036 untuk kelas normal dan sebesar 0.9176 untuk kelas TB, dengan nilai sensitivitasnya masih tergolong tinggi, yakni sebesar 0.9128 dan 0.9086 untuk masing-masing kelas. Nilai *F1-score* juga stabil di angka 0.9082 untuk kelas normal dan 0.9129 untuk kelas TB, dengan akurasi keseluruhan sebesar 91.08%. Hal ini menunjukkan bahwa model memiliki ketahanan yang cukup baik terhadap gangguan jenis *Salt and Pepper*.

Hasil evaluasi terhadap citra dengan *Motion Blur* pada Gambar 5.8c menunjukkan kinerja yang juga sangat baik. Sebanyak 371 citra kelas normal dan 376 citra kelas TB berhasil diklasifikasikan dengan benar, dengan kesalahan klasifikasi sebesar 19 dan 41 untuk masing-masing kelas. Metrik evaluasi menunjukkan presisi sebesar 0.9005 untuk kelas normal dan 0.9519 untuk kelas TB, dengan nilai sensitivitas yang sangat tinggi yakni 0.9513 untuk kelas normal dan 0.9014 untuk kelas TB. Nilai *F1-score* juga tetap tinggi yaitu sebesar 0.9251 dan 0.9252 untuk kedua kelas, sedangkan akurasi keseluruhan diperoleh sebesar 92.57%. Hasil ini menunjukkan bahwa gangguan *Motion Blur* tidak secara signifikan menurunkan kinerja model, bahkan kinerjanya mendekati kondisi tanpa gangguan.

Pengujian terhadap gangguan *Brightness and Contrast Distortion* pada Gambar 5.8d menunjukkan penurunan kinerja model yang paling signifikan di antara seluruh kondisi gangguan. Berdasarkan *confusion matrix* dapat dilihat bahwa hanya 246 citra kelas normal yang berhasil diklasifikasikan dengan benar, sementara 144 lainnya salah diklasifikasikan. Sebaliknya, model masih mampu mengenali 405 citra kelas TB secara benar dengan kesalahan klasifikasi hanya 12 citra. Presisi kelas normal cukup tinggi yaitu sebesar 0.9535, tetapi sensitivitasnya menurun signifikan menjadi 0.6308. *F1-score* untuk kelas normal juga hanya sebesar 0.7596. Sehingga akurasi keseluruhan menurun menjadi 80.67%. Hal ini menandakan bahwa gangguan pencahayaan dan kontras memengaruhi

kemampuan model terutama dalam mengenali citra kelas normal.

Pada pengujian dengan *Partial Occlusion*, model menunjukkan kinerja yang relatif stabil. Berdasarkan Gambar 5.8e dapat dilihat bahwa sebanyak 364 citra kelas normal dan 377 citra kelas TB berhasil diklasifikasikan dengan benar, dengan kesalahan masing-masing 26 dan 40. Nilai presisi sebesar 0.8900 untuk kelas normal dan 0.9347 untuk kelas TB, dengan nilai sensitivitas sebesar 0.9333 untuk kelas normal dan 0.8911 untuk kelas TB. *F1-score* juga tergolong tinggi dan seimbang di angka 0.91 untuk kedua kelas. Dengan akurasi keseluruhan sebesar 91.20%, model terbukti cukup tangguh terhadap gangguan akibat penutupan sebagian area citra.

Secara keseluruhan, hasil pengujian menunjukkan bahwa model *Focal-Base* memiliki ketahanan yang baik terhadap berbagai jenis gangguan pada citra. Meskipun terjadi penurunan kinerja pada kondisi gangguan tertentu, seperti *Brightness and Contrast Distortion*, model tetap mampu mempertahankan nilai presisi, sensitivitas, dan F1-score yang tinggi pada sebagian besar kondisi gangguan. Kinerja tertinggi dicapai ketika model diuji pada citra dengan gangguan *motion blur*, yang menunjukkan kemampuan model yang baik terhadap jenis gangguan tersebut.

## BAB VI

### KESIMPULAN DAN SARAN

Pada bab ini diberikan kesimpulan yang diperoleh berdasarkan penelitian yang telah dilakukan pada Tugas Akhir ini serta diberikan saran-saran yang dapat dijadikan pertimbangan dalam penilitian selanjutnya

#### 6.1 Kesimpulan

Adapun kesimpulan yang diperoleh dari penelitian Tugas Akhir ini sebagai berikut.

1. Penelitian Tugas Akhir ini telah menerapkan model *Focal Transformer* untuk Klasifikasi Tuberkulosis pada citra *X-ray* dada.
2. Model *Focal Transformer* ini dilatih dengan eksperimen tiga variasi model *Focal Transformer* yaitu *Tiny*, *Small*, dan *Base*. Variasi terbaik kemudian diujikan pada data uji dengan lima kondisi gangguan citra yang berbeda yaitu *Gaussian Noise*, *Salt and Pepper*, *Motion Blur*, *Brightness and Contrast Distorsion*, *Occlusion*. Dari semua variasi model, diperoleh Focal-Base sebagai variasi terbaik dengan performa skor metrik tertinggi yaitu akurasi 93.47% pada Citra Normal, 90.09% pada Citra *Gaussian Noise*, 91.08% pada Citra *Salt and Pepper*, 92.57% pada Citra *Motion Blur*, 80.67% pada Citra *Brightness and Contrast Distorsion*, 91.20% pada Citra *Occlusion*

#### 6.2 Saran

Adapun saran yang dapat dijadikan pertimbangan dalam penelitian selanjutnya adalah menggunakan dataset citra *X-ray* dada yang tidak hanya berkualitas tinggi, tetapi juga memiliki distribusi yang merata pada seluruh set data (latih, validasi, dan uji) agar model dapat fokus mempelajari fitur tuberkulosis yang relevan secara akurat, bukan sekadar mengenali *noise* yang timbul dari data berkualitas rendah. Dengan demikian, kemampuan generalisasi model untuk mengenali data baru akan meningkat.



## DAFTAR PUSTAKA

- Abd El-Ghany, S., Elmogy, M., Mahmood, M., & Ahmed, A. E.-A. (2024, 12). A robust tuberculosis diagnosis using chest x-rays based on a hybrid vision transformer and principal component analysis. *Diagnostics*, 14, 2736. doi: 10.3390/diagnostics14232736
- Bhalla, A. S., Goyal, A., Guleria, R., & Gupta, A. K. (2015). Chest tuberculosis: Radiological review and imaging recommendations. *The Indian journal of radiology & imaging*, 25(3), 213—225. Retrieved from <https://europepmc.org/articles/PMC4531444> doi: 10.4103/0971-3026.161431
- Das, G., Anwar, N., Chowdhury, S., & Rahman, K. A. (2016). Design and fabrication of an image processing based autonomous weapon. *International Journal of Engineering Research*, 5(12), 931–935. Retrieved from <https://doi.org/10.17950/ijer/v5s12/1212> doi: 10.17950/ijer/v5s12/1212
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, *abs/2010.11929*. Retrieved from <https://arxiv.org/abs/2010.11929>
- Esteva, A., Kuprel, B., Novoa, R., Ko, J., Swetter, S., Blau, H., & Thrun, S. (2017, 01). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542. doi: 10.1038/nature21056
- Gonzalez, R. C. (2009). *Digital image processing*. Pearson Education India.
- Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing*. Upper Saddle River, NJ 07458: Pearson Education India.
- Gonzalez, R. C., & Woods, R. E. (2018). *Digital image processing* (4th ed.). 330 Hudson Street, New York, NY 10013: Pearson Education India.
- Hamza, S. A., & Jesser, A. (2023). A hierarchical vision approach for enhanced medical diagnostics of lung tuberculosis using swin transformer. *Computer Science & Information Technology (CS & IT)*, 13(22), 111–121. Retrieved from <https://doi.org/10.5121/csit.2023.132209> doi: 10.5121/csit.2023.132209

Hansun, S., Argha, A., Liaw, S. T., Celler, B. G., & Marks, G. B. (2023). Machine and deep learning for tuberculosis detection on chest x-rays: Systematic literature review. *Journal of Medical Internet Research*, 25, e43154. Retrieved from <https://doi.org/10.2196/43154> doi: 10.2196/43154

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. Retrieved from <https://arxiv.org/abs/1207.0580>

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Retrieved from [https://proceedings.neurips.cc/paper\\_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)

Kumar, V., T R, M., Kumar, V., & Guluwadi, S. (2025, 03). Enhanced tuberculosis detection using vision transformers and explainable ai with a grad-cam approach on chest x-rays. *BMC Medical Imaging*, 25. doi: 10.1186/s12880-025-01630-3

Lin, M., Chen, Q., & Yan, S. (2014). *Network in network*. Retrieved from <https://arxiv.org/abs/1312.4400>

Majumder, R. I. (2024). Efficient classification of pulmonary pneumonia and tuberculosis alongside normal and non-x-ray images with minimal resources and maximum accuracy. *medRxiv*, 2024.12.31.24319820. Retrieved from <https://doi.org/10.1101/2024.12.31.24319820> doi: 10.1101/2024.12.31.24319820

Rahman, T., Khandakar, A., Kadir, M., Islam, K., Khandakar, F., Mazhar, R., ... Ayari, M. (2020, 10). Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization. *IEEE Access*, 8, 191586-191601. doi: 10.1109/ACCESS.2020.3031384

Swaminathan, S., & Tantri, B. R. (2024). Confusion matrix-based performance evaluation metrics. *African Journal of Biomedical Research*, 27(4s), 4023–4031. Retrieved from <https://doi.org/10.53555/AJBR.v27i4S.4345> doi: 10.53555/AJBR.v27i4S.4345

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/3f5ee243547dee91fb0d053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb0d053c1c4a845aa-Paper.pdf)

WHO. (2025, March 14). *Tuberculosis*. World Health Organization. Retrieved from <https://www.who.int/news-room/fact-sheets/detail/tuberculosis> (Retrieved March 22, 2023)

World Health Organization. (2022). *Global tuberculosis report 2022*. World Health Organization. Retrieved from <https://www.who.int/publications/i/item/9789240061729> (Accessed: 2024-03-22)

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., & Gao, J. (2021). *Focal self-attention for local-global interactions in vision transformers*. Retrieved from <https://arxiv.org/abs/2107.00641>



## **UCAPAN TERIMA KASIH**

Penyelesaian penulisan Tugas Akhir ini tidak lepas dari orang-orang terdekat penulis yang telah mendukung dan memotivasi penulis. Oleh sebab itu, penulis mengucapkan terima kasih kepada:

1. Razan, Andri, Indira, Nailah, Anto, Rama, Irvan, Rapop, dan teman-teman kongkow 24/7 lainnya yang telah menjadi saksi dan teman seperjuangan bagi penulis dalam suka maupun duka di perantauan.
2. Adienda Safira yang telah memberi semangat, menjadi pendengar yang baik, serta menjadi tempat berbagi keluh kesah di segala kondisi penulis.
3. Gloria, Kevin, Valldy, dan Altika sebagai teman seerbimbingan yang telah banyak berdiskusi dengan penulis selama penggerjaan Tugas Akhir.

Penulis juga mengharapkan kritik dan saran yang membangun dari berbagai pihak untuk penyempurnaan isi tugas akhir ini. Akhir kata, semoga tugas akhir ini bermanfaat bagi semua pihak yang bersangkutan.

Surabaya, Juli 2025

Muhammad Arief Rachman



## BIODATA PENULIS



Penulis bernama Muhammad Arief Rachman, lahir di Depok, 25 Mei 2003. Penulis menempuh pendidikan di SDIT At Taufiq Kota Depok (2009-2015), SMPIT Nurul Fikri Kota Depok (2015-2018), dan SMAIT Nurul Fikri Kota Depok (2018-2021). Setelah lulus SMA, penulis melanjutkan pendidikan S-1 di Departemen Matematika Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2021. Selama menjadi mahasiswa matematika, penulis aktif berorganisasi di Himpunan Mahasiswa Matematika ITS (HIMATIKA ITS) sebagai staf departemen *Sport and Art Development* pada tahun 2022 dan sebagai staf ahli *Sport and Art Development* pada tahun 2023. Selain itu penulis mengikuti kepanitiaan Olimpiade Matematika Institut Teknologi Sepuluh Nopember (OMITS) sebagai Kepala Sub-Divisi *Competition* pada tahun 2022. Penulis pernah juga menjadi Koordinator *Instructor Committee* pada kegiatan Padamu HIMATIKA 2023. Untuk informasi lebih lanjut mengenai Tugas Akhir ini dapat menghubungi email: ariefrachh25@gmail.com