



**IDENTIFIKASI PENYAKIT PARU-PARU BERDASARKAN  
SUARA DAN RIWAYAT PASIEN MENGGUNAKAN MODEL  
*CROSS-ATTENTION VIDEO VISION TRANSFORMER***

**NASKAH SKRIPSI**

**Husni Na'fa Mubarok  
121450078**

**PROGRAM STUDI SAINS DATA  
FAKULTAS SAINS  
INSTITUT TEKNOLOGI SUMATERA  
LAMPUNG SELATAN**

**2024**



**IDENTIFIKASI PENYAKIT PARU-PARU BERDASARKAN  
SUARA DAN RIWAYAT PASIEN MENGGUNAKAN MODEL  
*CROSS-ATTENTION VIDEO VISION TRANSFORMER***

**NASKAH SKRIPSI**  
**Diajukan sebagai syarat maju seminar hasil**

**Husni Na'fa Mubarok**  
**121450078**

**PROGRAM STUDI SAINS DATA**  
**FAKULTAS SAINS**  
**INSTITUT TEKNOLOGI SUMATERA**  
**LAMPUNG SELATAN**

**2024**

## **HALAMAN PENGESAHAN**

Naskah Tugas Akhir untuk Seminar Hasil dengan judul "**Identify lung disease based on sound and patient history using the Cross-Attention Video Vision Transformer model**" adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya, baik sebagian ataupun seluruhnya, baik oleh saya ataupun orang lain, baik di Institut Teknologi Sumatera maupun di institusi pendidikan lainnya.

Lampung Selatan, 06 April 2024

Penulis,

**Husni Na'fa Mubarok**  
**NIM. 121450078**



Diperiksa dan disetujui oleh,

Pembimbing I

Pembimbing II

**Christyan Tamara Nadeak, M.Si**  
**NRK. 1993120420211415**

**Luluk Muthoharoh, M.Si**  
**NIP. 199504112022032014**

Disahkan oleh,

Koordinator Program Studi Sains Data  
Fakultas Sains  
Institut Teknologi Sumatera

**Tirta Setiawan, S.Pd., M.Si**  
**NIP. 199008222022031003**

Penguji I : Mika Alvionita S, M.Si  
Penguji II : Penguji 2

Sidang Tugas Akhir :

## **HALAMAN PERNYATAAN ORISINALITAS**

**Skripsi ini adalah karya saya sendiri dan semua sumber baik yang dikutip  
maupun yang dirujuk telah saya nyatakan benar.**

**Nama : Husni Na'fa Mubarok**

**NIM : 121450078**

**Tanda tangan :**

**Tanggal : 06 April 2024**

## **HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Institut Teknologi Sumatera, saya yang bertanda tangan di bawah ini:

Nama : Husni Na'fa Mubarok  
NIM : 121450078  
Program Studi : Sains Data  
Fakultas : Sains  
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan Hak Bebas Royalti Noneksklusif (*Non-Exclusive Royalty Free Right*) kepada Institut Teknologi Sumatera atas karya ilmiah saya yang berjudul:

**Identifikasi penyakit paru-paru berdasarkan suara dan riwayat pasien  
menggunakan model *Cross-Attention Video Vision Transformer***

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Sumatera berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Lampung Selatan  
Pada tanggal : 06 April 2024

Yang menyatakan

Husni Na'fa Mubarok

## **ABSTRAK**

### **Identifikasi penyakit paru-paru berdasarkan suara dan riwayat pasien menggunakan model *Cross-Attention Video Vision Transformer***

Husni Na'fa Mubarok (121450078)  
Pembimbing I: Christyan Tamaro Nadeak, M.Si  
Pembimbing II: Luluk Muthoharoh, M.Si

Identifikasi penyakit paru-paru dapat dilakukan melalui analisis suara batuk dan riwayat pasien. Penelitian ini mengembangkan model berbasis Cross-Attention Video Vision Transformer yang memanfaatkan representasi fitur audio dan data riwayat pasien untuk klasifikasi penyakit paru-paru. Mel Frequency Cepstral Coefficient (MFCC) digunakan untuk ekstraksi fitur audio, sementara fitur riwayat pasien memperkaya informasi tanpa menambah beban komputasi. Dataset yang digunakan terdiri dari suara batuk, data riwayat pasien, dan label penyakit dengan tiga kelas: asma, COPD, dan sehat. Model Video Vision Transformer diterapkan dengan embedding spasial dan temporal, dioptimalkan melalui mekanisme Cross-Attention. Evaluasi dilakukan menggunakan metrik akurasi, presisi, recall, dan F1-score, serta analisis kurva ROC-AUC untuk memeriksa performa klasifikasi antar kelas. Hasil penelitian menunjukkan bahwa model mampu mencapai akurasi terbaik sebesar 82.57% dengan F1-score 81.95% menggunakan konfigurasi Big Model. Penggunaan Trainable Positional Encoding memberikan performa yang lebih unggul dibandingkan Sinusoidal Positional Encoding. Model ini menunjukkan potensi yang baik dalam klasifikasi penyakit paru-paru berdasarkan data multimodal, meskipun memerlukan pengembangan lebih lanjut untuk meningkatkan kemampuan dalam membedakan kondisi penyakit yang serupa.

**Kata kunci:** Identifikasi penyakit paru-paru, Video Vision Transformer, Cross-Attention, MFCC, klasifikasi suara.

## ***ABSTRACT***

### ***Identify lung disease based on sound and patient history using the Cross-Attention Video Vision Transformer model***

Husni Na'fa Mubarok (121450078)

*Advisor I : Christyan Tamaro Nadeak, M.Si*

*Advisor II: Luluk Muthoharoh, M.Si*

*Lung disease identification can be conducted through the analysis of cough sounds and patient history. This study developed a model based on the Cross-Attention Video Vision Transformer, leveraging audio feature representations and patient history data for lung disease classification. Mel Frequency Cepstral Coefficient (MFCC) was used for audio feature extraction, while patient history features enriched the information without adding computational overhead. The dataset comprised cough audio, patient history data, and disease labels with three classes: asthma, COPD, and healthy. The Video Vision Transformer model was implemented with spatial and temporal embeddings, optimized through the Cross-Attention mechanism. Evaluation metrics included accuracy, precision, recall, F1-score, and ROC-AUC curve analysis to assess inter-class classification performance. The results showed that the model achieved the best accuracy of 82.57% and an F1-score of 81.95% using the Big Model configuration. The use of Trainable Positional Encoding outperformed Sinusoidal Positional Encoding. This model demonstrates promising potential for multimodal data-based lung disease classification, though further improvements are needed to enhance its ability to distinguish between similar disease conditions.*

***Keywords :*** Lung disease identification, Video Vision Transformer, Cross-Attention, MFCC, sound classification.

## MOTTO

*Ini mottoku, mana motto-mu?.*

## **HALAMAN PERSEMBAHAN**

*Untuk Emak dan Bapak  
di kampung*

## **KATA PENGANTAR**

Puji syukur penulis panjatkan ke hadirat Allah SWT atas berkah dan rahmat-Nya sehingga skripsi ini dapat terselesaikan dengan baik. Skripsi ini dibuat untuk menyelesaikan pendidikan jenjang sarjana pada Institut Teknologi Sumatera. Penyusunan skripsi ini banyak mendapat bantuan dan dukungan dari berbagai pihak sehingga dalam kesempatan ini, dengan penuh kerendahan hati, penulis mengucapkan terima kasih kepada:

1. Prof. XXXX XXXX selaku Rektor Institut Teknologi Sumatera,
2. Prof. YYYY YYYY selaku Dekan Fakultas Sains Institut Teknologi Sumatera,
3. Dr. ZZZZ ZZZZ selaku Koordinator Program Studi,
4. Prof. DR. Nama selaku dosen pembimbing pertama yang telah membimbing,
5. Nama , S.Si., M.Si. selaku dosen pembimbing kedua yang selalu membantu, dan
6. Cantumkan pihak-pihak lain yang membantu penelitian tugas akhir, termasuk sumber data, tempat riset, rekan satu TA, dan-lain-lain.

Penulis menyadari bahwa penyusunan Skripsi ini jauh dari sempurna. Akhir kata penulis mohon maaf yang sebesar-besarnya apabila ada kekeliruan di dalam penulisan skripsi ini.

Lampung Selatan, 06 April 2024

**Husni Na'fa Mubarok**

## DAFTAR ISI

<b>HALAMAN JUDUL . . . . .</b>	<b>i</b>
<b>HALAMAN PENGESAHAN . . . . .</b>	<b>ii</b>
<b>HALAMAN PERNYATAAN ORISINALITAS . . . . .</b>	<b>iii</b>
<b>HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI . . . . .</b>	<b>iv</b>
<b>ABSTRAK . . . . .</b>	<b>v</b>
<b>ABSTRACT . . . . .</b>	<b>vi</b>
<b>MOTTO . . . . .</b>	<b>vii</b>
<b>HALAMAN PERSEMBAHAN . . . . .</b>	<b>viii</b>
<b>KATA PENGANTAR . . . . .</b>	<b>ix</b>
<b>DAFTAR ISI . . . . .</b>	<b>x</b>
<b>DAFTAR GAMBAR . . . . .</b>	<b>xii</b>
<b>DAFTAR TABEL . . . . .</b>	<b>xiii</b>
<b>I PENDAHULUAN . . . . .</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan Penelitian . . . . .	2
1.4 Batasan Masalah . . . . .	3
<b>II TINJAUAN PUSTAKA . . . . .</b>	<b>4</b>
2.1 Penelitian Terdahulu . . . . .	4
2.2 Penyakit Paru-paru . . . . .	4
2.2.1 <i>Chronic obstructive pulmonary disease (COPD)</i> . . . . .	4
2.2.2 Asma . . . . .	4
2.3 <i>Mel Frequency Cepstral Coefficient (MFCC)</i> . . . . .	5
2.4 Transformer . . . . .	5
2.4.1 <i>Positional Encoding</i> . . . . .	6
2.4.2 <i>Attention Mechanism</i> . . . . .	6
2.4.3 <i>Position-wise Feed-Forward Networks</i> . . . . .	7
2.4.4 <i>Layer Normalization</i> . . . . .	8
2.5 <i>Video Vision Transformer</i> . . . . .	8
<b>III METODE PENELITIAN . . . . .</b>	<b>10</b>
3.1 Deskripsi Data . . . . .	10

3.2 Rancangan Penelitian . . . . .	12
3.2.1 Pengumpulan Data . . . . .	12
3.2.2 Pemrosesan Suara . . . . .	12
3.2.3 Pemrosesan Data Tabel Riwayat Pasien . . . . .	15
3.2.4 Pipeline dan Pembagian Data . . . . .	16
3.2.5 Pemodelan . . . . .	17
3.2.6 Arsitektur . . . . .	17
3.2.7 <i>Loss Function</i> . . . . .	18
3.2.8 Pelatihan Model . . . . .	19
3.3 Evaluasi Model . . . . .	19
3.3.1 <i>Confusion Matrix</i> . . . . .	19
3.3.2 Kurva ROC-AUC . . . . .	20
<b>IV HASIL DAN PEMBAHASAN . . . . .</b>	<b>21</b>
4.1 Persiapan <i>Workspace Kaggle Notebook</i> . . . . .	21
4.1.1 Buat <i>Notebook Kaggle</i> . . . . .	21
4.1.2 Input Data . . . . .	21
4.1.3 Atur Sesi menggunakan GPU . . . . .	22
4.1.4 Import Package/Library . . . . .	22
4.2 <i>Exploratory data analysis</i> dan <i>Data Cleaning</i> . . . . .	23
4.3 Pembuatan Pipeline <i>Data Processing</i> . . . . .	25
4.3.1 <i>Audio Processing</i> . . . . .	25
4.4 Pelatihan Model . . . . .	26
4.4.1 Variasi Model . . . . .	26
4.4.2 Perbandingan <i>Sinusoidal Positional Encoding</i> dengan <i>Trainable Positional Encoding</i> . . . . .	27
4.5 Evaluasi Model . . . . .	27
4.5.1 Akurasi dan Loss Pelatihan . . . . .	27
4.5.2 Confussion Matrix . . . . .	30
4.5.3 Kurva AUC-ROC . . . . .	31
4.5.4 Prediksi Data baru . . . . .	34
<b>V KESIMPULAN DAN SARAN . . . . .</b>	<b>35</b>
5.1 Kesimpulan . . . . .	35
5.2 Saran . . . . .	35
<b>DAFTAR PUSTAKA . . . . .</b>	<b>36</b>
<b>LAMPIRAN . . . . .</b>	<b>40</b>
<b>LAMPIRAN . . . . .</b>	<b>40</b>

<b>A</b>	<i>Mel frequency Capstral Coefficients (MFCC)</i>	<b>41</b>
A.1	Pre-Emphasis	41
A.2	<i>Framing dan Windowing</i>	41
A.3	<i>Discrete Fourier Transform (DFT)</i>	42
A.4	<i>Mel-Frequency Filter Bank</i>	42
A.5	<i>Discrete Cosine Transform (DCT)</i>	43
<b>B</b>	<b>Perhitungan Encoder Cross-Attention</b>	<b>44</b>
B.1	<i>Self-Attention</i> dan <i>Multi-Head Attention</i>	44
B.2	<i>Feed-Forward Network</i>	46
B.3	Output Encoder $Y_s$ dan $Y_t$	47
<b>C</b>	<b>Optimizer Adam</b>	<b>49</b>
C.1	Optimizer Adam pada Transformers	49
C.1.1	Persamaan Oprimizer Adam	49
C.1.2	Contoh Perhitungan	50
<b>D</b>	<b>Summary Model</b>	<b>52</b>
D.1	Model Base 1	52
D.2	Model Base 2	54
D.3	Model Big 1	55
D.4	Model Big 2	57

## DAFTAR GAMBAR

Gambar 2.1	Arsitektur Transformer . . . . .	5
Gambar 3.1	Visualisasi gelombang suara tiap kelas . . . . .	10
Gambar 3.2	<i>Flowchart</i> rancangan penelitian . . . . .	12
Gambar 3.3	Desain rancangan arsitektur model. . . . .	18
Gambar 3.4	Kurva ROC-AUC . . . . .	20
Gambar 4.1	<i>Kaggle Workspace</i> . . . . .	21
Gambar 4.2	Tambahkan Dataset . . . . .	22
Gambar 4.3	Pengaturan Sesi . . . . .	22
Gambar 4.4	Pengecekan Data Kosong . . . . .	24
Gambar 4.5	Menghapus Kolom . . . . .	24
Gambar 4.6	Visualisasi Audio . . . . .	25
Gambar 4.7	Segmentasi Suara batuk . . . . .	25
Gambar 4.8	Visualisasi MFCC . . . . .	26
Gambar 4.9	Akurasi model per epoch . . . . .	28
Gambar 4.10	Loss model per epoch . . . . .	29
Gambar 4.11	Confusion Matrix . . . . .	30
Gambar 4.12	Kurva AUC-ROC antar kelas Base Model 1 . . . . .	31
Gambar 4.13	Kurva AUC-ROC antar kelas Base Model 2 . . . . .	32
Gambar 4.14	Kurva AUC-ROC antar kelas Big Model 1 . . . . .	33
Gambar 4.15	Kurva AUC-ROC antar kelas Big Model 2 . . . . .	34
Gambar B.1	Encoder Cross Attention . . . . .	44
Gambar B.2	<i>Self-Attention</i> atau <i>Scaled Dot-Product Attention</i> . . . . .	45
Gambar B.3	<i>Multi-Head Attention</i> . . . . .	45

## DAFTAR TABEL

Tabel 2.1	Penelitian Terdahulu . . . . .	4
Tabel 3.1	Tabel riwayat pasien . . . . .	11
Tabel 3.2	<i>Confusion Matrix</i> untuk 3 Kelas (a), Rumus metrik evaluasi (b) . . . . .	20
Tabel 4.1	Daftar Library Python dan Deskripsinya . . . . .	23
Tabel 4.2	Variasi parameter untuk model Transformer Base dan Big .	26
Tabel 4.3	Perbandingan evaluasi model dengan Sinusoidal Positional Encoding dan Trainable Positional Encoding. . . . .	27
Tabel 4.4	Akurasi, Presisi, Recall, dan F1-Score dari 4 Model . . . . .	31
Tabel B.1	Tabel Variabel untuk Encoder Transformator . . . . .	46
Tabel D.1	Summary Encoder model Base 1 . . . . .	52
Tabel D.2	Summary Classifier Model Base 1 . . . . .	53
Tabel D.3	Summary Model Base 1 . . . . .	53
Tabel D.4	Summary Encoder Block Model Base 2 . . . . .	54
Tabel D.5	Summary Model Classifier . . . . .	55
Tabel D.6	Summary Model Base 2 . . . . .	55
Tabel D.7	Summary Encoder Block . . . . .	55
Tabel D.8	Summary Classifier . . . . .	57
Tabel D.9	Summary Model Big 1 . . . . .	57
Tabel D.10	Summary Encoder Block . . . . .	57
Tabel D.11	Summary Classifier . . . . .	59
Tabel D.12	Summary Model Big 2 . . . . .	59

## BAB I

### PENDAHULUAN

#### 1.1 Latar Belakang

Paru-paru merupakan organ vital manusia yang berfungsi untuk pertukaran oksigen dan karbon dioksida pada darah[1]. Gangguan pada paru-paru dapat berakibat buruk pada sistem pernapasan sehingga dapat menimbulkan penyakit. Menurut *Lung Foundation Australia*, penyakit paru-paru dapat disebabkan oleh berbagai faktor seperti umur, perokok aktif atau pasif, lingkungan yang terpapar debu, gas, uap dan zat kimia[2]. Salah satu penyakit paru yang berisiko yaitu *Chronic obstructive pulmonary disease* (COPD) dan asma.

*Chronic obstructive pulmonary disease* (COPD) adalah gangguan pernapasan yang umum dan progresif yang ditandai dengan keterbatasan aliran udara yang terus-menerus dan sering dikaitkan dengan penyakit paru-paru lainnya seperti bronkitis kronis dan asma [3]. Menurut data *World Health Organization* (WHO) *Chronic obstructive pulmonary disease* (COPD) merupakan penyebab kematian keempat di seluruh dunia, menyebabkan 3.5 juta kematian pada tahun 2021 [4], dan diperkirakan COPD akan menjadi penyebab kematian ketiga di dunia pada tahun 2030 [5]. Berdasarkan survei data BPJS tahun 2024, hampir 19 juta jumlah pasien COPD yang berobat ke rumah sakit dengan penyakit tersebut [6]. Oleh sebab itu, diperlukan metode yang dapat mengidentifikasi jenis penyakit paru-paru berdasarkan gejala yang diketahui, salah satunya yaitu suara.

Suara batuk atau paru-paru mengandung banyak informasi tentang kondisi paru-paru dan dapat digunakan untuk menilai serta mendiagnosis penyakit pernapasan [7]. Penggunaan suara untuk identifikasi penyakit paru meningkatkan minat terhadap perawatan medis tanpa kontak untuk pemeriksaan paru-paru secara otomatis. Model Deep Learning banyak digunakan peneliti dalam menganalisis suara seperti LungRN+NL [8] yang menggunakan augmentasi data campuran dan arsitektur ResNet [9] untuk mengatasi ketidakseimbangan kelas data. RespireNet [10] menggunakan model *pre-trainer* pada ImageNet dengan strategi *fine-tuning device-specific*.

Model *General-Purpose* representasi audio lainnya seperti CLAP [11] menggunakan dua encoder untuk memproses input yaitu *Audio Encoder* yang memproses input suara dan *Text Encoder* yang memproses input berupa teks.

Namun, penggunaan dua encoder mengakibatkan beban komputasi yang tinggi sehingga memerlukan sumber daya komputasi yang besar.

Oleh sebab itu, pada penelitian ini mengadopsi konsep dari *Video Vision Transformer* [12] yang menggunakan embedding spasial  $Z_s$  dan temporal  $Z_t$  dengan input berupa representasi MFCC dan fitur riwayat pasien pada satu encoder yang sama sehingga mengurangi beban komputasi saat proses *training*. *Cross-Attention* [13] digunakan agar suatu *sequence* dapat memperhatikan informasi dari *sequence* lainnya.

Secara keseluruhan, kontribusi penelitian ini dapat dirangkum yaitu MFCC digunakan untuk merepresentasikan spektrum daya jangka pendek dari suatu suara yang membantu model memahami dan memproses suara manusia secara lebih efektif. Penelitian ini mengadopsi model *Video Vision Transformer* untuk memahami fitur temporal dan spasial dari data audio. Data riwayat pasien digunakan untuk memperkaya fitur tanpa menambah encoder fitur sehingga beban komputasi menjadi lebih ringan.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang 1.1 yang telah dijelaskan sebelumnya, berikut merupakan rumusan masalah pada penelitian tugas akhir ini:

1. Bagaimana penerapan konsep *Video Vision Transformer* pada data suara penyakit paru dengan penambahan fitur riwayat pasien menggunakan metode *Cross-Attention* dapat mengidentifikasi jenis penyakit paru-paru?
2. Bagaimana performa penerapan konsep *Video Vision Transformer* dalam mengklasifikasikan jenis penyakit paru-paru berdasarkan data suara?

## 1.3 Tujuan Penelitian

Tujuan dari penelitian ini berdasarkan rumusan masalah yang juga menjadi dasar dilakukannya penelitian ini adalah sebagai berikut:

1. Membuat model *Deep Learning* untuk mengklasifikasikan jenis penyakit paru-paru menggunakan konsep Model *Vision Transformer* dengan metode *Cross-Attention*.
2. Mengevaluasi performa model *Video Vision Transformer* dalam mengidentifikasi fitur suara dan riwayat pasien sehingga menghasilkan klasifikasi yang sesuai

#### **1.4 Batasan Masalah**

1. Penelitian ini hanya menggunakan data suara batuk dan riwayat pasien tanpa menyertakan identitas atau informasi lainnya.
2. Jenis penyakit paru-paru yang diidentifikasi terbatas pada dataset yang digunakan.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Penelitian Terdahulu

**Tabel 2.1** Penelitian Terdahulu

Author(s)	Metode	Hasil Penelitian
Hao Xue et al.[14]	Transformer-CP	Model berbasis Transformer dengan Constraintive Pre-training yang menggunakan random masking dengan fitur ekstraksi MFCC mendapatkan akurasi 87,74% pada data suara batuk pasien covid-19
Li Xiao et al.[15]	LungAdapter	Metode ini menggabungkan blok <i>trainable</i> ke dalam model AST yang telah dilatih sebelumnya, yang memungkinkan ekstraksi informasi penting tentang klasifikasi suara paru-paru dari model. Model mencapai kinerja yang baik dengan score 62,40%.
Victor Basu et al.[16]	GRU, MFCC	Lapisan GRU (gated recurrent unit) digunakan untuk memecahkan masalah gradien yang hilang dalam RNN standar. Akurasi: $95,67 \pm 0,77\%$ .

#### 2.2 Penyakit Paru-paru

Jenis penyakit paru-paru yang diamati pada penelitian ini terdapat pada 2.2.1 dan 2.2.2.

##### 2.2.1 *Chronic obstructive pulmonary disease (COPD)*

*Chronic Obstructive Pulmonary Disease* (COPD) memiliki berbagai gejala yang sering kali dapat disalahartikan sebagai kondisi pernapasan lainnya, sehingga mempersulit diagnosis dan penanganannya. Gejala utamanya meliputi batuk kronis [17], dispnea [18], dan produksi sputum [19], yang tumpang tindih dengan kondisi seperti asma dan bronkitis.

##### 2.2.2 Asma

Asma adalah suatu kondisi pernapasan kronis yang ditandai dengan gejala yang bervariasi, terutama mengi, batuk, dada terasa sesak, dan dispnea yang intensitas

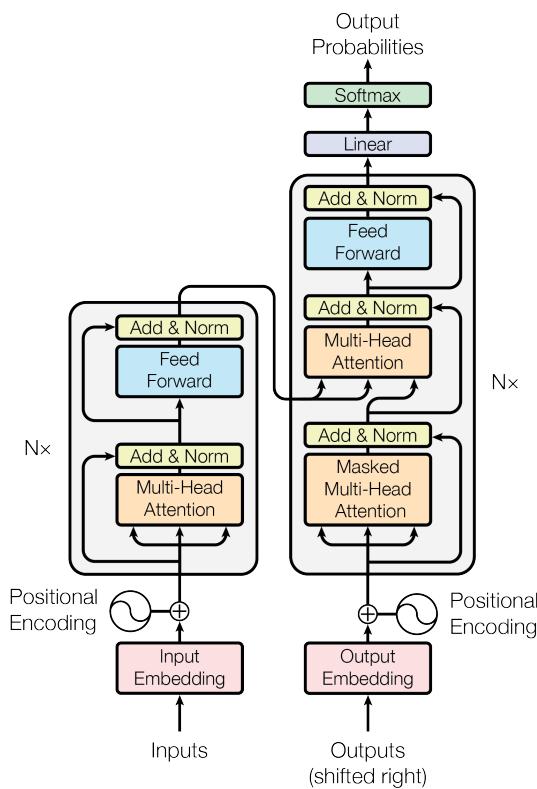
dan frekuensinya dapat berfluktuasi [20]. Gejala-gejala ini sering kali timbul dari pemicu seperti alergen, infeksi, atau olahraga, dan mungkin terjadi secara intermiten, dengan beberapa pasien mengalami periode bebas gejala [20], [21].

### 2.3 Mel Frequency Cepstral Coefficient (MFCC)

MFCC memanfaatkan persepsi frekuensi suara telinga manusia, menggunakan skala Mel non-linier untuk mengubah sinyal audio menjadi representasi yang lebih relevan secara persepsi. Transformasi ini dicapai melalui serangkaian langkah, termasuk *pre-emphasis*, *windowing* dan *framing*, transformasi Fourier, pemrosesan *Mel filter bank*, dan analisis *cepstral*[22], [23].

### 2.4 Transformer

Arsitektur Transformer pertama kali diperkenalkan oleh Vaswani et al. (2017) pada paper yang berjudul "*Attention Is All You Need*"[24] yang terdiri dari dua bagian utama yaitu encoder dan decoder. Karena fleksibilitasnya dalam menangani *sequence* dan kemampuannya dalam memahami konteks, peneliti banyak mengadopsi arsitektur ini pada berbagai media seperti suara [15], gambar [25] dan video [12].



**Gambar 2.1** Arsitektur Transformer

### 2.4.1 Positional Encoding

Arsitektur Transformer bersifat paralel sehingga diperlukan *Positional Encoding* untuk memberikan informasi urutan pada *sequence*. *Positional Encoding* memiliki ukuran dimensi yang sama dengan *embedding* input sehingga dapat dijumlahkan. Positional encoding dijelaskan pada 2.4.1 dan 2.4.1.

#### Sinusoidal Positional Encoding

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \quad (2.1)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \quad (2.2)$$

Keterangan:

- $PE_{(pos,2i)}$  adalah posisi pada indeks ganjil
- $PE_{(pos,2i+1)}$  adalah posisi pada indeks genap
- $pos$  adalah posisi sampel/data dalam *sequence*
- $i$  adalah dimensi data
- $d_{\text{model}}$  adalah ukuran embedding

$pos$  merupakan posisi dan  $i$  merupakan dimensi.  $2i$  dan  $2i + 1$  adalah indeks ganjil dan genap dalam embedding. *Positional encoding sinusoidal* dipilih karena memungkinkan model untuk mengekstrapolasi *sequence* yang lebih panjang daripada yang ditemui selama pelatihan.

**Trainable Positional Encoding** *Trainable Embedding* banyak digunakan pada *Large Language Model* (LLM) seperti BERT [26] dan GPT [27]. *Trainable Embedding* dapat memahami konteks data lebih baik karena nilainya di-update selama proses pelatihan. Pemahaman kontekstual ini membuat *Trainable embedding* jauh lebih optimal dalam tugas-tugas Deep Learning yang kompleks karena dapat memahami pola yang mungkin terlewatkan oleh *Embedding Statis*.

### 2.4.2 Attention Mechanism

Fungsi *Attention* dapat dideskripsikan sebagai pemetaan *Query* dan sekumpulan pasangan *Key-Value* ke suatu *Output*, di mana *Query*, *Key*, *Value*, dan *Output* semuanya merupakan vektor. *Self-Attention* menghubungkan semua posisi dengan jumlah operasi *sequence* yang konstan sehingga mempercepat operasi dibandingkan layer recurrent pada RNN. Fungsi *Attention* dijelaskan pada

persamaan 2.3

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{\text{Head\_DIM}}} \right) V \quad (2.3)$$

Keterangan:

- $\text{Attention}(Q, K, V)$  adalah mekanisme *self-attention*
- $Q$  adalah *Query*
- $K$  adalah *Key*
- $V$  adalah *Value*
- $\text{Head\_DIM}$  adalah dimensi embedding

*Multi-Head Attention* memungkinkan model untuk secara bersamaan memberikan *Attention* informasi dari subruang representasi yang berbeda pada posisi yang berbeda 2.4.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{concat}(\text{head}_1, \dots, \text{head}_h)W^O \\ \text{dengan } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (2.4)$$

Keterangan:

- $\text{MultiHead}(Q, K, V)$  adalah mekanisme multi-head attention
- $\text{head}_i$  adalah mekanisme attention
- $W_i^Q$  adalah bobot untuk *Query*
- $W_i^K$  adalah bobot untuk *Key*
- $W_i^V$  adalah bobot untuk *Value*
- $W^O$  adalah bobot untuk multi-head attention

#### 2.4.3 Position-wise Feed-Forward Networks

Masing-masing lapisan dalam encoder dan decoder terdapat *Feed-Forward Networks* yang saling terhubung, yang diterapkan ke setiap posisi secara terpisah dan identik. Fungsi *Feed-Forward Networks* dapat dilihat pada persamaan 2.5.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.5)$$

Keterangan:

- $\text{FFN}(x)$  adalah mekanisme Feed-Forward Networks
- $x$  adalah nilai input neuron
- $W_1$  adalah bobot pertama

- $b_1$  adalah bias pertama
- $W_2$  adalah bobot kedua
- $b_2$  adalah bobot kedua

Meskipun transformasi linier sama di berbagai posisi, FNN menggunakan parameter yang berbeda pada setiap lapisan.

#### 2.4.4 Layer Normalization

*Layer Normalization* digunakan untuk menormalkan output dari setiap jaringan, sehingga distribusi nilainya stabil selama *training*. Parameter  $\gamma$  dan  $\beta$  merupakan parameter pelatihan dan  $\epsilon$  merupakan nilai konstan  $10^{-6}$ .

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta \quad (2.6)$$

Keterangan:

- $\text{LayerNorm}(x)$  adalah Layer Normalisasi
- $x$  adalah nilai input layer
- $\mu$  adalah rataan
- $\sigma^2$  adalah standar deviasi
- $\epsilon$  adalah nilai galat agar pembagi tidak bernilai 0
- $\gamma$  dan  $\beta$  adalah parameter pelatihan

#### 2.5 Video Vision Transformer

Pada ViViT, video input  $V$  dengan dimensi  $D$  (depth),  $H$  (tinggi),  $W$  (lebar), dan  $C$  (channel) dibagi-bagi menjadi potongan-potongan kecil yang disebut *patch*[12]. Setiap *patch*  $P$  memiliki dimensi  $P_d \times P_h \times P_w \times C$ . Proses pembagian video menjadi *patch* ini dapat divisualisasikan sebagai persamaan 2.7

$$V \in \mathbb{R}^{D \times H \times W \times C} \rightarrow \text{Reshape} \rightarrow P \in \mathbb{R}^{N \times (P_d \times P_h \times P_w \times C)} \quad (2.7)$$

Keterangan:

- $V$  adalah input embedding video
- $P$  adalah input patch embedding
- Reshape adalah mengubah input video ke patch
- $D$  adalah ukuran *depth* kedalaman/frame
- $H$  adalah ukuran *height* atau tinggi
- $W$  adalah ukuran *width* atau lebar

- $C$  adalah jumlah *channel* atau kanal

di mana  $N = (D/P_d) \times (H/P_h) \times (W/P_w)$  adalah jumlah total *patch* yang dihasilkan. Selanjutnya, setiap *patch*  $P$  diproyeksikan ke dalam ruang *embedding* dengan dimensi  $d$  melalui sebuah lapisan linear. Hasil proyeksi ini membentuk sebuah *sequence* token  $Z$  dengan dimensi  $N \times d$  sesuai persamaan 2.8

$$Z = \text{Linear}(P) \in \mathbb{R}^{N \times d} \quad (2.8)$$

Keterangan:

- $Z$  adalah hasil proyeksi *patch embedding*
- $P$  adalah *patch embedding*

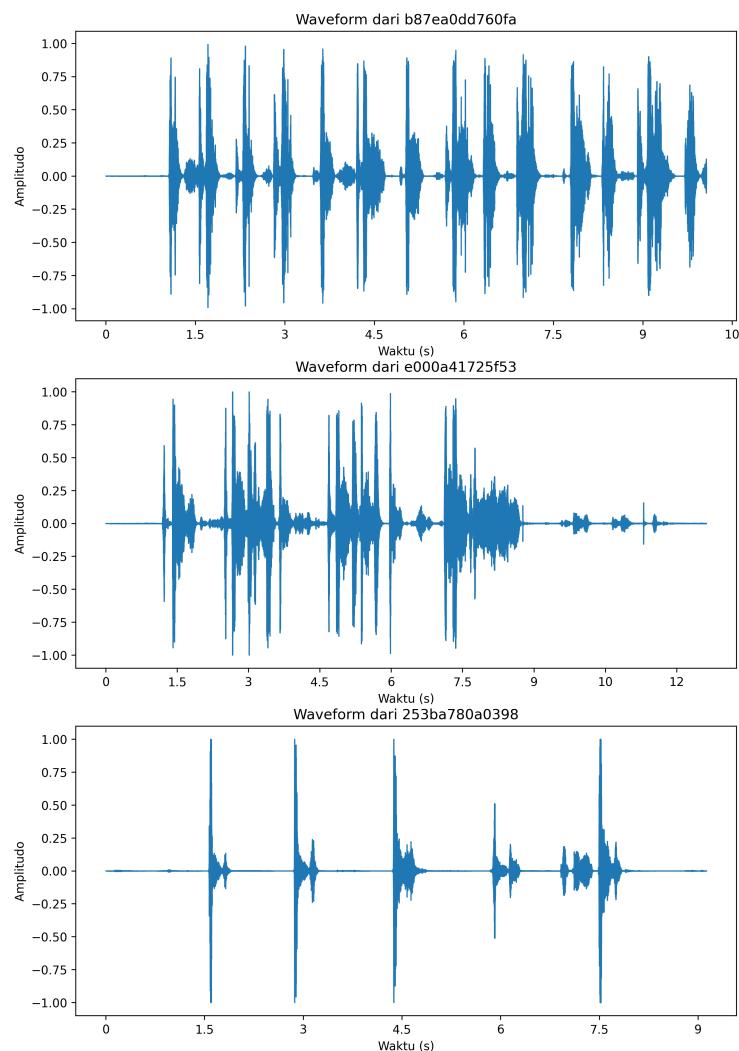
*Sequence* token  $Z$  ditambahkan dengan *positional encoding* yang kemudian menjadi input untuk *encoder transformer*. Terdapat dua metode utama untuk menghasilkan *patch embedding* yaitu, *Uniform Frame Sampling* yang hanya memiliki informasi spasial dan *Tubelet Embedding* yang menghasilkan *patch* yang mencakup informasi spasial dan temporal.

## BAB III

### METODE PENELITIAN

#### 3.1 Deskripsi Data

Dataset yang digunakan adalah *Medical Sound Classification Challenge* yang diselenggarakan oleh Himanshu Kaushik pada platform Kaggle[28]. Dataset ini memiliki 882 jumlah data yang terdiri dari tiga jenis data untuk setiap individu yaitu audio batuk, vowel dan fitur riwayat pasien. Data diidentifikasi menggunakan candidateID yang ditetapkan untuk setiap orang. File suara dan embedding terdapat di folder candidateID yang diberikan. Dataset terbagi menjadi dua jenis yaitu 544 untuk data latih dan 338 untuk data uji yang belum memiliki label. Label penyakit terdiri dari 3 kelas yaitu asma, COPD dan pasien sehat.

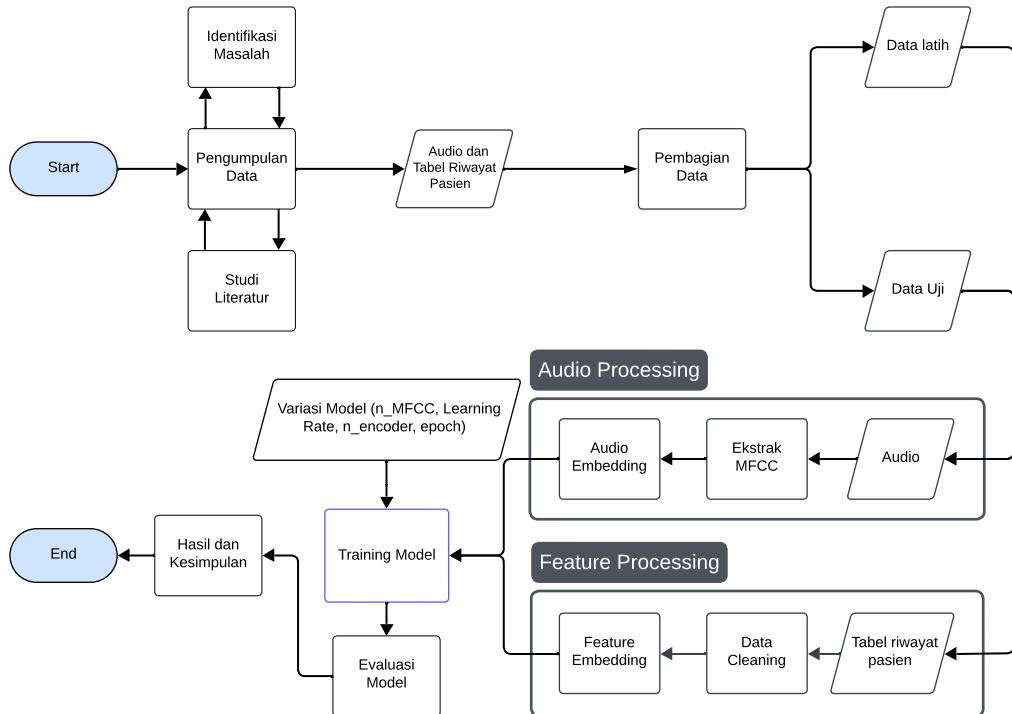


**Gambar 3.1** Visualisasi gelombang suara tiap kelas

**Tabel 3.1** Tabel riwayat pasien

<b>candidateID</b>	<b>age</b>	<b>gender</b>	<b>tbContact</b>	<b>wheezing</b>	<b>phlegmCough</b>	<b>familyAsthma</b>	<b>feverHistory</b>	<b>coldPresent</b>	<b>packYears</b>	<b>disease</b>
2bbd6c5ecf1ce	55	1	0.0	0.0	0.0	1.0	0.0	0.0	0	1
75fa6e335b5ca	65	0	0.0	1.0	0.0	0.0	0.0	1.0	560	2
7dc99cfcb5aa	43	0	0.0	1.0	0.0	0.0	0.0	0.0	0	1
59cf4a7821471	74	0	0.0	1.0	0.0	0.0	0.0	0.0	800	2
59f9fe56c2f12	28	0	0.0	0.0	1.0	0.0	0.0	0.0	0	0
caee891a86d8d	56	1	0.0	1.0	0.0	0.0	0.0	0.0	0	1
5b8578b39385f	31	0	0.0	0.0	1.0	0.0	0.0	0.0	0	0
f3e7d50ce7288	35	0	0.0	1.0	1.0	0.0	0.0	0.0	0	1
ad5fa122d4efb	56	1	0.0	0.0	0.0	1.0	0.0	0.0	0	0
14b58d18c66c7	57	0	0.0	1.0	1.0	0.0	0.0	0.0	0	0
7959121db060d	48	0	0.0	1.0	0.0	0.0	0.0	0.0	0	1
77aa6a34f6da1	21	0	0.0	0.0	1.0	1.0	0.0	0.0	0	1
069276518f6f	25	1	0.0	0.0	0.0	1.0	0.0	0.0	0	2
3d71862ec1801	48	1	0.0	0.0	1.0	1.0	0.0	0.0	0	1
1868d92d2db4	20	0	0.0	0.0	0.0	1.0	0.0	1.0	360	2
fbf8398bd0136	70	0	0.0	0.0	1.0	0.0	0.0	0.0	0	1
0343c366074ec	25	0	0.0	1.0	1.0	0.0	0.0	0.0	0	1
...	...	...	...	...	...	...	...	...	...	...
e17cb00bd9677	66	0	0.0	0.0	1.0	0.0	0.0	1.0	0	2

## 3.2 Rancangan Penelitian



Gambar 3.2 Flowchart rancangan penelitian

### 3.2.1 Pengumpulan Data

Data diunduh ke dalam *notebook* menggunakan kaggle-api dengan fungsi yang telah disediakan di laman kompetisi. File unduhan berupa .zip yang harus diekstrak terlebih dahulu.

### 3.2.2 Pemrosesan Suara

Data suara yang digunakan pada penelitian ini berupa suara batuk *cough.wav* pasien yang berisi minimal 3 kali batuk, dengan durasi rekaman maksimal 15 detik. Dilakukan ekstraksi fitur menggunakan *Mel frequency Cepstral Coefficients* (MFCC), yaitu dengan cara memfilter secara logaritmik pada frekuensi di atas 1000 Hz dan secara linier pada frekuensi di bawah 1000 Hz. MFCC dapat meningkatkan sensitivitas pada suara dengan frekuensi rendah dan sebaliknya, pada suara dengan frekuensi tinggi MFCC dapat mengurangi sensitivitas dalam menangkap suara [29]. Algoritma MFCC dapat dilihat pada Algoritma 2.

Kode `segment_cough_sound` pada Algoritma 1 berfungsi untuk mendeteksi dan memotong bagian-bagian sinyal audio yang mengandung suara batuk berdasarkan energi sinyalnya. Pertama, sinyal diubah menjadi satu channel jika berbentuk

stereo, lalu dihitung nilai Root Mean Square (RMS) energy untuk setiap segmen kecil audio. Energi ini dinormalisasi, kemudian dibandingkan dengan ambang batas tertentu (`cough_threshold`) untuk menentukan kapan sebuah kejadian (batuk) terjadi. Jika energi melewati ambang batas, bagian tersebut dianggap sebagai potensi batuk, dan dipotong dari sinyal asli dengan sedikit penambahan waktu sebelum dan sesudahnya (`padding`) untuk menangkap suara batuk secara lebih lengkap. Potongan-potongan audio hasil segmentasi ini kemudian dikembalikan sebagai daftar array sinyal-sinyal batuk.

---

**Algorithm 1** Segment Cough Sound

---

```
1: Input: signal, sr, cough_threshold, min_cough_duration, padding
2: hop_length ← int(min_cough_duration × sr)
3: if signal has more than 1 channel then
4:     signal ← mean across channels
5: end if
6: energy ← RMS energy of signal with hop_length
7: normalized_energy ← normalize(energy)
8: cough_threshold ← max(normalized_energy) × cough_threshold
9: min_cough_samples ← round(sr × min_cough_duration)
10: cough_segments ← empty list
11: event_start ← None
12: for each (i, value) in normalized_energy do
13:     if value ≥ cough_threshold then
14:         if event_start is None then
15:             event_start ← i × hop_length
16:         end if
17:     else
18:         if event_start is not None then
19:             cough_duration ← i × hop_length – event_start
20:             if cough_duration ≥ min_cough_samples then
21:                 event_end ← i × hop_length + padding × sr
22:                 event_start ← max(event_start – padding × sr, 0)
23:                 append      segment      signal[event_start:event_end+1]      to
24:                     cough_segments
25:             end if
26:         end if
27:     end if
28: end for
29: Return cough_segments
```

---

---

**Algorithm 2** Extract MFCC

---

```
1: Input: file_path, n_mfcc, target_length
2: audio, sr ← load audio from file_path
3: cough_segments ← Segment Cough Sound(audio, sr)
4: if no cough_segments found then
5:     segmented_audio ← full audio
6: else
7:     segmented_audio ← concatenate all cough_segments
8: end if
9: mfcc ← MFCC features from segmented_audio
10: transpose mfcc to shape (time_steps, n_mfcc)
11: if length of mfcc > target_length then
12:     truncate mfcc to target_length
13: else if length of mfcc < target_length then
14:     pad mfcc with zeros to reach target_length
15: end if
16: Return mfcc as Tensor
```

---

### 3.2.3 Pemrosesan Data Tabel Riwayat Pasien

Dilakukan *preprocessing* seperti pengecekan nilai kosong, penghapusan fitur yang tidak relevan untuk memastikan kualitas data yang baik agar model dapat menangkap informasi yang relevan pada data.

---

**Algorithm 3** Pemrosesan Data

---

```
1: function PREPROCESSFEATURES(row)
2:   Normalize numerical features
3:   Extract categorical features
4:   Combine all features into a vector
5:   return Feature vector
6: end function
7: function PROCESSRow(row)
8:   Get candidate ID: id  $\leftarrow$  row[”candidateID”]
9:   Generate audio path: audio_path  $\leftarrow$ 
   JOIN(SOUND_FOLDER, id, ”cough.wav”)
10:  Extract MFCC: mfcc  $\leftarrow$  EXTRACTMFCC(audio_path)
11:  Process features: features  $\leftarrow$  PREPROCESSFEATURES(row)
12:  One-hot encode label: label  $\leftarrow$  ONEHOTENCODE(row[”disease”], depth =
   3)
13:  return (mfcc, features, label)
14: end function
```

---

### 3.2.4 Pipeline dan Pembagian Data

Data dibagi menjadi data latih dan data uji dengan perbandingan 80% untuk data latih dan 20% untuk data uji. Digunakan parameter `random_state` agar proses pembagian data bisa direproduksi ulang. Dilakukan *stratified sampling* berdasarkan kolom target yaitu *disease* untuk memastikan distribusi kelas tetap sama di kedua set.

---

**Algorithm 4** PipeLine Model dan pembagian data

---

```
1: function DATAGENERATOR(data)
2:   for all row  $\in$  data do
3:     (mfcc, features, label)  $\leftarrow$  PROCESSRow(row)
4:     yield(mfcc, features, label)
5:   end for
6: end function
7: function CREATEDATASET(data, batch_size)
8:   Define dataset signature
9:   Create dataset using DATAGENERATOR(data)
10:  Batch, shuffle, and prefetch dataset
11:  return Dataset
12: end function
13: function SPLITDATA(data, test_size, random_S tate)
14:   Split dataset into train and validation sets with stratify target
15:   return (train_data, valid_data)
16: end function
17: (train_data, valid_data)  $\leftarrow$  SPLITDATA(data, 0.2, 42)
18: train_dataset  $\leftarrow$  CREATEDATASET(train_data, BATCH_SIZE)
19: valid_dataset  $\leftarrow$  CREATEDATASET(valid_data, BATCH_SIZE)
```

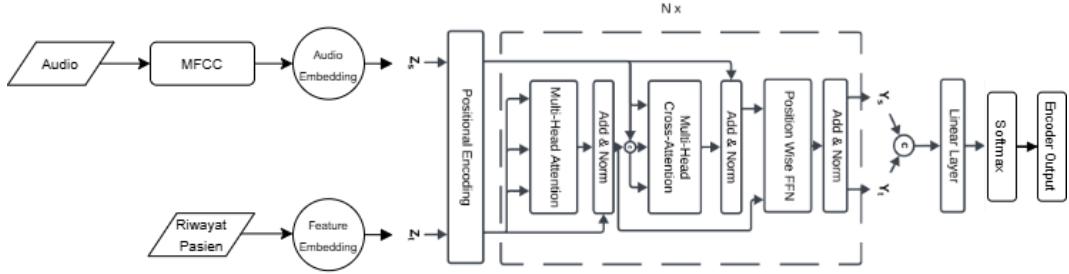
---

### 3.2.5 Pemodelan

Pada tahapan ini dilakukan rancangan model yang akan digunakan, seperti desain arsitektur, *loss function*, dan *optimizer* beserta *hyperparameter*-nya. Pada penelitian ini juga akan menggunakan beberapa desain arsitektur yang berfokus pada kedalaman seperti jumlah *block* dan *hyperparameter* jumlah *attention head*.

### 3.2.6 Arsitektur

Penelitian ini akan menggunakan MFCC untuk mengekstraksi fitur audio dan *embedding* data riwayat pasien.



Gambar 3.3 Desain rancangan arsitektur model.

Input berupa audio dengan fitur MFCC berukuran  $T \times F$  (dengan  $T$  adalah jumlah frame dan  $F$  adalah jumlah koefisien MFCC) dan data riwayat pasien berukuran  $N \times d_{feat}$  ( $N$  adalah jumlah data riwayat,  $d_{feat}$  adalah dimensi fitur) akan diubah menjadi embedding berukuran  $T \times d_{embed}$  dan  $N \times d_{embed}$  melalui *embedding layer*. *Embedding* ini ditambahkan dengan *positional encoding* dan diteruskan ke *encoder* berbasis transformer, di mana mekanisme *self-attention* dan *cross-attention* digunakan untuk menangkap hubungan antar fitur audio serta interaksi antara audio dan riwayat pasien. Setelah melalui *feedforward network* (FFN), hasil akhir *encoder* berupa *embedding multimodal* dikombinasikan dan diproyeksikan menggunakan linear layer, diikuti oleh *softmax* untuk menghasilkan output akhir dari token [cls] berupa representasi yang dapat digunakan untuk klasifikasi.

### 3.2.7 Loss Function

Fungsi kerugian yang digunakan *Categorical Cross-Entropy* untuk setiap kelas 3.1:

$$L(y, \hat{y}) = - \sum_{i=1}^C y_i \log(\hat{y}_i) \quad (3.1)$$

Keterangan:

- $L(y, \hat{y})$  adalah nilai *Categorical Cross-Entropy*
- $y$  adalah nilai target asli
- $\hat{y}_i$  adalah nilai prediksi

Di mana  $L(y, \hat{y})$  merupakan *categorical cross-entropy loss*.  $y_i$  adalah label sebenarnya (0 atau 1 untuk setiap kelas) dari *one-hot encoding* vektor target.  $\hat{y}_i$  adalah probabilitas yang diprediksi untuk kelas  $i$ .  $C$  merupakan jumlah kelas yang diprediksi.

### 3.2.8 Pelatihan Model

Penelitian ini akan menggunakan *optimizer* sama dengan yang digunakan Vaswani et al., yaitu *optimizer* Adam [30] yang menggabungkan momentum dengan RMSprop. Model akan dilatih menggunakan *Notebook Kaggle* dengan GPU NVIDIA Tesla P100-PCIE-16GB untuk model *Base* dan *Big*. Beberapa kedalaman *layer* dan jumlah *head attention* akan diuji performanya dengan tetap memperhatikan batasan komputasi.

---

#### Algorithm 5 Pelatihan Model

---

```
1: Inisialisasi:  $m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0, \beta_1 \leftarrow 0.9, \beta_2 \leftarrow 0.98, \epsilon \leftarrow 10^{-9}$ 
2: Inisialisasi: epochs, batch_size
3: while epoch → epochs do
4:   while step → N // batch_size do
5:      $lr \leftarrow (d_{\text{embed}}^{-0.5} \cdot \min(\text{step}^{-0.5}, \text{step} \times 4000^{-1.5}))$ 
6:      $t \leftarrow t + 1$ 
7:      $g_t \leftarrow \nabla_{\theta} \mathcal{L}(\theta_{t-1})$  (Gradien loss function)
8:      $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  (Update momentum pertama)
9:      $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update momentum kedua)
10:     $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Koreksi bias momentum pertama)
11:     $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Koreksi bias momentum kedua)
12:     $\theta_t \leftarrow \theta_{t-1} - lr \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameter)
13:  end while
14: end while
15: Return: Parameter  $\theta_t$ 
```

---

## 3.3 Evaluasi Model

### 3.3.1 Confusion Matrix

*Confusion matrix* adalah metode dalam pembelajaran mesin yang menyediakan representasi visual dari performa model dalam tugas klasifikasi [31]. Matriks ini merangkum prediksi yang benar dan salah yang dibuat oleh model, yang memungkinkan penghitungan berbagai metrik kinerja seperti akurasi, presisi, *recall*, dan *F1-score*. Matriks ini biasanya terdiri dari empat komponen: *True Positive*, *True Negative*, *False Positive*, dan *False Negative*.

**Tabel 3.2** Confusion Matrix untuk 3 Kelas (a), Rumus metrik evaluasi (b)

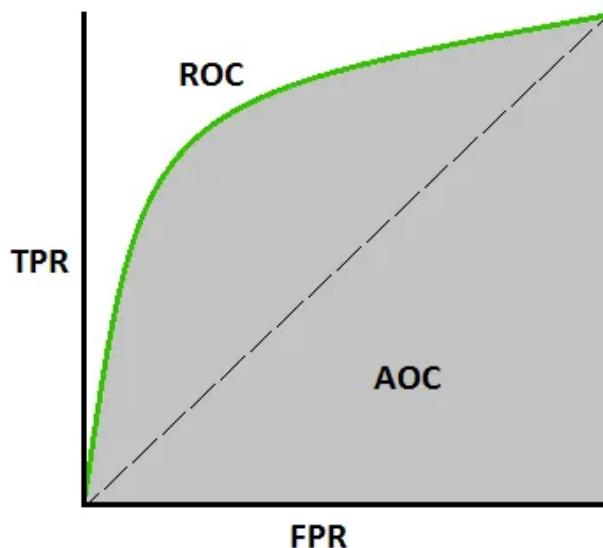
		Predicted Class			Metrik	Rumus
Actual Class		$C_1$	$C_2$	$C_3$		
Actual	$C_1$	$TP_1$	$FP_{12}$	$FP_{13}$	Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
	$C_2$	$FN_{21}$	$TP_2$	$FP_{23}$	Precision	$\frac{TP}{TP+FP}$
	$C_3$	$FN_{31}$	$FN_{32}$	$TP_3$	Recall	$\frac{TP}{TP+FN}$
(a)					F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
					(b)	

### 3.3.2 Kurva ROC-AUC

Kurva ROC-AUC merupakan metode untuk mengevaluasi keakuratan algoritma klasifikasi, yang memberikan wawasan tentang kinerjanya di berbagai ambang batas. Kurva ini mengukur keseimbangan antara sensitivitas (*true positive rate*) dan spesifitas (*false positive rate*), yang memungkinkan penilaian komprehensif terhadap efektivitas model [32].

$$\text{TPR/sensitivitas} = \frac{TP}{TP + FN} \quad (3.2)$$

$$\text{FPR/spesifitas} = \frac{FP}{FP + TN} \quad (3.3)$$



**Gambar 3.4** Kurva ROC-AUC

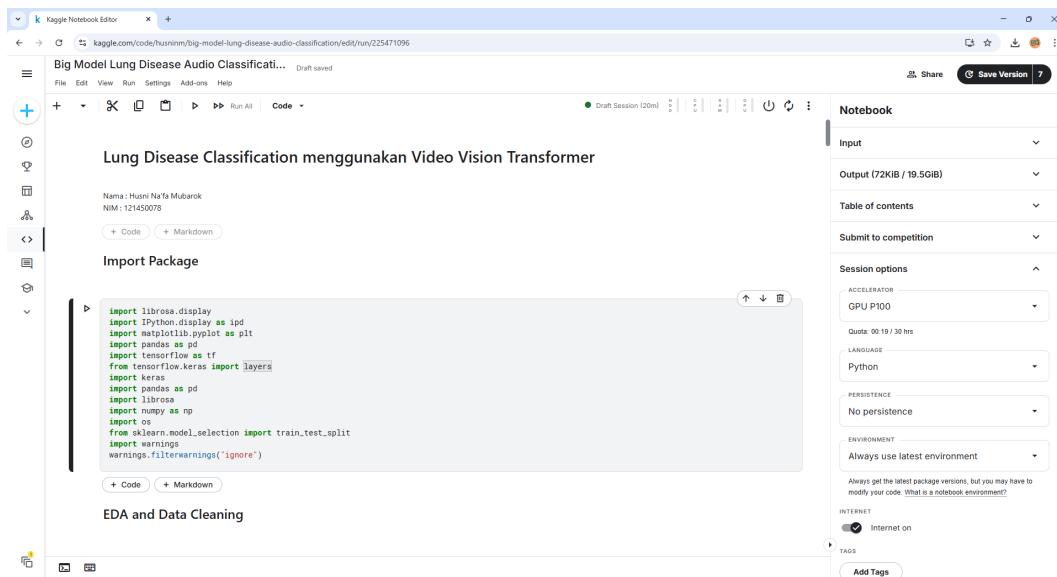
# BAB IV

## HASIL DAN PEMBAHASAN

### 4.1 Persiapan Workspace Kaggle Notebook

#### 4.1.1 Buat Notebook Kaggle

Pembuatan *Notebook* pada penelitian ini merupakan tahap yang penting karena semua proses komputasi akan dijalankan pada *platform* Kaggle. *Notebook Kaggle* dibuat pada laman [www.kaggle.com/code](http://www.kaggle.com/code), lalu klik "New Notebook" maka akan muncul tampilan *jupyter notebook* seperti pada gambar 4.1.



Gambar 4.1 Kaggle Workspace

#### 4.1.2 Input Data

Pada panel sebelah kanan terdapat kolom input untuk memasukkan data, masukkan kata kunci sesuai gambar 4.2 maka akan muncul dataset "Medical Sound Classification Challenge" lalu klik lambang tambah maka data berhasil diinput ke dalam notebook.

```

import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers
import keras
import pandas as pd
import librosa
import numpy as np
import os
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")

train = pd.read_csv("./kaggle/input/airs-ai-in-respiratory-sounds/train.csv")
train = train[train['candidateID'] != 'See502f2832c2']
print(train.info())
print(train.isnull().sum())
train

train = train.drop(columns=["coldPresent"])
train.isnull().sum()

SOUND_FOLDER = "/kaggle/input/airs-ai-in-respiratory-sounds/sounds/sounds"

```

**Gambar 4.2** Tambahkan Dataset

#### 4.1.3 Atur Sesi menggunakan GPU

Pengaturan *session notebook* terdapat pada panel sebelah kanan. Pada bagian "Session options" pilih akselerator menggunakan GPU P100 seperti pada gambar 4.3

Session options

- ACCELERATOR: None
- None
- GPU T4 x2
- GPU P100
- TPU VM v3-8

INTERNET

- Internet on

TAGS

- Add Tags
- GitHub

Schedule a notebook to run

Code Help

**Gambar 4.3** Pengaturan Sesi

#### 4.1.4 Import Package/Library

Library Python berisi fungsi-fungsi yang digunakan pada penelitian ini. Deskripsi library dijelaskan pada tabel 4.1

**Tabel 4.1** Daftar Library Python dan Deskripsinya

Library	Deskripsi
<code>librosa</code>	Digunakan untuk analisis dan pemrosesan audio, seperti ekstraksi fitur suara.
<code>librosa.display</code>	Modul dari <code>librosa</code> untuk menampilkan dan memvisualisasikan data audio.
<code>IPython.display</code>	Menyediakan fungsi untuk menampilkan media interaktif, seperti audio dan video di notebook Jupyter.
<code>matplotlib.pyplot</code>	Digunakan untuk membuat visualisasi data, termasuk grafik dan diagram.
<code>pandas</code>	Library untuk manipulasi dan analisis data dalam bentuk tabel (DataFrame).
<code>tensorflow</code>	Framework untuk machine learning dan deep learning.
<code>tensorflow.keras.layers</code>	Modul dari Keras dalam TensorFlow untuk membangun arsitektur jaringan saraf.
<code>keras</code>	High-level API untuk membangun dan melatih model deep learning.
<code>numpy</code>	Digunakan untuk komputasi numerik dan operasi array multidimensi.
<code>os</code>	Modul standar Python untuk berinteraksi dengan sistem file dan direktori.
<code>sklearn</code>	Modul dari <code>scikit-learn</code> yang digunakan untuk membagi data dan mengevaluasi model.
<code>random</code>	Modul yang digunakan untuk inisiasi nilai acak dengan <code>set_seed</code> agar nilai acak dapat di reproduksi.
<code>warnings</code>	Modul untuk menangani dan menyembunyikan peringatan di Python.

## 4.2 *Exploratory data analysis dan Data Cleaning*

Sebelum memulai pelatihan model, serangkaian proses pengecekan dataset dilakukan untuk mendapatkan kualitas dataset yang baik. Tahapan ini penting untuk menjamin validitas hasil penelitian serta memastikan model berjalan dengan baik selama proses pelatihan.

1. **Pengecekan Data Kosong** dilakukan untuk mengidentifikasi dan menangani nilai-nilai yang hilang dalam data. Ini penting untuk memastikan integritas data dan untuk mempersiapkan data sebelum analisis lebih lanjut. Dengan mengetahui lokasi dan kuantitas data kosong, kita dapat memutuskan strategi penanganan yang tepat, seperti pengisian nilai atau penghapusan data.

```
# Print info and null value summary
print(train.info())
print(train.isnull().sum())

# Display the updated dataframe
train

<class 'pandas.core.frame.DataFrame'>
Index: 545 entries, 0 to 545
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   candidateID      545 non-null    object  
 1   age               545 non-null    int64  
 2   gender            545 non-null    int64  
 3   tbContactHistory 545 non-null    float64 
 4   wheezingHistory   545 non-null    float64 
 5   phlegmCough       545 non-null    float64 
 6   familyAsthmaHistory 545 non-null    float64 
 7   feverHistory      545 non-null    int64  
 8   coldPresent       397 non-null    float64 
 9   packYears          545 non-null    int64  
 10  disease            545 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 51.1+ KB
None
candidateID      0
age              0
gender            0
tbContactHistory 0
wheezingHistory   0
phlegmCough       0
familyAsthmaHistory 0
feverHistory      0
coldPresent       148
packYears          0
disease           0
dtype: int64
```

**Gambar 4.4** Pengecekan Data Kosong

2. **Menghapus Kolom kosong** untuk membersihkan dataset dari kolom yang tidak memiliki informasi yang berguna. Hal ini dapat meningkatkan efisiensi analisis data dan menghindari gangguan dari data yang tidak lengkap atau tidak relevan.

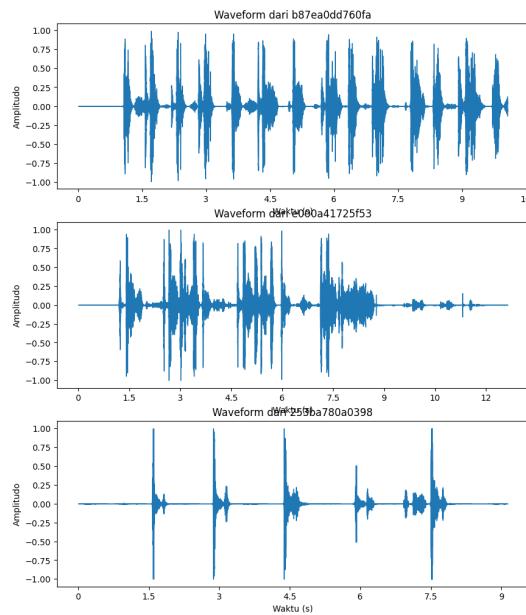
```
In [3]: train = train.drop(columns=["coldPresent"])
train.isnull().sum()

Out[3]: candidateID      0
age              0
gender            0
tbContactHistory 0
wheezingHistory   0
phlegmCough       0
familyAsthmaHistory 0
feverHistory      0
packYears          0
disease           0
dtype: int64
```

**Gambar 4.5** Menghapus Kolom

3. **Visualisasi Sinyal Audio** pada tahap eksplorasi data awal (EDA) memberikan manfaat penting dalam memahami karakteristik dan pola yang

terdapat dalam data audio. Dengan menggunakan grafik seperti spektrum frekuensi atau spektrogram, kita dapat mengidentifikasi fitur-fitur utama seperti frekuensi dominan, dinamika perubahan sinyal dari waktu ke waktu, dan potensi anomali atau noise dalam data. Hal ini membantu dalam proses analisis lebih lanjut, seperti pre-processing atau fitur ekstraksi, dan mendukung pengambilan keputusan yang lebih baik dalam pengembangan model atau aplikasi terkait audio.

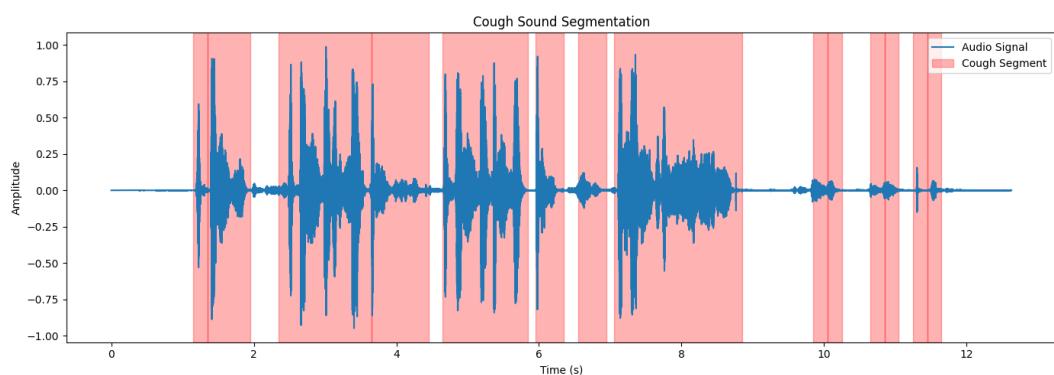


**Gambar 4.6** Visualisasi Audio

### 4.3 Pembuatan Pipeline *Data Processing*

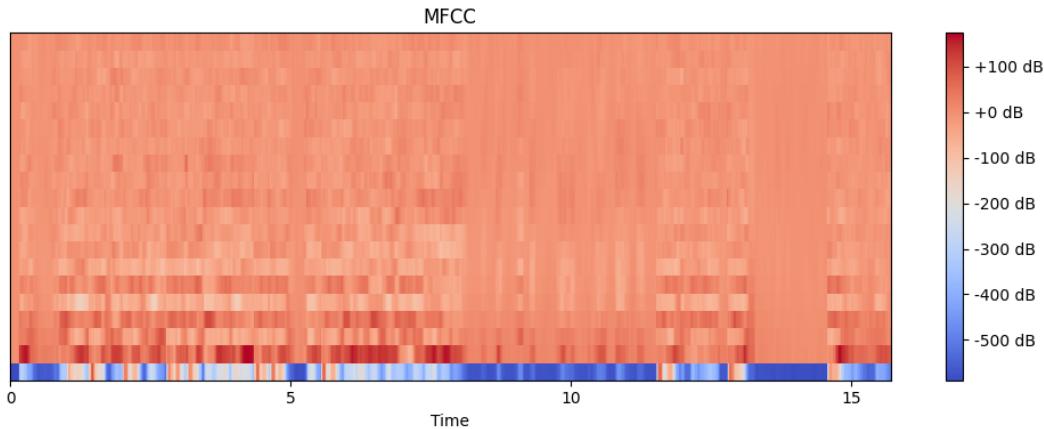
#### 4.3.1 *Audio Processing*

Data suara disegmentasi dan diekstraksi fiturnya menggunakan MFCC. Hasil segmentasi audio dapat dilihat pada 4.7



**Gambar 4.7** Segmentasi Suara batuk

Setelah diaugmentasi, fitur suara diekstrak menggunakan MFCC. Hasil ekstraksi audio menggunakan MFCC dapat dilihat pada 4.8



**Gambar 4.8** Visualisasi MFCC

## 4.4 Pelatihan Model

### 4.4.1 Variasi Model

Tabel 4.2 menunjukkan bahwa model Transformer dengan konfigurasi **Big (d\_model=512, d\_ff=1024, 8 heads, dropout=0.2)** memberikan performa terbaik dengan akurasi 82.57% dan F1-score 81.95%, meskipun jumlah parameternya (20.1 juta) masih tergolong efisien. Sebaliknya, peningkatan ukuran model secara drastis, seperti pada konfigurasi dengan d\_model=1024 dan 80 juta parameter, justru menurunkan performa, kemungkinan akibat overfitting atau kesulitan dalam pelatihan. Selain itu, model Base dengan parameter lebih sedikit justru dapat menghasilkan performa yang cukup kompetitif, terutama konfigurasi pertama yang mencapai akurasi 81.65%. Hal ini menunjukkan bahwa peningkatan kompleksitas model tidak selalu sebanding dengan peningkatan kinerja. Detail rangkuman model dan jumlah parameter terdapat pada lampiran D.

**Tabel 4.2** Variasi parameter untuk model Transformer Base dan Big

<b>Model</b>	<i>N</i>	<i>d<sub>model</sub></i>	<i>d<sub>ff</sub></i>	<i>h</i>	<i>P<sub>drop</sub></i>	<b>Accuracy</b>	<b>F1-Score</b>	Param ( $\times 10^6$ )
Base	3	512	1024	8	0.1	81.65	81.52	10.6
	3	768	1536	12	0.2	77.98	75.83	23.8
Big	6	512	1024	8	0.2	<b>82.57</b>	<b>81.95</b>	20.1
	6	1024	2048	16	0.3	77.98	75.60	80

#### **4.4.2 Perbandingan Sinusoidal Positional Encoding dengan Trainable Positional Encoding**

Tabel 4.3 menunjukkan bahwa model dengan Trainable Positional Encoding secara konsisten mengungguli Sinusoidal Positional Encoding pada semua metrik evaluasi, yaitu akurasi, presisi, recall, dan F1-score. Model dengan Trainable Positional Encoding mencapai akurasi 82.57%, presisi 81.79%, recall 82.48%, dan F1-score 81.95%, jauh lebih tinggi dibandingkan dengan model berbasis Sinusoidal yang hanya mencapai kisaran 61–66% di semua metrik. Hal ini mengindikasikan bahwa positional encoding yang dapat dilatih memungkinkan model untuk memahami representasi posisi secara lebih fleksibel dan sesuai dengan karakteristik data, sehingga menghasilkan performa klasifikasi yang jauh lebih baik.

**Tabel 4.3** Perbandingan evaluasi model dengan Sinusoidal Positional Encoding dan Trainable Positional Encoding.

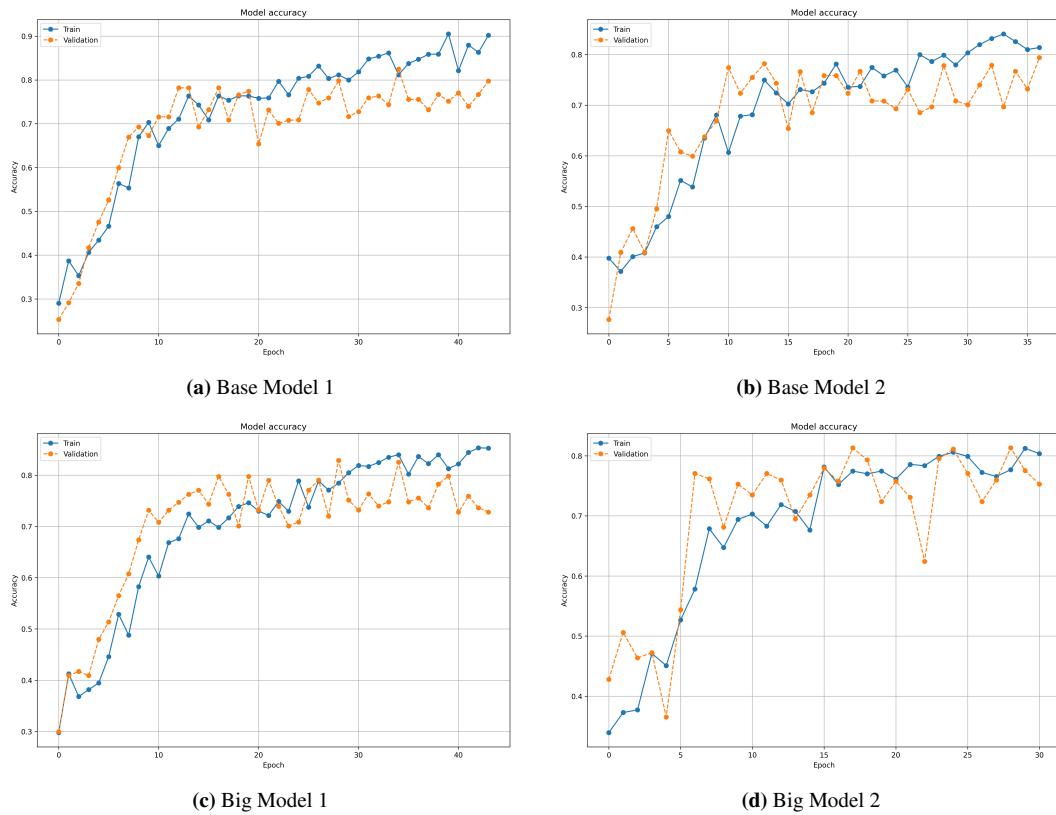
<b>Metrik</b>	<b>Sinusoidal Positional Encoding</b>	<b>Trainable Positional Encoding</b>
Akurasi	66.06%	82.57%
Presisi	66.22%	81.79%
Recall	61.68%	82.48%
F1-Score	62.36%	81.95%

### **4.5 Evaluasi Model**

#### **4.5.1 Akurasi dan Loss Pelatihan**

Selama proses pelatihan, akurasi dan loss terus dimonitor dengan menggunakan *Early Stopping* dengan meminimalkan `val_loss` dengan parameter `patience = 15` dan `restore_best_weights=True` yang berarti model akan memonitor nilai loss validasi dengan nilai terendah selama 15 epoch terakhir, jika selama 15 epoch terakhir tidak terdapat penurunan loss maka proses pelatihan akan dihentikan dan model akan memulihkan bobot terbaik. Penghentian awal digunakan selama pelatihan model deep learning untuk mencegah overfitting. *Early Stopping* memungkinkan proses pelatihan dihentikan setelah performa model pada set validasi mulai menurun, sehingga mempertahankan model yang berkinerja baik pada data yang tidak terlihat.

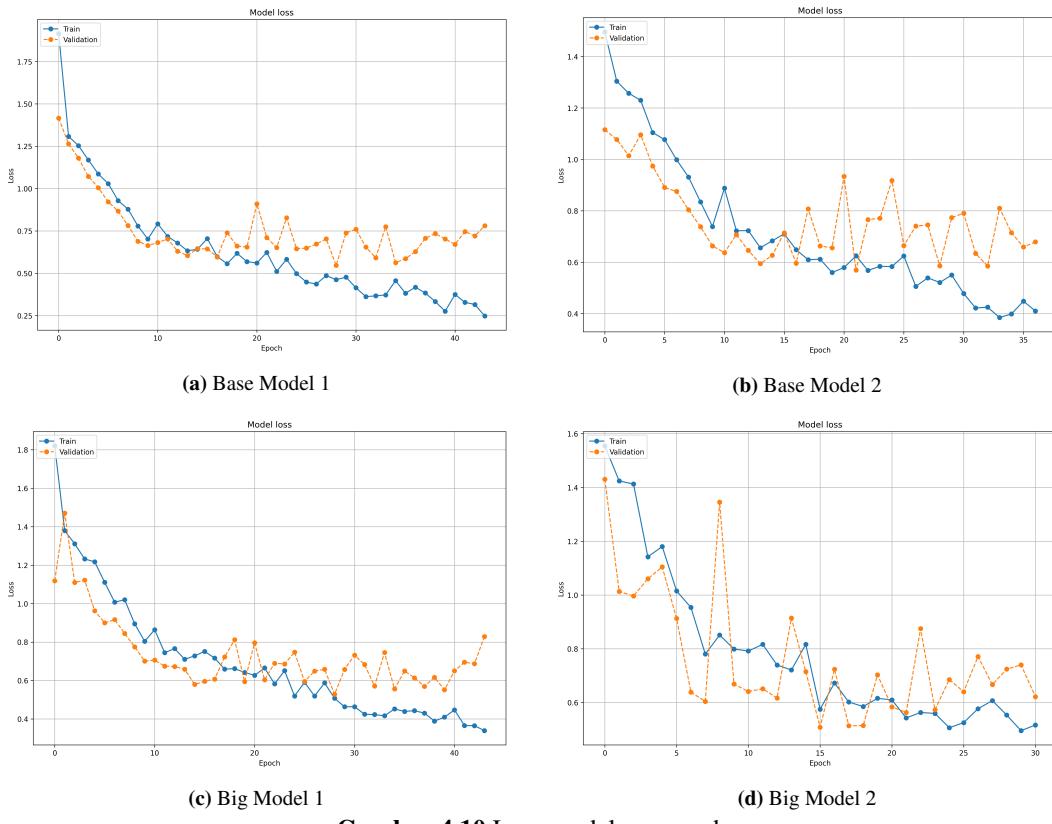
- **Akurasi**



**Gambar 4.9** Akurasi model per epoch

Keempat grafik 4.9 menunjukkan performa akurasi model selama pelatihan dan validasi. Base Model 1 memiliki akurasi pelatihan tinggi namun akurasi validasi stagnan, menunjukkan overfitting. Base Model 2 menunjukkan peningkatan akurasi yang lebih stabil namun performanya lebih rendah secara keseluruhan. Big Model 1 memiliki akurasi pelatihan yang tinggi namun akurasi validasi fluktuatif, mengindikasikan overfitting atau ketidakstabilan generalisasi. Sementara itu, Big Model 2 menunjukkan keseimbangan terbaik antara akurasi pelatihan dan validasi, dengan hasil yang tinggi dan stabil pada keduanya, sehingga merupakan model dengan performa generalisasi terbaik di antara keempat model yang diuji.

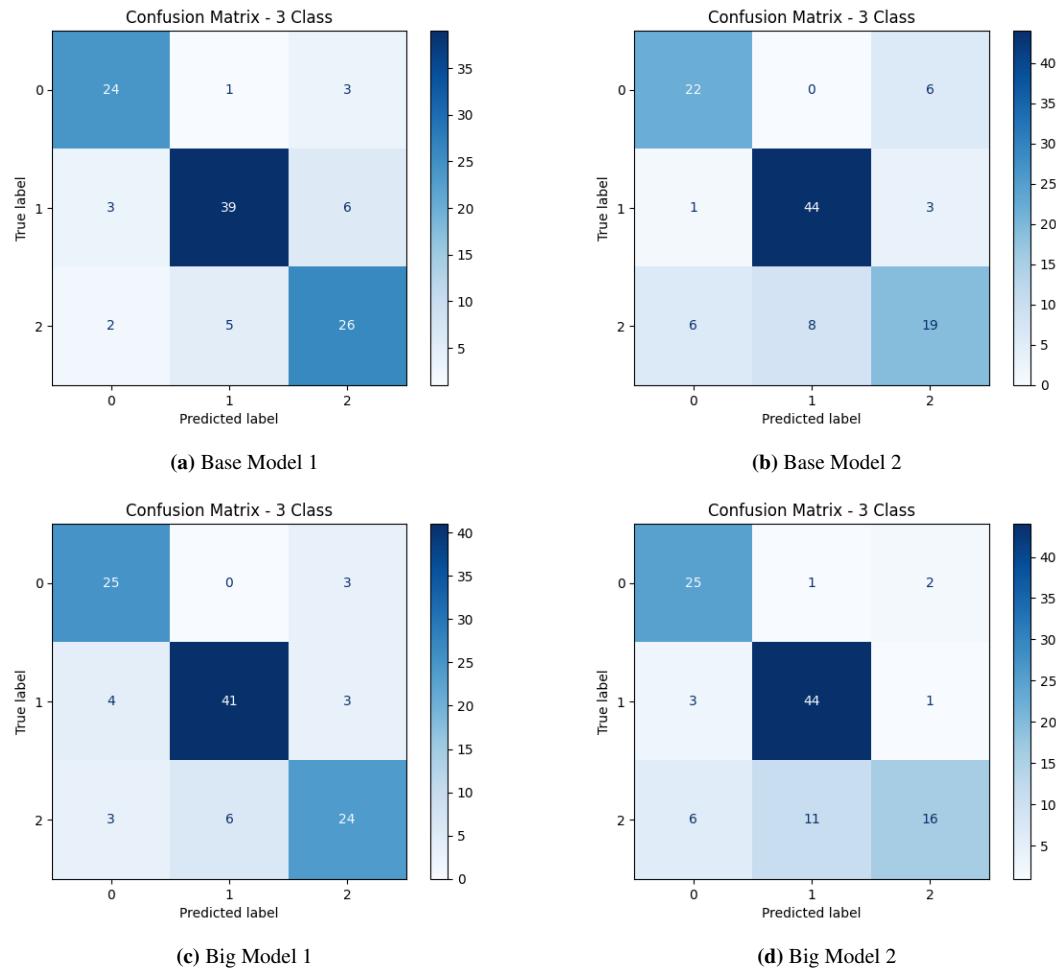
- **Loss**



**Gambar 4.10** Loss model per epoch

Keempat grafik loss 4.10 menunjukkan tren penurunan selama pelatihan, namun dengan pola validasi yang berbeda. Base Model 1 dan Big Model 1 mengalami overfitting yang cukup jelas, ditandai dengan loss pelatihan yang terus menurun sementara loss validasi stagnan dan fluktuatif. Base Model 2 menunjukkan penurunan loss pelatihan yang stabil, namun loss validasi masih cukup fluktuatif dan tidak menunjukkan peningkatan signifikan. Big Model 2 memiliki kinerja terbaik, dengan penurunan loss pelatihan yang konsisten dan loss validasi yang lebih stabil serta mengikuti tren yang serupa, menandakan generalisasi model yang baik. Dengan demikian, Big Model 2 kembali menunjukkan performa paling seimbang di antara semua model.

#### 4.5.2 Confusion Matrix



**Gambar 4.11** Confusion Matrix

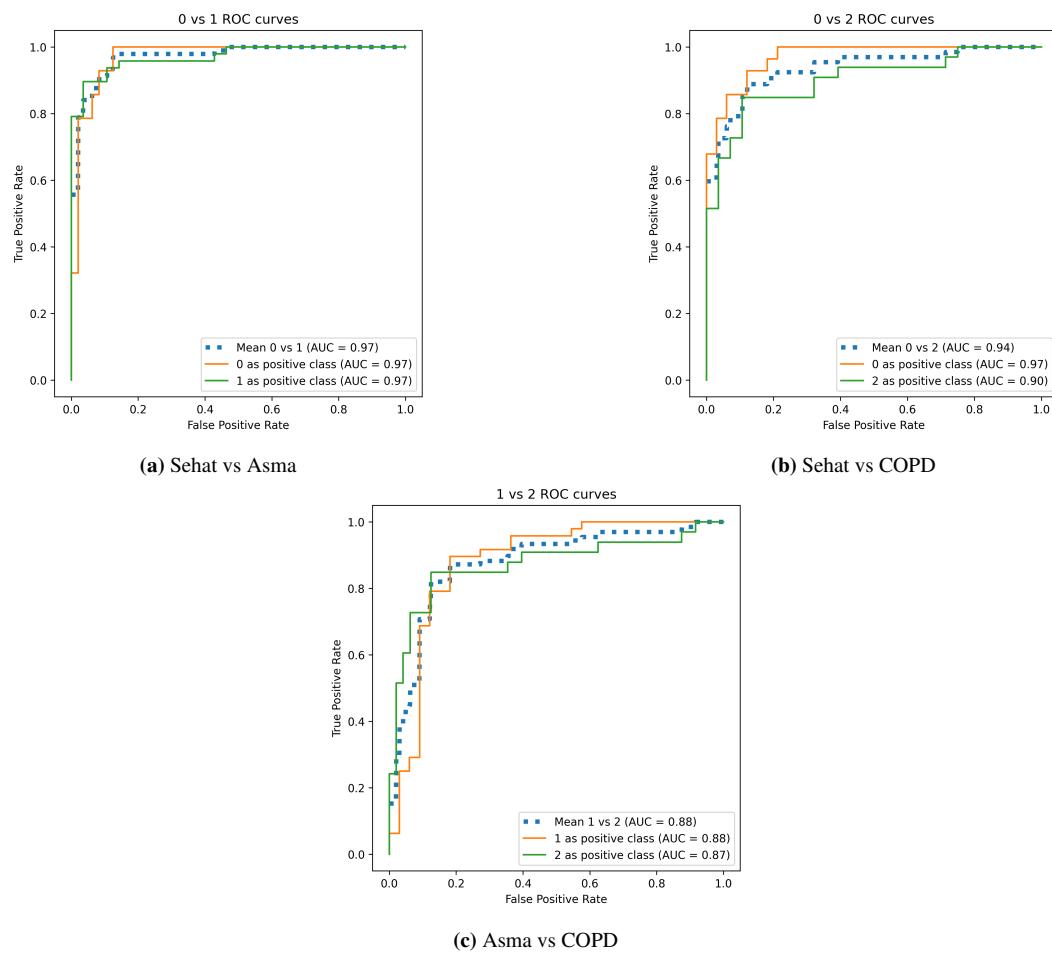
Tabel 4.4 menunjukkan bahwa **Big Model 1** memiliki performa terbaik di antara keempat model, dengan nilai **akurasi tertinggi (82.57%)**, **presisi (81.79%)**, **recall (82.48%)**, dan **F1-score (81.95%)**, yang menunjukkan keseimbangan yang sangat baik antara kemampuan model dalam mendeteksi kelas positif dan menghindari kesalahan klasifikasi. **Base Model 1** menempati posisi kedua dengan nilai yang cukup kompetitif, terutama pada recall (81.92%) dan F1-score (81.52%), menunjukkan bahwa meskipun lebih sederhana, model ini cukup andal. **Base Model 2** dan **Big Model 2** menunjukkan performa yang lebih rendah secara keseluruhan, dengan nilai akurasi yang sama (77.98%), tetapi Big Model 2 unggul sedikit dalam presisi (78.77%) dan recall (76.48%) dibandingkan Base Model 2. Hal ini mengindikasikan bahwa kompleksitas tambahan pada Big Model 2 tidak memberikan peningkatan signifikan dalam kinerja dan bahkan menunjukkan hasil F1-score terendah (75.60%). Secara keseluruhan, Big Model 1 menjadi model paling optimal dari sisi keseimbangan performa klasifikasi.

**Tabel 4.4** Akurasi, Presisi, Recall, dan F1-Score dari 4 Model

Model	Akurasi	Presisi	Recall	F1-Score
Base Model 1	81.65	81.24	81.92	81.52
Base Model 2	77.98	76.11	75.94	75.83
Big Model 1	<b>82.57</b>	<b>81.79</b>	<b>82.48</b>	<b>81.95</b>
Big Model 2	77.98	78.77	76.48	75.60

### 4.5.3 Kurva AUC-ROC

#### 1. Base Model 1

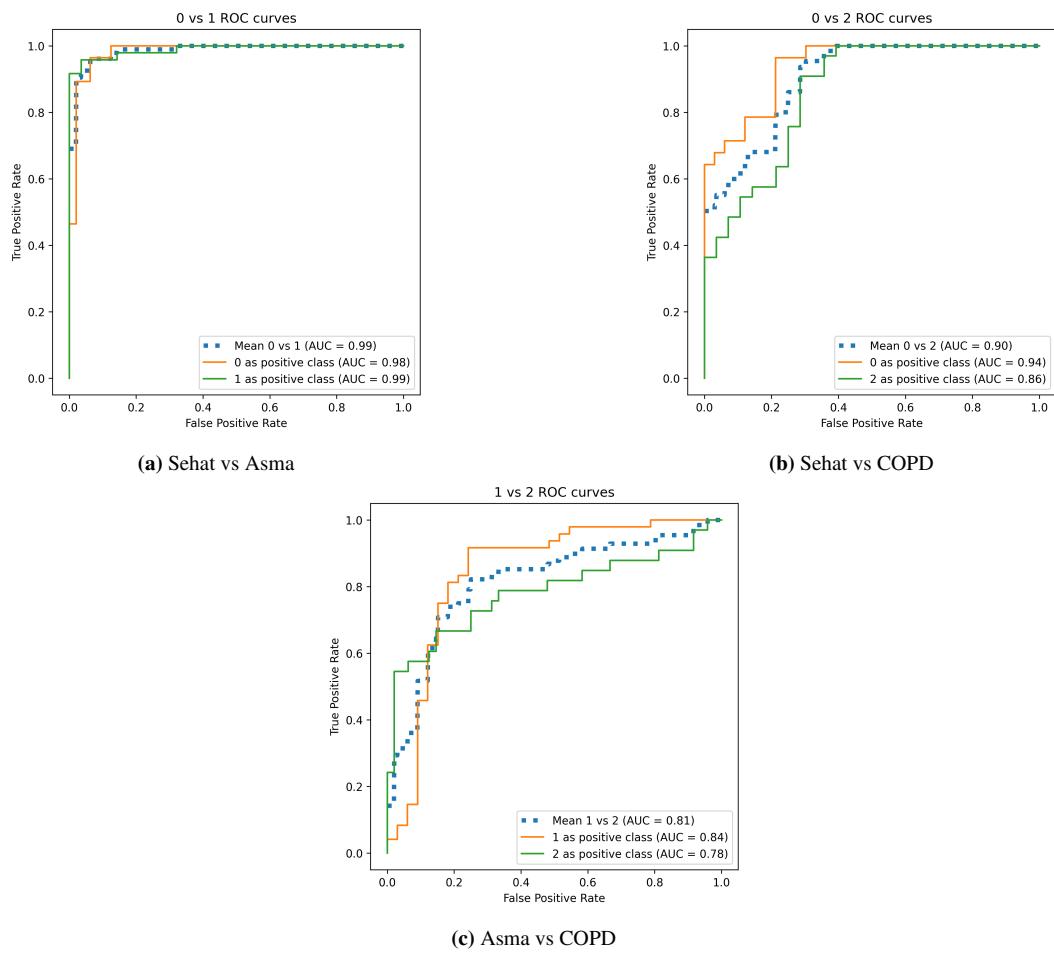


**Gambar 4.12** Kurva AUC-ROC antar kelas Base Model 1

Kurva AUC-ROC Base Model 1 pada gambar 4.12 menunjukkan performa klasifikasi yang sangat baik untuk membedakan kelas sehat dengan asma (AUC = 0.97) dan sehat dengan COPD (AUC = 0.94), menandakan bahwa model mampu mengenali kondisi sehat dengan sangat akurat. Namun, performa menurun saat membedakan antara asma dan COPD (AUC = 0.88), yang menunjukkan bahwa model kesulitan membedakan dua kondisi

penyakit yang memiliki karakteristik serupa. Secara keseluruhan, model unggul dalam klasifikasi antara kondisi sehat dan sakit, namun masih memerlukan peningkatan dalam membedakan jenis penyakit secara lebih spesifik.

## 2. Base Model 2

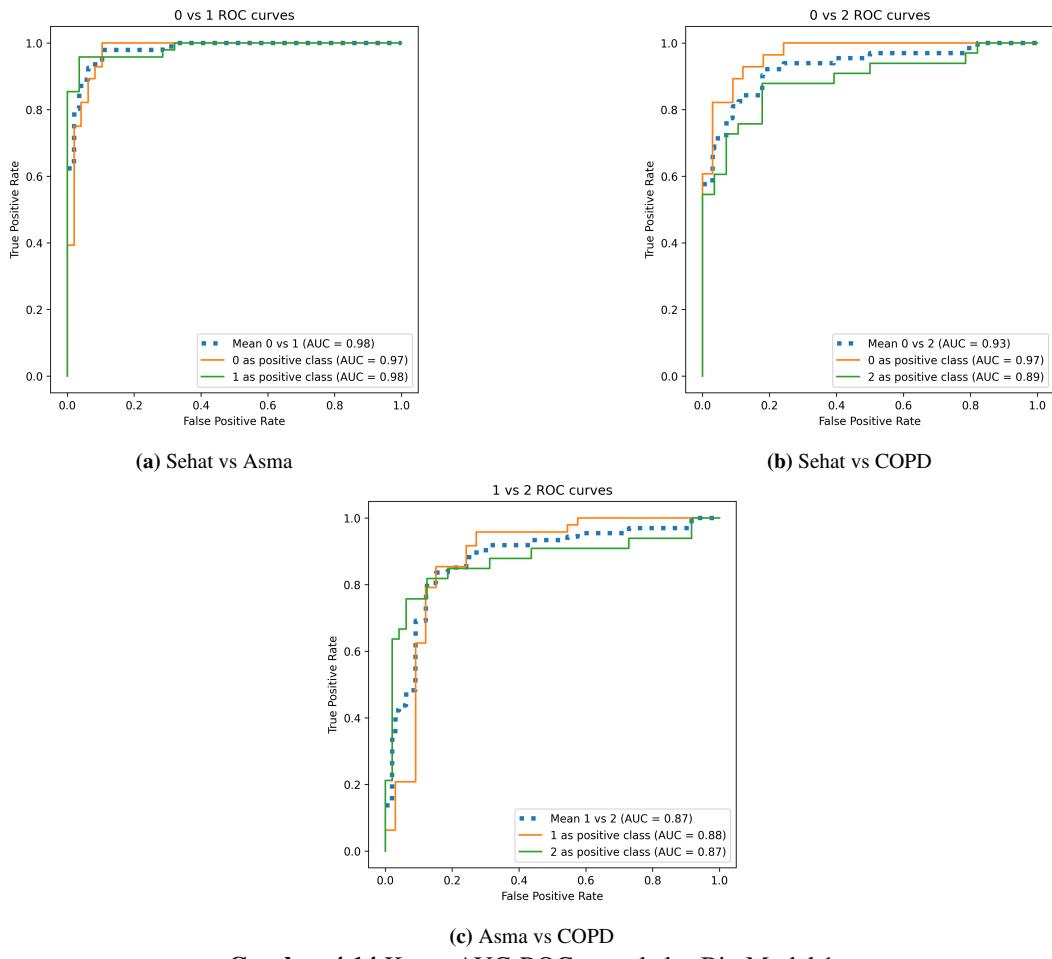


**Gambar 4.13** Kurva AUC-ROC antar kelas Base Model 2

Kurva AUC-ROC pada Base Model 2 4.13 menunjukkan performa yang sangat baik dalam membedakan kelas sehat dengan asma ( $AUC = 0.99$ ), menandakan kemampuan deteksi yang hampir sempurna. Namun, performa menurun untuk pasangan sehat vs COPD ( $AUC = 0.90$ ) dan lebih rendah lagi untuk asma vs COPD ( $AUC = 0.81$ ), terutama saat COPD menjadi kelas positif ( $AUC = 0.78$ ). Ini menunjukkan bahwa meskipun model sangat efektif dalam membedakan individu sehat dari yang sakit, kemampuannya untuk membedakan antara dua kondisi penyakit—terutama COPD—masih terbatas. Kesulitan ini kemungkinan disebabkan oleh kesamaan karakteristik antara asma dan COPD, serta representasi fitur yang kurang optimal untuk

kelas COPD.

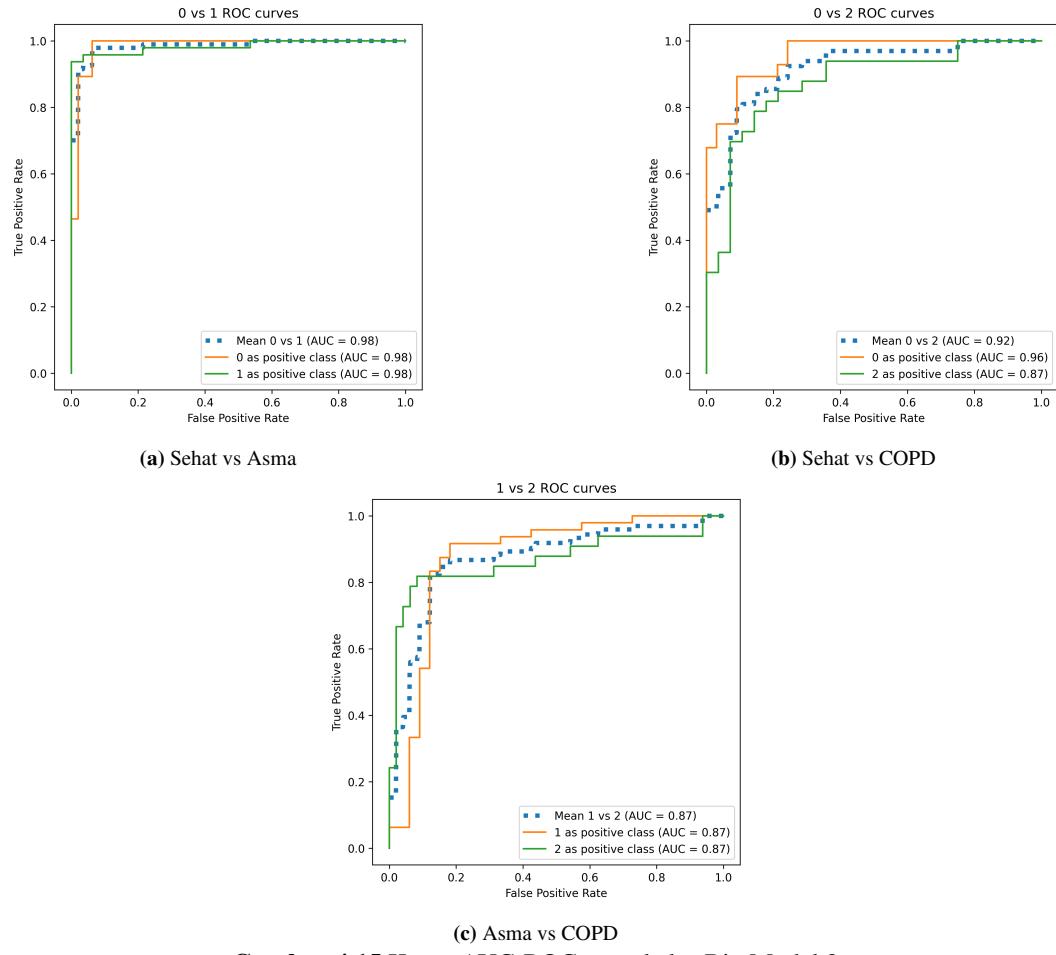
### 3. Big Model 1



**Gambar 4.14** Kurva AUC-ROC antar kelas Big Model 1

Kurva AUC-ROC dari Big Model 1 4.14 menunjukkan performa klasifikasi yang sangat baik dalam membedakan antara kelas sehat dan asma dengan AUC sebesar 0.98, serta antara sehat dan COPD dengan AUC sebesar 0.93. Hal ini mengindikasikan bahwa model mampu mengenali individu sehat dari penderita penyakit dengan akurasi tinggi. Untuk pasangan asma vs COPD, AUC sebesar 0.87 menunjukkan performa yang cukup baik namun tidak seakurat dua pasangan lainnya, kemungkinan karena tumpang tindih karakteristik antara kedua kondisi tersebut. Secara keseluruhan, Big Model 1 menunjukkan keseimbangan yang baik dalam mendeteksi ketiga kelas dengan tingkat ketelitian tinggi.

#### 4. Big Model 2



Gambar 4.15 Kurva AUC-ROC antar kelas Big Model 2

Kurva AUC-ROC dari Big Model 2 4.15 menunjukkan performa klasifikasi yang sangat baik dalam membedakan antara kelas sehat dan asma dengan AUC sebesar 0.98, mencerminkan kemampuan tinggi model dalam mengidentifikasi kedua kondisi tersebut. Untuk pasangan sehat dan COPD, AUC sebesar 0.92 juga menunjukkan performa yang kuat, meskipun sedikit lebih rendah dibanding klasifikasi sehat vs asma. Sementara itu, klasifikasi antara asma dan COPD memiliki AUC sebesar 0.87, yang menunjukkan performa cukup baik namun tetap menantang karena kesamaan karakteristik antara kedua kondisi. Secara keseluruhan, Big Model 2 menunjukkan kapabilitas klasifikasi yang konsisten tinggi di semua kombinasi kelas.

##### 4.5.4 Prediksi Data baru

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan hasil analisis dan evaluasi model pada Bab IV, didapat kesimpulan sebagai berikut:

1. Penerapan konsep *Video Vision Transformer* (ViViT) pada suara batuk penyakit paru-paru dan riwayat pasien menggunakan metode *Cross Attention* menghasilkan performa model yang baik sehingga dapat mengklasifikasikan penyakit paru-paru dengan akurat. Selain itu, *Trainable Positional Encoding* memungkinkan model untuk memahami representasi posisi secara lebih fleksibel dan sesuai dengan karakteristik data, sehingga menghasilkan performa klasifikasi yang jauh lebih baik.
2. Performa model terbaik didapatkan pada varian model Big 1 dengan blok encoder ( $N = 6$ ), Dimensi Embedding ( $d_{model} = 512$ , head attention ( $h = 8$ ) menghasilkan akurasi 82.57%, presisi 81.79%, recall 82.48% dan F1-score 81.95% pada semua kelas.

#### **5.2 Saran**

Saran dari penulis yang dapat dipertimbangkan untuk penelitian selanjutnya adalah sebagai berikut:

1. Melakukan eksperimen dengan menggunakan ekstraksi fitur atau augmentasi suara lainnya selain MFCC seperti Linear Predictive Coding (LPC) dan Pitch-Synchronous Zero-crossing peak-amplitude (PS-ZCPA).
2. Menggunakan dataset dengan jenis penyakit paru yang lebih beragam agar dapat mengakomodir berbagai kondisi penyakit paru-paru lainnya.
3. Penulis menyarankan untuk menggunakan perangkat komputasi yang lebih tinggi untuk mempercepat pelatihan model dan mengakomodir beban komputasi yang cukup tinggi.

## DAFTAR PUSTAKA

- [1] A. Chanda dan G. Singh, “Lung tissue simulants”, di dalam *Soft Tissue Simulants*. Singapore: Springer Nature Singapore, 2024, hlmn. 59–70. sumber: [https://doi.org/10.1007/978-981-97-3060-5\\_6](https://doi.org/10.1007/978-981-97-3060-5_6).
- [2] L. F. Australia, Mei 2023. sumber: <https://lungfoundation.com.au/lung-health/lung-disease/what-is-lung-disease/>.
- [3] M. A. Fadhilah, “Chronic obstructive pulmonary disease”, *Jurnal Medika Nusantara*, vol. 2, no. 2, hlmn. 117–125, Mei 2024. sumber: <https://jurnal.stikeskesdam4dip.ac.id/index.php/Medika/article/view/1127>.
- [4] WHO, *Chronic obstructive pulmonary disease (COPD) — who.int*, [https://www.who.int/news-room/fact-sheets/detail/chronic-obstructive-pulmonary-disease-\(copd\)](https://www.who.int/news-room/fact-sheets/detail/chronic-obstructive-pulmonary-disease-(copd)), [Accessed 27-11-2024], 2024.
- [5] W. EMRO, *WHO EMRO | Chronic obstructive pulmonary disease (COPD) | Health topics — emro.who.int*, <https://www.emro.who.int/health-topics/chronic-obstructive-pulmonary-disease-copd/index.html>, [Accessed 27-11-2024], 2024.
- [6] *Pasien PPOK RI 19 Juta di 2024, Diprediksi Terus Meningkat - Gaya Hidup — bloombergtechnoz.com*, <https://www.bloombergtechnoz.com/detail-news/55485/pasien-ppok-ri-19-juta-di-2024-diprediksi-terus-meningkat>, [Accessed 27-11-2024].
- [7] J. Heitmann, A. Glangetas, J. Doenz, dkk., “Deepbreath—automated detection of respiratory pathology from lung auscultation in 572 pediatric outpatients across 5 countries”, *NPJ digital medicine*, vol. 6, no. 1, hlmn. 104, 2023.
- [8] Y. Ma, X. Xu, dan Y. Li, “Lungrn+ nl: An improved adventitious lung sound classification using non-local block resnet neural network with mixup data augmentation.”, di dalam *Interspeech*, 2020, hlmn. 2902–2906.
- [9] K. He, X. Zhang, S. Ren, dan J. Sun, “Deep residual learning for image recognition”, di dalam *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, hlmn. 770–778.
- [10] S. Gairola, F. Tom, N. Kwatra, dan M. Jain, “Respirenet: A deep neural network for accurately detecting abnormal lung sounds in limited data setting”, di dalam *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, IEEE, 2021, hlmn. 527–530.

- [11] B. Elizalde, S. Deshmukh, dan H. Wang, *Natural language supervision for general-purpose audio representations*, 2024. arXiv: 2309.05767 [cs.SD]. sumber: <https://arxiv.org/abs/2309.05767>.
- [12] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, dan C. Schmid, *Vivit: A video vision transformer*, 2021. arXiv: 2103.15691 [cs.CV]. sumber: <https://arxiv.org/abs/2103.15691>.
- [13] H. Lin, X. Cheng, X. Wu, dan D. Shen, “Cat: Cross attention in vision transformer”, di dalam *2022 IEEE International Conference on Multimedia and Expo (ICME)*, 2022, hlmn. 1–6.
- [14] H. Xue dan F. D. Salim, “Exploring self-supervised representation ensembles for covid-19 cough classification”, di dalam *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Ser. KDD ’21, ACM, Agt. 2021, hlmn. 1944–1952. sumber: <http://dx.doi.org/10.1145/3447548.3467263>.
- [15] L. Xiao, L. Fang, Y. Yang, dan W. Tu, “Lungadapter: Efficient adapting audio spectrogram transformer for lung sound classification”, di dalam *Proc. Interspeech 2024*, 2024, hlmn. 4738–4742.
- [16] V. Basu dan S. Rana, “Respiratory diseases recognition through respiratory sound with the help of deep neural network”, di dalam *2020 4th International Conference on Computational Intelligence and Networks (CINE)*, 2020, hlmn. 1–6.
- [17] C. Barstow dan D. Forbes, “Respiratory conditions: Chronic obstructive pulmonary disease”, *FP essentials*, vol. 486, hlmn. 26–32, Nov. 2019. sumber: <http://europepmc.org/abstract/MED/31710455>.
- [18] D. Singh, M. Miravitles, dan C. Vogelmeier, “Chronic obstructive pulmonary disease individualized therapy: Tailored approach to symptom management”, *Advances in therapy*, vol. 34, hlmn. 281–299, 2017.
- [19] D. M. Mannino, “Chronic obstructive pulmonary disease: Epidemiology and evaluation”, *Hospital physician*, vol. 37, no. 10, hlmn. 22–40, 2001.
- [20] L. D. Benton, “Childhood respiratory conditions: Asthma”, *FP essentials*, vol. 513, hlmn. 11–19, Feb. 2022. sumber: <http://europepmc.org/abstract/MED/35143150>.
- [21] M. Malarvili, T. A. Howe, S. Ramanathan, M. Alexie, dan O. P. Singh, “Chapter two - asthma: The disease and issues in monitoring the asthmatic attack”, di dalam *Systems and Signal Processing of Capnography as a Diagnostic Tool for Asthma Assessment*, M. Malarvili, T. A. Howe, S. Ramanathan, M. Alexie, dan O. P. Singh, timed., Academic Press, 2023,

- hlmn. 25–50. sumber: <https://www.sciencedirect.com/science/article/pii/B9780323857475000073>.
- [22] S. M. Al Sasongko, S. Tsaury, S. Ariessaputra, dan S. Ch, “Mel frequency cepstral coefficients (mfcc) method and multiple adaline neural network model for speaker identification”, *JOIV: International Journal on Informatics Visualization*, vol. 7, no. 4, hlmn. 2306–2312, 2023.
  - [23] J. Sueur, “Mel-frequency cepstral and linear predictive coefficients”, di dalam *Sound Analysis and Synthesis with R*. Cham: Springer International Publishing, 2018, hlmn. 381–398. sumber: [https://doi.org/10.1007/978-3-319-77647-7\\_12](https://doi.org/10.1007/978-3-319-77647-7_12).
  - [24] A. Vaswani, N. Shazeer, N. Parmar, dkk., *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. sumber: <https://arxiv.org/abs/1706.03762>.
  - [25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, dkk., *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV]. sumber: <https://arxiv.org/abs/2010.11929>.
  - [26] J. Devlin, M.-W. Chang, K. Lee, dan K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL]. sumber: <https://arxiv.org/abs/1810.04805>.
  - [27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, dkk., “Language models are unsupervised multitask learners”, *OpenAI blog*, vol. 1, no. 8, hlmn. 9, 2019.
  - [28] H. Kaushik, *Medical sound classification challenge*, <https://kaggle.com/competitions/airs-ai-in-respiratory-sounds>, Kaggle, 2024.
  - [29] M. Avadhani, Jahnavi, A. P. Bidargaddi, dan T. S, “Multi-class urban sound classification with deep learning architectures”, di dalam *2024 5th International Conference for Emerging Technology (INCET)*, 2024, hlmn. 1–7.
  - [30] K. Ahn, Z. Zhang, Y. Kook, dan Y. Dai, *Understanding adam optimizer via online learning of updates: Adam is ftrl in disguise*, 2024. arXiv: 2402.01567 [cs.LG]. sumber: <https://arxiv.org/abs/2402.01567>.
  - [31] E. Manai, M. Mejri, dan J. Fattahi, “Confusion matrix explainability to improve model performance: Application to network intrusion detection”, di dalam *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2024, hlmn. 1–5.
  - [32] A. D. Rahajoe, Agussalim, R. Mumpuni, dkk., “Optimization of binary classification based on receiver operating characteristic area under the curve

- for supervised machine learning”, di dalam *2023 IEEE 9th Information Technology International Seminar (ITIS)*, 2023, hlmn. 1–6.
- [33] K. S. Rao dan K. Manjunath, *Speech recognition using articulatory and excitation source features*. Springer, 2017.
  - [34] H. A. Feldman, N. Kaiser, dan J. A. Peacock, “Power spectrum analysis of three-dimensional redshift surveys”, *arXiv preprint astro-ph/9304022*, 1993.
  - [35] H. Yin, V. Hohmann, dan C. Nadeu, “Acoustic features for speech recognition based on gammatone filterbank and instantaneous frequency”, *Speech Communication*, vol. 53, no. 5, hlmn. 707–715, 2011, Perceptual and Statistical Audition. sumber: <https://www.sciencedirect.com/science/article/pii/S01676393100009%2019>.
  - [36] G. Strang, “The discrete cosine transform”, *SIAM review*, vol. 41, no. 1, hlmn. 135–147, 1999.
  - [37] D. Hendrycks dan K. Gimpel, *Gaussian error linear units (gelus)*, 2023. arXiv: 1606.08415 [cs.LG]. sumber: <https://arxiv.org/abs/1606.08415>.

## LAMPIRAN

## LAMPIRAN A

### *Mel frequency Capstral Coefficients (MFCC)*

#### A.1 Pre-Emphasis

Pre-Emphasis merupakan salah satu praktik pra-pemrosesan umum dalam bidang pemrosesan sinyal yang digunakan untuk mengompensasi frekuensi tinggi sinyal yang ditekan selama produksi sinyal. Pra-penekanan merupakan langkah pertama selama adaptasi MFCC, yang dapat diadopsi hanya dengan menerapkan filter high-pass dengan pengaturan  $[1, -0.97]$ . Proses penyaringan mengubah distribusi energi di seluruh frekuensi, serta tingkat energi keseluruhan. Formula Pre-Emphasis dapat dilihat pada persamaan A.1.

$$\hat{y}[n] = x[n] - \alpha \cdot x[n - 1] \quad (\text{A.1})$$

Keterangan:

- $x[n]$  adalah sinyal input
- $\hat{y}[n]$  adalah sinyal keluaran
- $\alpha$  adalah faktor pre-emphasis

Keluaran  $\hat{y}[n]$  akan menjadi seperti persamaan A.2

$$\{x(0), x(1) - \alpha x(0), \dots, x(n - 1) - \alpha x(n - 2)\} \quad (\text{A.2})$$

#### A.2 *Framing dan Windowing*

Ide di balik pemisahan sinyal menjadi "frame" yang berbeda adalah memecah sinyal data mentah menjadi frame yang sinyalnya cenderung lebih stasioner. Untuk karakteristik akustik yang stabil, suara perlu diperiksa dalam jangka waktu yang cukup singkat. Oleh karena itu, pengukuran spektral jangka pendek biasanya dilakukan selama jendela 20 ms, dan setiap bingkai tumpang tindih 10 ms dengan bingkai berikutnya. Tumpang tindih bingkai sebesar 10 ms memungkinkan karakteristik temporal sinyal audio dilacak. Dengan tumpang tindih bingkai audio, representasi suara akan kira-kira terpusat pada beberapa bingkai. Pada setiap frame, jendela diterapkan untuk mempersempit sinyal ke arah batas frame. Secara umum, jendela Hanning dan Hamming [33] adalah salah satu metode yang paling banyak digunakan. Jendela ini dapat meningkatkan harmonik, menghaluskan tepi, dan mengurangi efek tepi saat melakukan DFT pada sinyal. Framing dan windowing

diterapkan dengan persamaan A.3.

$$x_n[m] = x[n] \cdot w[n] \quad (\text{A.3})$$

Keterangan:

- $w[n]$  adalah fungsi windowing
- $x_n[m]$  adalah hasil windowing
- $x[n]$  adalah sinyal input

### A.3 Discrete Fourier Transform (DFT)

*Discrete Fourier Transform* (DFT) banyak digunakan untuk menghitung spektrum daya. Spektrum daya dapat dideskripsikan sebagai distribusi dari daya pada komponen frekuensi pada sinyal [34]. Spektrum daya masing-masing frame dapat ditentukan dengan persamaan A.4.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N} \quad (\text{A.4})$$

Keterangan:

- $j$  adalah bilangan imajiner
- $N$  adalah panjang sinyal
- $x[n]$  adalah sinyal input

dengan  $x(n)$  adalah sinyal diskrit dan  $N$  adalah panjang dari sinyal.

### A.4 Mel-Frequency Filter Bank

*Mel-Frequency Filter Bank* adalah bank filter yang dibangun berdasarkan persepsi nada. Filter Mel awalnya dikembangkan untuk *speech recognition* dan seperti persepsi telinga manusia terhadap ucapan, filter ini menargetkan ekstraksi representasi nonlinier dari sinyal ucapan. *Mel-Frequency Filter Bank* konvensional dibangun dari 40 filter segitiga [35]. Fungsi transfer (TF) dari masing-masing filter ke- $m$  dapat dihitung melalui persamaan A.5,

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k < f(m) \\ 1 & k = f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \quad (\text{A.5})$$

dengan  $f(m)$  adalah pusat frekuensi dari filter segitiga dan  $\sum_m^{M-1} H_m(k) = 1$ . Skala Mel terhadap frekuensi respons dan sebaliknya dihitung dengan persamaan A.6 dan A.7.

$$m = 2595 \log_{10} \left( 1 + \frac{f}{700} \right) \quad (\text{A.6})$$

$$f = 700 \left( 10^{\frac{m}{2595}} - 1 \right) \quad (\text{A.7})$$

Keterangan:

- $f$  adalah frekuensi
- $m$  adalah skala mel

### A.5 Discrete Cosine Transform (DCT)

*Discrete Cosine Transform* (DCT) menyatakan *finite sequence* dari titik data mengenai penjumlahan fungsi kosinus yang berosilasi pada frekuensi yang berbeda. DCT diperkenalkan oleh Nasir Ahmed pada tahun 1972. Dalam proses MFCC, DCT diterapkan pada bank filter Mel untuk memilih koefisien yang paling akurat atau untuk memisahkan hubungan dalam besaran spektral logaritma dari bank filter [36]. DCT dihitung dengan persamaan A.8

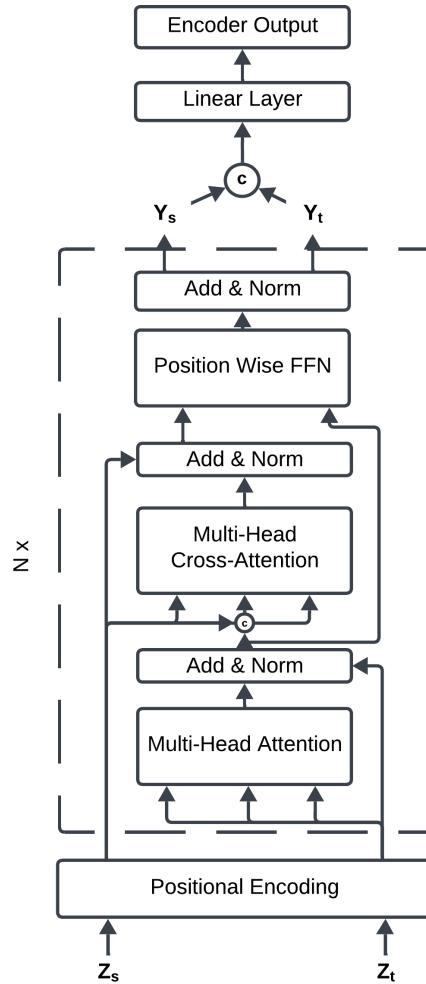
$$X_k = \sum_{n=0}^{N-1} x_n \cos \left( \frac{2\pi j nk}{N} \right), \quad k = 0, 1, \dots, N-1 \quad (\text{A.8})$$

Keterangan:

- $x_n$  adalah sinyal diskrit
- $N$  adalah panjang sinyal
- $X_k$  adalah koefisien MFCC

## LAMPIRAN B

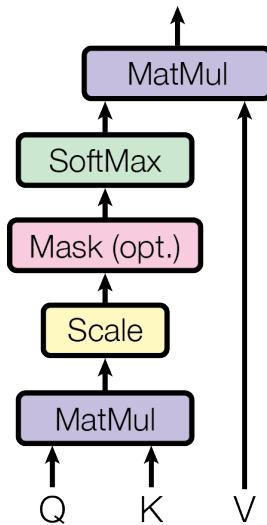
### Perhitungan *Encoder Cross-Attention*



Gambar B.1 Encoder Cross Attention

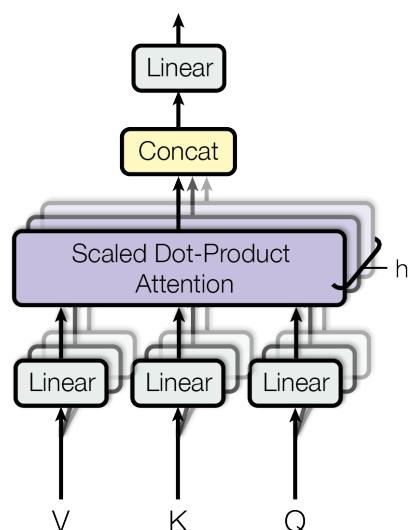
#### B.1 *Self-Attention* dan *Multi-Head Attention*

Proses dari *Self-Attention* atau yang memiliki nama lain *Scaled Dot-Product Attention* dapat dilihat pada Gambar B.2.



**Gambar B.2** *Self-Attention* atau *Scaled Dot-Product Attention*

Proses dari *Multi-Head Attention* dapat dilihat pada Gambar B.3.



**Gambar B.3** *Multi-Head Attention*

Inisiasi nilai variabel diperlukan untuk penentuan nilai awal pada pelatihan. Nilai yang digunakan adalah nilai yang sama dengan paper "*Attention Is All You Need*" dengan beberapa penyesuaian [24]. Nilai variabel inisiasi dapat dilihat pada tabel B.1.

**Tabel B.1** Tabel Variabel untuk Encoder Transformator

Nama Variabel	Nilai Variabel
EMBED_DIM	64
SEQ_LENGTH	512
NUM_FEATURE	8
D_MODELS	8
NUM_HEADS	8
DROPOUT_RATE	0.2
EPOCHS	30

Matriks fitur riwayat pasien  $Z_t$  berukuran  $8 \times 1$  dimasukkan ke dalam layer linear agar memiliki dimensi yang sama dengan matriks hasil ekstraksi fitur MFCC sehingga memiliki ukuran  $8 \times 64$ . Matriks fitur suara  $Z_s$  hasil ekstraksi fitur MFCC berukuran  $512 \times 64$  dan matriks fitur riwayat pasien berukuran  $8 \times 64$  ditambahkan dengan token [cls] menjadi masing-masing berukuran  $513 \times 64$  dan  $9 \times 64$ .

Matriks fitur riwayat pasien lalu masuk ke dalam mekanisme *Multi-Head Attention* dan *Layer Norm* dengan hasil matriks berukuran  $9 \times 64$ . Matriks ini selanjutnya di *concat* dengan matriks fitur suara yang telah ditambah token [cls] berukuran  $513 \times 64$  sehingga menghasilkan matriks berukuran  $521 \times 64$ .

Matriks fitur suara yang telah di *concat* dengan matriks fitur riwayat pasien yang telah melalui mekanisme *Multi-Head Attention* lalu masuk ke dalam mekanisme *Multi-Head Cross Attention*. Operasi yang digunakan pada *Multi-Head Cross Attention* sama dengan *Multi-Head Attention*, perbedaannya terletak pada input berupa gabungan antara fitur suara dan fitur riwayat pasien. Setelah melewati *Layer Norm* ukuran matriks hasil *Multi-Head Cross Attention* menjadi  $513 \times 64$ .

## B.2 Feed-Forward Network

Matriks hasil *Multi-Head Attention* dan *Multi-Head Cross Attention* masing-masing masuk ke dalam mekanisme *Feed-Forward Network*. mekanisme *Feed-Forward Network* terdiri dari tiga operasi utama yaitu *Linear Transformation*, *Activation Function* dan *Layer Normalization*. Transformasi Linear diimplementasikan sebagai *fully connected layer*, atau juga dikenal sebagai *dense layer*, yang menghubungkan setiap neuron masukan ke setiap neuron keluaran. Langkah berikutnya dalam operasi *Feed-Forward Network* adalah menerapkan fungsi aktivasi. Fungsi ini merupakan fungsi non-linier yang memungkinkannya mempelajari pola yang lebih kompleks. Fungsi aktivasi yang digunakan pada

model ini adalah GELU (*Gaussian Error Linear Unit*) yang memiliki kecepatan dan konvergensi yang lebih baik karena telah menggabungkan properti dari dropout, zoneout, dan ReLUs [37]. Persamaan fungsi aktivasi GELU dapat dilihat pada persamaan B.1.

$$\text{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2} \left[ 1 + \text{erf}(x/\sqrt{2}) \right] \quad (\text{B.1})$$

Fungsi aktivasi GELU dapat diaproksimasi dengan persamaan B.2

$$0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)]) \quad (\text{B.2})$$

atau persamaan B.3.

$$x\sigma(1.702x) \quad (\text{B.3})$$

Tahap terakhir pada *Feed-Forward Network* adalah *Layer Normalization*. *Layer Normalization* adalah teknik yang menormalkan masukan di seluruh dimensi fitur (bukan dimensi batch), menstabilkan jaringan dan mempercepat pelatihan. Output dari *Feed-Forward Network* berupa kedua matriks dengan ukuran  $513 \times 64$  dan  $9 \times 64$ .

### B.3 Output Encoder $Y_s$ dan $Y_t$

Proses dari keseluruhan di atas merupakan proses dalam satu blok encoder, proses ini diulang sebanyak N\_BLOCK. Didapatkan vektor  $Y_s$  dan  $Y_t$  dengan ukuran  $1 \times 64$  yang kemudian di *concat* secara horizontal/kolom sehingga didapatkan vektor berukuran  $1 \times 128$ . Vektor ini kemudian masuk ke dalam fungsi aktivasi GELU dan layer Linear sehingga didapatkan vektor berukuran  $1 \times 64$ . Fungsi aktivasi Softmax digunakan untuk mengakomodir klasifikasi multi-kelas dengan menghasilkan probabilitas setiap kelas dengan jumlah semua probabilitas berjumlah tepat 1. Fungsi aktivasi Softmax dapat dilihat pada persamaan B.4.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \quad \text{for } i = 1, \dots, K \quad (\text{B.4})$$

Keterangan:

- $\sigma$  adalah softmax
- $\mathbf{z}$  adalah input vektor
- $K$  adalah jumlah kelas
- $e^{z_i}$  fungsi eksponensial untuk vektor input

- $e^{z_j}$  fungsi eksponensial untuk vektor output

Fungsi B.4 menghasilkan vektor  $1 \times 3$  yang berisi nilai probabilitas dari ketiga kelas. Nilai probabilitas terbesar menunjukkan prediksi kelas dari data.

## LAMPIRAN C

### *Optimizer Adam*

#### C.1 Optimizer Adam pada Transformers

Adam menggabungkan kelebihan dari dua algoritma optimisasi yaitu momentum dan RMSProp dengan mengestimasi rata-rata dan varians dari gradien. Dalam model Transformers, parameter  $\theta$  mencakup bobot dan bias untuk:

- Lapisan *self-attention* dan *multi-head attention* (seperti bobot proyeksi query, key dan value).
- *Feed-Forward Network*.
- Parameter *Layer Normalization*.

Setiap parameter diperbarui menggunakan persamaan di bawah.

##### C.1.1 Persamaan Oprimizer Adam

1. Menghitung gradien dari loss  $L$  dengan parameter  $\theta$  dengan persamaan C.1:

$$g_t = \nabla_{\theta} L(\theta_t) \quad (\text{C.1})$$

2. Perbarui estimasi momentum bias dengan persamaan C.2 dan C.3:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (\text{C.2})$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (\text{C.3})$$

3. Terapkan koreksi bias pada momentum dengan persamaan C.4:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (\text{C.4})$$

4. Perbarui parameter dengan persamaan C.5:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (\text{C.5})$$

Keterangan:

- $g_t$  adalah fungsi gradien
- $L(\theta_t)$  adalah fungsi Loss

- $m_t$  adalah momentum
- $v_t$  adalah velocity
- $\eta$  merupakan *learning rate*.
- $\beta_1$  dan  $\beta_2$  adalah *hyperparameter* yang mengendalikan tingkat peluruhan estimasi momen.
- $\epsilon$  adalah konstanta kecil untuk mencegah pembagian dengan nol.

### C.1.2 Contoh Perhitungan

Asumsikan :

- Learning rate  $\eta = 0.001$ ,
- $\beta_1 = 0.9, \beta_2 = 0.999$ ,
- $\epsilon = 10^{-8}$ ,
- Gradien pada *timestep*  $t$ :  $g_t = 0.02$ ,
- Estimasi momentum sebelumnya:  $m_{t-1} = 0.01, v_{t-1} = 0.0001$ ,
- $t = 10$ .

Langkah-langkah perhitungan:

1. Hitung estimasi momen bias:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t = 0.9 \cdot 0.01 + (1 - 0.9) \cdot 0.02 = 0.011$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 = 0.999 \cdot 0.0001 + (1 - 0.999) \cdot (0.02)^2 = 0.0001004$$

2. Terapkan koreksi bias:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} = \frac{0.011}{1 - 0.9^{10}} \approx 0.0111$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} = \frac{0.0001004}{1 - 0.999^{10}} \approx 0.000102$$

3. Hitung pembaruan parameter:

$$\Delta\theta = \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Subsitusikan nilai:

$$\Delta\theta = 0.001 \cdot \frac{0.0111}{\sqrt{0.000102} + 10^{-8}} \approx 0.00109$$

4. Perbarui parameter:

$$\theta_{t+1} = \theta_t - \Delta\theta$$

## LAMPIRAN D

### Summary Model

#### D.1 Model Base 1

Tabel D.1 Summary Encoder model Base 1

Layer (type)	Output Shape	Param #
dense_60 (Dense)	(None, 512, 256)	2,304
dense_61 (Dense)	(None, 512, 512)	131,584
dense_62 (Dense)	(None, 512)	4,608
positional_encoding_6 (PositionalEncoding)	?	4,608
positional_encoding_7 (PositionalEncoding)	?	262,656
multi_head_attention_36 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_37 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_38 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_39 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_40 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_41 (MultiHeadAttention)	(None, 513, 512)	1,050,624
layer_normalization_36 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_37 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_38 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_39 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_40 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_41 (LayerNormalization)	(None, 513, 512)	1,024
dense_63 (Dense)	(None, 513, 1024)	525,312
dense_64 (Dense)	(None, 513, 1024)	525,312
dense_65 (Dense)	(None, 513, 1024)	525,312
dense_66 (Dense)	(None, 513, 512)	524,800
dense_67 (Dense)	(None, 513, 512)	524,800
dense_68 (Dense)	(None, 513, 512)	524,800
dropout_57 (Dropout)	?	0
dropout_58 (Dropout)	?	0
dropout_59 (Dropout)	?	0
dense_69 (Dense)	(None, 512)	524,800
dropout_60 (Dropout)	?	0
<b>Total params</b>		10,391,808

**Tabel D.2** Summary Classifier Model Base 1

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_70 (Dense)	(None, 512)	262,656
dense_71 (Dense)	(None, 3)	1,539
<b>Total params</b>		<b>264,195</b>

**Tabel D.3** Summary Model Base 1

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
transformer_encoder_block_3	?	10,391,808
classifier_3	?	264,195
<b>Total params</b>		<b>10,656,003</b>

## D.2 Model Base 2

**Tabel D.4** Summary Encoder Block Model Base 2

Layer (type)	Output Shape	Param #
dense_72 (Dense)	(None, 768, 256)	2,304
dense_73 (Dense)	(None, 768, 768)	197,376
dense_74 (Dense)	(None, 768)	6,912
positional_encoding_8 (PositionalEncoding)	?	6,912
positional_encoding_9 (PositionalEncoding)	?	590,592
multi_head_attention_42 (MultiHeadAttention)	(None, 2, 768)	2,362,368
multi_head_attention_43 (MultiHeadAttention)	(None, 2, 768)	2,362,368
multi_head_attention_44 (MultiHeadAttention)	(None, 2, 768)	2,362,368
multi_head_attention_45 (MultiHeadAttention)	(None, 769, 768)	2,362,368
multi_head_attention_46 (MultiHeadAttention)	(None, 769, 768)	2,362,368
multi_head_attention_47 (MultiHeadAttention)	(None, 769, 768)	2,362,368
layer_normalization_42 (LayerNormalization)	(None, 2, 768)	1,536
layer_normalization_43 (LayerNormalization)	(None, 2, 768)	1,536
layer_normalization_44 (LayerNormalization)	(None, 2, 768)	1,536
layer_normalization_45 (LayerNormalization)	(None, 769, 768)	1,536
layer_normalization_46 (LayerNormalization)	(None, 769, 768)	1,536
layer_normalization_47 (LayerNormalization)	(None, 769, 768)	1,536
dense_75 (Dense)	(None, 769, 1536)	1,181,184
dense_76 (Dense)	(None, 769, 1536)	1,181,184
dense_77 (Dense)	(None, 769, 1536)	1,181,184
dense_78 (Dense)	(None, 769, 768)	1,180,416
dense_79 (Dense)	(None, 769, 768)	1,180,416
dense_80 (Dense)	(None, 769, 768)	1,180,416
dropout_67 (Dropout)	?	0
dropout_68 (Dropout)	?	0
dropout_69 (Dropout)	?	0
dense_81 (Dense)	(None, 768)	1,180,416
dropout_70 (Dropout)	?	0
<b>Total params</b>		<b>23,254,272</b>

**Tabel D.5** Summary Model Classifier

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_82 (Dense)	(None, 768)	590,592
dense_83 (Dense)	(None, 3)	2,307
<b>Total params</b>		<b>592,899</b>

**Tabel D.6** Summary Model Base 2

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
transformer_encoder_block_4	?	23,254,272
classifier_4	?	592,899
<b>Total params</b>		<b>23,847,171</b>

### D.3 Model Big 1

**Tabel D.7** Summary Encoder Block

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_42 (Dense)	(None, 512, 256)	2,304
dense_43 (Dense)	(None, 512, 512)	131,584
dense_44 (Dense)	(None, 512)	4,608
positional_encoding_4 (PositionalEncoding)	?	4,608
positional_encoding_5 (PositionalEncoding)	?	262,656
multi_head_attention_24 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_25 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_26 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_27 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_28 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_29 (MultiHeadAttention)	(None, 2, 512)	1,050,624
multi_head_attention_30 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_31 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_32 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_33 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_34 (MultiHeadAttention)	(None, 513, 512)	1,050,624
multi_head_attention_35 (MultiHeadAttention)	(None, 513, 512)	1,050,624
layer_normalization_24 (LayerNormalization)	(None, 2, 512)	1,024

*Continued on next page*

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
layer_normalization_25 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_26 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_27 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_28 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_29 (LayerNormalization)	(None, 2, 512)	1,024
layer_normalization_30 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_31 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_32 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_33 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_34 (LayerNormalization)	(None, 513, 512)	1,024
layer_normalization_35 (LayerNormalization)	(None, 513, 512)	1,024
dense_45 (Dense)	(None, 513, 1024)	525,312
dense_46 (Dense)	(None, 513, 1024)	525,312
dense_47 (Dense)	(None, 513, 1024)	525,312
dense_48 (Dense)	(None, 513, 1024)	525,312
dense_49 (Dense)	(None, 513, 1024)	525,312
dense_50 (Dense)	(None, 513, 1024)	525,312
dense_51 (Dense)	(None, 513, 512)	524,800
dense_52 (Dense)	(None, 513, 512)	524,800
dense_53 (Dense)	(None, 513, 512)	524,800
dense_54 (Dense)	(None, 513, 512)	524,800
dense_55 (Dense)	(None, 513, 512)	524,800
dense_56 (Dense)	(None, 513, 512)	524,800
dropout_38 (Dropout)	?	0
dropout_39 (Dropout)	?	0
dropout_40 (Dropout)	?	0
dropout_41 (Dropout)	?	0
dropout_42 (Dropout)	?	0
dropout_43 (Dropout)	?	0
dense_57 (Dense)	(None, 512)	524,800
dropout_44 (Dropout)	?	0
<b>Total params</b>		<b>19,852,032</b>

**Tabel D.8** Summary Classifier

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_58 (Dense)	(None, 512)	262,656
dense_59 (Dense)	(None, 3)	1,539
<b>Total params</b>		<b>264,195</b>

**Tabel D.9** Summary Model Big 1

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
TransformerEncoderBlock	?	19,852,032
Classifier	?	264,195
<b>Total params</b>		<b>20,116,227</b>

## D.4 Model Big 2

**Tabel D.10** Summary Encoder Block

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_24 (Dense)	(None, 1024, 256)	2,304
dense_25 (Dense)	(None, 1024, 1024)	263,168
dense_26 (Dense)	(None, 1024)	9,216
positional_encoding_2 (PositionalEncoding)	?	9,216
positional_encoding_3 (PositionalEncoding)	?	1,049,600
multi_head_attention_12 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_13 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_14 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_15 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_16 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_17 (MultiHeadAttention)	(None, 2, 1024)	4,198,400
multi_head_attention_18 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
multi_head_attention_19 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
multi_head_attention_20 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
multi_head_attention_21 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
multi_head_attention_22 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
multi_head_attention_23 (MultiHeadAttention)	(None, 1025, 1024)	4,198,400
layer_normalization_12 (LayerNormalization)	(None, 2, 1024)	2,048

*Continued on next page*

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
layer_normalization_13 (LayerNormalization)	(None, 2, 1024)	2,048
layer_normalization_14 (LayerNormalization)	(None, 2, 1024)	2,048
layer_normalization_15 (LayerNormalization)	(None, 2, 1024)	2,048
layer_normalization_16 (LayerNormalization)	(None, 2, 1024)	2,048
layer_normalization_17 (LayerNormalization)	(None, 2, 1024)	2,048
layer_normalization_18 (LayerNormalization)	(None, 1025, 1024)	2,048
layer_normalization_19 (LayerNormalization)	(None, 1025, 1024)	2,048
layer_normalization_20 (LayerNormalization)	(None, 1025, 1024)	2,048
layer_normalization_21 (LayerNormalization)	(None, 1025, 1024)	2,048
layer_normalization_22 (LayerNormalization)	(None, 1025, 1024)	2,048
layer_normalization_23 (LayerNormalization)	(None, 1025, 1024)	2,048
dense_27 (Dense)	(None, 1025, 2048)	2,099,200
dense_28 (Dense)	(None, 1025, 2048)	2,099,200
dense_29 (Dense)	(None, 1025, 2048)	2,099,200
dense_30 (Dense)	(None, 1025, 2048)	2,099,200
dense_31 (Dense)	(None, 1025, 2048)	2,099,200
dense_32 (Dense)	(None, 1025, 2048)	2,099,200
dense_33 (Dense)	(None, 1025, 1024)	2,098,176
dense_34 (Dense)	(None, 1025, 1024)	2,098,176
dense_35 (Dense)	(None, 1025, 1024)	2,098,176
dense_36 (Dense)	(None, 1025, 1024)	2,098,176
dense_37 (Dense)	(None, 1025, 1024)	2,098,176
dense_38 (Dense)	(None, 1025, 1024)	2,098,176
dropout_19 (Dropout)	?	0
dropout_20 (Dropout)	?	0
dropout_21 (Dropout)	?	0
dropout_22 (Dropout)	?	0
dropout_23 (Dropout)	?	0
dropout_24 (Dropout)	?	0
dense_39 (Dense)	(None, 1024)	2,098,176
dropout_25 (Dropout)	?	0
<b>Total params</b>		<b>79,023,360</b>

**Tabel D.11** Summary Classifier

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
dense_40 (Dense)	(None, 1024)	1,049,600
dense_41 (Dense)	(None, 3)	3,075
<b>Total params</b>		<b>1,052,675</b>

**Tabel D.12** Summary Model Big 2

<b>Layer (Type)</b>	<b>Output Shape</b>	<b>Param #</b>
TransformerEncoderBlock	?	79,023,360
Classifier	?	1,052,675
<b>Total params</b>		<b>80,076,035</b>